BibPro: A Citation Parser Based on Sequence Alignment

Chien-Chih Chen, Kai-Hsiang Yang, Chuen-Liang Chen, and Jan-Ming Ho, *Member*, *IEEE*

Abstract—Dramatic increase in the number of academic publications has led to growing demand for efficient organization of the resources to meet researchers' needs. As a result, a number of network services have compiled databases from the public resources scattered over the Internet. However, publications by different conferences and journals adopt different citation styles. It is an interesting problem to accurately extract metadata from a citation string which is formatted in one of thousands of different styles. It has attracted a great deal of attention in research in recent years. In this paper, based on the notion of sequence alignment, we present a citation parser called BibPro that extracts components of a citation string. To demonstrate the efficacy of BibPro, we conducted experiments on three benchmark data sets. The results show that BibPro achieved over 90 percent accuracy on each benchmark. Even with citations and associated metadata retrieved from the web as training data, our experiments show that BibPro still achieves a reasonable performance.

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

1 Introduction

Citations play an important role in many scientific-publication digital libraries (DLs), such as CiteSeer, arXiv e-Print, DBLP, and Google Scholar. Users often use citations to find information of interest in DLs, while researchers depend on citations to determine the impact of a particular article. Evaluations of an individual's performance for promotion purposes or the allocation of grants may use citations as evidence of the competence of a researcher and the impact of his/her published work. Citations have also been used as auxiliary support in information retrieval tasks, e.g., automatic document classification [2], [3], indexing and ranking [10], and quality assessment [4]. Moreover, bibliographic measures that rely on citations have inspired recent web link analysis algorithms like PageRank [5]. In a broader sense, citations are the basis of DLs that specialize in scientific publications.

Parsing citations is essential for integrating bibliographical information published on the Internet. Most citation management techniques are based on the assumption that we can correctly identify the main components of a citation, such as authors' names, title, publication venue, date, and the number of pages. However, for a variety of reasons, it is difficult to design a parser that can automatically parse citations scattered over the Internet [1]. Potential problems include data entry errors, diverse citation formats, the lack of (enforcement of) a standard, imperfect citation gathering software, common author names, abbreviations of publication venues, and large-scale citation data.

In this paper, we propose a sequence-alignment-based citation parser, called BibPro, to extract components of citations in any given formats. The basic idea of BibPro is to capture the structural properties from semistructured format and transform these properties into a sequence template. The structural properties of a citation string include the order of punctuation marks and local structure in each field of a citation string. We use an encoding table and reserved words, which is automatically trained from the data set, to represent each semantic unit as a unique symbol; and use a blocking process to capture local structure in each citation field. After building up sufficient amount of encoded sequence templates, BibPro then applies a sequence alignment software, e.g., Basic Local Alignment Search Tool (BLAST), to match the query citation string with the sequence templates.

Using the sequence templates gives BibPro the following advantages:

1. By encapsulating the semantic units of a citation string into a simple symbol, BibPro reduces the complexity of a citation string and can easily utilize the generic structural properties of citations.
2. By transforming structural properties into sequence, we can use sequence techniques, such as search, matching and alignment to compare structural properties.
3. By adapting a suitable score matrix for sequence matching and sequence alignment, BibPro has a high fault tolerance of a citation whose format was not built in the template database.

• *C.-C. Chen and C.-L. Chen are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC. E-mail: rocky@iis.sinica.edu.tw, clchen@csie.ntu.edu.tw.*
• *K.-H. Yang is with the Department of Mathematics and Information Education, National Taipei University of Education, Taipei, Taiwan, ROC. E-mail: khyang@tea.ntue.edu.tw.*
• *J.-M. Ho is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC. E-mail: hoho@iis.sinica.edu.tw.*

Manuscript received 9 Apr. 2010; revised 7 June 2010; accepted 14 July 2010; published online 12 Nov. 2010.
Recommended for acceptance by J. Yang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-04-0209.
Digital Object Identifier no. 10.1109/TKDE.2010.231.

4. It is easy to verify the correctness of structural properties through the sequence format; thus, we can easily construct a template database from the web data. As a result, our technique is highly automated and extremely flexible, as shown by our experiment results.

To evaluate the performance and practicality of our approach, we conducted experiments on three benchmark data sets, namely, the Cora, Flux-CiM, and INFOMAP data sets. The Cora data set [14] is widely used as the benchmark for machine learning techniques. The Flux-CiM data set [13] contains citations from two domains: Computer Science (CS) and Health Science (HS); and the INFOMAP data set [11] contains 160,000 citation records in six different citation styles. The results of the experiments show that, on average, BibPro achieved over 80 percent field accuracy on each benchmark with the same template database, which was compiled automatically from the Internet. In addition, the average field accuracy can be further improved to 90 percent by constructing a partial template from the data set itself.

The remainder of this paper is organized as follows: Section 2 contains a review of related works. We reformulate citation parsing problem to explain the concepts behind BibPro in Section 3. In Section 4, we describe the architecture of BibPro. In Section 5, we describe the experiments and analyze the results. Section 6 contains some concluding remarks and suggests avenues for future research.

2 Related Work

The problem of citation extraction aims at automatically segmenting unstructured text strings into structured records. In recent years, several methods have been proposed to address a related problem of data extraction from textual documents, with the focus on documents available on the web. A brief survey of these methods is given in [6]. They include HTML structural analysis, natural language processing, machine learning, data modeling, and ontology. For citation domain, numerous works reported in the literature, e.g., [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] use similar concepts to extract metadata from citations. The approaches can be roughly classified into two categories: learning-based and knowledge-based approaches.

The learning-based approach transforms the citation extraction problem into a classification problem and uses machine leaning techniques to solve it. Currently, three major machine learning techniques, the Hidden Markov Model (HMM) [17], [18], [19], [20], [21], Support Vector Machines (SVM) [16], and Conditional Random Fields (CRF) [15], [29], are used to extract information from research papers. Although these approaches have good adaptability and perform well on the Cora data set, they have some limitations. For example, the quality of training data directly affects the performance of learning-based approaches; hence, correctness of the training data and whether that data are representative of the test data are important issues. As a result, compiling a suitable training data set is a challenging task for learning-based approaches.

Knowledge-based approaches utilize domain knowledge to derive ontology that describes the data of interest, where the knowledge includes relationships, lexical appearances, and context keywords. By parsing the ontology, several rules and an extractor can be generated, which are then used to perform information extraction. Such approaches are widely used in real-world applications. For example, CiteSeer [7], [8], [9], [10] is a well-known search engine and digital library that uses heuristics to extract metadata from citations. It identifies titles and author names in citations with roughly 80 percent accuracy and page numbers with roughly 40 percent accuracy [7]. However, it requires a domain expert to maintain the knowledge database. Other approaches include CRAM [12], INFOMAP [11], and FLUX-CiM [13]. CRAM mines tables in data warehouses and relational databases to develop an automatic segmentation system. INFOMAP is a tree-like knowledge representation scheme that organizes the knowledge of reference concepts in a hierarchical fashion. It has an overall average accuracy rate of 92.39 percent for the six major citation styles. Meanwhile, Flux-CiM automatically constructs ontology from an existing set of sample metadata records for a given area. The precision and recall rates are above 94 percent for both the HS data set and the CS data set.

In this paper, we propose a new citation parser called BibPro, which retains the advantages of our previous work [22], [23] (e.g., it uses protein sequences to represent citations and employs BLAST to find similar templates), and integrate the concept of knowledge-based approach and learning-based approach. Instead of relying on a knowledge database and heuristic rules, BibPro developed a canonicalization algorithm to systematically capture the structural features of a citation string and store these features in a sequence template. To maintain a sequence template database, BibPro needs some labeled citation string to construct sequence template and trains a score matrix for sequence alignment like other supervised machine learning approach. While the role of sequence template database is like an external reference to aid citation parsing, it only partially influences the performance of BibPro. Besides, we choose the structure of citations (such as the order of fields and how many fields appear in citations) as the global features used in sequence template, while other learning-based approaches focus on features extracted from content of citations. Hence, the diversity of global features of citation strings is much less than the textual features of citation strings; we can easily obtain sufficient global feature information from few labeled citation strings. Thus, only few sequence templates can help BibPro to achieve a stable performance. A research initiative closely related to our work is called ParaCite,[1] which has been integrated with the EPrints.org software.

3 Problem Formulation

In this paper, we refer to a citation string as a textual string which is used to present the metadata of an academic publication in a specific formatting style. Typical elements of a metadata include author, title, and publication

1. http://paracite.eprints.org/.

information. These fields are usually separated by punctuation marks in a citation string. We then formulate citation parsing problem as follows: given a semistructured citation string S, S = {Field$_1$ Delimiter$_1$ Field$_2$ Delimiter$_2$ Field$_3$ Delimiter$_3$ ... }, where Field$_i$ are fields in the metadata, and Delimiter$_i$ are some symbols to separate fields. In this paper, we are interested in retrieving the following seven citation fields in a citation string: Author, Title, Venue, Volume, Issue, Page, and Date.

4 BibPro: A Citation Parser

In our observation, given its metadata, a citation string can be rewritten into a canonical string consisting of symbols corresponding to its Fields and Delimiters. Two citation strings of the same citation format usually have similar canonical strings. On the other hand, canonical strings of two citation strings of different format are not quite similar.

Our strategy of solving the citation parsing problem, thus, is to rewrite a given citation string into a canonical form. However, without prior knowledge of its metadata, the boundary between fields and delimiters of a citation string are not trivial. For example, some punctuation marks used as delimiters may also appear inside a certain field. Besides, sometimes it is also hard to determine the type of a field by its content, and the order of fields varies among different citation formats. Thus, our strategy is to rewrite one such citation string such that structured information associated with its citation style is preserved as much as possible while textual information of the specific citation string is removed as much as possible.

Our methodology, call BibPro, then stores a set of citation strings with given metadata into the template database by storing for each citation string its canonical string and addition style information denoted as "sequence template." A query citation string can then be processed into its canonical string and matched against canonical strings stored in the sequence template database. Style information of the similar canonical string is then retrieved and used to parse the query string into its metadata.

BibPro can be divided into two primary modules, i.e., template database construction and query processing as shown in Fig. 1. In constructing template database, we are given a set of pairs of citation string and associated metadata. In the template database, structural features information, i.e., INDEX FORM and STYLE FORM, of each citation string is stored. A STYLE FORM is a symbolic representation of a citation string in which each field of metadata as well as each punctuation mark is represented by a single symbol, respectively. INDEX FORM is a canonical symbolic representation of a citation string so that it will be used later to align with a given query citation string. In query process module, a given query string is first converted into its INDEX FORM using the canonicalization algorithm. The query INDEX FORM is then used to search the template database for similar INDEX FORMS. A detailed alignment algorithm is performed on STYLE FORM of templates with INDEX FORMS similar to that of the given query citation string. We then make use of detail alignment information to give the final parser output.

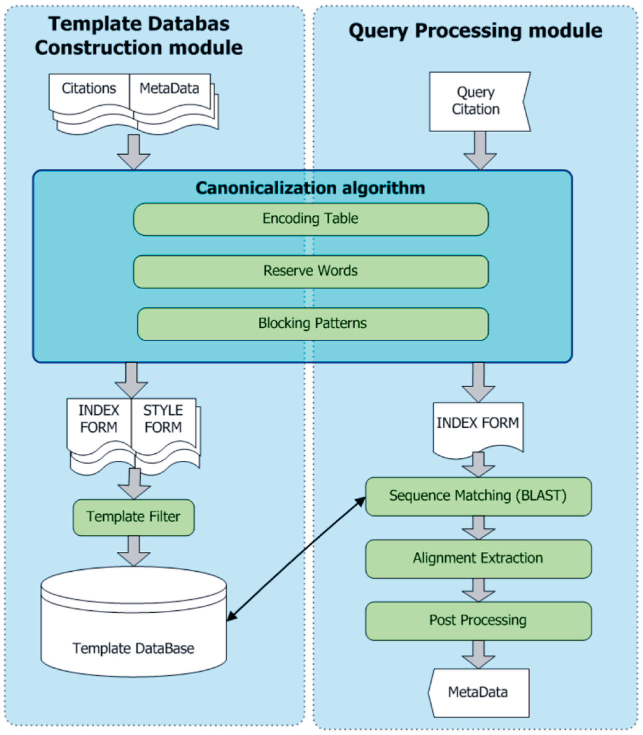


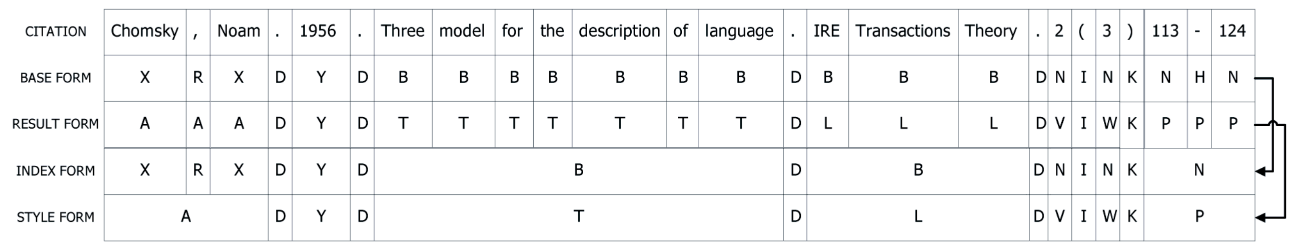
Fig. 1. The workflow of BibPro.

In Section 4.1, we present the canonicalization algorithm; more details of template database construction modules and query processing module are described in Sections 4.2 and 4.3, respectively.

4.1 Canonicalization Algorithm

The format of a citation string is semistructured. Although it is difficult to find regular rules to parse the structure precisely, we can easily recognize some common features in citation strings; for example, double quotation marks are often used as separator to indicate the title field. However, these common features cannot be used directly to devise rules because they contain various levels of noise. For example, double quotation marks also appear inside the title field, which confuses the parser when extracting that field. Therefore, to reinforce the structural features and filter out redundant noise, we use a canonical string to represent structural feature of citation string. The structural features include the fields that appear in the citation string, the kind of delimiter between each field, and the combination order of fields in the citation string. We considered the many sequence alignment techniques in bioinformatics and choose protein sequence as canonical string to represent structural feature of citation string.

To capture the structural properties from semistructured citation string, we design a canonicalization algorithm in BibPro. The canonicalization algorithm is based on the observation: the combination of some term in a field (reserved word) and punctuation mark may have enough knowledge to describe the structure of citation. Thus, the canonicalization algorithm is designed to extract this knowledge in a systemically way. The design issue includes: 1) Representing citation string in a simple format



CITATION	Chomsky	,	Noam	.	1956	.	Three	model	for	the	description	of	language	.	IRE	Transactions	Theory	.	2	(	3	)	113	-	124
BASE FORM	X	R	X	D	Y	D	B	B	B	B	B	B	B	D	B	B	B	D	N	I	N	K	N	H	N
RESULT FORM	A	A	A	D	Y	D	T	T	T	T	T	T	T	D	L	L	L	D	V	I	W	K	P	P	P
INDEX FORM	X	R	X	D	Y	D	B							D	B			D	N	I	N	K	N		
STYLE FORM	A			D	Y	D	T							D	L			D	V	I	W	K	P		

Fig. 2. An example of a BASE FORM, RESULT FORM, INDEX FORM, and STYLE FORM.

to reduce the diversity of textual feature and keep it structural information. 2) Finding the reserved words which is representative and includes the structural knowledge. 3) Finding the dependency between reserved word and punctuation mark to reconstruct the structure of citation. We divide canonicalization algorithm into three components: 1) encoding table, 2) reserved words, and 3) blocking patterns. The encoding table assigns each meaningful semantic unit a symbol to simplify the complexity of data format (see Section 4.1.1). Reserved words and blocking patterns are used to capture the contextual features and local structure inside fields (see Sections 4.1.2 and 4.1.3). According to the canonicalization algorithm, we define the BASE FORM and INDEX FORM to represent a query citation string, and define the RESULT FORM and STYLE FORM to represent structural information of a citation style. Fig. 2 shows an example of BASE FORM, RESULT FORM, INDEX FORM, and STYLE FORM of a citation string. More details will be described in the following section.

4.1.1 Encoding Table

Encoding table is used to map tokens in citation strings to symbols in protein sequences. We use the symbol to encapsulate the semantic units of the citation string, including metadata elements and punctuation marks. To do this, we needed to resolve the following questions:

- How many symbols can be used in a protein sequence?
- How many fields should be extracted from a citation string?
- Which kinds of punctuation marks should be treated as delimiter?
- How can we transform a citation string into a protein sequence and retain its structural features?

To address above questions, we collected an annotated citation strings data set from the web and created an encoding table according to the occurrence statistics of tokens, shown as Table 1. The design of the encoding table is based on the following principles:

- We classify tokens of citation strings into two categories: field token and delimiter token. We considered BLAST only processes sequences with 23 symbols, so we use the 23 symbols to represent different fields and delimiters to keep the citation's structural feature in the sequence.
- Fields tokens. The most common fields in a citation string are: author, title, venue, volume, issue, page, and date. We focus on extracting these fields from a citation string and assign a symbol for each field. Note that, we also use symbols to present the editor, institution, and publisher fields because they may confuse the parser and cause mistakes. For example, the contents of author field and editor field are both personal name, so we need to use additional symbols to discriminate them. Besides, when processing a citation string, we do not know its metadata in advance; so many tokens cannot be recognized directly. For example, a token "of" may belong to the title field or the venue field, while a token "11" may belong to the volume field or the

TABLE 1
Encoding Table

Category	Symbol	Representation Field
Field Token	A	Author
	T	Title
	L	Venue (Journal, BookTitle, Technical Report)
	V	Volume
	W	Issue
	P	Page
	Y	Date (Year Month)
	F	Editor
	S	Institution
	M	Publisher
	X	Unknown Single Token
	B	Unknown Continuous Token
	N	Numeral Token
Delimiter Token	R	,
	D	.
	G	"
	E	'
	C	:
	Z	;
	H	-
	I	([< {
	K	)] > }
	Q	/ _ ! @ # $ % ^ & * + = \ | ? ◦ ~

page field. However, these tokens still provide some information that could help the parser make decisions. Therefore, we assign a symbol to each of these unknown tokens, such as a numeral token "N," a single unknown token "X," or continuous unknown tokens "B."

- Delimiter tokens. In citation format, punctuation marks are always treated as delimiters between different fields. We calculate the occurrence times of all punctuation marks which be treated as delimiter in the annotated citation string data set and assign different symbols to the top nine most appeared ones (here, Parentheses are a pair of curved marks that we assign two symbols I and K). However, some punctuation marks are used within a field: for example, the punctuation marks "!," "?" may appear in the title field, while "-" always appears in the page field. To address this issue, we use one symbol "Q" to represent punctuation marks that do not have a partitioning function.

Note that, when we have a citation string and its associated metadata, we can directly transform each token of a citation string into its corresponding symbol. We call this sequence RESULT FORM as shown in Fig. 2. A citation parsing problem is equal to transform a citation string into a RESULT FORM.

4.1.2 Reserved Words

In the encoding process of a query citation string, we do not know the metadata in advance. We use reserved word to recognize the token belongs to which fields. A reserved word defined here is a term that only appears in a specific field. For example, "pp" is a reserved word for the page field, and "vol" is a reserved word for the volume field. However, not all fields have explicit reserved words. We design an automatic way to select reserved words for each field of a citation string. The selection principle is that words which have higher term frequency in a specific field than in other fields should be treated as reserved words for that field; while words that occur frequently in all fields should be avoided. The selection process computes a weight for each term in an annotated citation string data set by using the following formula. For a term t_i within the particular citation field f_j, the term frequency is defined as follows:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

where $n_{i,j}$ is the number of occurrences of the term t_i in the field f_j, and the denominator is the sum of number of occurrences of all terms in field f_j.

The inverse field frequency (*iff*) is a measure of the general importance of the term (obtained by dividing the number of all fields by the number of fields containing the term, and then taking the logarithm of that quotient).

$$iff_i = \log \frac{|F|}{|\{f : t_i \in f\}|},$$

where $|F|$ is the total number of fields in the data set, $|\{f : t_i \in f\}|$ is the number of fields where the term t_i appears. Then, the weight of term t_i for field f_j can be computed as follows:

$$(tf - iff)_{i,j} = tf_{i,j} \times iff_i.$$

A high weight in *tf-iff* is reached by a high term frequency (in a specific field) and a low field frequency of the term in the whole citation string data set; the weights hence tend to filter out common terms.

If we use too many reserved words in the process, the encoding results would be data set dependent. To reduce such effect, we only select few reserved words in each field. Besides, we also use common knowledge for some specific fields. For example, we adopt the name abbreviation as an important feature in the author field. Table 2 shows the reserved words we use for each field. Note that journal, book title, and technical report are subdivisions of the venue field, and the month and year are subdivisions of the date field.

According to the reserved word, we can encode a citation string into a protein sequence without metadata. We call this sequence BASE FORM, as shown in Fig. 2. Thus, the goal of the parsing process is to transform a citation string from its BASE FORM into its RESULT FORM.

TABLE 2
The Reserved Words Used in BibPro

Field	Encoded Knowledge or Keyword
Author	Name abbreviation
Journal	"Transactions", "Trans", "Journal"
Booktitle (Conference)	"Proceedings", "Proc", "Workshop", "Conf", "Conference", "Symposium", "Sympos", "Symp", "International", "Intern", "Annual", "Annu"
Technical Report	"Tech", "rep", "Rpt", "TR", "Master", "Masters", "Ph", "PhD", "Thesis", "thesis", "Dissertation", "dissertation"
Volume	"Volume", "volume", "Vol", "vol", "Vo", "vo"
Issue (Number)	"Number", "number", "Nr", "nr", "No", "no", "NO", "Nos"
Page	"pp", "page", "pages", "PP", "Page", "Pages", "pg", "PG"
Month	"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December", "Jan", "Feb", "Mar", "Apr", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec", "Sept"
Year	1900-2010
Editor	"eds", "Eds", "editors", "Editors", "editor", "Eds", "ED", "Ed", "ed", "edited"
Institution	"University", "Univ", "Department", "Dept", "Corporation"
Publisher	"Press", "Pub", "Publishers", "Inc", "Publications"

TABLE 3
REGULAR EXPRESSION for Author, Venue, Volume, PAGE and ISSUE Fields

Field	Regular Expression
Author	A[^LVWPYSF]+A
	A[^LVWPYSF]*[BX]
	[BX][^LVWPYSF]*A
Venue	L[^AVWPYSF]+L
	L[^AVWPYSF]*[BX]
	[BX][^AVWPYSF]*L
Volume	V[^ALWPYSF]*N
Page	P[^ALVWYSF]*N
Issue	W[^ALVPYSF]*N

4.1.3 Blocking Patterns

After defining the reserved words, our next step is to find the dependency between reserved words and punctuation mark inside each citation field. We treat these dependencies as local structure of citation string, because some regulations always are used inside citation fields, like the abbreviation of author name and the format of page number. By encoding citation string into BASE FORM, these local structures can be easily recognized as some specific patterns often appear in specific fields. For example, "ADXRA" is a pattern that often appears in the author field, and its structure starts with an "A" and ends with an "A." We can easily recognize that pattern "ADXRA" belongs to the author field; similarly, title field is always composed of continuous unknown tokens, such as "BBB . . . ," and page field always has patterns like "PDN." Since these patterns express common structure rules for all citation styles, we can use them to reinforce the structural features of a citation string. According to these patterns, we can group tokens in the same field into a block unit and assign a representative symbol, which can keep the integrity of semistructured information and can simplify the complexity of the problem. We call it blocking pattern.

In order to efficiently collect block patterns, we transform an annotated citation strings data set into BASE FORM, and use two criteria to select block patterns: pattern frequency and regular expression. Since blocking patterns always appear in higher frequency in specified field, we can use the occurrence of frequency as a simple filter to select block patterns. However, some high frequency patterns like the combination of unknown token and punctuation mark cannot be directly blocked as one unit, because it may loss the structural information. Therefore, we need some domain knowledge to distinguish between blocking patterns and unblocking patterns. According to the classification of encoding table, we can use regular expression to describe simple domain knowledge of a set of patterns. For example, "$A[\wedge LVWPYSF]+A$" expresses the structure starting with an "A" and ending with an "A," and the structure does not include other field's reserved word. The physical meaning of this regular expression is that reserved words in the author field may have punctuation marks between each other, but these punctuation marks do not have partition function; thus, we can use the regular expression to collect blocking patterns in author field, which are used to deal with punctuation marks inside author field. Similarly, the same concept can be used to collect blocking patterns in other field. Table 3 shows the regular expression we used in each field to search blocking patterns. According to the pattern frequency and regular expression, we can easily obtain blocking patterns from an annotated citation string data set (see Supplemental Material for block patterns we used in our experiment, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.231).

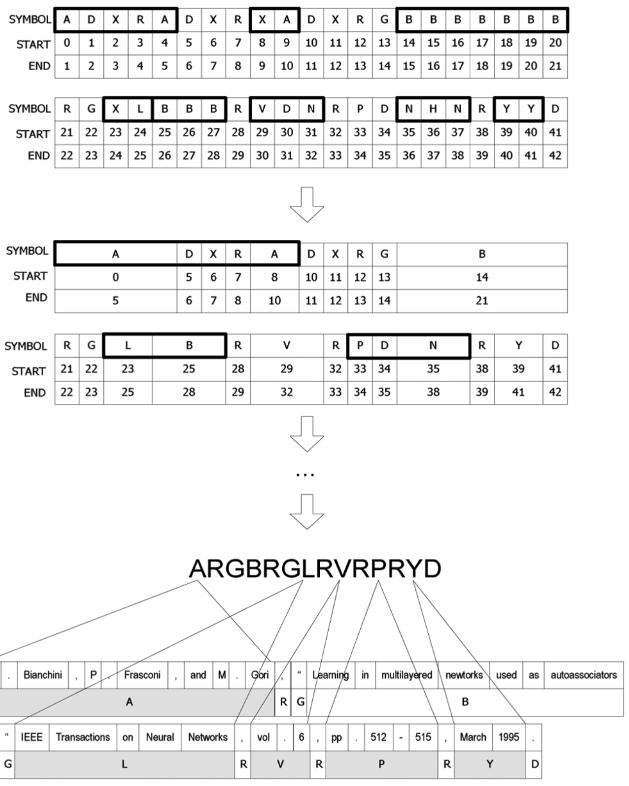

SYMBOL
START
END
ARGBRGLRVRPRYD
M . Bianchini , P . Frasconi , and M . Gori , " Learning in multilayered newtorks used as autoassociators
" IEEE Transactions on Neural Networks , vol . 6 , pp . 512 - 515 , March 1995 .

Fig. 3. An example of the blocking process.

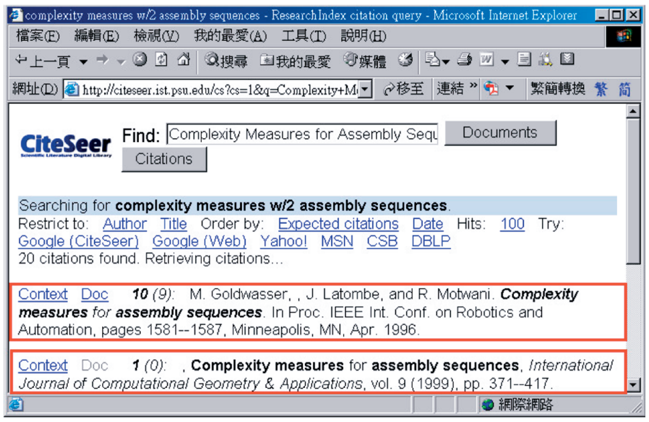
Fig. 3 shows an example of the blocking process. If a sequence has a substring equal to the block pattern, we use a representative symbol to replace the pattern and record its start position and end position. Then, we repeat the above process until no more patterns can be found in the sequence. For example, "ADXRA" and "XA" are block patterns in author field. The substring "ADXRADXRXA" can be transformed to "ADXRXA" in step 1, and then be transformed to "ADXRA" in step 2. Finally, it will be transformed to "A." If the sequence has enough reserved words and pattern information, several fields can be recognized after the blocking process; for example, in Fig. 3, the author, venue, volume, page, and date fields can be recognized.

In canonicalization algorithm, a citation string can be encoded to a BASE FORM or a RESULT FORM. The difference is that the RESULT FORM is encoded with its associated metadata, whereas the BASE FORM is encoded according to reserved words. The BASE FORM and RESULT FORM can be further processed to INDEX FORM and STYLE FORM. The INDEX FORM is transformed from the BASE FORM according to the blocking process. Since the INDEX FORM is constructed without metadata and it contains information about the order of punctuation marks, which can be used to partially represent the structure of a citation string, we can use it to determine which citation string has a similar structure. Because it is like an index used in a conventional database, we call this sequence "INDEX FORM." The STYLE FORM is transformed from the RESULT FORM and is used to maintain the citation's structural features. Although the RESULT FORM can represent style information for a specific citation string, it contains a great deal of excess information, such as the length of the author, title, and journal fields. To address this problem, we adopt the concept of blocking process. Specifically, we condense the redundant information in the RESULT FORM by using a symbol to represent each field in the citation string. Since this sequence generalizes the citation style information of RESULT FORM, we call it "STYLE FORM." INDEX FORM and STYLE FORM enable us to store information about a citation's style in a database.

4.2 Template Database Construction

We define a sequence template to store structural feature of citation format. A sequence template is a pair of two sequences: INDEX FORM and STYLE FORM which are transformed from the same citation string and associated metadata. The goal of template database construction modules is to construct a database of sequence templates automatically. We divide the template database construction modules into two phases. The first phase focuses on collecting data, including citation strings and their associated metadata. The second phase processes these data to compile sequence templates. In the first phase, we can let the user input data or we can collect data from the Internet. We use a BibTeX file as the initial input because it is easy to retrieve such files from the Internet. Since the files are field based, we can easily parse them to obtain the metadata for a citation string. Then, we use the title field as a query to search for a citation in CiteSeer or other search engine, e.g., Google. In this way, we can obtain citation strings and their associated metadata.

In the second phase, using the metadata of each citation, it is easy to transform the citation string into its STYLE FORM and INDEX FORM (see the canonicalization algorithm). We treat these two sequences as one record in the template database. However, we do not store the record in the database directly because the data collected from Internet may be inconsistent with the metadata of BibTeX file. For example, Fig. 4 shows two citation strings retrieved from CiteSeer by inputting the same title and they have different metadata. Thus, we should check which citation string is consistent with the metadata of the BibTeX file. Moreover, even if only one citation string found by CiteSeer, the format of metadata of BibTeX file may be inconsistent with the format of metadata in found citation string. Since the construction of RESULT FORM and STYLE FORM depends on the correct metadata information, we


complexity measures w/2 assembly sequences - ResearchIndex citation query - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://citeseer.ist.psu.edu/cs?cs=1&q=Complexity+M
移至 連結 繁簡轉換 繁 简
CiteSeer
Find:
Documents
Citations
網際網路

Find: Complexity Measures for Assembly Sequ

Searching for **complexity measures w/2 assembly sequences**.
Restrict to: Author Title Order by: Expected citations Date Hits: 100 Try:
Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP
20 citations found. Retrieving citations...

Context Doc ***10*** *(9)*: M. Goldwasser, , J. Latombe, and R. Motwani. ***Complexity measures*** *for* ***assembly sequences***. In Proc. IEEE Int. Conf. on Robotics and Automation, pages 1581--1587, Minneapolis, MN, Apr. 1996.

Context Doc ***1*** *(0)*: , **Complexity measures** for **assembly sequences**, *International Journal of Computational Geometry & Applications*, vol. 9 (1999), pp. 371--417.

Fig. 4. The citation interface of CiteSeer. Note that more than one citation string may be found in the same query title.

designed a template filter to check whether a citation string is consistent with its metadata.

The filter processes sequence templates by evaluating their accuracy and applying some simple rules. Specifically, we parse the citation string with the created sequence template, and use the metadata in the BibTeX file to evaluate the parsed answer. We use F-Measure >80 percent as criterion (see Section 5.1 for more details) to check if the accuracy is acceptable, and we check whether the STYLE FORM of the template satisfies the following rules: 1) the author, title, and journal symbols cannot appear more than once in a citation string; and 2) the symbols which represent different fields should be partitioned by punctuation marks. Then, we can store qualified sequence templates into the database.

4.3 Query Processing

In the query processing modules, we use sequence alignment techniques to transform a citation string into its RESULT FORM and further map each field to the given citation string. When a query citation string is input to the query processing module, it first convert to INDEX FORM by the canonicalization algorithm, then it passes through three stages: 1) sequence matching, 2) alignment extraction, and 3) post processing. In the sequence matching stage, BibPro uses BLAST to search the sequence templates database by matching similar INDEX FORMs. A BLAST search enables a researcher to compare a query sequence with a database of sequences that uses a score matrix to evaluate the search result. We use BLAST as a template search tool by modifying the score matrix.

The construction of the score matrix is based on the concept of BLOSUM [26], which are used to score alignments between evolutionarily divergent protein sequences. The BLOSUM are constructed from a large and diverse sample of sequence alignments. In our case, we treat the sequence template database as the sample of sequence alignment. Since one sequence template is transformed from BASE FORM and RESULT FORM as shown in Fig. 2, we take BASE FORM and RESULT FORM as one pair of alignment. To calculate our score matrix, the following equation is used:

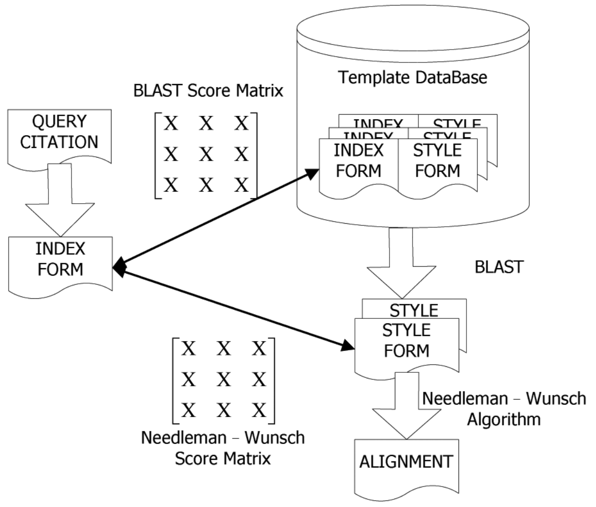


Fig. 5. The relationship between a query citation string and the sequence templates in the parsing process.

$$S_{ij} = \left(\frac{1}{\lambda}\right) \log\left(\frac{p_{ij}}{p_i \times q_j} + 0.5\right).$$

Here, p_{ij} is the probability of two symbols i and j align in the same column in BASE FORM and RESULT FORM, and q_i and q_j are the background probabilities of finding the symbol i and j in any sequence at random. We compute the overall occurrence of a symbol in the alignment of BASE FORM and RESULT FORM from template database to obtain the background probabilities. The constant 0.5 is used to smooth to avoid zero probability of p_{ij}. The factor λ is a scaling factor, set such that the matrix contains easily computable integer values. With the score matrix, BLAST assigns each sequence template a score which determines the degree of similarity with query citation string. Based on the score, we find a group of sequence templates that have similar citation styles. Note that, BLAST only gives quick and rough results of possible sequence templates. Since a sequence template is composed of an INDEX FORM and a STYLE FORM, we obtain several STYLE FORMs which represent possible citation format for the query citation string.

In the alignment extraction stage, we perform a detailed alignment for the query citation string's INDEX FORM and sequence template's STYLE FORM as a second-round analysis. We adopt the Needleman-Wunsch algorithm [24] as our detailed alignment algorithm. Needleman-Wunsch algorithm is an example of dynamic programming which is guaranteed to find an optimal alignment with given score matrix. Note that, we use the same score matrix used in BLAST, the difference is that the dynamic programming is more accurate than BLAST but needs more computation time. Besides, we take length of gap into account. Considering that gaps in the alignment of INDEX FORM and STYLE FORM always represent attributes of a citation string. In our observation, INDEX FORM align to STYLE FORM may have a larger gap, rather than many small gaps. To account for this tendency, we adopt affine gap penalty in Needleman-Wunsch algorithm. Affine gap penalty is to have a gap penalty for opening a gap, o, and a different gap penalty for extending the gap, e, which is smaller then gap opening penalty. A gap of length l is then given a penalty $o + (l - 1)e$. To let a few large gaps that are better than many small gaps, we set open gap penalty 1 and extended gap penalty 0.1. According to the detailed alignment algorithm, we can choose the STYLE FORM with the best fit and label the correct symbols for the query citation string. Fig. 5 shows the relationship between a query citation string and the sequence templates in the sequence matching and alignment extraction. With the alignment information, BibPro changes unknown tokens in the INDEX FORM into its corresponding STYLE FORM, as shown in Fig. 6. Since each symbol of the INDEX FORM keeps the start and end positions of a specific field in a query citation string, it is straightforward to map the symbols of the INDEX FORM to the RESULT FORM and extract all the metadata correctly.

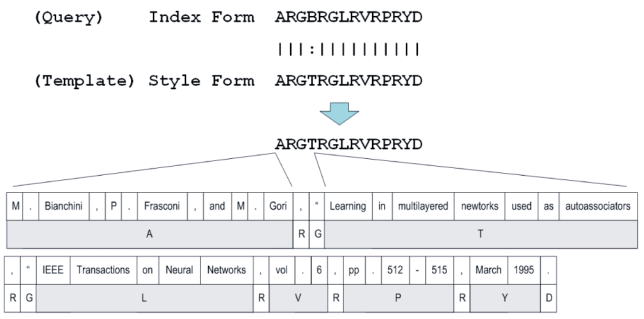

(Query) Index Form ARGBRGLRVRPRYD
|||:||||||||||
(Template) Style Form ARGTRGLRVRPRYD
ARGTRGLRVRPRYD
M . Bianchini , P . Frasconi , and M . Gori , " Learning in multilayered newtorks used as autoassociators
A R G T
, " IEEE Transactions on Neural Networks , vol . 6 , pp . 512 - 515 , March 1995 .
R G L R V R P R Y D

Fig. 6. Aligning a query's INDEX FORM and a template's STYLE FORM to obtain the RESULT FORM.

In the postprocessing stage, BibPro analyzes the extracted metadata; for example, the date field can be subdivided into year and month, and the venue field can be divided into journal, book title, and technical report. We match reserved words to derive a simple classification. For the date field, if the token is a numeral and its range is from 1900 to 2010, we classify it into the year field; meanwhile, if the token matches the month reserved word shown in Table 2, we classify it into the month field. Similarly, for the venue field, we decide the venues of journals, book titles, and technical reports based on their reserved words. Postprocessing allows us to utilize the symbols more efficiently.

5 Experiments and Results

To give a comprehensive evaluation, we use several benchmarks to measure the performance and design different experiments to verify the feasibility and flexibility of BibPro. In Section 5.1, we describe benchmarks used in our experiments. In Section 5.2, we described the detail of our experiments. In Section 5.3, we compare BibPro with existing tools and analyze factors which can influence the performance of BibPro.

5.1 Evaluation Metrics

We chose three data sets for our experiments. The first data set, used in [11], contains 160,000 citations comprised of six citation styles, namely, JMIS, ACM, IEEE, APA, MISQ, and ISR. The second data set was created by the Cora project [14], [15]. It comprises 500 citation strings, each of which contains 13 fields: author, title, editor, book title, date, journal, volume, tech report, institution, pages, location, publisher, and notes. The third data set, obtained from [13], contains 300 citation strings from the computer science domain and 2,000 citation strings from the health science domain. The CS citations contain 10 fields and the HS citations contain six fields. We denote the three data sets as D1, D2, and D3, respectively.

Each data set has its own characteristics. D1 has the most citations; however, they are more regular than those in D2 and D3 because they were generated from known metadata. The citation strings in D1 only differ in the order of the fields and the separators in the fields. In other words, the data set has fewer variations in the citation formats. D2, which is a widely used data set, comprises multiple styles and the diversity of the citation strings is much like that of real-world data; however, it only contains 500 citation strings and there are some errors in the labeling. D3 focuses on specific domains. The CS domain's citations contain some labeling errors, e.g., the publisher field is labeled as the journal field. To evaluate BibPro more precisely, we rework the D2 data set and the CS domain of D3. The changes include subdividing the volume field into volume and issue in D2; relabeling the journal field in D3 as the book title field; and correcting some labeling errors. We have made our experimental data sets available at http://csclws.iis.sinica.edu.tw:8080/input.jsp to facilitate further comparison.

We use two measurements to assess the performance of BibPro on the data sets. The first measurement which is defined in [11] is (1).

$$\text{Accuracy} = \frac{\text{Number of correctly extracted fields}}{\text{Total number of fields}}. \quad (1)$$

This measurement focuses on the field level, which means that a field can only be deemed correctly extracted if every token in the field is correctly extracted from the citation string.

The second measurement focuses on the token level. It is comprised of three well-known metrics: the precision rate, recall rate, and F-measure, which are also used in [13], [21]. They are calculated as follows:

Let A be the number of true positive tokens, B be the number of false negative tokens, and C be the number of false positive tokens. The formulas for the precision, recall, and F-measure are (2), (3), and (4).

$$\text{Precision} = \frac{\text{A}}{\text{A} + \text{C}}, \quad (2)$$

$$\text{Recall} = \frac{\text{A}}{\text{A} + \text{B}}, \quad (3)$$

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (4)$$

By using these two measurements, we can compare BibPro with other systems and derive more reliable experiment results.

TABLE 4
Evaluation Results for D1 with 10,000 Training—10,000 Testing

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	99.37%	99.33%	99.35%	98.13%
Title	99.54%	96.9%	98.21%	94.93%
Venue	98.14%	98.14%	98.14%	96.80%
Volume	96.37%	98.07%	97.21%	97.59%
Issue	99.04%	94.79%	96.87%	93.38%
Page	99.65%	98.85%	99.25%	98.10%
Date	99.31%	99.50%	99.41%	99.10%
Average	98.77%	97.94%	98.35%	96.86%
Journal	99.12%	92.80%	95.86%	92.20%
BookTitle	N/A	N/A	N/A	N/A
TechReport	N/A	N/A	N/A	N/A

5.2 Experiment Results

We design three experiments to verify the performance of query process module, practicality of template database construction module, and improvement compared with previous version.

5.2.1 Performance Evaluation

In the first experiment, we evaluate the performance of the query process; thus, we construct the template database from the data set itself and derive the score matrix from the template database. For the D1 data set, since each citation has six definite citation styles, we randomly selected 10,000 citations as the training set to build the template database and another 10,000 citations for testing. D2 and D3 data sets mix different citation styles, which are difficult to differentiate; therefore, we use fivefold cross validation to validate both data sets. Tables 4 and 5 show the results for D1 and D2, respectively; and Tables 6 and 7 show the results for D3. Since journal, book title, and tech report are extracted from the venue field during post processing, we list their results at the end of the table to show the performance of post processing. From these results, we observe that BibPro performs best on the D1 data set, with an average F-measure of 98.35 percent and an average field accuracy rate of 96.86 percent, followed by the D3 data set and then the D2 data set. These results imply that the query processing performs better when the template database is consistent with the test data. Since D1 is generated from known metadata and targets six specific citation styles, the constructed template database is sufficient to cover the six styles. Similarly, D3 is derived from a specific domain; thus, the citation styles in the template database fit the test data well. On the other hand, D2 is a more diverse data set, so it is more difficult to cover all the citation styles in the template database. Even so, the average F-measure is

TABLE 5
Evaluation Results for D2 with Fivefold Cross Validation

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	96.01%	97.47%	96.72%	89.55%
Title	97.12%	94.35%	95.71%	90.07%
Venue	92.48%	88.54%	90.39%	78.79%
Volume	84.81%	88.41%	86.41%	86.75%
Issue	77.77%	93.85%	84.92%	78.51%
Page	97.28%	96.71%	96.99%	95.08%
Date	97.76%	97.34%	97.04%	94.25%
Average	91.89%	93.81%	92.60%	87.57%
Journal	66.65%	88.13%	75.70%	85.60%
BookTitle	97.39%	82.38%	89.14%	65.86%
TechReport	86.93%	78.07%	81.72%	58.53%

TABLE 7
Evaluation Results for the HS Data Set in D3
with Fivefold Cross Validation

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	95.20%	99.71%	97.40%	93.20%
Title	99.85%	95.91%	97.84%	93.06%
Venue	97.26%	99.44%	98.30%	98.90%
Volume	99.85%	98.64%	99.24%	98.95%
Issue	N/A	N/A	N/A	N/A
Page	100.00%	97.07%	98.51%	96.99%
Date	98.38%	100.00%	99.18%	99.00%
Average	97.53%	98.31%	97.89%	96.48%
Journal	97.97%	89.26%	93.39%	93.19%
BookTitle	N/A	N/A	N/A	N/A
TechReport	N/A	N/A	N/A	N/A

92.60 percent and the field accuracy rate is 87.57 percent, which is good enough for practical use.

5.2.2 *Practicality Evaluation*

In the second experiment, we evaluate the practicality of the template database construction. Since it is difficult to collect every citation style to construct a universal template database, we focus on the automatic construction of the template database. In this experiment, we construct a default template database by the following steps. First, we obtain a BibTeX file from CiteSeer and choose the title field to search for citation strings in CiteSeer. Then, we use the extracted citation strings and their corresponding metadata to construct a template database automatically. Similarly, the score matrix is derived from this constructed template database. Next, we use BibPro to evaluate the 10,000 citations of D1 used in Experiment 1, and the complete D2 and D3 data sets with this template database. Tables 8 and 9 detail the results for D1 and D2, respectively; and Tables 10 and 11 detail the results for D3. The results show that, even though the template

TABLE 6
Evaluation Results for the CS Data Set in D3
with Fivefold Cross Validation

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	97.41%	99.51%	98.43%	98.67%
Title	99.41%	98.75%	99.07%	98.32%
Venue	100.00%	99.53%	99.76%	98.31%
Volume	98.97%	98.00%	98.42%	98.18%
Issue	100.00%	96.00%	97.78%	97.50%
Page	99.66%	98.11%	98.88%	97.84%
Date	92.16%	99.39%	95.61%	90.96%
Average	98.23%	98.47%	98.28%	97.11%
Journal	95.20%	90.68%	92.49%	93.70%
BookTitle	98.74%	99.06%	99.89%	97.51%
TechReport	N/A	N/A	N/A	N/A

TABLE 8
Evaluation Results for D1 with the
Default Template Database

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	98.67%	88.12%	93.10%	82.99%
Title	95.39%	94.61%	95.00%	89.44%
Venue	95.33%	69.65%	80.49%	69.32%
Volume	94.68%	82.45%	88.14%	78.73%
Issue	97.77%	86.58%	91.39%	84.24%
Page	99.69%	97.27%	98.46%	96.71%
Date	97.75%	98.61%	98.18%	97.44%
Average	97.04%	88.18%	92.11%	85.55%
Journal	96.55%	66.10%	78.48%	66.59%
BookTitle	N/A	N/A	N/A	N/A
TechReport	N/A	N/A	N/A	N/A

TABLE 9
Evaluation Results for D2 with the Default Template Database

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	96.79%	97.14%	96.96%	89.14%
Title	98.21%	95.35%	96.76%	91.67%
Venue	95.33%	84.26%	89.45%	70.69%
Volume	82.62%	95.67%	88.67%	89.01%
Issue	89.83%	92.98%	91.38%	92.39%
Page	97.78%	93.62%	95.65%	91.23%
Date	96.08%	97.30%	96.68%	93.21%
Average	93.81%	93.76%	93.65%	88.19%
Journal	78.09%	84.38%	81.11%	81.33%
BookTitle	97.30%	81.16%	88.50%	59.64%
TechReport	87.79%	66.23%	75.50%	36.67%

TABLE 11
Evaluation Results for the HS Data Set in D3 with the Default Template Database

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	90.42%	96.46%	93.34%	85.69%
Title	93.89%	84.34%	88.86%	72.96%
Venue	84.89%	75.17%	79.73%	52.15%
Volume	98.61%	90.31%	94.28%	90.02%
Issue	N/A	N/A	N/A	N/A
Page	99.64%	90.87%	95.05%	90.81%
Date	96.49%	76.94%	85.61%	74.94%
Average	93.99%	85.68%	89.48%	77.76%
Journal	84.25%	66.37%	74.25%	48.80%
BookTitle	N/A	N/A	N/A	N/A
TechReport	N/A	N/A	N/A	N/A

database is not derived from the data set itself, the average field accuracy is higher than 80 percent. Note that the average field accuracy of D2 is 88.19 percent, which is higher than the field accuracy under fivefold cross validation; on the other hand, the average field accuracy of the HS domain in D3 is 77.76 percent, which is much lower than the field accuracy under fivefold cross validation. This observation implies that the distribution of the citation styles in the default template database is consistent with that in D2, but it differs substantially from the distribution in the HS domain of D3. To verify the flexibility of BibPro, we add 100 labeled citation strings from the HS domain in D3 to the default template database without changing the score matrix and use it to evaluate the HS domain. To prevent an imbalance in the distribution of the citation styles, we partition the HS domain data set into 20 subsets, each containing 100 labeled citation strings. Each subset is added to default template database and used to evaluate the complete HS domain data set. The 20 results are then averaged to produce a single estimate. The average F-measure is 97.80 percent and the average field accuracy is 96.18 percent. Fig. 7 compares the original default template database with the default template database containing the 100 labeled citation strings. Based on the experiment results, we conclude that BibPro can quickly add new templates that are consistent with the test data to achieve a reasonable performance.

TABLE 10
Evaluation Results for the CS Data Set in D3 with the Default Template Database

	Token-Level			Field-Level
	Precision	Recall	F-Measure	Accuracy
Author	97.78%	95.47%	96.61%	92.64%
Title	97.77%	94.55%	96.13%	91.64%
Venue	98.87%	97.55%	98.21%	87.88%
Volume	90.15%	95.20%	92.61%	85.96%
Issue	96.43%	88.52%	92.31%	87.23%
Page	99.52%	96.26%	97.86%	95.24%
Date	91.35%	98.48%	94.78%	88.85%
Average	95.98%	95.15%	95.50%	89.92%
Journal	91.47%	76.62%	83.39%	72.58%
BookTitle	97.97%	98.29%	98.13%	90.28%
TechReport	N/A	N/A	N/A	N/A

5.2.3 *Blocking versus Nonblocking*

In the third experiment, we compare BibPro with its previous version [23]. The main difference between the two versions is that the previous version does not have the blocking mechanism and the score matrix is constructed

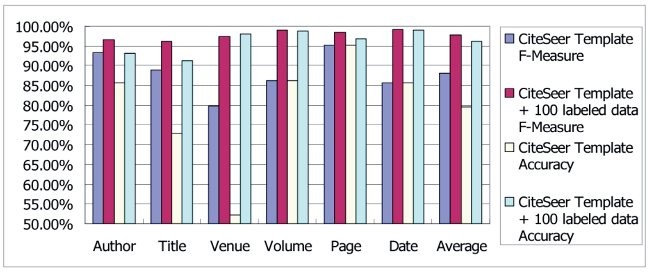


Fig. 7. Comparison of the performance for the original template database and the template database with additional 100 labeled citations.

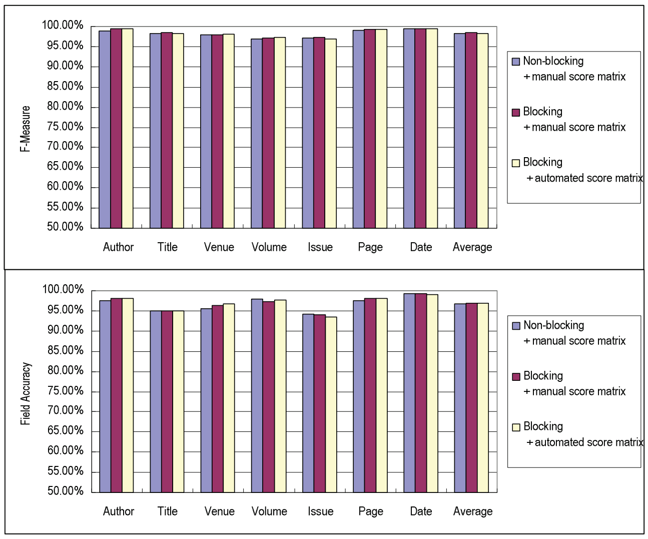


Fig. 8. Comparison of the performance for the variation of the blocking mechanism and score matrix on the D1 data set.

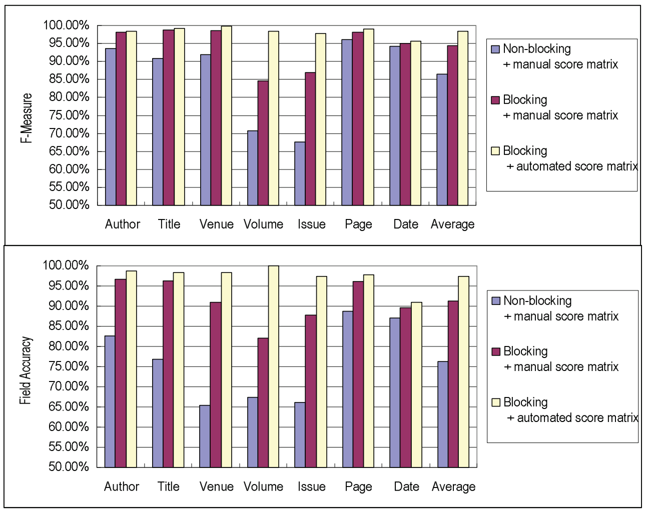


Fig. 10. Comparison of the performance for the variation of the blocking mechanism and score matrix on the CS data set in D3.

manually according to domain knowledge; where the score matrix in current version is constructed automatically.

We apply the validation method used in Experiment 1 to compare the performance with three conditions: 1) Without blocking mechanism and use manually constructed score matrix. 2) With blocking mechanism and use manually constructed score matrix. 3) With blocking mechanism and use score matrix which is automatically derived from template database. Figs. 8 and 9 compare their performance on D1 and D2, respectively; and Figs. 10 and 11 compare their performance on D3. The results demonstrate that blocking mechanism significantly improves the performance of BibPro, and the more diverse the data set, the greater will be the improvement in the performance, especially in terms of the field-level accuracy. Interestingly, with the automatically constructed score matrix, BibPro performs better for some data set (D3 CS); while the performance of other data sets is quite similar with which used manually constructed score matrix. It is clarified that we efficiently reduce the manual tuning effort of BibPro without losing the performance.

5.3 Analysis

Table 12 offers a summary of existing citation parsers and their provided benchmarks and evaluated metrics. To compare with other exist tools, we have to obtain all implementation codes and benchmarks of those approaches. However, we cannot obtain all the implementation codes. The only things we can obtain are their benchmarks. Hence, we choose to compare BibPro with each approach by using its own benchmark. To evaluate system performance, we use both field-level and token-level evaluation metrics.

Fig. 12 summarized the performance of BibPro for different benchmark and we add result of related works in this figure as a reference. In the field-level aspect, it is easy to observe that BibPro outperforms INFOMAP with an

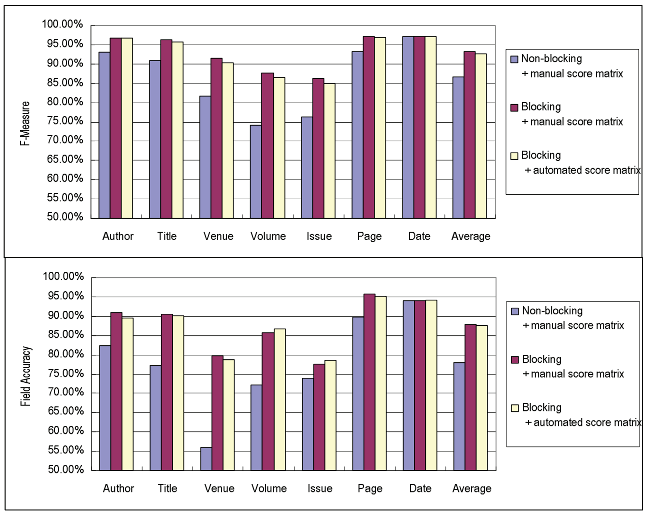


Fig. 9. Comparison of the performance for the variation of the blocking mechanism and score matrix on the D2 data set.

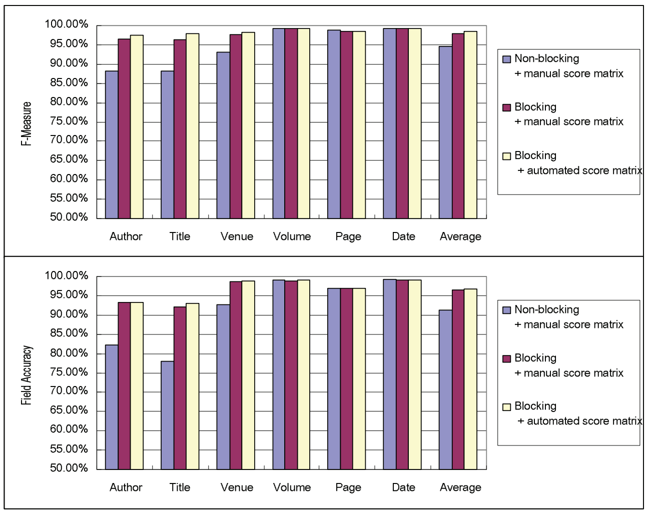


Fig. 11. Comparison of the performance for the variation of the blocking mechanism and score matrix on the HS data set in D3.

TABLE 12
Existing Citation Parser with Benchmark

	algorithm feature	benchmark	evaluation metric
INFOMAP	knowledge-base	INFOMAP Data (D1 DataSet)	Accuracy (field level)
Peng (CRF)	machine learning	Cora Data (D2 Dataset)	F-Measure (token level)
Flux-CiM	knowledge-base	Flux-CiM Data (D3 DataSet)	F-Measure (token level)
ParsCit (CRF)	machine learning	CiteSeer^X Data	F-Measure (token level)

average accuracy of 96.86 percent versus 91.63 percent for INFOMAP. In the token-level aspect, BibPro seems to perform better than Flux-CiM and ParsCit for their provided data set, and seems to perform worse than Peng (CRF) for Cora data set (see Supplemental for actual experimental data, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.231). However, Councill et al. [29] report that ParsCit is largely comparable with Peng (CRF). According to these results, we cannot justify which system is better. It is worthy to mention that BibPro provides a stable performance for all benchmarks in both field-level and token-level evaluations, where the average field accuracy is over 80 percent and average F-measure is over 90 percent for all benchmarks. From the figure, we observe that BibPro's accuracy is highest on data set of INFOMAP, followed by Flux-CiM and Cora, and performs worse in CiteSeer^X data. The results verify that D1 (INFOMAP) is the most regular data set and D2 (Cora) and CiteSeer^X are more diverse data sets.

To demonstrate the robustness of our approach, we conduct a simple experiment on the Cora data set. We take the first 350 citations in the data set as training data, and the remaining 150 citations as test data. Then, we use

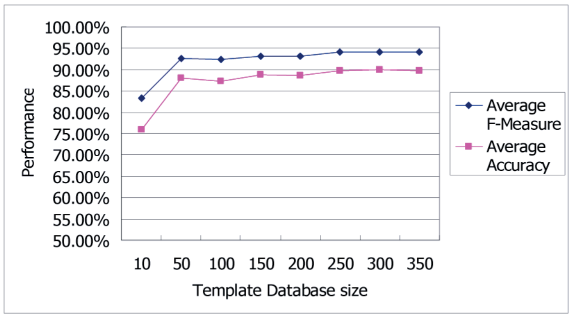


Fig. 13. The learning curve of BibPro on the D2 data set.

different amounts of training data (from 10 to 350) to construct the template database and evaluate the performance. Fig. 13 shows the result of this experiment, which is like the learning curve graph of the general learning-based approach. The figure shows that even though the template database only has 10 templates, the average F-measure result is over 80 percent. Moreover, the performance quickly becomes stable as the size of the template database increases.

Three important factors can influence BibPro's performance, namely, the score matrixes, domain knowledge (reserved words and the blocking patterns), and the template database. These factors can be automatically derived from labeled data. However, score matrixes derived from different data set are quite similar between each other (see Supplemental Material for constructed score matrix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.231). Thus, we can fix the score matrix for practical use, such as our online service. For domain knowledge, we use the reserved words to replace knowledge databases, such as author databases and venue databases used in previous work. Since the performance of a knowledge database is dependent on the query data, using reserved words can achieve a more stable performance than ambiguous knowledge databases. Our experiment results show that even without a knowledge

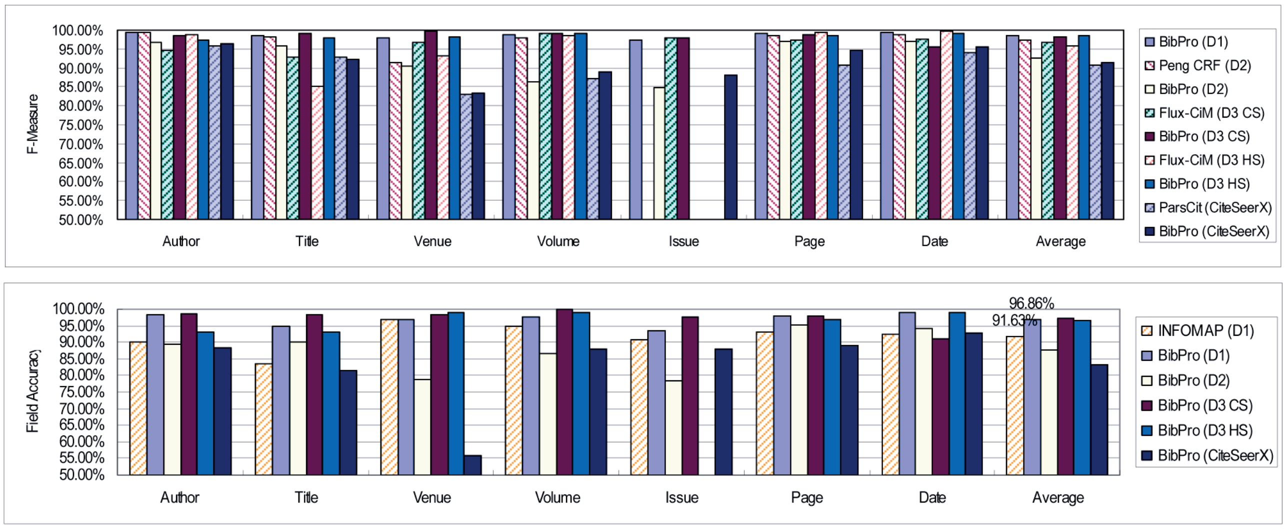


Fig. 12. Comparison of the field accuracy rates of BibPro on different benchmarks.

database, BibPro still yields a reliable performance. However, a knowledge database can be used to complement for particular types of query data. For example, Chinese Pinyin characters can be treated as reserved words in author field, because they rarely have ambiguous meanings in other fields. In fact, we use Chinese Pinyin characters as reserved words in our online service, and they obviously improve the performance on data containing Chinese names. With regard to the blocking patterns, the experiment results show that the blocking mechanism can effectively improve the overall performance, especially the field-level accuracy. Meanwhile, for the template database, the experiment results show that the distribution of citation styles in the database directly influences the performance of the query processing. In addition, with the support of the template database construction module, users can easily add labeled data to ensure that the template database is consistent with the query data. That is, users can optimize the performance of BibPro for specific requirement by adjusting these factors.

6 Conclusion

Parsing citations is challenging due to the diverse nature of citation formats. In this paper, we present a sequence-alignment-based citation parser called "BibPro." The basic concept of BibPro is to transform semistructured properties of a citation string into a sequence template, and apply sequence alignment techniques to further resolve the structured information. The effectiveness and applicability of BibPro is demonstrated by experiments on three benchmark data sets. Specifically, BibPro achieved over 90 percent field-level accuracy on all three benchmarks. In addition, BibPro can automatically construct a template database using not-so-accurately labeled citation strings retrieved from the web, and experiments show that its field-level accuracy is over 80 percent for the given data sets. BibPro is implemented to allow users to maintain a profile of templates in order to meet specific user requirements.

Our methodology can be applied to similar information extraction problems in which a semistructured text can be formulated as a string of multiple fields, $S = \{\text{Field}_1\ \text{Delimiter}_1\ \text{Field}_2\ \text{Delimiter}_2\ \text{Field}_3\ \text{Delimiter}_3 \ldots\}$. For example, a web-based information provider usually stores information in database and presents information structurally on webpages markup with additional html tags. By using the same idea to transform semistructured text into a sequence template, our approach can be applied to solve the recognition problem of extracting the structured information embedded in the webpages [27], [28]. We are interested in applying the approach presented in this paper on some of these problems. We are also interested in designing a system to automatically extract publication information from webpages containing researchers' publication lists to provide a more comprehensive citation management service. Some preliminary studies can be found in [25], [28].

Acknowledgments

This work is supported by National Science Council (NSC), Taiwan, for financially supporting this research under Contract No. NSC98-2221-E-001-010-MY3. The authors thank the editor and anonymous reviewers for their helpful advice.

References

[1] D. Lee, J. Kang, P. Mitra, C.L. Giles, and B.-W. On, "Are Your Citations Clean?," *Comm. ACM*, vol. 50, pp. 33-38, 2007.

[2] M. Cristo, P. Calado, M.A. Goncalves, E.S. de Moura, B. Ribeiro-Neto, and N. Ziviani, "Link-Based Similarity Measures for the Classification of Web Documents," *J. Am. Soc. for Information Science and Technology*, vol. 57, pp. 208-221, 2006.

[3] T. Couto, M. Cristo, M.A. Goncalves, P. Calado, N. Ziviani, E. Moura, and B. Ribeiro-Neto, "A Comparative Study of Citations and Links in Document Classification," *Proc. Sixth ACM/IEEE-CS Joint Conf. Digital Libraries,* 2006.

[4] M.A. Goncalves, B.L. Moreira, E.A. Fox, and L.T. Watson, "'What Is a Good Digital Library?' - A Quality Model for Digital Libraries," *Information Processing and Management,* vol. 43, pp. 1416-1437, 2007.

[5] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. Seventh Int'l Conf. World Wide Web 7,* 1998.

[6] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, and J.S. Teixeira, "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record,* vol. 31, pp. 84-93, 2002.

[7] C.L. Giles, K. Bollacker, and S. Lawrence, "CiteSeer: An Automatic Citation Indexing System," *DL '98: Proc. Third ACM Conf. Digital Libraries,* pp. 89-98, 1998.

[8] K.D. Bollacker, S. Lawrence, and C.L. Giles, "CiteSeer: An Autonous Web Agent for Automatic Retrieval and Identification of Interesting Publications," *Proc. Second Int'l Conf. Autonomous Agents,* 1998.

[9] S. Lawrence, C.L. Giles, and K.D. Bollacker, "Autonomous Citation Matching," *Proc. Third Ann. Conf. Autonomous Agents,* 1999.

[10] S. Lawrence, C.L. Giles, and K.D. Bollacker, "Digital Libraries and Autonomous Citation Indexing," *Computer,* vol. 32, no. 6, pp. 67-71, June 1999.

[11] M.-Y. Day, R.T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, and W.-L. Hsu, "Reference Metadata Extraction Using a Hierarchical Knowledge Representation Framework," *Decision Support Systems,* vol. 43, pp. 152-167, 2007.

[12] E. Agichtein and V. Ganti, "Mining Reference Tables for Automatic Text Segmentation," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2004.

[13] E. Cortez, A.S. da Silva, M.A. Gon calves, F. Mesquita, and E.S. de Moura, "FLUX-CiM: Flexible Unsupervised Extraction of Citation Metadata," *Proc. Seventh ACM/IEEE-CS Joint Conf. Digital Libraries,* pp. 215-224, 2007.

[14] Andrew McCallum's Code and Data, http://www.cs.umass.edu/~mccallum/code-data.html, 2005.

[15] F. Peng and A. McCallum, "Accurate Information Extraction from Research Papers Using Conditional Random Fields," *Proc. Human Language Technology Conf. and North Am. Chapter of the Assoc. for Computational Linguistics (HLT-NAACL),* pp. 329-336, 2004.

[16] H. Han, C.L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E.A. Fox, "Automatic Document Metadata Extraction Using Support Vector Machines," *Proc. Third ACM/IEEE-CS Joint Conf. Digital Libraries,* 2003.

[17] K. Seymore, A. McCallum, and R. Rosenfeld, "Learning Hidden Markov Model Structure for Information Extraction," *Proc. Workshop Machine Learning for Information Extraction (AAAI '99),* pp. 37-42, 1999.

[18] V. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic Segmentation of Text into Structured Records," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* 2001.

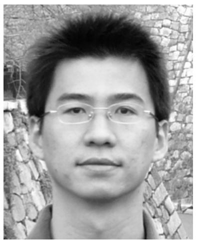
[19] A. Takasu, "Bibliographic Attribute Extraction from Erroneous References Based on a Statistical Model," *Proc. Third ACM/IEEE-CS Joint Conf. Digital Libraries,* 2003.

[20] P. Yin, M. Zhang, Z. Deng, and D. Yang, "Metadata Extraction from Bibliographies Using Bigram HMM," *Proc. Seventh Int'l Conf. Asian Digital Libraries,* pp. 310-319, 2004.

[21] E. Hetzner, "A Simple Method for Citation Metadata Extraction Using Hidden Markov Models," *Proc. Eighth ACM/IEEE-CS Joint Conf. Digital Libraries,* 2008.

[22] I.-A. Huang, J.-M. Ho, H.-Y. Kao, and W.-C. Lin, "Extracting Citation Metadata from Online Publication Lists Using BLAST," *Proc. Eighth Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD '04),* Jan. 2004.

[23] C.-C. Chen, K.-H. Yang, and J.-M. Ho, "BibPro: A Citation Parser Based on Sequence Alignment Techniques," *Proc. Int'l Conf. Advanced Information Networking and Applications (AINA '08),* 2008.
[24] S.B. Needleman and C.D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *J. Molecular Biology,* vol. 48, pp. 443-453, 1970.
[25] K.-H. Yang, J.-M. Chung, and J.-M. Ho, "PLF: A Publication List Web Page Finder for Researchers," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence,* 2007.
[26] S. Henikoff and J.G. Henikoff, "Amino Acid Substitution Matrices from Protein Blocks," *Proc. Nat'l Academy of Sciences USA,* vol. 89, pp. 10915-10919, 1992.
[27] C. Chang, M. Kayed, M. Girgis, and K. Shaalan, "A Survey of Web Information Extraction Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 10, pp. 1411-1428, Oct. 2006.
[28] K.-H. Yang, S.-S. Chen, M.-T. Hsieh, H.-M. Lee, and J.-M. Ho, "CRE: An Automatic Citation Record Extractor for Publication List Pages," *Proc. 12th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD '08),* 2008.
[29] I.G. Councill, C. Lee Giles, M.-Y. Kan, "ParsCit: An Opensource CRF Reference String Parsing Package," *Proc. Language Resources and Evaluation Conf.,* 2008.



Chien-Chih Chen received the BS degree from National Taiwan Normal University in 2002, and the MS degree from the Department of Computer Science and Information Engineering, National Taiwan University in 2005. He is working toward the PhD degree at National Taiwan University. He is currently a research assistant in the Computer Systems and Communication Lab, which is lead by Dr. Jan-Ming Ho, of Institute of Information Science, Academia Sinica. His research interests include program language, information extraction, and bioinformatics.

Kai-Hsiang Yang received the BA degree from the Department of Mathematics, National Taiwan University in 1997, and the PhD degree from the Department of Computer Science and Information Engineering, National Taiwan University in 2004. He is an assistant professor at National Taipei University of Education. His research interests include web mining, peer-to-peer computing, network security, and information retrieval.

Chuen-Liang Chen received the BS degree in electrical engineering from National Taiwan University in 1980 and the PhD degree from the Department of Computer Science at National Chiao Tung University in 1985. He is an associate professor at National Taiwan University. His research interests include compiler, programming language, and computer system.

Jan-Ming Ho received the BS degree in electrical engineering from National Cheng Kung University in 1978 and the MS degree from the Institute of Electronics of National Chiao Tung University in 1980. He received the PhD degree in electrical engineering and computer science from Northwestern University in 1989. He joined the Institute of Information Science, Academia Sinica, as an associate research fellow in 1989, and was promoted to research fellow in 1994. He visited IBM's T. J. Watson Research Center in Summer 1987 and Summer 1988, and the Leonardo Fibonacci Institute for the Foundations of Computer Science, Italy, in Summer 1992. In 2004-2006, he was jointly appointed by National Science Council, Taiwan, where he served as the director general of Division of Planning and Evaluation. His research interests cover the integration of theory and applications, including information retrieval and extraction, combinatorial optimization, multimedia network protocols and their applications, bioinformatics, and digital library and archive technologies. He also published results in VLSI/CAD physical design. He is a member of the IEEE.