TableSeer: Automatic Table Metadata Extraction and Searching in Digital Libraries*

Ying Liu, Kun Bai, Prasenjit Mitra, C. Lee Giles

The College of Information Sciences and Technology
Pennsylvania State University
University Park, PA 16801
{yliu, kbai, pmitra, giles}@ist.psu.edu

ABSTRACT

Tables are ubiquitous in digital libraries. In scientific documents, tables are widely used to present experimental results or statistical data in a condensed fashion. However, current search engines do not support table search. The difficulty of automatic extracting tables from un-tagged documents, the lack of a universal table metadata specification, and the limitation of the existing ranking schemes make table search problem challenging. In this paper, we describe *TableSeer*, a search engine for tables. *TableSeer* crawls digital libraries, detects tables from documents, extracts tables metadata, indexes and ranks tables, and provides a user-friendly search interface. We propose an extensive set of medium-independent metadata for tables that scientists and other users can adopt for representing table information. In addition, we devise a novel page *box-cutting* method to improve the performance of the table detection. Given a query, *TableSeer* ranks the matched tables using an innovative ranking algorithm – *TableRank*. *TableRank* rates each <query, table> pair with a tailored vector space model and a specific term weighting scheme. Overall, *TableSeer* eliminates the burden of manually extract table data from digital libraries and enables users to automatically examine tables. We demonstrate the value of *TableSeer* with empirical studies on scientific documents.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval – search process

General Terms

Algorithms, Experimentation, Documentation, Performance, Design

*This work was partially supported by NSF grants 0454052 and 0535656.

JCDL'07, June 17–22, 2007, Vancouver, British Columbia, Canada.

Keywords

accessibility, architectures, data management, information retrieval, knowledge organization, scientific applications, system design

1. INTRODUCTION

Tables are ubiquitous in scientific publications, web pages, financial reports, news papers, magazine articles, etc. Tables present structural data and relational information in a two-dimensional format and in a condensed fashion. Researchers always use tables to concisely display their latest experimental results or statistical data. Other researchers, for example, who are conducting the empirical studies in the same topic, can quickly obtain valuable insights by examining these tables. Along with the rapid expansion of digital libraries, tables become an important information source for information retrieval and the demand for searching table is increasing. For example, a bio-chemist may want to search tables containing experimental results about "mutant genes" or an economist may look for the tables about "the GDP growth of USA in 2000-2007." However, existing search engines do not support table search. When applying a table search query to the popular search engines, we observe that a flood of unwanted and sometimes unsolicited results will be returned. Moreover, we also notice that within the top n returned results, the ranking order does not precisely reflect the relevance to the queries.

Automatic table extraction is critical to the success of table search. However, current search engines do not support it. Automatic table extraction is important because there exists no table markup language that scientists and other users have adopted for representing table information in documents. There is not yet a universal metadata specification for tables. Scientists and scholars who now want to extract data from tables in published papers have to do it manually, establishing their own metadata formats. In order to eliminate the burden and enable users to automatically examine these tables, we design *TableSeer*, a system to automatically extract table metadata and search tables in digital libraries.

Automatic table extraction and search in digital libraries is a challenging problem for several reasons. First, although considerable research has been done to extract tables from HTML or Image documents, extracting un-tagged tables (e.g. in PDF format) in digital libraries is difficult. Second, tables present unique challenges to information retrieval (IR)

systems because of diverse media, different press layouts, cell types, affiliate table elements, etc. Along with the increasing demands of information sharing, an extensive and universal table metadata specification is needed to facilitate the table information processing, extracting, and reusing. Additionally, such a standard representation can heavily reduce the workload of table information extraction and increase the accuracy of the table indexing and searching. Third, the ranking schemes of current search engines are inadequate and not designed for table search. The returned ranking order does not correctly reflect the relevance of the tables to the queries.

Our paper has four main contributions: a table search engine *TableSeer*, a set of universal medium-independent metadata specification for tables, an automatic table detector and table metadata extractor with a novel page box-cutting method, and an innovative table ranking algorithm *TableRank*. *TableSeer* crawls scientific documents from digital libraries, identifies documents with tables, extracts tables from documents, represents each table with a unique table metadata file, indexes tables and provides an interface to enable the end-users to search for tables. *TableRank* considers multiple features of a table and the document it appears in, and aggregates these features to determine the final ranking of each table with respect to an user query.

In this paper, we focus on PDF documents because of two reasons. First, PDF gains popularity in digital libraries due to the compatibility of output on a variety of devices. Second, PDF documents are overlooked in table extraction field. Please note that *TableSeer* can be extended to deal with documents in other media, e.g., HTML, Word, PDF, Image, etc.

Extensive experimental studies have been conducted to provide support for the table metadata extraction and the table ranking function. In order to evaluate the performance of our *TableRank* schema, we compare *TableSeer* with several popular web search engines. Because all the popular web search engines make no efforts at treating tables specially, we propose two methods to set up the common test-beds (see Section 5.4). Empirical results show that *TableSeer* achieves encouraging results compared to *Google Scholar*[1] and *CiteSeer*[2].

The remainder of the paper is organized as follows. Section 2 is the related work. Section 3 presents the architecture of *TableSeer* and a proposal for medium-independent table metadata. In Section 4, we describe the table ranking algorithm. Section 5 discusses the experiments and an analysis of the results and Section 6 makes a conclusion.

2. RELATED WORK

A search engine is an information retrieval system designed to help find information[1]. Most commonly search engines are Web search engine, which searches for information on the public Web. For example, *Google*, *Yahoo! search*, *Microsoft MSN Search*, *ASK.com*, etc. Other kinds of search engines are enterprize search engines, which search on intranets, personal search engines, and mobile search engines. Different selection and relevance criteria may apply in different environments, or for different uses. More recently, more and more lights are shed on specialty search engines. Some of them support search on various kinds of documents (e.g., map search[1], video search[3]image search[1], etc.), as well as on document components (e.g., citation search[2], acknowledgement search[2], etc). However, none of them meets the table search demand.

Although table-related research has recently received considerable attention, most of the research focuses on the table extraction. Some researchers try to associate the table extraction with question answering (QA) or information retrieval. For example, Pyreddy and Croft [14] design a character alignment graph (CAG) to extract tables for information retrieval. TINTIN [14] is a system that incorporates indexing and searching concepts into the table detection field. However, it only roughly divided the information of a table into two parts: table captions or table entries. Hu [17] designs a system that extracts table-related information, stores them in databases, and generates a man-machine dialog to access the table data via a spoken language interface. None of above provide a real web search engine to tables. To the best of our knowledge, *TableSeer* is the first table search engine, which supports the automatic table metadata extraction and table search.

Researchers in the automatic table extraction field largely focus on analyzing the table structure in a specific document media. Zanibbi [15] provides a survey with detailed description of each method. All the methods can be divided into three categories: pre-defined layout based [10], heuristics based [7][9][11][16], and statistical based [14]. Pre-defined layout based algorithms usually work well for one domain, but is difficult to extend. Heuristics based methods need a complex post-processing and the performance relies largely on the choice of features and the quality of training data. Most approaches described so far utilize purely geometric features (e.g. pixel distribution, line-art, white streams) to determine the logical structure of the table, and different document mediums require different process methodologies: OCR [5], X-Y cut [8], tag classification and keyword searching [3][4][19] etc. Most of them use trial-and-error methods and no general table ground-truth data set is publicly available to train and test these algorithms [20]. For a table search query, the matched tables may be contained in diverse media, which require different methods to extract. It is inconvenient to divide the documents into different groups and apply the corresponding algorithms. Thus, good table representation schemes are needed. Wang [18] proposed a table model to show the table layout. The well-known table representation schemes are designed by the World Wide Web Consortium (W3C) in the specification of XHTML (The Extensible HyperText Markup Language) and by the Organization for the Advancement of Structured Information Standards (OASIS). Wang [20][21] develops a software tool to generate documents that include similar table elements based on the given table ground truth. Akira Amano [2] proposes a representation of table form document based on XML. However, none of them covers table structure and layout information, as well as the table-related information and the document background. Our table metadata can provide a universal metadata specification to represent the information and all the necessary facts of a table in any document medium for the table searching purpose.

[1]http://www.google.com/
[2]http://citeseer.ist.psu.edu/
[3]http://www.youtube.com/

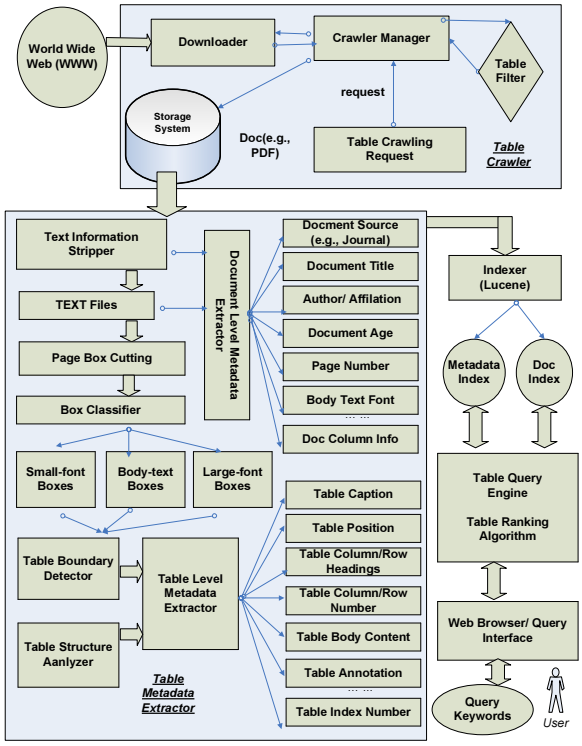

Figure 1: The Architecture of TableSeer

3. THE ARCHITECTURE OF TABLESEER

3.1 System Overview

Figure 1 shows the architecture of *TableSeer*, which consists of a number of important components: 1) a table crawler, 2) a table metadata extractor, 3) a table metadata indexer, 4) a table ranking algorithm, and 5) a table searching query interface. In summary, *TableSeer* crawls scientific documents from the digital libraries, identifies the documents with tables, detects each table using a novel document *page box-cutting* method, extracts the metadata for each identified table, ranks the matched tables against the end-user's *query* with the *TableRank* algorithm, and displays the ordered results in a user-friendly interface.

3.2 The Table Crawler

TableSeer harvests online scientific documents by crawling open-access digital libraries and scientists' webpages. The crawler supports a number of document media, such as PDF, HTML, WORD, PowerPoint, etc. *TableSeer* is designed to be able to handle all the document media listed above. In this paper, the table crawlers pay attention to the scientific documents in the PDF format because they gain more and more popularity in digital libraries. In addition, comparing to other document media that have been extensively studied in the table detection field, PDF documents have been overlooked. Our table crawlers use a depth-first crawling policy with a maximum depth of five (from the seed URLs).

3.3 The Table Metadata Extractor

The table metadata extractor comprises of three key parts: 1) a text information stripper (*TIS*), 2) a table box detector with the *page box-cutting* method, and 3) a table metadata extractor from both the document level and the table level.

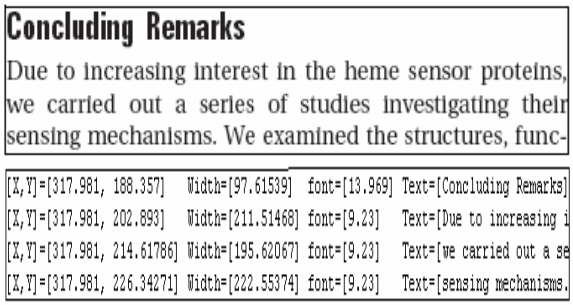

Figure 2: An Example of a *PDF* Document Segment and the Corresponding *Document Content File*

3.3.1 Text Information Stripper (TIS)

Initially, for each PDF document, *TIS* strips out the text information from the original PDF source file word by word through analyzing the *text operators*[4] and the related *glyph*[4] information. *TableSeer* reconstructs these words into lines with the aid of their position information and saves the lines into a *Document Content File* in the *TXT* format. For each document page, *TableSeer* analyzes the text information and merges them into different physical component levels (lines, paragraphs, boxes, pages) according to their font and position information. Figure 2 shows an example segment of a PDF document and its corresponding *Document Content File*. The text file also records the coordinates of the left-most word (X_0, Y_0), the line width W, the line height H, the font size F, as well as the text content of each line. All the lines are ordered in the same sequence as shown in the PDF document.

3.3.2 Table Box Detection and Table Metadata Extraction

Automatic table metadata extraction includes two tasks: table detection and metadata extraction. To effectively detect tables, we design a novel *page box-cutting* method as following. The details of metadata extraction are addressed in [13].

The input of the *box-cutting* method is a *Document Content File* shown in Figure 2. We define a *page box* as a rectangle of adjacently connected *lines* with a uniform font size in the same document page. Whether two lines merge into a same *page box* is decided by two factors: the font size and the position. We treat a line l_η in a *Document Content File* as the seed line l_{seed} of a *box* b_θ. Initially, $\eta = 1, \theta = 1$, and l_η is the only line in b_θ. If the next line $l_{\eta+1}$ in the same *Document Content File* satisfies the following three conditions, we combine $l_{\eta+1}$ into b_θ and set $l_{\eta+1}$ as the new l_{seed}. Otherwise, $l_{\eta+1}$ will be the l_{seed} of a new *box* $b_{\theta+1}$. C1) $Font(l_{\eta+1}) = Font(l_{seed})$; C2) $l_{\eta+1}$ is adjacent to l_{seed}; C3) $l_{\eta+1}$ is close enough to l_{seed}.

TableSeer defines "*adjacent*" as no other lines exist between $l_{\eta+1}$ and l_{seed}. Different definitions of the "*close enough*" may generate different box cutting results. We define the "*close enough*" as: $Y_0(l_{\eta+1}) - (Y_0(l_{seed}) + H(l_{seed})) \leq \delta^l$. δ^l is the maximal space between two adjacent lines in a same paragraph. Figure 3 displays an example of a PDF document page together with the boxes segmented by our

[4]PDF Reference Fifth Edition, Version 1.6

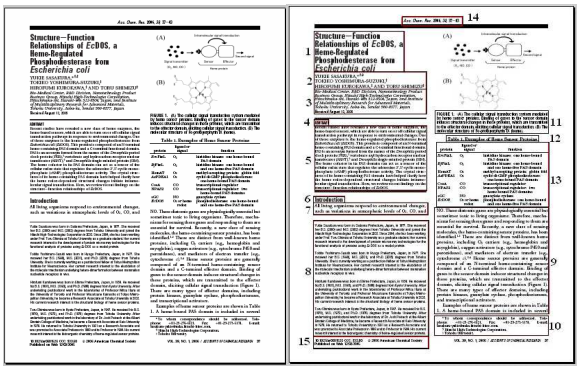

Figure 3: An Example of the Page Box-Cutting Approach

page box-cutting method.

For all the segmented boxes in a document, *TableSeer* classifies them into three categories: small-font boxes B^{SF}, large-font boxes B^{LF}, and regular-font boxes B^{RF}, whose font sizes are smaller than, larger than, or equal to the font size of the *document body text* F_b. Based on the observation and statistical study on the proceeding/journal templates, we summarize a set of heuristic rules (see *Table 1*), which are crucial for mapping boxes to different logical components (*titles*, *authors*, *affiliations*, *abstract*, *references*, etc.) and specific physical components (*tables*, *figures*, etc.).

Table 1: Heuristic Rules for Table Box Dections

Rules	Content
1	• *Document title, author, affiliation, heading* $\in B^{LF}$.
2	• *Document title, author, affiliation, abstract* $\in$ Page 1, in a fixed order.
3	• *Figure caption, table caption, table body, footnote, Reference* usually $\in B^{SF}$.
4	• $\nexists$ *Table caption, table body, footnote, Reference* $\in B^{LF}$.
5	• $\forall$ *Figure* and *table* have captions.
6	• $\forall$ *Figure* captions are beneath the figure; $\forall$ *table* captions are above the table.
7	• *Table captions* start with keywords "Table" or "TABLE" while *Figure captions* start with "Figure" or "FIGURE".

TableSeer detects tables in at most three iterations (see algorithm 1). In each iteration, we use a *keyword matching* method and a *whitespace checking* method to examine one group of the boxes. We create a predefined *keyword list* K^l that records all the possible starting keywords of the table captions in scientific documents, such as "Table", "TABLE", "table", "Form", "form", "FORM", "Figure", "FIGURE", "figure", etc. In the first iteration, we process all B^{SF} in the document page by page. If we find a B^{SF} that starts with a keyword listed in the K^l, we treat this box as a table candidate. If we detect the tabular structure from its white space information, we confirm this B^{SF} as a real table. Once we get the font size of a table in a document, the font size of all tables in this document are fixed, and we ignore all the other boxes in different font sizes. If we do not find any table candidates in the first iteration, we start the second iteration on all B^{RF} because there still is a percentage of documents that contain tables displayed in F_b. If we fail in this run again, we start checking the boxes with font size B^{LF}. Usually we will success in detecting tables in the first iteration. The table caption and the table body may be detected in two adjacent boxes because of the sparse layout or the trivial font difference. *TableSeer* deals with such cases as well. When we detects a box started with words in the K^l without tabular structures, we check the whitespace information of the boxes $b'_n s$ neighbor boxes b_{n-1}, b_{n+1}, respectively.

Using figure 3 as an example, the body text font fize F_b is *9.23*. Boxes *1, 2, 3, 4, 6, 11* are B^{LF}, boxes *5, 10, 8, 12, 13, 14, 15* are B^{SF}, and boxes *7, 9* are B^{RF}. In the first iteration, we detect the keyword "*Table*" in box *12* without tabular structures. We check its neighbor – box 11 and 13. Box 13 is confirmed as a real table. With the fixed table font size 8.76, we only have to check the boxes with the same font size to detect other tables in the same document. With the *box-cutting* method, *TableSeer* easily exclude more than 93.6% of the document content in the beginning, which impressively increases the efficiency and accuracy of the table detection.

Algorithm 1: Table Detection using the Page Box-cutting Method

```
begin
    η ← 1, θ ← 1, l_s ← l_η, found ← 0;
    for l_η ∈ l do
        if l_{η+1} does not satisfy C_{1-3} then
            compare b_θ.fontsize() with F_b;
            classify b_θ into one of B^SF, B^LF, B^RF;
            θ ← θ + 1;
        l_{η+1} ∈ b_θ, l_seed ← l_{η+1}, l_η ← l_{η+1};
    while !found do
        for b ∈ B^(SF,RF,LF) do
            if ∃ b[StartWord] ∈ K^l then
                if there is tabular structure in b or its
                neighbor boxes then
                    get F_table; found ← 1; break;
        foreach b with b.fontsize() = F_table do
            if ∃ b[StartWord] ∈ K^l and has a tabular
            structure then
                b is a table;
end
```

3.4 TABLE METADATA

Some characteristics of our table metadata extraction are: 1) The table metadata should have meaningful names. 2) They should be easily stored for indexing and searching. 3) They can be combined differently according to the users' purposes. For example, only sharing the text or displaying the original layout.

Tables present unique challenges to IR systems because of the diverse media, different press layouts, cell types, table elements, etc. No formal rules/standards exist for designing table or for searching them. In order to be able to characterize tables occurring in very diverse composite documents, we have designed a rich and flexible representation scheme for table metadata that describe tables in digital documents. We classify the table metadata into six mutually exclusive categories: 1) table environment/geography (Document-level), 2) table-frame metadata, 3) table affiliated metadata, 4) table-layout metadata, 5) table cell-content metadata, 6) and table-type metadata. Figure 4 shows a segment of a table metadata file.

Table Environment/Geography Metadata. The *table environment/geography metadata* includes the information of the document where a table is located, such as *Document Medium Type* (HTML, PDF, image, PS, text, email, etc), *Document Page Number* of the table (the index number of the page where the table is located), *Document Title* (the paper title shown in the journal or conference proceedings, usually in a large font size), *Document Author*, *Document Origination* (the name of the journal or conference), *Document Age*, *Table Starting Position* (the X and Y-axis coordinates of the starting place of the table), etc. This metadata can facilitate the table searching if users only know pieces of the document information or wish to restrict the search to certain types of documents.

Table-Frame Metadata. The Table Frame metadata records whether there are frames in the four sides around a table. The values can be left, right, top, bottom, all, none, top and bottom, left and right.

Table Affiliated Metadata. A table has several affiliated elements. *Table Caption* is the caption (sentence(s)) that appears along with the table, e.g., *"Table 1. Molecular Properties of Tested Polymers"*. *Table Caption Position* is the position of the caption with relation to the body of the table: above or below. *Table Footnote* is text that explains the information in the table and usually appears below the table body. *Table Reference Text* is the text in the document body that refers to the table and discusses the content of the table.

Table Layout Metadata helps to capture the visualization of the original table. It is composed of *Table Width* (the width of the table boundary), *Table Length* (the length of the table boundary), *Number of Columns*, *Number of Rows*, *Stub Separator* (vertical ruling), *Boxhead Separator* (horizontal ruling), and *Column Width*, *Row Length*, *Column Headers*, *Row Headers*, and *Horizontal Alignment* (the values can be "flush left, right, and central).

Table Cell Content Data refers to the values in each cell of a table, and enables people to search tables based on the contents of their cells. Content in *Cell[i, j]* is the content in the cell that is located in the i^{th} row and the j^{th} column of a table.

Table Cell Type Metadata records the type of a table based on the type of its cells: *Numerical* and/or *Symbolic*. If the table contains cells with numeric information, it gets a type *Numerical*, if it has symbols, like, text, equations, etc., then it is marked *Symbolic*, Numerical tables can also be further divided into number tables, mathematical equation tables, percentage tables, and so on. Symbolical tables include character tables, image tables, formula tables, and so on. A table can have both numerical and symbolic cells.

3.5 The Table Ranking Algorithm – TableRank

One key question in information retrieval is how to rank matched results based on their relevance to a query. However, The existing ranking schemes are inadequate and are not designed for table search. Our *TableSeer* search engine has an innovative table ranking algorithm – *TableRank*. Given a user query, *TableRank* returns the matched tables in a descendant order according to their relevance scores. Different from the popular web search engines, our *TableRank* rates the <query, table> pairs instead of the <query, document> pairs.

```
<table-metadata>
<property>
 <name>Paper Title</name>
 <value>Dissolution of albite glass and crystal</value>
</property>

<property>
 <name>Table Caption</name>
 <value>Table 2. Comparison of crystalline and amorphous albite dissolution rates</value>
</property>

<property>
 <name>Table Column Head</name>
 <value>Type of experiment Initial pH Final pH Temperature</value>
 <description>……</description>
</property>
        ……
</table-metadata>
```

Figure 4: An Example of the Table Metadata File

TableRank tailors the traditional vector space model to rate the <query, table> pair by replacing the document vectors with the table vectors. As shown in Table 1, each row is a query or a table vector. To determine the weight for each term in the vector space, we design an innovative term weighting scheme: Table Term Frequency - Inverse Table Term Frequency (TTF-ITTF), a tailored *TF-IDF*[6] weighting scheme. Compared with *TF-IDF, TTF-ITTF* demonstrates two major advantages. First, it calculates the term frequency in the table metadata file instead of the whole document, which prevents the false positive results. Second, it calculates the weight of a term with a comprehensive view.

We divide the features we consider into two groups: *query-dependent* features and *query-independent* features. Query-dependent features include the traditional features of a document (e.g., the document title, the author/affiliation, the abstract, etc) and the structural features of a table (e.g., the table caption, the table column header, etc). Query-independent features include document citation, document freshness, and document origination, etc. We consider the query-independent features because when several similar tables match the end-user's query, intuitively, the end-user is most interested in the tables in newly published articles (document freshness) with the high quality, e.g., a top conference (document origination) or with a large number of citations (document citation). They are the tables that the user probably has not seen before and is seeking. Overall, *TableRank* considers features from three levels: 1) the term level, 2) the table level, and 3) the document level to determine the final ranking score of a table.

TableRank algorithm first applies each impact feature to weight the terms in the vector space, then aggregates all these features to determine the final weight values. Cosine measure is used to determine the similarity between the query vectors and the table vectors. Section 4 describes the ranking algorithm in great details.

3.6 The Table Index and the Search Interface

Most existing text retrieval techniques rely on indexing keywords. Unfortunately, keywords or index terms alone cannot adequately capture the document contents, resulting in poor retrieval performance. In *TableSeer*, we eliminate this risk by indexing table metadata files, which concentrates the table-related information into a small-size file. *TableSeer* uses the Lucene Index Toolbox[5] to index tables. A "document" is created where the table metadata fill the

[5]http://lucene.apache.org/java/docs/index.html

"fields". *TableSeer* offers two levels of searches: basic search and advanced search. A basic search allows a search with simple keywords and then the matched results are returned in ranked order. For advanced search, users can set more complex queries (see Figure 5). Unlike the current search engines that index and check whole documents, *TableSeer* indexes and checks only the table metadata files.

In order to facilitate the browsing of results, *TableSeer* provides a user-friendly interface to present the sorted results. Figure 6 shows that, for each matched table, *area 1* lists the basic document information, e.g., the document title, author, and affiliation. *Area 2* lists the table location information and highlights the references to the table in the text. *Area 3* provides links for the original whole PDF document and the table metadata file. *Area 4* includes the snapshot of the matched tables. Once clicked, table information is enlarged for better visualization.

Figure 5: An Example of the Advanced Search Interface

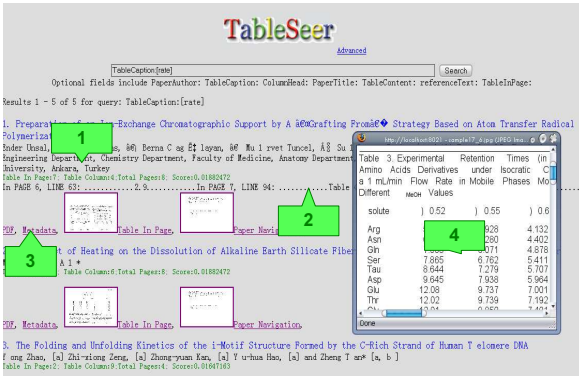

Figure 6: An Example of the Query Result by Advanced Search

4. TABLERANK – THE TABLE RANKING ALGORITHM

4.1 Similarity Measurement with a Tailored Vector Space Model

We first formalize the notations of the table ranking problem used throughout this paper. We represent the set of crawled and extracted tables as $T = \{\cup\ tb_j, j \in [1, b]\}$, where b is the total number of tables in a set of documents, $D = \{\cup d_\alpha, \alpha \in [1, N]\}$, where N is the total number of documents. $F(T, D)$ is the mapping function from $T \rightarrow D$. $\forall tb_j \in T, \exists d_\alpha \in D$ where the table tb_j comes from the document d_α. Also, $\forall tb_j \in T$, $\exists\ TM_j = \{\cup\ m_k, k \in [1, d]\}$, where, m_k is the metadata proposed to represent table tb_j, and d is the number of metadata.

A term t_i may repeatedly appear in different table metadata m_k (e.g., table caption, table column/row header, table reference text, even the document title). In particular, terms reflecting the core idea of an article usually appear more frequently and broadly. We believe that different m_k have different influences on the results and then each m_k has an assigned weight MW_k [12]. The same terms appearing in different m_k should be treated separately. Likewise, the same terms appearing in the same m_k should be treated equally.

Let $sim(tb_j, Q)$ denote the similarity between the table tb_j and the query Q. Both of the table and the query can be represented as a vector. Each vector is composed of a set of alphabetically ordered terms. All the table vectors and query vectors construct a vector matrix as shown in *Table 2*. Each row in the matrix represents the vector of a table tb_j ($j \in [1, b]$) or the query Q. Each table tb_j has k metadata elements describing it. $w_{i,j,k}$ is the term weight of the i^{th} term in the k^{th} metadata of the table tb_j and $w_{i,q,k}$ refers the term weight of the i^{th} query term in the k^{th} metadata. If the i^{th} term does not occur in the metadata m_k of the table tb_j, $w_{i,j,k} = 0$. Otherwise, $w_{i,j,k} > 0$. The term weight $w_{i,q,k}$ follows the same rule. As discussed, a term t_i can appear in multiple metadata. If the user specifies the query term location, e.g., the metadata *"Table Caption,"* the query term will only be filled in the metadata *"Table Caption"* columns. Otherwise, it will be filled to every metadata in *Table 2*.

TableRank constructs the vector matrix and measures the similarity between the query and each table by computing the cosine of the angle between these two vectors (the equation is shown in Equation 1).

$$sim(tb_j, Q) = cos(tb_j, Q) = \frac{\sum_{i=1}^{s} w_{i,j,k} w_{i,q,k}}{|tb_j||Q|} \qquad (1)$$

where, the final weight $w_{i,j,k}$ of the i^{th} term in the k^{th} metadata of the table tb_j is computed by Equation 2. The weighting scheme is also applicable to the query term $w_{i,q,k}$.

4.2 Term Weighting

Equation 2 shows that the final weight of a term $w_{i,j,k}$ comes from three levels: the term-level weight $w_{i,j,k}^{TermLevel}$, the table-level weight boost $TLB_{i,j}$, and the document-level weight boost DLB_j.

$$w_{i,j,k} = w_{i,j,k}^{TermLevel} * TLB_{i,j} * DLB_j \qquad (2)$$

In addition to the term-level weight, which contributes a great proportion to the final weight of a term, there are two levels of boost factors. *TLB* refers to those table-level features that improve the weight of a term (e.g., table frequency) while *DLB* refers to those document-level features that improve the weight of a term (e.g., journal rank, document citation).

4.2.1 The Term-Level Weight

TableRank uses an innovative weighting scheme: Table Term Frequency - Inverse Table Term Frequency (TTF-

Table 2: The Vector Space Model for rating <Query, Table> pairs

	$m_1(MW_1)$			$m_2(MW_2)$			...	$m_k(MW_k)$			TLB	DLB
	$t_{1,1}$	...	$t_{x,1}$	$t_{1,2}$	...	$t_{y,2}$	...	$t_{1,k}$	...	$t_{z,k}$	...	...
$tb1$	$w_{1,1,1}$	...	$w_{x,1,1}$	$w_{1,1,2}$	...	$w_{y,1,2}$	...	$w_{1,1,k}$	...	$w_{z,1,k}$	...	...
$tb2$	$w_{1,2,1}$	...	$w_{x,2,1}$	$w_{1,2,2}$	...	$w_{y,2,2}$	...	$w_{1,2,k}$	...	$w_{z,2,k}$	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	...	⋮	...	...	⋮	...	⋮	...	⋮	...	...	...
tb_b	$w_{1,b,1}$	...	$w_{x,b,1}$	$w_{1,b,2}$	...	$w_{y,b,2}$	...	$w_{1,b,k}$	...	$w_{z,b,k}$	...	...
Q	$w_{1,q,1}$			$w_{1,q,2}$			...	$w_{1,q,k}$			...	...
$ITTF$	...	...	...		...	...	...	...	...	...	...	...

ITTF) scheme. $TTFITTF$ is adapted from the TF-IDF[6] weighting scheme, a widely used weighting method for free-text documents. However, it is not suitable for table ranking because it only considers the term frequency and inverse document frequency and ignores many other important impact factors, e.g, term position. In contrast to TF-IDF, TTF-ITTF demonstrates two major differences. First, TTF-ITTF calculates term frequency in a table metadata file instead of the whole document. Second, the position of a term in the table and the document is considered. Therefore, we have

$$w_{i,j,k}^{TermLevel} = TTFITTF_{i,j,k} = TTF_{i,j,k} * ITTF_{i,j,k} \quad (3)$$

where $TTF_{i,j,k}$ is the term frequency of the table term t_i in the metadata m_k of the table tb_j and $ITTF_{i,j,k}$ is the inverse table term frequency. Like in the case of inverse document frequency, the idea behind using the $ITTF_{i,j,k}$ is that a term that occurs in a few tables is likely to be a better discriminator than a term that appears in most or all tables.

4.2.2 The Table-Level Boost (TLB)

Intuitively, a table itself is also important and influences the weight of terms in it. Besides term-level features, *TableRank* also considers table-level features, such as: 1) the table frequency, 2) the length of the text that exposits the table content in the document (namely the table reference text), and 3) the table position. These factors are embodied in the Table Level Boosting *(TLB)* factor as follows:

$$TLB_{i,j} = B_{tbf} + B_{trt} + (r * B_{tp}) \quad (4)$$

where B_{tbf} is the boost value of the table frequency, B_{trt} is the boost value of the table reference text, B_{tp} is the boost value of the table position, and r is a constant $\in$[0,1]. If users specify the table position in the query, $r = 1$, otherwise $r = 0$.

4.2.3 The Document-Level Boost (DLB)

Unlike *TTF-ITTF* and *TLB*, *Document Level Boosting* (*DLB*) considers query-independent (static) factors. *DLB* indicates the overall importance of a document where a table appears. For a high quality document, its tables are also inclined to be important and the terms should receive a high document-level boosting.

Given a document d_j, IV_j denotes its document Importance Value (IV). *TableRank* sets IV_j for the document d_j by considering the following three factors: 1) the inherited citation value (IC) from the ones who cite the document, 2) the document origin (DO, and 3) the document freshness(DF_j). Suppose there are x documents $(d_1, d_2, ..., d_x)$ that cite the document d_j, then IV_j is defined using the following recursive equation 5.

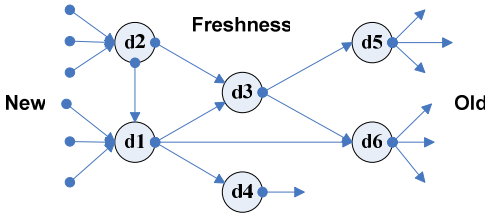

Figure 7: An Example of the Citation Network

$$DLB_j = IV_j = IC_j * DO_j * DF_j == (\frac{\sum_{v=1}^{x} IV_v}{x}) * DO_j * DF_j \quad (5)$$

IC relies on the nature of the scientific document itself by using its crucial citation link structure as a major indicator. In essence, a citation link from one document D_α to another document D_β can be seen as an endorsement of D_α. We represent it as $D_\alpha \rightarrow D_\beta$ and the document D_β is considered older than the document D_α. The number of times D_α is cited and the quality of the ones that cite D_α are indicative of the importance of D_α. By this way, academic documents construct a citation network which is a *Directed Acyclic Graph (DAG)* as shown in *Figure 7*. Our algorithm computes the document importance of each document using this citation network. Each node in the *DAG* highlights the fact that the importance of a node is essentially obtained as a weighted sum of contribution coming from every path entering into the node.

DO_j can be set based on the journal/conference prestige. In each research field, scholarly journals or conferences are scored based on the opinions of field experts. We get this information from three sources: the impact ranking for the computer science publications estimated by *CiteSeer*, for chemistry papers listed in *Wikipedia*[6], and a comprehensive journal impact factor for all the fields in *CNCSIS*[7]. Different scoring schemes have different score ranges from $[0, 3.31]$ to $[0, 53.055]$. We normalize the different ranges into the same measure range $[1, 10]$.

DF_j is set according to the publication date of the document. *TableRank* assigns more weight to fresher documents for two reasons: 1) Researchers usually are inclined to search for the more recently published documents because these documents typically reflect the latest research trends and results. 2) There is a need to eliminate "The Rich Get Richer" phenomenon which is begot by the boosting value of the citation frequency (the more citation, the higher boosting is). The documents with high citation counts are inclined to be cited again because they are more likely to be found eas-

[6]http://en.wikipedia.org/wiki/List_of_scientific_journals_in_chemistry/
[7]http://www.cncsis.ro/PDF/IF_2004.pdf

ily. However, newly published documents are unlikely to attract attention and be cited within a short period of time. Considering these two reasons, we boost the rankings of recent documents. The more recent the document, the greater the boost is. On the other side, because of the high-speed of information propagation and the convenient source sharing (e.g., DBLP bibliography[8]), in our *TableRank*, we only apply the *DF* boost to brand-new documents, which are published within recent two years.

5. EXPERIMENTS AND RESULTS ANALYSIS

In this section, we demonstrate the experimental results of evaluating the table search engine, *TableSeer*. Before describing the experimental details, we first discuss document collection.

5.1 Document Collection

Although *TableSeer* can be used to search for tables contained in documents of varying formats and media types, we focus on tables in scientific documents in PDF format in this work. The document collection comes from three sources: 1) scientific digital libraries (Royal Chemistry Society[9]), 2) the web pages of research scientists in chemistry departments in universities, for example, UCLA[10] lists numerous addresses of academic institutions in the field of chemistry, which are set as seeds for the table crawler, and 3) the CiteSeer archive. The total number of collected PDF documents is approximately 10,000. These documents belong to more than 50 journals and conferences in a variety of research fields, e.g., chemistry, biology, computer science, etc. All the documents span the years 1990 to 2006. In addition, the original hierarchy of documents is also recorded, e.g., coming from the same journal, or the same web site. A random selection of 100 documents indicates that 78% of them has at least one table and most of them have more than one.

5.2 Experimental Results of Table Detection

We perform a five-user study to evaluate the quality of the table detection and the table metadata extraction separately. Each user randomly checks 20 documents and all the corresponding table metadata files. The evaluation metrics are precision and recall. The experiment on table detection is conducted on a document set with 200 randomly selected PDF documents, which are from journal/conference categories of Chemistry, Biology, and Computer Science from RSC and CiteSeer. Given the number of true tables extracted by *TableSeer* A, the number of true positive tables but overlooked B, and the number of true negative tables that is misidentified as tables C, the *Precision* is $\frac{A}{A+C}$, and the *Recall* is $\frac{A}{A+B}$. Within the 200 documents, 397 tables exist (see Table 3). *TableSeer* recognizes 371 tables and all of them are real tables, which means the number of true negative tables is 0. Based on these limited experimental results, the precision value is 100% and the recall value is 93.5%.

Three main reasons explain why we miss 26 true positive tables. First, in some PDF documents, tables are created as embedded images. In the current state, *TableSeer* only processes the text-based tables and simply filters out the image tables. However, these images and their location information are stored. Existing approaches dealing with image table extraction [5] [8] can be easily integrated into the *TableSeer* to fix this problem. Second, we use predefined keyword list to detect table candidates. This predefined keyword list contains all the possible starting keywords of table captions, such as "Table, TABLE, Form, FORM," etc. Most tables have one of these keywords in their captions. However, some tables are labeled using other keywords, especially the documents in computer science field, which usually names the tables with the keyword "Figure." This confuses with real image figures. To avoid real figures and to keep high efficiency, *TableSeer* overlooks such "wrongly" labeled tables in current stage. However, this problem can be overcome by heuristics that identify the grid-structure cells by white-space analysis. Third, the performance of the text information stripper directly affects the table extraction results. Currently, *PDFBOX*[11] is used to fetch all the text information in the documents. Characters missing and space insertions are two typical errors inherited from *PDFBOX*, which may tamper with the table caption. For example, the word "Table" can be splintered into "a ble." This problem is orthogonal to our work and we hope that these problems will be addressed independently by the designers of the text stripper.

Table 3: The testing set for table detection

Fields	Documents	Tables
Chemistry	100	245
Biology	50	84
Computer Science	50	68

5.3 Experimental Results of Table Metadata Extraction

We evaluate the table metadata extractor of *TableSeer* using the 371 recognized tables from the document set. Precision and recall remains the measurement functions. A is the number of true positive metadata extracted by our algorithms and labeled with the correct metadata labels, B is the number of true positive metadata but overlooked by *TableSeer*, C is the number of true negative metadata that are misidentified as another metadata.

Table 4 demonstrates the performance of the metadata extraction over the test set. Apparently, *TableSeer* has good performance on table metadata extraction. The main reasons for inaccuracy of the recall are: first, some journal names and volume information are mislabeled as "Document Title" metadata. Most journals put the journal names and the publication information in the page header or footer. However, some of the documents display the journal/volume information in the ordinary position where the document title is. This problem can be solved by adding heuristic rules that look for patterns to identify journal/volume information. Second, some documents omit the keyword "Abstract" when they start the abstract section using the same font size as the size of the author/affiliation section. This can be fixed by restricting the size of the author/affiliation section and by looking for text similarity. The abstract usually has similar keywords as in the main text, whereas the author/affiliation will have different words than the main text.

[8]http://www.informatik.uni-trier.de/ ley/db/
[9]http://www.rsc.org/
[10]http://www.chem.ucla.edu/VL/Academic.html
[11]http://www.pdfbox.org/

Table 4: Table Metadata Extraction Performance

Table Metadata	Precision(P %)	Recall(R %)
Document Type	100.00	100.00
Document Page Number	100.00	100.00
The page number of table	100.00	100.00
Document Title	95.65	95.65
Document Author	92.58	92.58
Table Caption	95.96	95.96
Table Column Header	93.79	93.79
Table Content	90.15	90.15
Table Caption Position	100.00	100.00
Table Footnote	82.77	82.77
Table Reference text	100.00	100.00

Third, some table footnotes, especially short ones, are recognized as a row of the table because some tables have such cross-column rows.

In order to obtain the table reference text metadata, a "keyword matching" method is used to collect all the sentences where the table keywords appear. Some explanations also exist around without the keywords. However, it is extremely difficult to separate them from other texts. Furthermore, for those tables that use a cross-line to separate different parts, we only recognize the first part because of the difficulty in telling whether the cross-line belongs to the table or the body text of the document. This can be solved by using the font-size difference, existence of cells, and the indication of the white spaces.

5.4 Experimental Results of Table Ranking

Comparison of ranking methods is complicated for two main reasons. First, it is difficult to find a common test-bed for different search engines. Although Google Co-op[12] can enable the comparison between TableRank and Google by harnessing the power of Google search technology, the dynamic web and continued crawlers imply that the two sets of documents may not be the same. Second, even with a common test-bed, no universally recognized measurement of quality for a table ranking scheme exists.

In order to set up a well-accepted measurement, we establish a "*golden standard*" to define the "*correct*" ranking based on human judgement. A survey of six experienced testers, who frequently use search functions of different search engines, generated the "*golden standard*" of each document set in the test-bed. For each issued query, the testers determine how many results, among the returned hits, the testers order the relevant lists.

For each ranking scheme, we apply *pairwise accuracy* to evaluate the ranking quality. If H_R is the ranking decided by human judgement and T_R is ranking decided by the search engines, the *pairwise accuracy* can be defined as the fraction of times that search engines and human judges agree on the ordering of tables: *pairwise accuracy* $= \frac{|H_R \bigcap T_R|}{H_R}$. In this section, we provide quantitative comparisons between *TableRank* and other popular web search engines based on experimental results. The average ranks of the six human testers is obtained and the results are ranked using this average to obtain H_R. We will use a distance based measure among the ranked orders in the future.

Before evaluating the performance of the table ranking, we identify the returned true tables because most search engines can not identify documents with tables. We should filter out all the false tables and only compare the ranking on these real tables. We randomly select 100 terms such as "gene", "protein", "query,", and use each term as a query in *TableSeer*. For the other two search engines: *Google Scholar* and *CiteSeer*, the query is set as the term together with an additional keyword "Table." An example is applying "Alkaline" to *TableSeer* and applying the combination of the term "alkaline" and "Table" to *Google Scholar*. The combination includes "alkaline, Table", "alkaline Table", "Table, alkaline", "Table alkaline", etc. Because it is not easy to know the total number of tables in the test-bed, we use the *precision* value as the measurement. Another reason is that with the growing size of the Web collections, users are now primarily interested in high accuracy, defined as high precision. High precision is very important because users typically look at very few results and mostly look at the top N items of the returned results from the search engines. Recognizing this trend, we manually examine the first 20 results returned by both each search engine. Comparing with the precision value of our *TableSeer* (100%), the precision value of *Google Scholar* is 73.4% and the precision value of *CiteSeer* is 76.7%. Thus, it can be seen that *TableSeer* outperforms the other two search engines.

Table 5: The Basic Ranking Results on the Manually Created Document Sets

Ranking	The Method to set-up the test-bed	Accuracy (%)
Google	Custom search engine	51.8
Google Scholar	bottom-up method	52.72
CiteSeer	bottom-up method	55.35
TableSeer	Both methods	69.61

The remaining question is which search engine has the best ranking scheme? We adopt two methods to set up the common test-bed: the manually "bottom-up" method and the custom search engine method. The manually "bottom-up" method constructs a test-bed using the following two steps: 1) applying a query (e.g., DNA) together the keyword "Table" to an existing search engine, e.g., *CiteSeer*; 2) from the numerous returned results, we pick out a set of "real" hits in PDF format together with their ranking orders. We set these PDF documents as the common test-bed for both *TableRank* and other search engines. In the second custom search engine method, we set up a common test bed for *TableRank* and *Google* using the *Google* Custom Search engine API. We register a *Google* account and use the *Google* custom search engine API[6] to build a customized search engine. We set the seed URLs (e.g., a chemistry journal website[13]) as one or several websites where *TableSeer* crawls. All the documents in the seed URLs construct the common test-bed. For all the documents in the common test-beds, *TableSeer* extracts and indexes the metadata file for each table. We try 20 randomly selected search queries on both *TableSeer* and the *Google* custom search engine to compare their search results. We collect 20 document sets for 20 random queries. Table 5 displays the average pairwise accuracy results made by six testers.

TableRank decides the relevance score for each result by comprehensively considering multiple impact factors from different perspectives. In order to find out how well each impact factor performs and how heavily each of them influence the final ranking order, we implement *TableRank* algorithm

[12]http://www.google.com/coop/

[13]http:/www.rsc.orgdelivery_ArticleLinking DisplayArticleForFree.cfm?doi=b2*

Table 6: Results for Individual Impact Factor in TableRank Algorithm

Impact Factors	Accuracy (%)
TFIDF	50.19
TTFITTF without MW	61.46
TTFITTF with MW	63.55
TLB	29.60
DLB	40.33
All factors	69.61

Table 7: The Ablation Experiment to Learn the Real Contribution of Each Factor of TableRank Algorithm

Without Factor	Accuracy (%)	Decrease in Accuracy (%)
DLB	68.50	1.11
TLB	69.46	0.15
MW	68.19	1.42
TTFITTF, MW	58.05	9.56

on each impact factor, independently, and apply varied combination of the impact factors by gradually adding one new factor at a time. Such implementation can not only reveal how sensitive *TableRank* is to each impact factor, but also show how to adjust the parameters for better results.

The results for individual impact factors are shown in Table 6, which shows the application of a single factor each time. The first run with the applied traditional *TFIDF* weighting scheme on the whole document shows the accuracy rate compared to the "golden standard". In the second and the third runs, we update the *TFIDF* to the *TTFITTF* weighting scheme with/without the metadata weighting scheme. Next, the *TLB* factors are applied, and in the last run, the *DLB* factors are tested.

It is not surprising to see that *TTFITTF* with/without metadata weight contribute heavily to the final ranking quality, and *TLB* and *DLB* do not perform well in isolation. Even table 6 shows the effectiveness of each individual factor. In order to have a better measurement of the contribution of each factor, in the context of all the other factors, we perform an ablation experiment. The basic idea is to compare the ranking results based on all the factors except the studied ones with the ranking results generated by *TableRank* with all the factors. The detailed results are shown in table 7.

The results in table 6 are consistent with the results of the ablation experiment in table 7. Table 7 reconfirms the vital role of the *TTFITTF* weighting scheme. *TTFITTF* acts as the most important impact factor, and second and third in importance are the metadata weight and *DLB* respectively.

6. CONCLUSIONS AND FUTURE WORK

To facilitate table extracting and searching, we devise a novel table-specific search engine, *TableSeer*. An extensive set of table metadata is proposed to precisely represent a table. Because the traditional TF-IDF approach is no longer suitable for table search, we also studies how to calculate the ranking scores of tables contained in scientific documents based on multiple ranking factors from three levels: the term level, the table level, and the document level. We propose a novel table ranking algorithm, *TableRank*, which combines the query-dependent features with the query-independent features of a document to rank tables and their containing documents in response to an end-users query. Experimental results demonstrate that *TableSeer* outperforms the widely used search engine, e.g. *Google Scholar* on searching information contained in tables. Scientific documents are the focus of our current work. However, there are a large numbers of tables in documents in other areas. Current parameter settings are based on empirical results and further work is needed to establish optimal settings for ranking and searching. Future user studies will indicate the validity of the ranking methods.

7. REFERENCES

[1] http:/en.wikipedia.orgwikisearch_engine.

[2] N. A. A. Amano. Graph grammar based analysis system of complex table form document. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 916–920, 2003.

[3] W. G. B. Krupl, M. Herzog. Using visual cues for extraction of tabular data from arbitrary html documents. In *In Proc. of the 14th Int'l Conf. on World Wide Web*, pages 1000–1001, 2005.

[4] S. T. H. Chen and J. Tsai. Mining tables from large scale html texts. In *In Proc. 18th Int'l Conf. Computational Liguistics, Saarbrucken, Germany*, 2000.

[5] X. W. W. B. D. Pinto, A. McCallum. Table extraction using conditional random fields. In *In proceeding of Proceedings of the 26th ACM SIGIR, Toronto, Canada*, July 2003.

[6] C. B. G. Salton. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management 24(5)*, pages 513–523, 1988.

[7] J. T. K. H.T. Ng, C. Y. Lim. Learning to recognize tables in free text. In *In Proc. of the 37th Annual Meeting of the Association of Computational Linguistics on Computational Linguistics*, pages 443–450, 1999.

[8] R. H. J. Ha and I. Philips. Recursive x-y cut using bounding boxes of connected components. In *In Proc. Third Int'l Conf. Document Analysis and Recognition*, pages 952–955, 1955.

[9] N. G. J. Shin. Table recognition and evaluation. In *In Proc. of the Class of 2005 Senior Conf., Computer Science Department, Swarthmore College*, pages 8–13, 2005.

[10] T. W. J.H. Shamilian, H.S. Baird. A retargetable table reader. In *In Proc. of the 4th Int'l Conf. on Document Analysis and Recognition*, pages 158–163, 1997.

[11] T. G. Kieninger. Table structure recognition based on robust block segmentation. In *In Proc. Document Recognition V, SPIE, volume 3305*, pages 22–32, January 1998.

[12] Y. Liu, K. Bai, P. Mitra, and C. L. Giles. Tablerank: A ranking algorithm for table search and retrieval. In *AAAI*, 2007.

[13] Y. Liu, P. Mitra, C. L. Giles, and K. Bai. Automatic extraction of table metadata from digital documents. In *JCDL*, pages 339–340, 2006.

[14] P. Pyreddy and W. Croft. Tintin: A system for retrieval in text tables. In *In Proceedings of the Second International Conference on Digital Libraries*, pages 193–200, 1997.

[15] D. B. R. Zanibbi and J. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. In *Int'l J. Document Analysis and Recognition, Vol. 7, No.1*, pages 1–16, 2004.

[16] A. D. T. Kieninger. Applying the t-rec table recognition system to the business letter domain. In *In Proc. of the 6th Int'l Conf. on Document Analysis and Recognition*, pages 518–522, September 2001.

[17] J. Wang and J. Hu. A machine learning based approach for table detection on the web. In *Proceedings of the 11th Int'l Conf. on World Wide Web (WWW'02)*, pages 242–250, Nov 2002.

[18] X. Wang. Tabular abstraction, editing, and formatting. In *Ph.D. Thesis, Dept. of Computer Science, University of Waterloo*, 1996.

[19] J. H. Y. Wang. Detecting tables in html documents. In *In Proc. of the 5th IAPR Int'l Workshop on Document Analysis Systems, Princeton, NJ*, 2002.

[20] R. H. Y. Wang, I.T. Philips. Automatic table ground truth generation and a background-analysis-based table structure extraction method. In *In Proc. of the 6th Int'l Conference on Document Analysis and Recognition*, page 528, September 2001.

[21] R. M. H. Y. Wang, L.T. Phillips. Table structure understanding and its performance evaluation. In *Pattern Recognition, 37(7)*, pages 1479–1497, July 2004.