TableRank: A Ranking Algorithm for Table Search and Retrieval

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

College of Information Sciences & Technology
Pennsylvania State University
University Park PA, USA 16802
Email: {yliu, kbai, pmitra, giles}@ist.psu.edu

Abstract

Tables are ubiquitous in web pages and scientific documents. With the explosive development of the web, tables have become a valuable information repository. Therefore, effectively and efficiently searching tables becomes a challenge. Existing search engines do not provide satisfactory search results largely because the current ranking schemes are inadequate for table search and automatic table understanding and extraction are rather difficult in general. In this work, we design and evaluate a novel table ranking algorithm – TableRank to improve the performance of our table search engine TableSeer. Given a keyword based table query, TableRank facilities TableSeer to return the most relevant tables by tailoring the classic vector space model. TableRank adopts an innovative term weighting scheme by aggregating multiple weighting factors from three levels: term, table and document. The experimental results show that our table search engine outperforms existing search engines on table search. In addition, incorporating multiple weighting factors can significantly improve the ranking results.

Introduction

With the rise of the World Wide Web to the biggest pool of information, tables that often are used to present the latest experimental results, to list the statistical data, and to show the financial activities of a business, etc., have gradually accumulated a significant amount of valuable information. In order to unlock the information in tables, table search is a highly desirable feature for the Web search. Due to the huge volume, seeking a relevant table is more difficult than finding a needle in the haystack. Unfortunately, current search engines are deficit in helping the end-users to find satisfactory table results. The difficulty of automatic table extraction, especially from the un-tagged documents, makes automatic table extraction and table search problems challenging. Moreover, the ranking schemes embedded in the existing search engines are inadequate and are not designed for table search.

The performance of a ranking scheme is crucial to the success of a search engine. In the literature, most approaches primarily focus on the similarity of a query and a page, as well as the overall page quality using the vector space mechanism to calculate the relevant score and use the ordinary TFIDF (Salton & Buckley 1988) technique to determine the weight of each term in the vector space. There are several other methods to calculate the similarity score between a document and the query. For example, the vector space model with pivoted document length normalization(Singhal, Buckley, & Mitra. 1996), the language modeling approach (LM)(Ponte & Croft. 1998), the Okapi BM25 formula(Robertson, S.Walker, & Beaulieu 1998), etc. PageRank is another popularly-used ranking technique designed by Brin and Page(Sergey Brin 1999). With increasing popularity of search engines, many works focus on how to improve the rankings by tailoring the PageRank algorithm, e.g., incorporating implicit feedback (i.e., the actions users take when interacting with the search engine)(Agichtein, Brill, & Dumais 2006), using historical data(Fernández, Vallet, & Castells 2006), overcoming the drawback of the random walk model(Feng et al. 2006), optimizing scoring functions and indexes for proximity search in type-annotated corpora(Chakrabarti, Punjani, & Das 2006), re-ranking using links induced by language models(Kurland & Lee 2005), using annotations in enterprise search(Dmitriev et al. 2006), etc.

In this paper, we explore the issue about how tables should be ranked. We propose TableRank, a table ranking algorithm to facilitate our table search engine TableSeer to bring orders to the relevant tables. TableRank considers multiple features of a table and the document it appears in, and aggregates these features to determine the final ranking of the table with respect to a query. In order to evaluate the performance of TableRank, we compare TableSeer with several popular web search engines. Because the popular web search engines make no effort at treating tables specially, we propose two methods to set up the common test-beds. Our experiments on scientific documents show that TableRank can significantly improve the ordering of the top results in table search by incorporating features from different levels.

The remainder of the paper is organized as follows. Section 2 overviews the TableSeer system. Section 3 elaborates the table ranking algorithm. Section 4 discusses the setting of several important parameters and the experimental results. Section 5 concludes this paper and our future work.
TableRank: The Table Ranking Algorithm

Equation 1 shows that our TableRank algorithm calculates the final weight of a term \( w_{i,j,k} \) in three levels: the term-level weight \( w_{i,j,k}^{\text{Term Level}} \), the table-level weight boost \( TLB_{i,j} \), and the document-level weight boost \( DLB_j \).

\[
    w_{i,j,k} = w_{i,j,k}^{\text{Term Level}} \cdot TLB_{i,j} \cdot DLB_j
\]

TableRank algorithm first weights the terms in the vector space in each level, then aggregates the three levels to determine the final weight values.

Term Level Weighting: TTF-ITTF

TableRank uses an innovative weighting scheme in the term level: Table Term Frequency - Inverse Table Term Frequency (TTF-ITTF) scheme, a tailored TF-IDF (Salton & Buckley 1988). TF-IDF is 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 advantages. First, TTF-ITTF calculates term frequency in a table metadata file instead of the whole document, which prevents the false positive hits.

Table 1: The Vector Space for Tables and Queries

<table>
<thead>
<tr>
<th>Table 1, Row</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{1,1} )</td>
<td>( f_{1,2} )</td>
<td>( f_{1,3} )</td>
<td>( f_{1,4} )</td>
<td>( f_{1,5} )</td>
<td>( f_{1,6} )</td>
</tr>
<tr>
<td>( b_1 )</td>
<td>( b_{1,1} )</td>
<td>( b_{1,2} )</td>
<td>( b_{1,3} )</td>
<td>( b_{1,4} )</td>
<td>( b_{1,5} )</td>
</tr>
<tr>
<td>( b_2 )</td>
<td>( b_{2,1} )</td>
<td>( b_{2,2} )</td>
<td>( b_{2,3} )</td>
<td>( b_{2,4} )</td>
<td>( b_{2,5} )</td>
</tr>
<tr>
<td>( q )</td>
<td>( q_{1,1} )</td>
<td>( q_{1,2} )</td>
<td>( q_{1,3} )</td>
<td>( q_{1,4} )</td>
<td>( q_{1,5} )</td>
</tr>
<tr>
<td>( t )</td>
<td>( t_{1,1} )</td>
<td>( t_{1,2} )</td>
<td>( t_{1,3} )</td>
<td>( t_{1,4} )</td>
<td>( t_{1,5} )</td>
</tr>
</tbody>
</table>
Second, the position of a term both in the table and the document is considered. Therefore, we have

\[ w_{i,j,k}^{TermLevel} = TTF^{ITFF}F_{i,j,k} = TTF_{i,j,k} + ITTF_{i,j,k} \]  \hspace{0.5cm} (2)

where \( TTF_{i,j,k} \) is the term frequency of the table term \( t_i \) in the metadata \( m_k \) of the table \( t_j \) and \( ITTF_{i,j,k} \) is the inverse table term frequency. The idea behind the \( ITTF_{i,j,k} \) is that a term occurring in a few tables is likely to be a better discriminator than a term appearing in most or all tables.

For free-text document retrieval, \( \frac{f_{i,j,k}}{tf_{i,j}} \) is widely used to calculate the Term Frequency (TF), where \( f_{i,j,k} \) denotes the occurrence times of the term \( t_i \) in the metadata \( m_k \) of the table \( t_j \). However, this fraction is not suitable for the table texts. Based on our observation of hundreds of table metadata files, we notice that we should treat the texts of the table metadata files as semi-structured texts instead of the the free texts. Essentially, the semi-structured text is typically a short summary of an object, rarely more than a few words long. Table metadata files, search queries, and document abstracts are, many times, similar in nature: all of them tend not to be complete sentences. They are a few words long and express a summary of a subject that the user is interested in. According to (Salton & Buckley 1988), we have the following Equation 3 to compute the TTF.

\[ TTF_{i,j,k} = (p + (1 - p)) \frac{f_{i,j,k}}{tf_{i,j}} MW_k \]  \hspace{0.5cm} (3)

where \( p \) is a parameter between 0 and 1. To weight terms in the table and query, \( p \) is set to be 0.5 according to (Salton & Buckley 1988). \( ITTF_{i,j,k} \) is computed by Equation 4:

\[ ITTF_{i,j,k} = \log_2(\frac{b}{IDF_{i,j,k}}) + 1 \]  \hspace{0.5cm} (4)

where \( b \) is the number of tables in set \( T \). \( IDF_{i,j,k} \) denotes the number of tables that term \( t_i \) occurs in the metadata \( m_k \).

Table Level Boosting (TLB)

Intuitively, a table itself is also important and influences the weight of terms in it. Besides the term-level features, TableRank also considers table-level features like: 1) the table frequency, 2) the length of the text that elaborates 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:

\[ TLB_{i,j} = B_{tb_i} + B_{trt} + (r \times B_{tp}) \]  \hspace{0.5cm} (5)

where \( B_{tb_i} \) 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, \( r = 1 \), otherwise \( r = 0 \). The principle behind the Equation 5 is that if the query terms appear in several tables of a same document, the document has a high relevance potential and more credits should go to the terms appearing in the tables of this document. Section 4 displays some experimental results to show the effects with and without these boosts. In order to calculate \( B_{tb_i} \), we propose another weighting scheme – Table Frequency Inverse Table Frequency (TFBF-ITBF). Both TBF-ITBF and \( TTF-ITTF \) derive from \( TFIDF \). The former works on the table level while the later works on the table level.

\[ B_{tb_i} = TBF_{i,j} \times ITBF_i = \frac{tb_{i,j}}{tb_f} \times (\log_2 \frac{b_j}{b} + 1) \]  \hspace{0.5cm} (6)

where \( tb_{i,j} \) denotes the number of tables that include the term \( t_i \) in the document \( d_j \), and \( tb_f_j \) denotes the total number of tables in the document \( d_j \). \( b \) is the total number of tables in the table set \( T \), and \( b_i \) denotes the number of tables in \( T \) in which the term \( i \) appears.

If the query terms appear in a table with a long table reference text or the size of the table itself is large, users may have a great possibility for achieving their desired results.\( nlr_j \) denotes the normalized total length of the table size and the reference text of \( tb_i \) over the entire document length.

\[ B_{trt} = nlr_j \]  \hspace{0.5cm} (7)

Tables usually appear in the document sections named “Experiment”, “Evaluation”, “Result Analysis”, etc. Sometimes, tables appear in “Related Work” or “Architecture” or “System Designing” sections. The position factor is not considered unless users specify the table position because we cannot make a conclusion about the priority of different sections. For those tables in the specified document section, terms should also have a table position boosting value \( B_{tp} \).

Combining these factors, we have

\[ TLB_{i,j} = \begin{cases} \frac{tb_{i,j}}{tb_f} \times (\log_2 \frac{b_j}{b} + 1) + nlr_j + B_{tp} & \text{if } r = 1 \\ \frac{tb_{i,j}}{tb_f} \times (\log_2 \frac{b_j}{b} + 1) & \text{if } r = 0 \end{cases} \]  \hspace{0.5cm} (8)

Document Level Boosting (DLB)

Unlike the \( TTF-ITTF \) and TLB, Document Level Boosting (DLB) is a query-independent (static) ranking. 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. The terms should receive a high document-level boosting. The DLB effectively complements the dynamic table ranking algorithm.

Given a document \( d_j \), setting its document importance value \( IV_j \) considers three factors: 1) the inherited citation value (IC) from the ones who cite the document, 2) the document origin value(DO), and 3) the document freshness(DF). Suppose there are \( x \) documents \((d_1, d_2, \ldots, d_x)\) that cite the document \( d_j \), then \( IV_j \) is defined using the following recursive equation:

\[ DLB_j = IV_j = IC_j \times DO_j \times DF_j = \sum_{x=1}^{\infty} \frac{IC_i}{x} \times DO_i \times DF_i \]  \hspace{0.5cm} (9)

The inherited citation value (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 \( D_\alpha \) to another document \( D_\beta \) can be seen as an endorsement of \( D_\alpha \). We represent it as \( D_\alpha \rightarrow D_\beta \) and the document \( D_\alpha \) is considered older than the document \( D_\beta \). By this way, academic documents construct a citation network which is a Directed Acyclic Graph (DAG) as shown in Figure 3. The number of times \( D_\alpha \) is cited and the quality of the ones that cite \( D_\alpha \) are indicative of the importance of \( D_\alpha \). 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.

Similar to the PageRank approach (Brin & Page 1998), \( \forall d_j \in D, IC_j \) is defined recursively based on all the “incoming links” – the documents that cite \( d_j \). The influence
Figure 3: An Example of the Citation Network of a document may be repeatedly considered because of the nature of the citation structure. For example, \( d_6 \) in Figure 3 inherits \( IV \) from \( d_3 \) and \( d_1 \) but \( d_1 \) also affects \( d_3 \). An exponential decay can be useful to deal with the impact of the propagated importance. Unlike the PageRank approach, the inherited \( IV \) should not be divided by the number of the outbound links. We believe that \( IV \) of a document should not be decreased as the number of references increase. In Figure 3, for each document in \( D = \{d_1, d_2, d_3, d_4, d_5, d_6\} \), the initial \( IC \) of these six documents is \( 1/6 = 0.167 \).

\( DO_j \) can be set based on the journal/conference ranking while \( DF_j \) is set according to the publication date of the document. Recent published documents are preferred. The \( IC \) will be impacted by the publisher/Journal/Proceeding quality of the documents that cite it. Citation cast by documents that are themselves “important” weigh more heavily and help to make other document “important”. Those terms in a document with a high \( IV \) should receive more credits.

**Ranking Measure**

After setting the final term weights in the vector matrix, TableRank measures the similarity between the \(<\text{table}, \text{query}>\) pairs by computing the cosine of the angle between these two vectors. Give the table vector \( \vec{tb}_j \) and the query vector \( \vec{q} \), the similarity measure is computed as:

\[
sim(\vec{tb}_j, \vec{q}) = \frac{|\vec{tb}_j| \cdot |\vec{q}|}{|\vec{tb}_j| \times |\vec{q}|} = \cos(\vec{tb}_j, \vec{q}) = \frac{\sum_{i=1}^{n} w_{i,j} \cdot w_{i,q,k}}{|\vec{tb}_j| \times |\vec{q}|}
\]

(10)

**Experimental Results**

**Parameter Settings**

Before demonstrating the experimental results, we discuss the parameter settings in TableRank algorithm.

**Parameter Setting of the Metadata Weight**

Intuitively, the weight of a metadata \( MW_k \) is proportional to the occurrence frequency of the meaningful words in the metadata. Meaningful words refer to those representative terms, which reflect the end users’ query interests. We determine each metadata’s weight \( MW \) based on a study of the keyword distribution over different metadata. First, we randomly select a set of tables from different research fields, e.g., chemistry, computer science, etc. For each table, we delete the stop list words and create a Term Dictionary that list all the terms in the table in a descendant order according to term-occurrence frequency. The top \( \varphi \) meaningful terms from the ordered Term Dictionary are manually collected to construct a Popular Term List. Although different tables have different Popular Term Lists, we believe that they have maximum likelihood for use to construct queries. We tried 10 different sample table sets, the term occurrence distributions over the metadata are similar to each other.

In our experiment, to determine the metadata weights, 100 tables are randomly selected as the sample set. Half of them come from the field of chemistry and the other half come from the field of computer science. For each table, \( \varphi \) is set to 10 and the term-occurrence frequency of the top 10 keywords within each metadata is calculated. The empirical results of the term distribution over each table metadata is shown in Figure 4. The X-axis lists seven main text-based table metadata, ordered by the ratio that the collected keywords occur. The Y-axis shows the percentage of the term occurrence frequency of each metadata over all the metadata. From Figure 4, we can see that the metadata “Document Title”, “Table Column Heading”, and “Table Caption” are the top three metadata and should get the highest metadata weight. We keep the metadata with the (“Table Footnote” and “Author” in Figure 4) intact by assigning their \( MW \) as 1. All the other \( MW \)’s are calculated according to their ratio comparing with lowest percentage.

**Parameters for Document Origination**

In each research field, scholarly journals or conferences are scored and ranked based on the opinions of field experts. CiteSeer gives an estimation of the impact ranking for the computer science publications and Wikipedia\(^2\) estimates for chemistry papers. A comprehensive journal impact factor list spanning the years 2002-2004 for all the fields can be found in CNCSIS\(^3\). Different schemes have different score ranges from \([0, 3.31]\) to \([0, 53.055]\). We normalized the different ranges into the same range \([1, 10]\). For those journals or conferences that appear in both Wikipedia and CNCSIS, we normalize them respectively, then take the average value as the final score. Table 2 lists examples of the document origins with the corresponding importance scores.

**Parameter Setting of Document Freshness**

Document freshness \( (DF) \) refers to the age of a 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

\(^2\)http://en.wikipedia.org/wiki/List_of_scientific_journals_in_chemistry/

Table 2: An Example of The Importance of Conference/Journal

<table>
<thead>
<tr>
<th>Conference/Journal</th>
<th>Field</th>
<th>Importance Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>CS</td>
<td>7.9</td>
</tr>
<tr>
<td>Analytical Chem</td>
<td>Chemistry</td>
<td>7.5</td>
</tr>
<tr>
<td>Nature</td>
<td>Biology</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Table Ranking Results

In this work, we focus on tables in scientific documents in (PDF) format. The document collection comes from three sources: 1) scientific digital libraries (Royal Chemistry Society), 2) the web pages of research scientists in chemistry departments in universities, e.g., UCLA\(^4\) 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 crawler uses a depth-first crawling policy with a maximum depth of five (from the seed URLs). 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. A random selection of 100 documents indicates that 78% of them has at least one table and most of them have more than one.

The commonly used comparison methods Kendall or Spearman distance (Fagin, Kumar, & Sivakumar 2003) are not sufficient for determining which ranking scheme is better. 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 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.

Comparative Retrieval Effectiveness of TableRank

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

\(^1\)http://www.informatik.uni-trier.de/ley/db/

\(^2\)http://www.chem.ucla.edu/VL/Academic.html

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 are relevant among the returned hits, and in what orders.

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 = \(\frac{[H_R \cap T_R]}{H_R}\). 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.

Among the results returned by current search engines, a large part does not have table or match the query because they can not identify the tables in the documents. We call these results as false tables, filter out them and only compare the ranking on the true tables. We randomly select 100 terms from the Popular Term List, such as “gene”, “protein”, “query,”, and then use each term as a query in TableSeer. In addition, we also select 20 terms beyond the Popular Term List because users may pose free queries with any possible terms, such as “result”, “document”, etc. For the other two search engines: Google Scholar and CiteSeer, the query is set as the term together with an additional keyword “Table” to indicate the search engines to search for tables. An example is applying “Alkaline” to TableSeer and applying the combination of the term “alkaline” and “Table” to Google Scholar. Because it is difficult 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 \(n\) results returned from the search engines. Thus, we manually examine the first 20 results returned by both popular search engines. As shown in Table 3, we can see that TableSeer outperforms the other two search engines.

The remaining question is which search engine has the best ranking scheme. We adopted 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 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 CiteSeer. In the second custom search engine method, we

Table 3: The average precision of TableSeer compared with other popular search engines

<table>
<thead>
<tr>
<th>Search engines</th>
<th>Google Scholar</th>
<th>CiteSeer</th>
<th>TableSeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Precision</td>
<td>0.734</td>
<td>0.767</td>
<td>1.00</td>
</tr>
</tbody>
</table>

\[^3\]http://www.chem.ucla.edu/VL/Academic.html
set up a common test bed for TableRank and Google using the Google Custom Search engine API to build a customized search engine. We set the seed URLs (e.g., a chemistry journal website) 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 tried 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 4 displays the average pairwise accuracy results made by six users.

**Factor Influence in TableRank** The TableRank algorithm decides the relevance score for each table 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, we implement TableRank algorithm 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 5, 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 with the “golden standard”. On the second and the third runs, we updated the TFIDF to the TTFITTF weighting scheme with/without the metadata weighting scheme. Next, the TLB factors are tried, and in the last run, the DLB factors are tested.

**Table 4:** The Basic Ranking Results on the Manually Created Document Sets

<table>
<thead>
<tr>
<th>Ranking</th>
<th>The Method to set-up the test-bed</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Custom search engine</td>
<td>51.8</td>
</tr>
<tr>
<td>Google Scholar</td>
<td>bottom-up method</td>
<td>52.72</td>
</tr>
<tr>
<td>CiteSeer</td>
<td>bottom-up method</td>
<td>55.35</td>
</tr>
<tr>
<td>TableSeer</td>
<td>Both methods</td>
<td>69.61</td>
</tr>
</tbody>
</table>

**Table 5:** Results for Individual Impact Factor in TableRank Algorithm

<table>
<thead>
<tr>
<th>Impact Factors</th>
<th>TFIDF</th>
<th>TTFITTF with MW</th>
<th>TTFITTF without MW</th>
<th>TLB</th>
<th>DLB</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>50.19</td>
<td>61.46</td>
<td>63.35</td>
<td>29.60</td>
<td>40.33</td>
<td>69.61</td>
</tr>
</tbody>
</table>

**Conclusions and Future Work**

In this paper, we present the TableSeer system that arms with a novel table ranking algorithm, TableRank, to retrieve the tables contained in Web and digital libraries. We use a modified vector space model to rate the \(<query, table>\) pairs by substituting the document vectors with the table vectors. TableRank applies a new term weighting scheme TTF-ITTF, which calculate the weight of a term in the vector space based on multiple factors. In addition to TTF-ITTF, TableRank also have boosting factors from two levels: the table level (TLB) and the document level (DLB). Our experimental results on common test-beds demonstrate that TableRank outperforms several popular web search engines for table retrieval. There are several areas in which we still hope to make progress. First, currently we focus on the scientific documents in PDF format. Next, we will extend to handle other kinds of documents in the Semantic Web. Second, although we present preliminary results showing the effect of the impact factors proposed, many parameter settings are based on empirical studies. In the future, more extensive experiments are needed to determine more suitable parameter settings.

**Acknowledgments**

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

**References**


