Table Header Detection and Classification

Jing Fang¹,², Prasenjit Mitra², Zhi Tang¹, C. Lee Giles²

¹Institute of Computer Science & Technology, Peking University, Beijing, China
Email:{fangjing, tangzhi}@pku.edu.cn
²Coll. of Infor. Scien. & Techn., Depar. of Comp. Scien. & Engin., The Pennsylvania State University, University Park, PA, U.S.A. 16803.
Email:{pmitra, giles}@ist.psu.edu

Abstract
In digital libraries, a table, as a specific document component as well as a condensed way to present structured and relational data, contains rich information and often the only source of that information. In order to explore, retrieve, and reuse that data, tables should be identified and the data extracted. Table recognition is an old field of research. However, due to the diversity of table styles, the results are still far from satisfactory, and not a single algorithm performs well on all different types of tables. In this paper, we randomly take samples from the CiteSeerX to investigate diverse table styles for automatic table extraction. We find that table headers are one of the main characteristics of complex table styles. We identify a set of features that can be used to segregate headers from tabular data and build a classifier to detect table headers. Our empirical evaluation on PDF documents shows that using a Random Forest classifier achieves an accuracy of 92%.

Introduction
Digital libraries usually contain a large collection of digital documents, many of which contain tables. Tables, as significant document components, store and present relational data in a condensed way, i.e. experimental results in scientific documents, statistical data in financial reports, etc. In short, tables contain rich sources of information that can be very useful and are only available in the table. Automatic table extraction is of great importance to exploring, retrieval and making full use of this data.

Table extraction and indexing have been popular but open issues still continue, primarily due to the diversity of table styles. It is not easy for a single algorithm to perform well on all the different types of tables. A table processing survey (Lopresti & Nagy 1999) shows 15 examples for tables and demonstrates how much tables may be different from each other in actual documents. Table headers may be complex. For example, Figure 1 is a table with both row and column headers; the header has multiple text lines and multiple levels. As such, we randomly collect samples from CiteSeerX – a public scientific search engine and digital library, to investigate table categories and table header types. Note that, we focus on PDF documents, which are widely used in today’s digital libraries.

![Figure 1. An example of table with complex header](image)

We first delineate what kinds of tables that exist in actual documents, and what are their structures, header types, etc; and, importantly, what kinds of tables can and cannot be easily recognized. Table headers may be complex. For example, Figure 1 is a table with both row and column headers; the header has multiple text lines and multiple levels. As such, we randomly collect samples from CiteSeerX – a public scientific search engine and digital library, to investigate table categories and table header types. Note that, we focus on PDF documents, which are widely used in today’s digital libraries.
We then propose and evaluate algorithms that automatically detect table row headers and column headers. First, we apply a forwards weighted average score strategy to calculate similarity between consecutive table rows/columns and then find the first local minimum top-down/left-right to be the separation between header and data. Second, a similar backwards strategy is applied from the last row/column to the first to find separations. Additionally, we treat the problem as header and data binary classification problem, and apply three classifiers—support vector machine (SVM), logistic regression, and random forests. In the experiments, we elaborate on feature selection, analyze parameter impact, and compare the performance of these three models, as well as with the rule-based method.

This research is based on an existing table extraction tool, which is part of the search engine system TableSeer (Liu et al. 2007). It automatically identifies tables in PDF digital documents, detects table boundaries (Liu, Mitra, & Giles 2008) and extracts the contents in the table cells (Liu et al. 2006). The contents are then stored in a queryable table in a database. It also indexes the tables and provides a novel ranking function to enable end-user table search. However, existing work on extracting table structure stops after finding cells, segmenting rows and columns. This work extends previous work and in many ways explores automated methods for topological discovery.

Related Work

Previous surveys (Zanibbi, Blostein, & Cordy 2004) and (Silva, Jorge, & Torgo 2006) summarized approaches for table recognition, which can be divided into physical and logical structure analysis. The former refers to segmenting table cells, rows, and columns, while the latter aims at analysis of index relations (Hirayama 1995) - (Seth et al 2010).

For logical structure analysis, (Nagy et al 2010) presented a grammatical framework for parsing a linear string representation of column headers of tables in a range of specified formats for web pages. They focused on grammar-based header hierarchy extraction using already known column headers, but didn’t mention how the headers were detected. Wong (Wong, Martinez, & Cavedon 2009) classified tabular information for the task of named entity detection for genetic mutations. HTML documents and tables were broken up into row headers, column headers and data cells by making use of the hr tags. Then, a machine learning method and simple bag-of-word features were used to extract mutation classes. They focused on the extraction task rather than the actual detection of row/column headers. Moreover, the processed dataset is also an HTML format instead of a PDF format, which is harder since there is no tag information.

There are numerous methods proposed for table detection and table structure extraction. However, very few have addressed table classification or related topics. (Wang & Hu 2002) classified web tables into genuine tables and non-genuine tables. The former refers to real tables, while the latter means those using table tags only to organize and arrange content. This kind of classification is special for web tables, where <Table> </Table> tags do not necessarily mean that there is a table. Kim and Liu proposed a function-based table categories detection method (Kim & Liu 2011). They classified scientific document tables into three topical categories: background, system/method, and experiments, and two functional categories: commentary and comparison. This function-based table classification is beneficial to table interpretation. But it is not designed for improving table recognition accuracy.

More importantly, to the best of our knowledge, our work is the first to detect how frequent different styles of headers are used in computer and information science academic documents, which is a topological investigation.

The main contributions of our work are: i) we investigate document samples in a digital library to identify the frequency of different styles of headers and categorize them; ii) we design table header detection methods and compare their performances empirically to demonstrate their efficacy.

Dataset

We randomly collect two dataset samples from CiteSeer\(^*\) document repository; both contain 200 PDF scientific documents. TableSeer (Liu et al. 2007) is used to extract tables from these files. The distribution of the number of tables in each file (shown in Figure 2) is similar across both samples. Overall, 76 documents in dataset sample1 and 87 documents in sample2 contain tables. The total numbers of detected tables are 151 and 130 respectively. Through manual judgment, we find that some document components such as algorithms, code and, figures are mistakenly detected as tables; some tables with image cells are detected as empty, and some do not contain any text content due to the errors made by the PDFBOX\(^1\) parse engine used in TableSeer. These cases are treated as invalid and are not counted when doing header detection evaluation. Finally, we get 135 valid tables from the first sample, and 120 from the second.

\(\text{Figure 2. Table distribution in two dataset samples}
\)

\(^1\) http://pdfbox.apache.org/

\[\text{Number of file}\]

\[\text{Number of tables in a file}\]
We look at table header types and layout complexity. During this process, only valid tables are counted. If only a row header or a column header exists, the table is called a one-dimensional table (1-D); if both headers exist, a two-dimensional table (2-D). Some tables do not contain either kind of header. Multi-dimensional tables (M-D) contain multi-dimensional data flattened into a two-dimensional form, e.g., in Figure 3 the table contains data related to: i) different states, ii) different sexes, iii) different age categories, and iv) different years. The columns represent different years, but the other three dimensions are flattened out into the rows.

![Figure 3. An example of multi-dimensional table](image)

Figure 3. An example of multi-dimensional table

Figure 4 shows the proportion of different header types in our samples.

![Figure 4. Proportion of different header types](image)

Figure 4. Proportion of different header types

We also classify tables into complex and simple types based on our observations of table layout characteristics. The complexity is caused by the following: multi-line cells, multi-level headers, multi-dimension, long and folded table, and other irregular cases. Some examples are shown in Figure 5. Tables without such complexity will be referred to as simple tables. The sub classes of complex tables are not mutually exclusive. For instance, Figure 5 (b) shows a long, folded table with a multi-level header.

![Figure 5. Examples of complex tables](image)

(c) Irregular layout

Table 1 presents the distributions of complex and simple tables.

<table>
<thead>
<tr>
<th>Layout complexity</th>
<th>Sample1</th>
<th>Sample2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex</td>
<td>81 (60%)</td>
<td>76 (63.3%)</td>
</tr>
<tr>
<td>Multi line cell</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>Multi level header</td>
<td>36</td>
<td>33</td>
</tr>
<tr>
<td>Multi dimensional</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Long and folded</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Other irregular layout</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Simple</td>
<td>54 (40%)</td>
<td>44 (36.7%)</td>
</tr>
</tbody>
</table>

The data presented in this section demonstrates that, sample1 and sample2 are similar in document attributes and different kinds of fractions. This suggests that these two samples are stable and representative of the digital library. Otherwise, more samples should be collected to make sure the samples have no bias.

### Heuristic Methods

In this section, we propose two heuristic strategies to detect the separation between the header and data part of a table. We use the first row and first column as default headers to be the baseline method, and compare their performance.

#### Local Minimum Methods

Table headers are often set in bold or italic fonts or have a larger font size. Ruling lines and special background color may also be applied to separate table headers from the data. In this paper, only text and their coordinate information are utilized; graphic lines and colors require image processing techniques and are hence not considered.

Data rows normally share similar data type, cell alignment, character overlap, number of cells, etc. For a multi-line header, header rows also share similar font style and data type. A header row and a data row usually differ with respect to these features. We apply a weighted average score to calculate the similarities between each pair of consecutive rows. Considering that headers usually exist in the beginning one or a few rows by default and some irregular data rows layout may also cause low similarity, we choose the first local minimum, i.e. the first valley to be the separation. Alternatively, we use a backward local minimum strategy that chooses the first local minimum while making an upward pass from the end of the table.

First, we define two cells from consecutive rows as corresponding cells if their bounding boxes overlap.
horizontally. The top row cell is denoted as \( C_T \), and the cell down in the next row is \( C_D \). The following attribute scores are calculated between two corresponding cells:

- Font size score: \( S_1 = \text{minimum font size of } C_T \text{ and } C_D \text{ divided by the maximum font size of this cell pair}. \)
- Character number score: \( S_2 = \min(\text{Num chars of } C_T, C_D) / \text{maximum number of characters of this cell pair}. \)
- Overlap score: \( S_3 = \text{the width of bounding box overlap (as the arrow range shown below) divided by their minimum cell bounding box width}. \)
- Data type score: if \( C_T \text{ and } C_D \text{ have the same data type (numeric or alphabetic), set } S_4 \text{ to } 1, \text{ otherwise to } 0. \)
- Alignment score: if the two cells are found to be aligned (left, right, or center), \( S_5 \) is set to be 1, otherwise 0.

Each pair of corresponding cells gets a weighted score and the rows’ CellScore is the average of the scores of all its cells as follows:

\[
\text{CellScore} = \frac{1}{n} \sum_{i=1}^{n} \left( u \cdot S_1 + v \cdot S_2 + w \cdot S_3 + x \cdot S_4 + y \cdot S_5 \right) 
\]

where \( S_1-5 \) respectively represent the scores obtained from the above attributes, and \( u, v, w, x \text{ and } y \) are their corresponding weights.

We calculate similarity on the row level as follows:

- RowScore: the minimum number of cells divided by the maximum number of cells among the two rows.

The final score is computed as follows:

\[
\text{FinalScore} = \frac{\alpha \cdot \text{CellScore} + \beta \cdot \text{RowScore}}{\alpha + \beta} 
\]

Similarly, the same mechanism is applied to table columns to find the local minimum as the separation between header columns and data columns. Since columns do not possess the repetition characteristic as rows, only data type, font style, font size, and number of cells are explored.

Results and Analysis

The heuristic methods are executed on dataset sample1 and sample2 respectively. The results are shown in Figure 6. Errors can be classified into three types: false positive (the detected header is not a header), partial (for multi-line headers, only part of them have been detected), and expanded (some data are mistakenly detected as header). We evaluated table boundary detection manually to avoid table boundary detection errors. The data presented in Figure 6 are all based on correctly detected tables or partially detected tables but with complete headers.

In order to improve the accuracy of the header and data separation task, we cannot rely on a simple single heuristic as stated above. We propose supervised learning algorithms in the next section for classifying table content into header and data classes.

<table>
<thead>
<tr>
<th>three-year-olds</th>
<th>26.2%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>four-year-olds</td>
<td>56.3%</td>
<td>55.2%</td>
</tr>
</tbody>
</table>

Machine Learning Techniques

In this section, we first define the feature set for classifying header and data row/column, and then use a SVM based classifier and an ensemble classifier, a Random Forest.

Feature Sets

First, we list the features used for table row classification. We classify our features into two categories: single row features and neighboring row features.

Single row features

- Number of cells. Typically, data rows have the same number of cells, while header rows often have missing cells, especially for multi-level or hierarchical headers, where one cell in the first row expands into multiple cells in the next.
- Average cell length. For numeric tables, data rows usually have shorter average cell length than header rows. Therefore the cell length could be applied to differentiate the header and data.
- Number of characters.
- Percentage of numeric characters.
- Percentage of alphabetical characters.
- Percentage of symbolic characters (characters other than "A" to "Z", "a" to "z", and "0" to "9").
- Percentage of numeric cells. We define a cell containing only "0" to "9", ",", and "]" as a numeric cell.
- Average font size. Typically, the header row has a slightly larger font size.
- Percentage of bold cells.
- Percentage of italic cells. Bold and italic font styles are often used to highlight table headers.
- Row number. Headers appear before data.
Neighboring row features

- Percentage of spanning cells. We define a cell spanning over multiple cells in the lower row as a spanning cell.
- Number of cells difference = abs (number of cells upper – number of cells lower) / (number of cells upper + number of cells lower). Header rows often have missing cells, similar to the “number of cells” for single rows.
- Average alignment. (left 1, right 2, center 3, other 4). This is a category feature instead of value feature.
- Average overlap proportion. The average overlaps proportion of corresponding cells (calculated the same way as the overlap score described in section 4) between two rows. Data-data rows often have larger value than header-data rows.
- Percentage of same cell data type. Here data type refers to alphabetical, digit and symbol.
- Percentage of same cell font style. Here font style refers to bold and italic. It calculates the proportion of the same font style between the corresponding cells.
- Overall content repetition of corresponding cells.
- Average content repetition of corresponding cells. As a pre-processing, we replace all number characters with “#”, so that numeric strings with the same length are treated as with equal content. Then the Levenshtein distance is used to calculate similarity of two cell strings.

The above features are selected for header and data row classification. In terms of header and data column, some characteristics change. For instance, the repetition or consistency property does not apply for neighboring characteristics change. For instance, the repetition or classification. In terms of header and data column, some

Classification Models

In our paper, we use i) an SVM (Burges 1998) based classifier with the popular RBF kernel, and the popular “grid-search” and cross validation to find the optimal soft margin parameter C, ii) a logistic regression (Balakrishnan 1991) based classifier, and iii) a random forest based classifier (Breiman 2011) to separate the header from the data.

Experimental Results and Analysis

Data Set

The two dataset samples described in section 3 were utilized. We show the number of tables (with correct labels for header and data), number of rows and columns, and number of header rows and header columns for the dataset sample1 (D1) and sample2 (D2) in Table 2.

Impact of Learning Models

We use Libsvm\(^2\) for SVM, a popular library for support vector machines. For logistic regression, we use the statistical computing tool R\(^3\), with its generalized linear regression function. For random forest, we use the Weka (Witten and Frank, 2005) toolkit, which contains a wide selection of in-built algorithms. For this experiment, it is built with 100 trees, and m = int (log₂ (20 + 1)) = 4 as suggested by Breiman.

We use a 10-fold cross-validation method to evaluate our algorithms. Since both D1 and D2 are random samples selected from the same set, and share similar number of tables, rows/columns and headers, we simply combine them together to do classification. The experimental results of table row and column type classification are listed in Table 3. The evaluation metrics are precision, recall, and F-measure. Given the number of the correctly-labeled true table rows A, the number of true positive header rows B, and the number of true negative data rows C, the precision is \( \frac{A}{A+C} \), the recall is \( \frac{A}{A+B} \) and the F-measure is \( \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \).

The metrics also apply to table column type classification evaluation.

### Table 2. Machine learning training set

<table>
<thead>
<tr>
<th>Type</th>
<th>Learning model</th>
<th>Precision</th>
<th>Recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>SVM</td>
<td>0.921</td>
<td>0.918</td>
<td>0.919</td>
</tr>
<tr>
<td></td>
<td>Logistic regression</td>
<td>0.843</td>
<td>0.90</td>
<td>0.871</td>
</tr>
<tr>
<td></td>
<td>Random forest</td>
<td>0.974</td>
<td>0.978</td>
<td>0.976</td>
</tr>
<tr>
<td>Column</td>
<td>SVM</td>
<td>0.861</td>
<td>0.86</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Logistic regression</td>
<td>0.968</td>
<td>0.967</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td>Random forest</td>
<td>0.982</td>
<td>0.982</td>
<td>0.982</td>
</tr>
</tbody>
</table>

Within the experiments with the same method, the same training dataset, and the same features, random forest outperforms the other two classifiers. This is probably because given an unrefined feature set, random forest is able to automatically choose the most useful features. Besides, random forest has bagging mechanism to select training samples, which could reduce variance and avoid overfitting. The SVM may probably be affected by the unbalanced number of header and data cases. The low performance of logistic regression may be caused by relative limited size of datasets and also the non-selected feature set. Note that, we have also tried other classifiers

\(^2\)http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

\(^3\)http://www.r-project.org/
integrated in Weka, such as NaiveBayes, BayesNet, J48 (C4.5 decision tree algorithm), AdaBoost, etc. Overall, Random Forest still performs the best.

**Impact of Feature Set**
We use the best performing random forest classifier to evaluate the impact of different feature sets. For table header and data row classification, we design three sets of experiments: i) random forest model using only single row feature; ii) experiment using only neighboring feature; iii) add all features together. The results are shown in Table 4.

Table 4. Results of header and data row classification on different feature set, but all training set

<table>
<thead>
<tr>
<th>Features</th>
<th>Precision</th>
<th>Recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single row feature</td>
<td>0.961</td>
<td>0.962</td>
<td>0.961</td>
</tr>
<tr>
<td>Neighbor row feature</td>
<td>0.961</td>
<td>0.962</td>
<td>0.961</td>
</tr>
<tr>
<td>Together</td>
<td>0.974</td>
<td>0.978</td>
<td>0.976</td>
</tr>
</tbody>
</table>

Then, we use the ‘InfoGain’ attribute evaluator of the Weka toolkit to choose the most important features. The results showed that Number of characters, Percentage of alphabetical characters, Average font size, Average alignment, Number of cells difference, Percentage of consistent cell data type, and Percentage of consistent cell font style are the most effective features.

**Impact of Parameters**
The random forest classifier has two effective parameters: the number of trees to grow, and the number of features to consider when splitting each node. We have set the second value to be \( \log_2(M +1) \). Figure 7 shows the increase in accuracy with an increase in the number of trees. Increasing the number of trees beyond 100 improves the result very slightly at the cost of a huge increase in run-time.

Table 5. Comparison of learning method, rule-based method and baseline on table header row and column detection

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Heuristic</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table header row detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>0.402</td>
<td>0.609</td>
<td>0.920</td>
</tr>
<tr>
<td>Partial</td>
<td>0.587</td>
<td>0.25</td>
<td>0.043</td>
</tr>
<tr>
<td>Expanded</td>
<td>0</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>Fake</td>
<td>0.011</td>
<td>0.011</td>
<td>0.007</td>
</tr>
<tr>
<td>Table header column detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct</td>
<td>0.735</td>
<td>0.598</td>
<td>0.904</td>
</tr>
<tr>
<td>Partial</td>
<td>0.24</td>
<td>0.186</td>
<td>0.053</td>
</tr>
<tr>
<td>Expanded</td>
<td>0</td>
<td>0.181</td>
<td>0.025</td>
</tr>
<tr>
<td>Fake</td>
<td>0.025</td>
<td>0.025</td>
<td>0.018</td>
</tr>
</tbody>
</table>

**Conclusion**
We investigate the classification of diverse table styles for effective table header information extraction using random samples from CiteSeerX. From the analysis of table categories and their relationship with table header types, we propose heuristic methods and machine learning techniques to address the table header detection problem in order to better classify table categories. Empirically, a Random Forest classifier outperforms all the other methods.

Future work could focus on header hierarchy extraction in order to distinguish single-line single-level, multi-line single-level and multi-line multi-level table headers. Furthermore, enhanced table header detection could not only improve table classification and understanding, but can also improve table search. For instance, given a table, we could extract the headers and use them as a query instead of the entire PDF table to search tables with the same or semantically related headers.

**Acknowledgement**
We gratefully acknowledge the support of the co-supervised Ph.D. student scholarship program of the China Scholarship Council (CSC), the National Basic Research Program of China (No. 2012CB724108) and the National Science Foundation under Grant Nos. 0845487.
References


Liu, Y.; Bai, K.; Mitra, P.; and Giles, C.L. 2007. TableSeer: Automatic Table Metadata Extraction and Searching in Digital Libraries Categories and Subject Descriptors. In JCDL, 91 100.


Liu, Y.; Mitra, P.; Giles, C.L.; and Bai, K. 2006. Automatic Extraction of Table Metadata from PDF Documents. In JCDL, 11 15.


Silva, A.C.e.; Jorge, A.M.; and Torgo, L., 2006. Design of an end to end method to extract information from tables. In IJDAR, 144 171.


Oro, E.; and Ruffolo, M. 2009. PDF TREP an Approach for Recognizing and Extracting Tables from PDF Documents. In ICDAR, 906 910.


Kim, S.; and Liu, Y. 2011. Functional Based Table Category Identification in Digital Library. In ICDAR, 1364 1368.

