Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Table Header Detection and Classification

Jing Fang[1,2], Prasenjit Mitra[2], Zhi Tang[1], C. Lee Giles[2]

[1]Institute of Computer Science & Technology, Peking University, Beijing, China

Email:{fangjing, tangzhi}@pku.edu.cn

[2] Coll. 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 CiteSeer^X 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%.

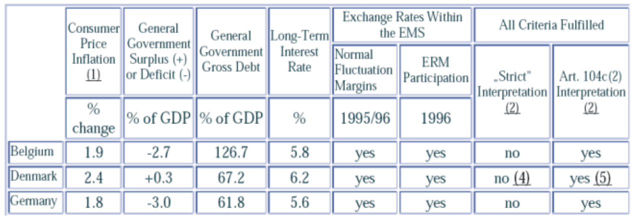
Introduction

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. The document medium can be plain text, image, handwritten, or web pages; from the functional or context aspect, financial, schedule, vote tables, etc. can be found. While the majority of existing methods explore table layout characteristics for table recognition, what impacts the extraction performance most is the diversity of table structures. Just like the previously mentioned table examples, there are tables with and without headers, nested tables (whose certain cells are small tables themselves), and even figure-like tables.

A reasonable assumption is that if tables could be automatically classified into several categories according to their structure, then targeted algorithms should work. We observed that table headers are one of the key factors that determines the structure of tables and determines the complexity in tables. We define the lines at the top of a table (header rows) or at the left of the table (header columns) as the table headers. Table header detection is also important for other applications. For example, in the domain of environmental chemistry, previous surveys published ground water levels at a location inside tables. Current surveyors want to extract that data from old documents and compare with current findings. Identifying the header accurately allows the end-user to query a database containing that data.

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 CiteSeer^X – 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.



	Consumer Price Inflation (1)	General Government Surplus (+) or Deficit (-)	General Government Gross Debt	Long-Term Interest Rate	Exchange Rates Within the EMS		All Criteria Fulfilled	
					Normal Fluctuation Margins	ERM Participation	„Strict" Interpretation (2)	Art. 104c(2) Interpretation (2)
	% change	% of GDP	% of GDP	%	1995/96	1996		
Belgium	1.9	-2.7	126.7	5.8	yes	yes	no	yes
Denmark	2.4	+0.3	67.2	6.2	yes	yes	no (4)	yes (5)
Germany	1.8	-3.0	61.8	5.6	yes	yes	no	yes

Figure 1. An example of table with complex header

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

physical vs logical

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 finding out about table headers, extraction of header hierarchy, and 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^X 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.

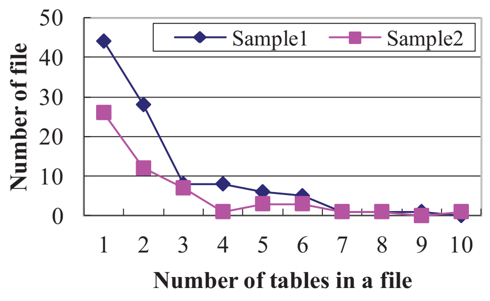
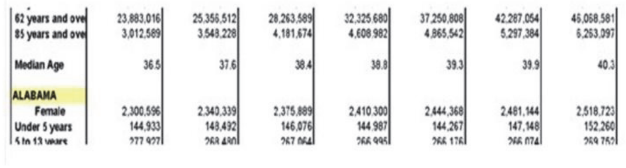


Figure 2. Table distribution in two dataset samples

[1] http://pdfbox.apache.org/

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.



62 years and ove	23,883,016	25,356,512	28,263,589	32,325,680	37,250,808	42,287,054	46,068,581
85 years and ove	3,012,589	3,543,228	4,181,674	4,608,982	4,865,542	5,297,384	6,263,097
Median Age	36.5	37.6	38.4	38.8	39.3	39.9	40.3
ALABAMA							
Female	2,300,596	2,340,339	2,375,889	2,410,300	2,444,368	2,481,144	2,518,723
Under 5 years	144,933	148,492	146,076	144,987	144,267	147,148	152,260
5 to 13 years	277,927	268,480	267,064	266,995	266,176	266,074	259,752

Figure 3. An example of multi-dimensional table

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

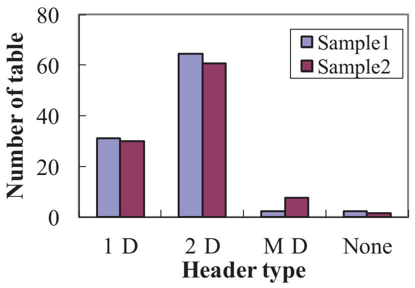
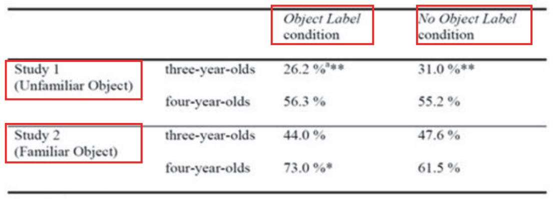


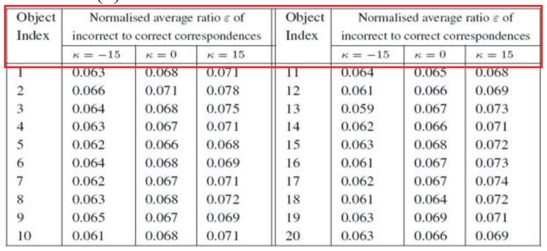
Figure 4. Proportion of different header types

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.



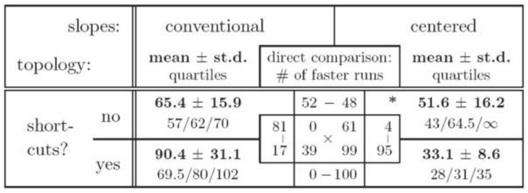
		Object Label condition	*No Object Label* condition
Study 1 (Unfamiliar Object)	three-year-olds	26.2 %ª**	31.0 %**
	four-year-olds	56.3 %	55.2 %
Study 2 (Familiar Object)	three-year-olds	44.0 %	47.6 %
	four-year-olds	73.0 %*	61.5 %

(a) Table with multi line cells



Object Index	Normalised average ratio ε of incorrect to correct correspondences			Object Index	Normalised average ratio ε of incorrect to correct correspondences		
	$\kappa = -15$	$\kappa = 0$	$\kappa = 15$		$\kappa = -15$	$\kappa = 0$	$\kappa = 15$
1	0.063	0.068	0.071	11	0.064	0.065	0.068
2	0.066	0.071	0.078	12	0.061	0.066	0.069
3	0.064	0.068	0.075	13	0.059	0.067	0.073
4	0.063	0.067	0.071	14	0.062	0.066	0.071
5	0.062	0.066	0.068	15	0.063	0.068	0.072
6	0.064	0.068	0.069	16	0.061	0.067	0.073
7	0.062	0.067	0.071	17	0.062	0.067	0.074
8	0.063	0.068	0.072	18	0.061	0.064	0.072
9	0.065	0.067	0.069	19	0.063	0.069	0.071
10	0.061	0.068	0.071	20	0.063	0.066	0.069

(b) Long and folded table with a multi level header



slopes:		conventional				centered
topology:		**mean ± st.d.** quartiles	direct comparison: # of faster runs			**mean ± st.d.** quartiles
short-cuts?	no	**65.4 ± 15.9** 57/62/70		52 − 48	*	**51.6 ± 16.2** 43/64.5/∞
			81 – 17	0 × 61 / 39 × 99	4 – 95	
	yes	**90.4 ± 31.1** 69.5/80/102		0 − 100		**33.1 ± 8.6** 28/31/35

(c) Irregular layout table

Figure 5. Examples of complex tables

Table 1 presents the distributions of complex and simple tables.

Table 1. Table layout complexity

Layout complexity	**Sample1**	**Sample2**
Complex	81 (60%)	76 (63.3%)
Multi line cell	42	41
Multi level header	36	33
Multi dimensional	3	9
Long and folded	2	4
Other irregular layout	11	6
Simple	54 (40%)	44 (36.7%)

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.

super simple baseline

Local Minimum Methods

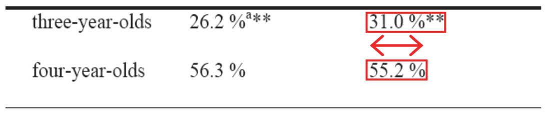
Table headers are often set in bold or italic fonts or have a larger font size. Ruling lines and special background color may also be applied to separate table headers from the data. In this paper, only text and their coordinate information are utilized; graphic lines and colors require image processing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

Potential Features
Font form, color, background
graphic lines

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: S1 = minimum font size of C_T and C_D, divided by the maximum font size of this cell pair.
- Character number score: S2 = min(Num chars of C_T, C_D) / maximum number of characters of this cell pair.
- Overlap score: S3 = the width of bounding box overlap (as the arrow range shown below) divided by their minimum cell bounding box width.



three-year-olds	26.2 %[a]**	31.0 %**
four-year-olds	56.3 %	55.2 %

- Data type score: if C_T and C_D have the same data type (numeric or alphabetic), set S4 to 1, otherwise to 0.
- Alignment score: if the two cells are found to be aligned (left, right, or center), S5 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:

$$CellScore = \frac{1}{n}\sum_{i=1}^{n} \frac{u*S1_i+v*S2_i+w*S3_i+x*S4_i+y*S5_i}{u+v+w+x+y}$$

where S1-S5 respectively represent the scores obtained from the above attributes, and u, v, w, x 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:

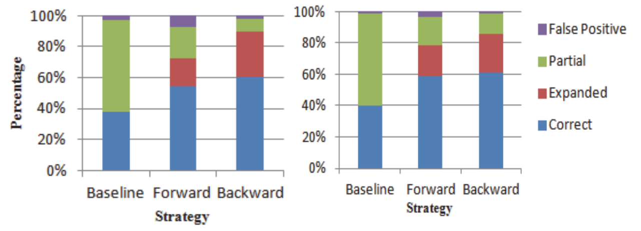
$$FinalScore = \frac{\alpha*CellScore+\beta*RowScore}{\alpha+\beta}$$

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.

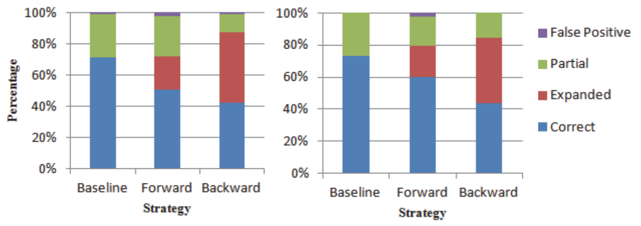
Results and Analysis

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.



(a) Row header detection results on both samples



(a) Column header detection results on both samples

Figure 6. Heuristic header detection methods evaluation

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 "%" 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 a *spanning cell*.

✓ • Number of cells difference = fabs (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 section4) 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 columns. Hence, features are adjusted as follows: *Number of cells, Number of characters, Percentage of digital characters, Percentage of alphabetical characters, Percentage of symbol characters*, *Percentage of numeric cells, Average font size, Percentage of bold cells, Percentage of italic cells, Column number.*

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.

Table 2. Machine learning training set

	Numbers	D1	D2
Row	Table	94	81
	Row	1382	1108
	Header row	240	195
Column	Table	99	90
	Column	758	704
	Header column	139	126

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 $m = \text{int}(\log_2(20 + 1)) = 4$ as suggested by Breiman.

description of models

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 $\frac{2*\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$. The metrics also apply to table column type classification evaluation.

Table 3. Results of header and data row / column classification on all features and all training set.

Type	Learning model	Precision	Recall	F
Row	SVM	0.921	0.918	0.919
	Logistic regression	0.843	0.90	0.871
	Random forest	0.974	0.978	0.976
Column	SVM	0.861	0.86	0.84
	Logistic regression	0.968	0.967	0.967
	Random forest	0.982	0.982	0.982

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 NiaveBayes, 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

Features	Precision	Recall	F
Single row feature	0.961	0.962	0.961
Neighbor row feature	0.961	0.962	0.961
Together	0.974	0.978	0.976

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.

most important features

Impact of Parameters

tuning the random forest

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.

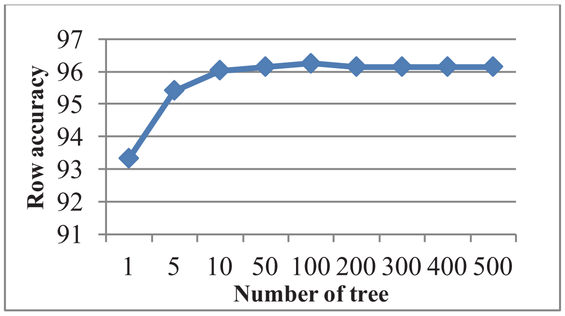


Figure 7. Impact of number of trees for random forest learning

Evaluation on Table Level

Our machine-learning-based methods can classify individual table rows as headers and table columns as headers with respectively 97% and 98% accuracy. Table header detection is not only a classification problem, but also an information extraction problem of determining where the header ends and the data begins. In section 4, we addressed heuristic methods and provided their results at the table level. Here, we evaluated the accuracy of our machine learning algorithms at the table level by comparing the predicted and labeled classes per table, and make a comparison with the proposed heuristic method which has better results as well as with the baseline method.

Since the random forest (RF) has the best results for rows and columns, we use its predicted results for table level accuracy, including the proportions of tables with completely correct header, partially detected header, expanded and false positive header. The results are shown in Table 5. It is interesting to note that, although the column classification accuracy has fewer errors, these erroneous columns are distributed over many tables but the erroneous rows are distributed over fewer tables. Thus, the table-level accuracy is better for the rows-case.

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

	Baseline	Heuristic	RF
Table header row detection			
Correct	0.402	0.609	0.920
Partial	0.587	0.25	0.043
Expanded	0	0.13	0.03
Fake	0.011	0.011	0.007
Table header column detection			
Correct	0.735	0.598	0.904
Partial	0.24	0.186	0.053
Expanded	0	0.181	0.025
Fake	0.025	0.025	0.018

Conclusion

We investigate the classification of diverse table styles for effective table header information extraction using random samples from CiteSeer^X. 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

Lopresti, D.; and Nagy, G. 1999. A Tabular Survey of Automated Table Processing. In *GREC*, 93 120.

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

Liu, Y.; Mitra, P.; and Giles, C.L. 2008. Identifying Table Boundaries in Digital Documents via Sparse Line Detection. In *CIKM*, 1311 1320.

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

Zanibbi,R.; Blostein, D.; and Cordy, J.R.. 2004. A survey of table recognition: Models, observations, transformations, and inferences. In *IJDAR*, 1 16.

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.

Hirayama, Y. 1995. A method for table structure analysis using DP matching. In *DAR*, 583 586.

Kieninger, T.; and Dengel, A. 2001. Applying The T Recs Table Recognition System To The Business Letter Domain. In *ICDAR*, 113 120.

Yildiz,B.; Kaiser,K.; and Miksch, S.. 2005. pdf2table: A Method to Extract Table Information from PDF Files. In *IICAI*, 1773 1785.

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

Hassan, T.; and Baumgartner, R. 2007. Table Recognition and Understanding from PDF Files. In *ICDAR*, 1143 1147.

Hurst, M.; Douglas, S. 1997. Layout and Language: Preliminary Investigations in Recognizing the Structure of Tables. In *ICDAR*, 1043 1047.

Seth, S.; Jandhyala, R.; Krishnamoorthy, M.; and Nagy, G. 2010. Analysis and taxonomy of column header categories for web tables. In *DAS*, 81 88.

Nagy, G.; Padmanabhan, R.; Krishnamoorthy, M.; Jandhyala, R.C.; and Silversmith, W. 2010. Table Metadata Headers, Augmentations and Aggregates. In *DAS,* 507 510.

Nagy, G.; Seth, S.; and Jin, D. 2011. Data Extraction from Web Tables: the Devil is in the Details. In *ICDAR,* 242 246.

Wong, W.; Martinez, D.; and Cavedon, L. 2009. Extraction of named entities from tables in gene mutation literature. In *BioNLP*, 46 54.

Wang, Y.; and Hu, J. 2002. A machine learning based approach for table detection on the web. In *WWW*, 242 250.

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

Burges, CJC. 1998. A tutorial on support vector machines for pattern recognition. In *DMKD*, 121 167.

Balakrishnan, N. 1991. Handbook of the Logistic Distribution. Marcel Dekker, Inc. ISBN 978 0824785871

Breiman, L.. 2011. Random Forests. Machine Learning.

Witten, I.H.; and Frank, E. 2005. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, 2nd edition.