TABLE HEADERS: AN ENTRANCE TO THE DATA MINE

George Nagy
Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY, USA
nagy@ecse.rpi.edu

Sharad Seth
Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE,USA
seth@cse.unl.edu

***Abstract*—Algorithmic methods are demonstrated for information extraction from table header elements, including data categories and data hierarchies. The table headers are found with the Minimum Index Point Search algorithm. The header-path alignment and header completion algorithms yield database-ready table content and configuration statistics on a random sample of 400 diverse tables with ground truth and 1120 tables without ground truth from international statistical data sites.**

Keywords—table headers; category hierarchies; spanning cells

I. INTRODUCTION

Almost any known quantitative information can be found in some table on the web. Web tables are semi-structured data types, designed and formatted for human reading convenience. While the table formats that have evolved for printing and publishing are admirably suited for quick human interpretation of a single table, the cornerstone of Big Data analytics is structured, platform-independent data representation across diverse sources. The success of ontological approaches to decision support is also directly dependent on the extent, accuracy and completeness of the underlying data. Consequently research on the conversion of tables into structured collections of relational tables and Resource Definition Framework (RDF) triples is of growing interest.

Estimates of the number of tables accessible to computer search (mainly spreadsheets, HTML and PDF tables) range into the billions. Only a minority originate in scanned documents, which until recently were the focus of most table research. The rest are either static computer-generated tables or dynamic on-demand tables, derived in either case from a database. In spite of the evident potential value of information extraction from tables, we know of only a few curated, publicly available table corpora. Quantitative analysis of such corpora can help prioritize pertinent research.

The essential structural characteristic of tables is the indexing relationship between row and column headers and data values. We present algorithms for detailed analysis of table headers that (1) augment our previously reported methods of converting ordinary tables to relational tables and RDF triples and (2) automate the derivation of the relevant descriptive statistics and observations. We report experimental results on 200 tables from six international agencies that we collected several years ago (cf. http://www.iapr-tc11.org/mediawiki/index.php/Datasets_List) and on a corpus of 1320 Statistical Abstracts of the United States (SAUS) spreadsheets posted by Professor Michael Cafarella in 2013. Both corpora have been used by other groups. Although research results have been published on much larger datasets, these have not been released for public access.

We consider the CSV (Coma Separated Values) file format as the greatest common denominator of all table formats, including scanned printed tables, word-processing or page layout formats, HTML, proprietary spread sheets and minimally formatted ASCII tables from email and other text transformed into grid tables by earlier methods. Requiring only universally accessible CSV input gives our table processing methods the widest applicability. We therefore convert HTML and spreadsheets culled from the web into CSV grid files.

Although tables are laid out in two spatial dimensions, they can, as first pointed out by X. Wang, represent any number of data categories or logical dimensions. From a database perspective, tables are therefore equivalent to an n-cube. As shown in [1], we can extract such categories and export them to relational databases and RDF triples. The current work extends the analysis to the hierarchy of labels within each category.

Another aspect of our methodology that further differentiates it from earlier work is the retention of the original cell coordinates through the entire analysis. This not only facilitates visual verification but also provides traceable connections (*provenance*) between header and data cells, and between footnote references and footnotes.

II. PRIOR WORK

We first acknowledge Wang's pioneering research on the two-dimensional representation of multi-category data hierarchies that has long guided our approach to table understanding [2]. Hurst and Douglas were early advocates of converting tables into relational form [3]. Hurst provided a taxonomy of category attributes and emphasized the importance of natural language analysis for table understanding, including the syntax of within-cell strings [4]. He presented many examples of thought-provoking tables [5].

This work was reviewed and augmented by Costa e Silva et al. [6], who analyzed prior work in detail in terms of contributions to the tasks of table location, segmentation, functional analysis (tagging cells as *data* or *attribute*), structural analysis (header index identification), and interpretation (semantics).

Partially overlapping surveys of earlier table research include [7,8]. Kim and Lee reviewed web table analysis from 2000 to 2006 and found logical hierarchies in HTML tables using cell formats and syntactic coherency [9]. They extracted the table caption and divided spanning cells correctly. Like us, but in contrast to many other researchers, they handled vertical and horizontal column headers symmetrically.

Pivk et al. "cleaned and canonicalized" HTML tables into a matrix representation similar to our grid table, but they depended on cell formats (letters, numerals, capitalization, and punctuation) rather than indexing properties [10]. Like Hurst they labeled cells as *access* or *data cells* and assembled them into a Functional Table Model. The final output was a semantic (F-logic) frame constructed with WordNet. Their complex evaluation scheme was hampered by human disagreement over the appropriate description of the frames

After Halevy, Norvig, and Pereira pointed out the importance of sheer data size [11], Google researchers collected and analyzed (necessarily with very limited manual verification) millions of tables harvested from the web [12,13]. Their general approach has been to treat table rows as tuples with attributes specified by the top row. Extending this work beyond simple relational tables, Adelfio and Samet generated interpretations for spreadsheet and HTML tables [14]. Using Conditional Random Fields (CRF), they classified each table row as *header, data, title, group header, aggregate, non-relational metadata,* or *blank*. With their test set of 1048 spreadsheet tables and 928 HTML tables, they achieved an accuracy of 76.0% for classifying header and data rows for spreadsheet tables and 85.3% for HTML tables, and for classifying all rows, 56.3% and 84.6% respectively. These heuristic approaches, unlike ours, do not reveal the precise alignment of column-header cells with data elements. Furthermore, they depend on appearance features, whereas we use indexing properties for further analysis.

Also heavily row-oriented is V. Long's analysis of a large sample of tables from Australian Stock Exchange financial reports [15]. A valuable aspect of this work is the detection and verification of the scope and value of aggregates like totals, subtotals, and averages. The analysis is based on a blackboard framework with a set of cooperating agents. Other work dealing with aggregates in tables includes [16].

Chen and Cafarella transformed spreadsheet tables into relational database tables [17]. Like Adelfio and Samet and Pinto et al. [18], they adapt a CRF to label each row as *title, header, data,* and *footnote*, using similar row features. (Rows labeled as "data" also include the cells in the stub, hence to distinguish between the two, they must assume, unlike us, that the data region is purely numeric.) Their hierarchy extractor builds parent-child candidates of cells in the header region using formatting, syntactic, and layout features. The candidate list is pruned by an SVM classifier. In the experiments reported below we benefited from their observations and insights and gratefully made use of their posted SAUS corpus [19].

Logical analysis of table header structure, as we do here, has received scant attention in the literature. Fang et al. identify a set of features to segregate and detect table headers in a random sample drawn from the CiteSeer's PDF files [20]. The rule-based table segmentation and header analysis of [21] includes experiments on our data with similar canonical output.

Interactive methods based on expert advice and user feedback [22,23,24] will remain of interest until wholly automated and error-free table analysis is demonstrated.

The foundations for analyzing table headers that we developed over the years have been published in a succession of conference papers cited in our report at ICPR 2014 [25] and updated in our recent IJDAR article [1].

III. TABLE STRUCTURE

The most important part of a table, and the subject of most prior research to date, is the *principal region* that consists of the stub head, column header, row header and data The essential structure of human-readable tables is that *every data cell is indexed by a sequence of header-path cells in the row header and by a sequence of header-path cells in the column header*. Our table analysis is based on algorithmic determination of this indexing structure [26]. It is the analysis and exploitation of the indexing structure, instead of dependence on formatting and appearance features that differentiates our work from all other prior table research that we are aware of.

The size of the *stub head*, at the top-left of the principal region, is determined by the height of the column header and the width of the row header. The bottom-right corner of the stub header is the *minimum indexing point.* The rest of the table (*ancillary regions*) may include a *table title*, rows or columns of *notes* (often source, date of origin or explanations), *footnote references*, *footnot*es and *footnote markers,* and empty rows or columns for improved legibility. Notes may occur anywhere. In very long or wide tables, headers may be repeated for visibility under scrolling. Peculiarities of the table generation process may produce superfluous external empty rows or columns.

Conversion to CSV loses most formatting information: color, typeface, type size, and style (bold, italic, small caps). When a spanning cell is split into elementary cells, its content is copied into the top-left elementary cell and the other elementary cells are left empty. The textual and numerical content of other cells are retained as within-cell strings. Excel right-justifies numeric strings unless they contain thousands-separators symbols, and displays some date strings as text.

The indexing structure is preserved by the above transformation, but the precise alignment of the labels of subdivided spanning cells requires algorithmic *prefixing*. Our Minimum Indexing Point Search (MIPS) algorithm finds the *minimal indexing headers* (i.e., the minimum number of rows/columns in the column/row header) that suffice to index the data [26]. However, the column headers often contain additional information that can be recovered by *header*

extension via analysis of spanning cell configuration. Prefixing (§IV), header extension (§V), and detailed table configuration statistics (§VI) are the main contributions of this report.

IV. PREFIXING

In an ideal table, the labels necessary to uniquely index any data cell are directly to the left or above that data cell. Header labels within the same header row or column may be repeated, like "First Quarter" for different years. In column headers the distinguishing labels can be placed in the row above the one that contains repeated labels. However, table designers are reluctant to add extra columns to the row header, and therefore usually insert the distinguishing labels in the same column, with some formatting convention, like indentation, bold face, color, or type case and size. These conventions vary from source to source (and even within the same source), are often inconsistent and, except for some indentation, cannot be preserved in CSV files of ASCII or Unicode strings.

In order to avoid depending on formatting or appearance features for locating the minimal indexing headers, our algorithm prefixes each repeated label with a unique label. Even though only a few labels may be repeated, an entire column or row must be added in order to maintain the rectangularity of the table grid. For deeper label hierarchies, more than one row or column must be added. Although the process is the same for row and column headers, we illustrate it only for row headers where it is far more often necessary.

The prefixing algorithm processes successive columns left to right as long as the current columns contain any duplicate rows, as shown by the following pseudo-code. A check (not shown) for a new column identical to some previous column prevents a conceptually possible infinite loop.

```
Prefix(RowHeader)
# RowHeader is an Nrows × Ncols array of table cell values.
# PrefixedRowHeader, the return argument, has no duplicate rows.
# Prefixing the rightmost column replaces it by two new columns,
    the second of which can also be prefixed.

NLastCol ← Ncols  # Ncols is the number of columns in RowHeader
while there are any duplicates among the rows of RowHeader:
    LastCol ← null;   NewCol ← null;   Prefix ← 'NOPREFIX'
    LastCol ← RowHeader(:, NLastCol)
    SinglesList ← set of cell values that occur only once in LastCol
    for krow = 1:Nrows:   # Check for duplicates in rightmost column
    Target ← RowHeader(krow, NLastCol)
        if Target is in SinglesList:   Prefix ← Target;
            Append Target to LastCol;   Append 'DITTO' to NewCol
        else:  Append Prefix to LastCol;   Append Target to NewCol
        end if-else      # Target not in Singles, therefore prefixed.
    end for      # new columns constructed for RowHeader
    delete RowHeader(NLastCol);  append LastCol to RowHeader
    append NewCol to RowHeader;  NLastCol ← NLastCol + 1
end while
PrefixedRowHeader ← RowHeader
Return PrefixedRowHeader
```

Fig. 1 shows an HTML table with a column added by prefixing to allow indexing the data despite duplicate labels. The four categories of the table in Fig. 2 (families/individuals, quantiles, years, and transfers) are found after prefixing repeated row-header labels with "Economic Families" and "Unattached Individuals".

1. Buildings, dwellings and persons by type of building and number of storeys 31.12.2008

Open this table as a bigger version

Type of building, number storeys	Buildings	Dwellings	Persons (dwelling population)
Total	1 421 188	2 767 925	5 219 284
1 - 2 storeys	1 349 630	1 657 771	3 608 304
3 - 9 storeys	51 529	1 086 426	1 577 577
10 + storeys	292	12 771	17 315
unknown number of storeys	19 737	10 957	16 088
Detached hiouses	1 082 511	1 112 142	2 674 740
Attached houses	75 109	382 273	696 753
Blocks of flats	55 925	1 218 633	1 758 564
1 - 2 storeys	16 769	141 329	199 568
3 - 9 storeys	38 543		
10 + storeys	186		
unknown number of storeys	427		
Other than residential buildings	207 643		
1 - 2 storeys	180 191		
3 - 9 storeys	8 874		
10 + storeys	99		
unknown number of storeys	18 479		

Source: Buildings and Free-time Residences, Statistics Fin

Inquiries: Marja Hermiö (09) 1734 3211, Arja Tiihonen (09) Huohvanainen (09) 1734 3232

Director in charge: Jari Tarkoma

```
final_R_header:
['Total       ', 'DITTO     ']
['Total       ', '1 - 2 storey']
['Total       ', '3 - 9 storey']
['Total       ', '10 + storeys']
['Total       ', 'unknown numb']
['Detached hio', 'DITTO     ']
['Attached hou', 'DITTO     ']
['Blocks of fl', 'DITTO     ']
```

Fig. 1. A web table from Statistics Finland (left) and part of the the two-column row header created by prefixing (right). "1 - 2 storey" is one of several duplicate labels that are prefixed by a previous unique label. The printout truncates all labels to 12 characters.

Government transfers and income tax						
(Government transfers)						
	2006			2007		
	Government transfers					
	Average $ constant 2007	Implicit transfer rates1 %	Shares %	Average $ constant 2007	Implicit transfer rates1 %	Shares %
Economic families,	8900	10.7	100	9000	10.4	100
Lowest quintile	12500	47.2	28.1	12800	45.7	28.4
Second quintile	11000	22.8	24.7	11300	22.7	25.1
Third quintile	9200	13.2	20.7	9200	12.9	20.4
Fourth quintile	7000	7.2	15.6	6900	6.9	15.4
Highest quintile	4,800E	2.7E	10.9E	4900	2.6	10.8
Unattached individu	5500	15.8	100	5700	15.9	100
Lowest quintile	3600	49	13.2	4300	53.3	15
Second quintile	9100	54.3	32.7	9600	55.7	33.7
Third quintile	8000	31	28.9	7300	27.6	25.9
Fourth quintile	4300	10.6	15.6	4700	11.3	16.5
Highest quintile	2700	3.2	9.6	2600	3	9
E : use with caution.						
Note: After-tax income quintiles constitute one method of categorising a population by in						
1. The "implicit" rate of either transfers or taxes demonstrates the relative importance of						
Source: Statistics Canada, CANSIM, table (for fee) 202-0301 and Catalogue no. 75-202-X.						

Fig. 2. CSV version of a four category HTML table from Canada Statistics. Cells with footnote references are hilighted yellow.

The column header needs no prefixing because after being copied into the empty cells to their right (cf. Section V), the unique 2006 and 2007 labels distinguish the duplicate labels in the fifth row. The program also finds the two references to footnote E and to footnote 1, and labels all the notes rows. Here we can use only very small tables for illustration.

V. Spanning Cells and Header Extension

Spanning Cells: The information about which groups of cells constitute spanning cells in the original table is lost during the CSV conversion. Although a left filling algorithm, which fills each empty cell with the label of the nearest non-empty cell in the same row, is sufficient for segmentation, it cannot correctly recreate the spanning cells when the header includes both horizontal and vertical spanning cells, as illustrated in Fig. 3. The correct filling for this example, using the pseudo-code below, is shown in Fig. 4.

```
Fill(Table):
# Table is the original table with surrounding blank rows/columns removed.
# Row_range  (Col_range) are the set of non-empty rows (columns)
     in the table, in top-to-bottom (left-to-right) order.

for c in Col_range:     flag =          # Top Fill
    for r in Row_range:
        if Table[r][c] != null: top_fill = Table[r][c]; flag = 1
        else if flag == 1: Table[r][c] = top_fill
        if r+1 in list_empty_rows: flag = 0
for r in Row_range:    flag = 0        # Left Fill
    for c in Col_range:
        if Table[r][c] != null:
            if (Table[r+1][c]  != Table[r][c]) or (Table[r-1][c] != Table[r][c]):
                left_fill = Table[r][c]; flag = 1
            else: flag = 0
        else if flag == 1: Table[r][c] = left_fill
        if c+1 in list_empty_cols: flag = 0
return Table
```

After the columns are filled, the algorithm shifts to filling the rows from top to bottom. For every row, an initial group of leftmost blank cells, if any, is skipped. Then, the label of a non-blank cell that differs from the label of the cell immediately below it is used to fill the group of any blank cells immediately to its right.

A row of blank cells (for column filling) or a column of blank cells (for row filling) is treated as a barrier to the filling described above. That is, it resets the process to the beginning of the column or row.

Header Extension: The minimal headers produced by our segmentation algorithm do not always capture their full extent. For the column header of the SAUS-200 table shown in Fig. 5, for example, MIPS identifies the column header as the blue cells in row 8 because they suffice to index the data columns. The header extension algorithm identifies any additional neighboring rows to create the intended header. In the example, rows 7 and 9, shown in tan, are added to the header found by MIPS.

The input to the algorithm is a CSV table and the critical cells CC1 and CC2 that define the extent of the stub head determined by MIPS. The header extension of rows above CC1 is carried out one row at a time using the following rule:

Add the new row if it does not consist of cells with identical values and if it adds at least one non-blank cell that has a value different from the cell immediately below it.

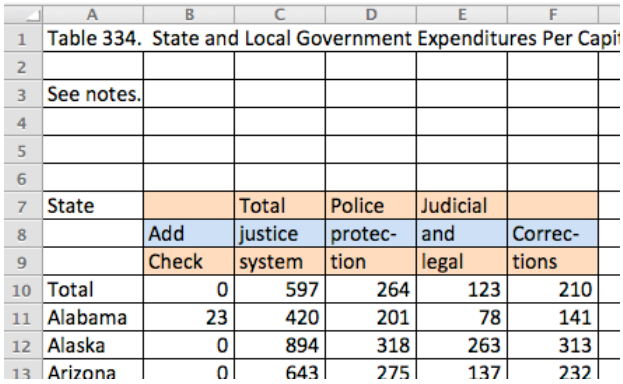
The header extension of rows below CC2 is done using a symmetrical rule.

	A	B	C	D	E	F	G	H	I	J
1	Pupils in comprehensive schools and with leaving certificates from comprehensive schools 1990-2009									
2	Year	Schools	Pupils	Pupils	Pupils	Pupils	Pupils	Grade 1	Leaving certificates	
3			Pre-primary	Grades	Grades	Additional ec	Total	Total	Total	
4				6-Jan	9-Jul	9-Jul	9-Jul	9-Jul	9-Jul	
5										
6	1990	4,869	2,189	389,410	197,719	3,602	592,920	67,427	61,054	

Fig. 3. The left fill algorithm, applied to the Troy Table 0058, with the filled labels shown in red. Note: Excel renders Grades "1-6" in cell D4 and "7-9" in cell E4 as dates, but they can be preserved correctly if the conversion from HTML is carried out into text-formatted Excel CSV tables..

	A	B	C	D	E	F	G	H	I	J
1	Pupils in comprehensive schools and with leaving certificates from comprehensive schools 1990-2009									
2	Year	Schools	Pupils	Pupils	Pupils	Pupils	Pupils	Grade 1	Leaving certificates	
3		Schools	Pre-primary	Grades		Additional ec	Total	Grade 1	Leaving certificates	
4		Schools	Pre-primary	6-Jan	9-Jul	Additional ec	Total	Grade 1	Leaving certificates	
5										
6	1990	4,869	2,189	389,410	197,719	3,602	592,920	67,427	61,054	

Fig. 4. The vertical-first filling of the Troy Table 0058 that captures both the horizontal spanning cell "Pupils" and the other vertical spanning cells.



	A	B	C	D	E	F
1	Table 334. State and Local Government Expenditures Per Capi					
2						
3	See notes.					
4						
5						
6						
7	State		Total	Police	Judicial	
8		Add	justice	protec-	and	Correc-
9		Check	system	tion	legal	tions
10	Total	0	597	264	123	210
11	Alabama	23	420	201	78	141
12	Alaska	0	894	318	263	313
13	Arizona	0	643	275	137	232

Fig. 5. Column header for the SAUS Table 0334 determined by MIPS (blue) and extended to include additional rows (hilighted tan).

VI. Experimental Results

Table 1 shows our observations on the CSV versions of 200 ("Troy") HTML tables from 6 international statistical sites, 200 randomly selected SAUS spreadsheet tables with ground truth, and 1120 SAUS spreadsheets for which we don't have ground truth. VeriClick [23] allows rapid and accurate interactive ground-truthing of segmentation for tables that fit on the screen. However, ground-truthing some SAUS tables with hundreds of rows and columns and complex headers proved to be extremely time consuming and required resolving many disagreements between independent human observers. Over 1.5 million table cells were classified into ten categories.

On average, the tables in the three data sets have three times as many rows as columns. The SAUS tables are about twice as wide and 2.5 times as long as the HTML tables. The average number of data cells (85) is about a third the total number of cells (290) in the csv version of the HTML tables, and about one half (676/1184) in the spreadsheets. The largest tables (with 183 columns and 828 rows, respectively) have 10-25 times n cells as the average. Run time is roughly proportional to the number of cells: about 12 HTML tables per second on a 2.4 Ghz Dell Optiplex under Windows 7 running Python 2.7.

More than 10% of both HTML and spreadsheet tables have multi-category headers, with multiple column categories more common than multiple row categories. Row headers with three or more columns and column headers with three or more rows are far more frequent in spreadsheets than in HTML tables.

Discriminating labels above or below the minimal indexing column headers were found in 2 Troy tables and 44 SAUS_200 tables. These headers were extended by adding the necessary rows. We found no instances of row headers that require extension.

We converted to CSV only the first worksheet of the SAUS XLSX workbook files. The footnotes are on separate worksheets and never appear in our CSV files. The program found 56 Troy tables with footnotes ant 158 references to the footnotes. Fifteen footnote markers did not appear in the tables above them or were missed. Our HTML tables have an average of 5 note rows, with typically only one filled cell in each row. Spreadsheets average 8 note rows and 1 notes column, and also have many empty rows and columns.

The segmentation of HTML row and column headers and data region was 98% correct versus 96% for the spreadsheets. The Minimum Indexing Point was located with 99% accuracy on both sets. The errors in delimiting the data region are due to a few notes rows mistaken for data or vice versa. Since all of the above observations (except for the gross size of the tables) are based directly on the results of the segmentation, we believe that they are as reliable as the segmentation. Furthermore, the consistency of the header sizes and of prefixing and category counts suggests that the segmentation accuracy of the 1120 tables without ground truth does not differ much from that of the validated 200 SAUS tables.

TABLE I. CHARACTERISTICS OF SEGMENTED TABLES

Observations	Troy	SAUS_ 200	SAUS_ 1120
Number of tables	200	200	1120
Number of tables processed	199	198	1107
Trivial tables	1	2	5
Segmentation errors			
Minimum indexing point	2	2	NA
Critical cells	4	9	NA
Gross size of tables			
Rows average	25	64	61
(maximum)	(183)	(453)	(828)
Columns average	11	17	17
(maximum)	(80)	(81)	(201)
Cells average	290	1184	1117
(maximum)	(7320)	(14094)	(21294)
Net size of tables			
Data rows average	15	45	41
Data columns average	5	15	15
Data cells average	85	676	664
Categories			
Multi-category row headers	7	12	75
Multi-category col headers	14	13	105
Prefixed headers			
Row headers	23	63	378
Column headers	3	0	7
Size of headers			
1-col 1-row header	145	56	273
Row headers with ≥3 columns	1	9	51
Column headers with ≥3 rows	3	44	301
Tables with extended headers	2	44	192
Footnotes			
Footnoted tables	56	NA	NA
Reference markers (total)	91	NA	NA
References found (total)	158	NA	NA
References not found (total)	15	NA	NA
Notes			
Rows (average)	5.13	8.00	8.45
Columns (average)	0.06	0.89	0.73
Total run time (sec) w/o file output	15.6	61.9	308.2

In total, the program segmented 1504 of 1520 tables. Of the remaining 16, 8 were trivial tables (with only one row or column of data) that the program did not attempt to analyze. Visual examination of the remaining 8 revealed some source errors, as did the 4 tables with MIP errors. Two of the SAUS worksheets had side-by-side tables, some had repeated columns or undifferentiable repeated column labels, and we found two spreadsheets under constructions with repeated data rows and informal notes to staff members.

VII. SUMMARY

Tables provide compact data display with subtle means of representing complex relationships. Formalization and exploitation of intrinsic structural table constraints opens the way for algorithmic conversion of vast amounts of tabulated data to uniform standard data analysis formats.

Segmentation and classification based on the minimum indexing point provide effective means for inferring category hierarchies and explicit header-data relationships, and for labeling ancillary table constituents. In contrast to recent machine learning approaches based on formatting and appearance features, methods based on 2-D table syntax offer the advantage of language, domain, and format independence.

Experiments on 1520 diverse tables demonstrate a practical and reliable solution to important aspects of information extraction from HTML tables and spreadsheets.

We are confident that algorithmic analysis can be extended to scanned and OCR'd tables transformed to a CSV grid format. It should be considered as a complement, rather than an alternative, to published methods based on format and semantics. For example, indexing cannot separate the header from data when there is only a single row or a column of data, and our program (and even the authors) are occasionally stumped by the ambiguity of some *notes* or *data* cells.

Any claim of automatically derived descriptors raises questions about the validity of the results. The accuracy of the segmentation and category extraction, verified against laboriously collected ground truth on a random subset of 400 heterogeneous tables, provides some confidence in the applicability of the proposed algorithms to arbitrary tables. The most important remaining task is to flag for human editors the inevitable residue of tables that cannot be automatically parsed.

REFEREMCES

1 D.W. Embley, M. Krishnamoorthy, G. Nagy, and S. Seth, "Converting Heterogeneous Statistical Tables on the Web to Searchable Databases," Int'l J. Document Analysis and Recognition, Vol. 19, 2, 119-138, June

2 X. Wang, Tabular abstraction, editing, and formatting, Ph.D. thesis, University of Waterloo 1996.

3 M. Hurst, S. Douglas, "Layout and language: Preliminary investigations in recognizing the structure of tables," Procs. Int. Conf. Doc't Analysis & Recog'n (ICDAR'97), 1043–1047, 1997.

4 M. Hurst, "Towards a theory of tables," 8 (2-3), Springer, Heidelberg, 66-86, 2006.

5 M. Hurst, The Interpretation of Tables in Texts. Ph.D. thesis, University of Edinburgh, 2000.

6 A. Costa e Silva, A. M. Jorge and L. Torgo, "Design of an end-to-end method to extract information from tables," Int. J. Doc. Anal. Recognit. 8 (2-3), Springer, Heidelberg, 66-86, 2006.

7 R. Zanibbi, D. Blostein, J. R. Cordy: "A survey of table recognition," Int. J. Doc. Anal. Recognit., 7(1): 1-16, 2004.

8 D.W. Embley, D. Lopresti, M. Hurst, and G. Nagy, "Table Processing Paradigms: A Research Survey," Int. J. Doc. Anal. Recognit. 8 (2-3), 66-86, Springer, June 2006.

9 Y-S Kim, K-Y Lee, "Extracting logical structures from HTML tables," Computer Standards & Interfaces, 30, 5 296-308, July 2008.

10 A. Pivk, P. Cimiano, Y. Sure, M. Gams,V. Rajkovic, R. Studer, "Transforming arbitrary tables into logical form with TARTAR," Data & Knowledge Engineering 60, 567–595, 2007.

11 A. Halevy, P. Norvig, and F. Pereira, "The Unreasonable Effectiveness of Data," IEEE INTELLIGENT SYSTEMS. March/April 2009.

12 P. Venetis, A. Halevy. J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu, "Recovering Semantics of Tables on the Web," Proceedings of the LDB Endowment, Vol. 4, No. 9, 2011.

13 H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, J. Goldberg-Kidony, "Google Fusion Tables: Web-Centered Data Management and Collaboration," SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.2010.

14 M.D. Adelfio and H. Samet, "Schema Extraction for Tabular Data on the Web," Procs.The 39th International Conference on Very Large Data Bases, (Proceedings of the VLDB Endowment, Volume 6, Number 6), Riva del Garda, Trento, Italy 26–30 August, 2013.

15 V. Long, An Agent-Based Approach to Table Recognition and Interpretation, Macquarie University PhD dissertation, May 2010.

16 N. Astrakhantsev, "Extracting Objects and Their Attributes from Tables in Text Documents," Proceedings of the Institute for System Programming Volume 21, 297-310., Moscow, Russia, 2011.

17 Z. Chen and M. Cafarella, "Automatic Web Spreadsheet Data Extraction," Proceedings of the 3rd International Workshop on Semantic Search over the Web (SSW 2013), Riva del Garda, Trento, Italy, 30 August 2013.

18 D. Pinto, A. McCallum, X. Wei, W.B. Croft, "Table extraction using conditional random fields," Procs 26th Annual Int. ACM SIGIR Conference on R&D in Information Retrieval, 235–242 2003.

19 SAUS dataset: http://wwweb.eecs.umich.edu/db/sheets/datasets.html (accessed 12/23/2015).

20 J. Fang, P. Mitra, Z. Tang, L. Giles, "Table Header Detection and Classification," Proceedings of the Twenty-Sixth AAAI Conference on AI, 2012.

21[A.O. Shigarov et al. "Rule-Based Canonicalization of Arbitrary Tables in Spreadsheets," accepted, 22nd Int'l Conf. on Information and Software Technologies, Druskininkai, Lithuania, October 2016,

22 N. Astrakev, D. Turdakov, N. Vassilieva, "Semi-automatic Data Extraction from Tables," Proceedings of the 15th All-Russian Conf. on Digital Libraries: Advanced Methods and Technologies, Digital Collection — RCDL, Yaroslavl, Russia, 2013.

23 G. Nagy, M. Tamhankar, "VeriClick, an efficient tool for table format verification," Proc. SPIE 8297, Document Recognition and Retrieval XIX, 82970M, January 23, 2012.

24 T. Kasar, T. K. Bhowmik and A. Belaid, "Table information extraction and structure recognition using query patterns," Procs.13th International Conference on Document Analysis and Recognition, ICDAR 2015, Vol. 1, pp. 1086-1080, 2015.

25 D.W. Embley, S. Seth, G. Nagy, "Transforming Web tables to a relational database," Procs. ICPR 2014, Stockholm, Sweden, 2014.

26 S. Seth, and G. Nagy, "Segmenting Tables via Indexing of Value Cells by Table Headers," Proceedings ICDAR 2013, Washington, D.C., August 2013.