G. Nagy, M. Tamhankar, VeriClick, an efficient tool for table format verification, Procs. SPIE/EIT/DRR, San Francisco, Jan. 2012.

VeriClick, an efficient tool for table format verification

George Nagy[*], Mangesh Tamhankar[†]
DocLab, Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute,
Troy, NY, USA 12180

ABSTRACT

The essential layout attributes of a visual table can be defined by the location of four critical grid cells. Although these critical cells can often be located by automated analysis, some means of human interaction is necessary for correcting residual errors. VeriClick is a macro-enabled spreadsheet interface that provides ground-truthing, confirmation, correction, and verification functions for CSV tables. All user actions are logged. Experimental results of seven subjects on one hundred tables suggest that VeriClick can provide a ten- to twenty-fold speedup over performing the same functions with standard spreadsheet editing commands.

Keywords: table analysis, well-formed table, interactive table layout verification, critical cells

1. INTRODUCTION

After two decades of experimentation on various aspects of table processing, research is reaching the stage where large end-to-end experiments on information extraction are practicable. Attention has gradually shifted from scanned printed tables to HTML and PDF tables that obviate the need for OCR. It is also becoming increasingly clear how much the complexity of the formatting conventions developed and refined for human access to tables hampers the attainment of perfectly accurate automated information extraction[1]. For some time to come, most practical applications will require some human intervention to produce acceptable results.

Here we report the development of an interactive tool, VeriClick, which improves the accuracy of layout analysis of spreadsheets containing tables imported from the web. In addition to confirming or correcting the results of computer analysis of tables, VeriClick can also be used as a ground-truthing tool for large table data sets. Since human table analysis is also imperfect, an add-on, Merge-Diff, allows comparison, arbitration, merger and concatenation of results reached by several computer or human experts, thereby producing unified results for further downstream analysis of table contents. These developments were also motivated by our long-term goal of human-machine symbiosis, where the machine will take advantage of human interaction to avoid repeating the same mistake again and again.

Because many processing steps are common to tables, forms, and lists laid out on a grid, the word “table” is often used for all three. Here we consider only *well-formed tables* where the data values can be indexed by row and column headers. This definition includes tables with a single row of data cells indexed by a (possibly hierarchical and multi-row) column header and only an implicit row header, and single-column tables indexed by a row header. Our definition excludes nested and concatenated tables, and multi-column/row lists with only a column/row header.

1.1 Prior Work

Although there has been considerable research on table processing, especially on HTML tables, we are not aware of any published research on interactive correction of computer-created errors in table layout analysis since the comprehensive surveys of Zanibbi et al. and Embley et al[2,3]. The present work is part of the larger TANGO project, Table Analysis for Growing Ontologies[4], where we addressed similar goals of information extraction and aggregation from tables[5,6], attempted to formulate an analytical framework for characterizing tables[7], proposed the notion of header paths[8] , and

[*] nagy@ecse.rpi.edu
[†] tamham@rpi.edu

demonstrated an end-to-end table processing pipeline that yielded relational tables and 34,110 subject-predicate-object RDF triples from 200 tables[9].

Our earlier experiments indicated that correcting the residual errors of even relatively accurate automated processing, using standard table and text preparation and editing tools, requires an intolerable amount of human time. We constructed two previous interactive table processing systems[10,11]. VeriClick, the interactive spreadsheet described here, is our third and by far most successful endeavor to minimize the human effort necessary to verify or correct automated layout analysis. VeriClick is freely available from the authors and is small enough for dissemination via email (if built-in security measures don't strip it from the attachment because of the embedded macros!). The macrocode embedded in a VeriClick spreadsheet can be edited if necessary. We hope that other researchers can make use of it to ground-truth and verify their own collections of tables.

Since nothing in VeriClick is dependent on our downstream analysis, we present the entire data flow (Fig. 1) only by way of context. The selected tables are exported from the HTML pages to Excel and automatically converted to Comma Separated Value (CSV) format. The critical cells extracted by Python layout analysis routines are verified corrected using VeriClick. The header paths to each row and column of data cells are extracted and "factored" by open-source Sis software designed for switching algebra. Sis outputs canonical sum-of-products expressions that are turned into Relational Tables and Resource Description Framework (RDF) triples by a Java program. Duplicate row or column headers can be detected by Sis. Beyond VeriClick, only the heuristic path extraction step can produce incorrect output, but given correct layout information (i.e., the location of the critical cells) this step seldom fails. Therefore in our methodology, careful interaction through VeriClick is the key to complete and accurate end-to-end information extraction. For details, please see our earlier reports[8].

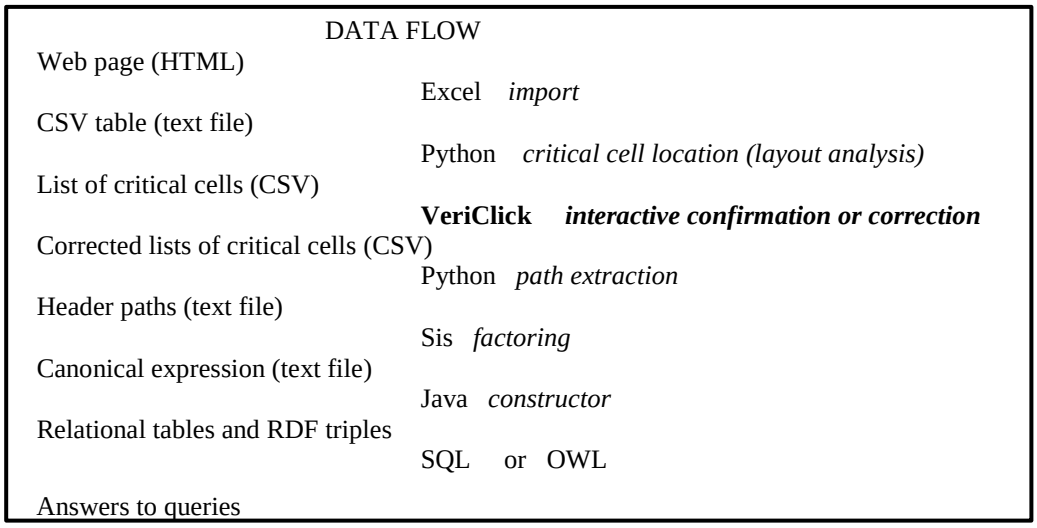


Fig. 1. Data flow for a web table processing pipeline. VeriClick provides an intermediate interactive step

2. WELL-FORMED TABLES AND CRITICAL CELLS

A well-formed table (WFT) has a rectangular array of data value cells, each uniquely indexed by a (column-header, row-header) pair path. A simple configuration with a single-row column header and a single-column row header is shown in Fig. 2. The array of data value cells is called the *delta region*. The part of the table above the delta region is the *column-header region*, and the part to the left is the *row-header region*. The area to the left of the column header and above the row header is the *stub*.

	A	B	C
a	*d11*	*d12*	*d13*
b	*d21*	*d22*	*d23*
c	*d31*	*d32*	*D33*

Fig. 2. A simple table configuration with an empty stub. The delta region is shaded.

Row headers and column headers are often arranged in a hierarchy, as in the well-known table of Fig. 3. Although in principle horizontal and vertical indexing is symmetric, this table displays the common practice of placing the roots of row header trees in the stub. Therefore the row-header path to the top-left data cell ("85") is Year-1991 + Term-Winter, while the column-header path is Mark-Assignments-Ass1. Determining whether the contents of the stub contain the title of the paper, or belong to the column header, or to the row header, requires semantics. Because the last configuration is most common, our programs insert the contents of the stub at the top of the row category tree(s).

This table is often used to illustrate the concept of Wang categories[12]. There is only a single column category here, Mark. The two row categories are Year and Term. The category trees are extracted from the header paths to rows and columns by Sis for downstream processing.

Year	Term	Mark					
		Assignments			Examinations		Grade
		Ass1	Ass2	Ass3	Midterm	Final	
1991	Winter	85	80	75	60	75	75
	Spring	80	65	75	60	70	70
	Fall	80	85	75	55	80	75
1992	Winter	85	80	70	70	75	75
	Spring	80	80	70	70	75	75
	Fall	75	70	65	60	80	70

Fig. 3. Prototypical Wang table.

A well-formed table can be partitioned into four (not necessarily contiguous) regions by four critical cells. Fig. 4 shows the location of these critical cells for a general table layout. The critical cells are numbered according to the obvious partial order. One pair of critical cells defines the bounding box of the stub, and the other pair defines the bounding box of the delta region. The vertical dimension of the column header is governed by the height of the stub, and the horizontal dimension of the row header is governed by the width of the stub. The other dimensions of the headers must be commensurate with the height and width of the delta region. When the stub contains only a single cell, CC1 and CC2 are identical. In single-row and single-column tables, CC3 and CC4 differ only in one of their coordinates. The table title is usually in the cells above CC1, and footnotes are in the cells below CC4.

Average Weight							
CC1		YEAR					
	CC2	1991	1992	1993	1994	1995	
Adult	M	CC3					
*	F						
Child	M						
*	F					CC4	
*Adults are persons over 16 years old							

Fig. 4. Location of the critical cells CC1, CC2, CC3 and CC4. When a table is imported into a spreadsheet from a web page, it may contain, in addition to the table proper, the title of the table, notes or footnotes, and sometimes empty columns on the right. Here the stub could be empty, or it could contain row header roots like "Age" and "Gender".

The first step in the pipeline extracts the critical cells from the CSV tables that were manually selected and imported from randomly chosen HTML pages of large statistical web sites. Fig. 5 shows part of the file of critical cell lists created by our Python program. When the program fails to find the critical cells in some table, it reports *z0, z0, z0,* z0 for that table. The critical cells for each table are inspected and either confirmed or corrected via the VeriClick program described in the next section.

Filename	CC1	CC2	CC3	CC4
…..				
C10112.csv	A2	A3	B4	F26
C10113.csv	A2	A2	B3	F19
C10114.csv	A2	A3	B4	J13
C10115.csv	A1	A2	B3	D22
C10116.csv	A5	A5	B6	G7
C10117.csv	A8	A9	B10	T20
C10118.csv	A3	A3	B4	F36
C10119.csv	A2	A3	B4	G5
C10120.csv	z0	z0	z0	z0
C10121.csv	A2	A2	B3	D19
C10122.csv	A5	A5	B6	G7
C10123.csv	A5	A5	B6	G7
C10124.csv	A5	A5	B6	G7
C10125.csv	A3	A3	B4	F35
C10126.csv	A3	B3	C4	C11
C10127.csv	A3	A3	B4	F17
C10128.csv	A3	A3	B4	F24
…..				

Fig. 5. Partial output of Python program that finds the critical cells. Each row corresponds to one table. The spreadsheet addresses of the critical cells follow the filename of the table.

The current python program can locate the critical cells only if the delta region consists of numerical values, or if the stub is empty. The search algorithm for these conditions has several parameters. The first learning step we envision is adapting these parameters to produce correct output for tables corrected via VeriClick (and, more importantly, for similar uncorrected tables). More general aspects of learning will include appropriate processing of parameterized title, header, data and footnote cell formats, partly-blank rows or columns, and data-frames for common table words like TABLE, YEAR, and TOTAL. Although the Python programs could be hard-coded for these situations, we would like to be able to parameterize them in order to demonstrate the effectiveness of operational human feedback.

3. VERICLICK

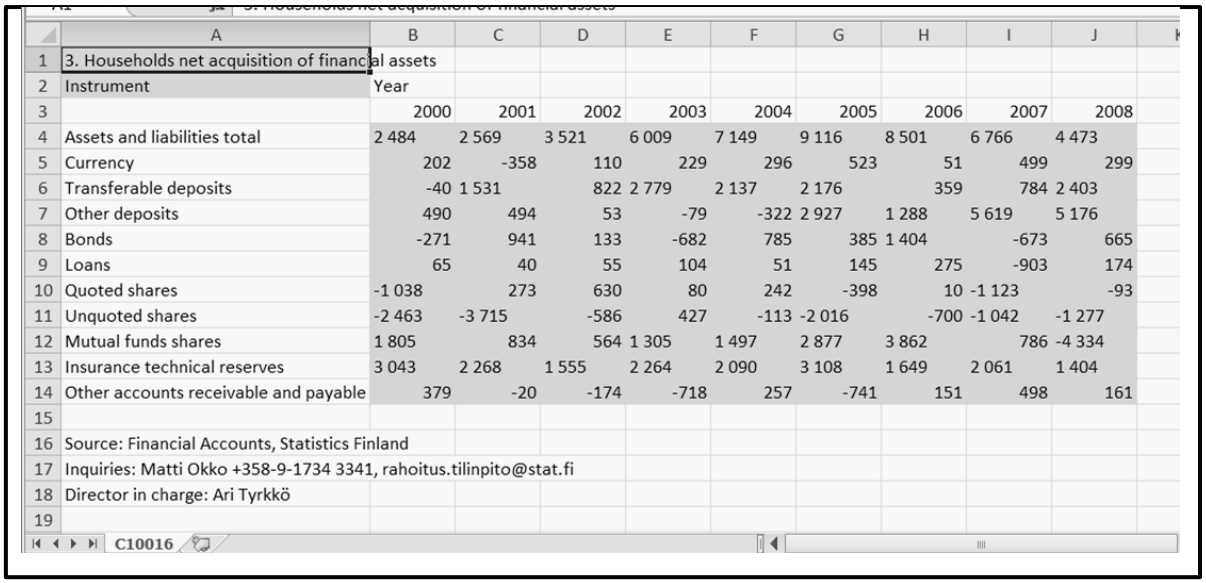
A new user is introduced to VeriClick with a short slide presentation, a five-minute video, and a set of a dozen practice tables. The slide show explains the notion of critical cells, gives a preview of VeriClick, and gives some examples where the choice of critical cells may require a close look at the table. The video demonstrates the method of correction of misplaced critical cells. After the neophyte completes the practice, the results are compared to the correct results and any discrepancy is noted.

VeriClick itself is a spreadsheet with embedded VBA code for (1) reading a file of critical cell coordinates, the parameter file, (2) reading CSV tables from a designated directory, (3) translating operator clicks into the addresses of the critical cell, (4) writing out the results, and (5) creating a log file.

When VeriClick is opened, it first gives the operator a choice between starting with a new set of files, or continuing interrupted work on an earlier set. For a new set of files, browser windows are presented to designate the directory of CSV files to be processed, the file containing the automatically assigned critical cells, and the directory in which the file of corrected critical cells is to be saved. To continue earlier work, it is sufficient to indicate the log filename that contains all necessary information. After these preliminaries, VeriClick loads all the tables and displays them one at a time.

Fig. 6 shows a table as displayed by VeriClick. Here C1 is wrong because the top left cell A1 is part of the title instead of the stub. The operator clicks Cell A1 and the cell below (A2) in turn to correct the error. The location of any critical cell

is always corrected by first clicking on it, and then clicking on the correct location. The current location of a pair of critical cells is indicated by the top-left and bottom-right corners of the highlighted region.



	A	B	C	D	E	F	G	H	I	J
1	3. Households net acquisition of financial assets									
2	Instrument	Year								
3		2000	2001	2002	2003	2004	2005	2006	2007	2008
4	Assets and liabilities total	2 484	2 569	3 521	6 009	7 149	9 116	8 501	6 766	4 473
5	Currency	202	-358	110	229	296	523	51	499	299
6	Transferable deposits	-40	1 531	822	2 779	2 137	2 176	359	784	2 403
7	Other deposits	490	494	53	-79	-322	2 927	1 288	5 619	5 176
8	Bonds	-271	941	133	-682	785	385	1 404	-673	665
9	Loans	65	40	55	104	51	145	275	-903	174
10	Quoted shares	-1 038	273	630	80	242	-398	10	-1 123	-93
11	Unquoted shares	-2 463	-3 715	-586	427	-113	-2 016	-700	-1 042	-1 277
12	Mutual funds shares	1 805	834	564	1 305	1 497	2 877	3 862	786	-4 334
13	Insurance technical reserves	3 043	2 268	1 555	2 264	2 090	3 108	1 649	2 061	1 404
14	Other accounts receivable and payable	379	-20	-174	-718	257	-741	151	498	161
15										
16	Source: Financial Accounts, Statistics Finland									
17	Inquiries: Matti Okko +358-9-1734 3341, rahoitus.tilinpito@stat.fi									
18	Director in charge: Ari Tyrkkö									
19										

C10016

Fig. 6. VeriClick display of a table from a Finnish statistical web site.

Fig. 7 shows the top-left portion of the table after the correction described above. If all the critical cells are deemed correct – either because the program located them correctly or because the operator has already corrected them – a double click anywhere prompts the display of the next table. Except for the preliminary file selection, no buttons are shown because larger tables cover the entire display. The operator may be required to scroll down to verify that CC4 is correct (i.e., to ensure no footnotes are included in the delta-cell region). The session can be interrupted any time, without loss of information, by closing the VeriClick spreadsheet.

3. Households net acquisition of financial assets										
Instrumen	Year									
	2000	2001	2002	2003	2004	2005	2006	2007	2008	
Assets and	2 484	2 569	3 521	6 009	7 149	9 116	8 501	6 766	4 473	
Currency	202	-358	110	229	296	523	51	499	299	

Fig. 7. Partial display of the table of Fig. 5 after the operator has corrected the misplaced critical cell.

The logging subsystem records the time spent by the operator on each table. The date, start time and end time of every session time is also recorded, along with the directory paths and filenames of every file used in the session. In addition to its use for analysis of human effort, this information allows a user to resume an interrupted session

VeriClick contains about 500 lines of VBA macro code. Before loading any CSV tables, the "empty" VeriClick spreadsheet is about 260 KB. Fig. 8 on the next page shows its top-level pseudo-code.

4. EXPERIMENTAL RESULTS

Critical cells were extracted from 100 tables randomly selected from our corpus of 1000 web tables from ten large statistical sites. Seven subjects used VeriClick to correct the locations of critical cells in the 100-table test set. Five subjects are members of the TANGO team, one is a recent CS graduate, and one is an engineering student.

The time taken by the subjects is shown in Table 1. When the critical cells were correctly extracted, the confirmation time was close to the minimum except when large tables required scrolling to verify CC4. The Python program did not

find the critical cells in two of the tables, so the subjects had to correct all four critical cells. Corrections took about twice as long as confirmations (usually only one critical cell had to be corrected), but a few tables where the correct segmentation was not obvious took much longer.

```
        Initialize()
                Prompt user for Files and Directories
                Open all necessary Files
                Initialize Variables
        End Function

        Workbook_SheetSelectionChange()
                If SelectClick                                          //First Click
                        If ActiveCell is Critical Cell
                                Save ActiveCell Coordinate
                        Endif
                Endif

                If MoveClick                                                    //Second Click
                        Change Saved Coordinate to ActiveCell Coordinate

                        Remove Highlighting()
                               If Cell Highlighting includes Flipped coordinates  //rectangle
                                                        defined by top left and bottom right
                                Unflip coordinates
                        Endif
                        ReHighlight Cells()
                Endif

        End Function

        Workbook_SheetBeforeDoubleClick
                StopTimer
                Update LogFile
                Advance to Next Sheet
                Restart Timer
End Function
```

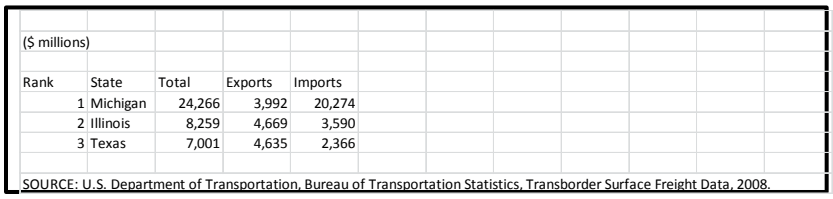
Fig. 8. Pseudo-code of VeriClick macro.

Table 1. Seven subjects' performance using VeriClick

Subject	Average time per table (s)	STD time per table (s)	Minimum time (s)	Maximum time (s)	Session time (min)	Number confirmed	number corrected
S 1	3.2	2.6	0.77	14.8	6:06	87	13
S 2	9.5	9.8	0.02	74.5	19:03	80	20
S 3	14.8	18.0	2.67	105.2	24:47	84	16
S 4	8.4	7.6	2.19	49.8	14:50	65	35
S 5	5.1	8.1	0.75	61.1	9:13	89	11
S 6	4.2	4.1	1.15	22.8	8.27	81	19
S 7	8.3	11.2	1.37	74.2	14:50	85	15

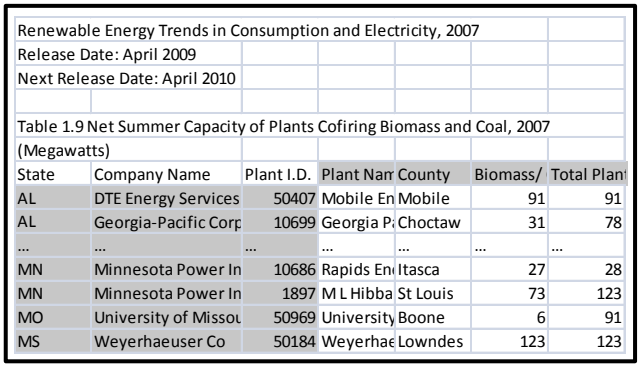
Subjects were puzzled by the table of Fig. 9, where the Python program included the State column with the stub. According to our definition, the rank column is sufficient for unique row headers. Subjects also had difficulty with the table of Fig. 10. Here a quick look shows that the entries in the first column are not unique, but it requires closer

inspection to reveal that the second ("Company") column also has duplicates. Therefore the first three columns must be considered row headers. One of the subjects took a long time on the unusual single-column table of Fig. 11.



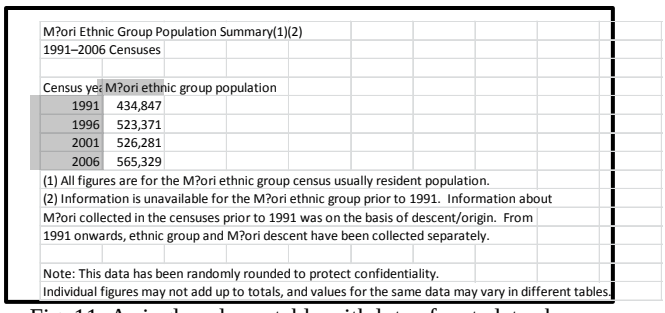
($ millions)				
Rank	State	Total	Exports	Imports
1	Michigan	24,266	3,992	20,274
2	Illinois	8,259	4,669	3,590
3	Texas	7,001	4,635	2,366
SOURCE: U.S. Department of Transportation, Bureau of Transportation Statistics, Transborder Surface Freight Data, 2008.				

Fig. 9. What is the correct row header here?



Renewable Energy Trends in Consumption and Electricity, 2007						
Release Date: April 2009						
Next Release Date: April 2010						
Table 1.9 Net Summer Capacity of Plants Cofiring Biomass and Coal, 2007						
(Megawatts)						
State	Company Name	Plant I.D.	Plant Nam	County	Biomass/	Total Plan
AL	DTE Energy Services	50407	Mobile En	Mobile	91	91
AL	Georgia-Pacific Corp	10699	Georgia P	Choctaw	31	78
...	...	...	...	...	...	...
MN	Minnesota Power In	10686	Rapids En	Itasca	27	28
MN	Minnesota Power In	1897	M L Hibba	St Louis	73	123
MO	University of Missou	50969	University	Boone	6	91
MS	Weyerhaeuser Co	50184	Weyerhae	Lowndes	123	123

Fig. 10. Part of a long table where stub is three columns wide (the Python program reported four columns because it was tricked by the numerical values further to the right, and the max stub width was set to 4.)



M?ori Ethnic Group Population Summary(1)(2)	
1991–2006 Censuses	
Census ye	M?ori ethnic group population
1991	434,847
1996	523,371
2001	526,281
2006	565,329
(1) All figures are for the M?ori ethnic group census usually resident population.	
(2) Information is unavailable for the M?ori ethnic group prior to 1991. Information about	
M?ori collected in the censuses prior to 1991 was on the basis of descent/origin. From	
1991 onwards, ethnic group and M?ori descent have been collected separately.	
Note: This data has been randomly rounded to protect confidentiality.	
Individual figures may not add up to totals, and values for the same data may vary in different tables.	

Fig. 11. A single column table with lots of metadata above and below the table proper.

Table 2. Differences between the corrected critical cell files of the six subjects, and between subjects and automated analysis.

	S1	S2	S3	S4	S5	S6	S7	Auto
S1	0	11	10	28	11	14	12	13
S2	11	0	15	36	17	25	11	20
S3	10	15	0	33	15	22	14	16
S4	28	36	33	0	31	31	35	34
S5	11	17	15	31	0	20	15	11
S6	14	25	22	31	20	0	24	19
S7	12	11	14	35	15	24	0	5
Auto	13	20	16	34	11	19	15	0

Table 2 shows that human subjects, even among table "experts", often disagree on how to segment a table. Every subject made at least one outright mistake; other differences are due to lack of precise definitions. Between the experts, 5 or 6 disagreements involved tables similar to that of Fig. 9, where either one or two columns can be reasonably considered part of the row header. In several others (e.g. Figs. 10 and 11) the disagreement involved a blank row. None of these affect downstream analysis.

5. DISCUSSION

VeriClick improves the accuracy of table layout analysis without requiring excessive human time. With experience, we believe that an average speed of 12 tables per minute, or 720 tables per hour, is readily attainable. This brings within reach verifiable experiments on a few thousand tables. On an operational basis, 5 seconds per table corresponds to 4 cents per table under an assumption of $25 per hour operating cost. This may be cost-effective for

many applications. (These figures do not include the cost of automated harvesting of web tables, for which reliable methods are already available[13].)

Throughput per hour would of course improve if the extraction program made fewer mistakes, and therefore increase the ratio of confirmations to corrections. We believe that the current ~14% error rate of automated critical cell extraction can be halved (on similar data!) with relatively little programming effort. Progress beyond that would, however, require developing heuristics for many rare special cases. VeriClick can, however, be easily extended to additional critical cells for table titles and footnotes which will be useful for combining information from multiple tables.

Rather than making an effort in improving our heuristics through intuitive notions, we propose to experiment with machine learning techniques to improve critical cell extraction through feedback from normal interactive verification and correction. We have been successful in exploiting operator feedback in other visual recognition tasks[14, 15] and have also followed other researchers' progress in learning document classification and document recognition tasks from human corrections[16, 17]. The necessary "training" information, i.e., the critical cells chosen by the program and verified by the operator, is available in a simple format from the VeriClick log. The proposed features for machine learning are format characteristics of each cell and of its row and column neighbors. Examples of readily accessible features in CMS tables include alpha/digit distinctions, capitalization, decimal points and commas, indentation, within-cell punctuation, special symbols ($, #, %), and data frames for recognizing dates, monetary amounts, and so on.

"Green computing" means not wasting operator interactions. If we want to design systems that improve with use, then human corrections must be integrated into an operational feedback loop instead of relegated to mere post-processing,. VeriClick is a step in that direction.

ACKNOWLEDGMENTS

We gratefully acknowledge our collaborators David Embley (BYU), Sharad Seth (UNL), Dan Lopresti (Lehigh), and Mukkai Krishnamoorthy (RPI) as our main source of ideas about tables over the years. We also thank students P. Jha, R. Padmanathan, R. Jandhalaya (RPI), D. Jin (UNL), and S. Machado (BYU) for their important contributions to refining and testing these ideas. The project was generously supported by the Rensselaer Open Software Foundation.

REFERENCES

[1] Lopresti, D., and Nagy, G., "A Tabular Survey of Automated Table Processing," Graphics Recognition: Recent Advances, Springer-Verlag, Berlin, LNCS 1941, pp. 93-120, (2000).

[2] Zanibbi, R., Blostein, D., and Cordy, J.R., "A Survey of Table Recognition: Models, Observations, Transformations, and Inferences," J. Doc. Anal. Recognit. 7(1), 1–16, (2004).

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

[4] Tijerino, Y. A. Embley, D.W., Lonsdale, D.W., and Nagy, G., "Towards Ontology Generation from Tables," World Wide Web Journal, vol. 6(3), 261-285, (2005).

[5] Krüpl, B., Herzog, M., and Gatterbauer, W., "Using Visual Cues for Extraction of Tabular Data from Arbitrary HTML Documents," Procs. of the 14th Int'l Conf. on World Wide Web, 1000-1001, (2005).

[6] Pivk, A., Ciamiano, P. , Sure, Y., Gams, M. Rahkovic, V., and Studer, R., "Transforming arbitrary tables into logical form with TARTAR," Data and Knowledge Engineering 60(3), 567-595, (2007).

[7] Jandhyala, R.C., Nagy, G., Seth, S., Silversmith, W. Krishnamoorthy, M., and Padmanabhan R.K., From "Tessellations to Table Interpretation," L. Dixon et al. (Eds.): Calculemus/MKM 2009, Springer-Verlag, Berlin, vol. 5625 LNCS, 422-437, (2009).

[8] Embley, D.W. Krishnamoorthy, M., Seth, S., and Nagy, G. "Factoring WebTables," Procs. ACM EIA/AIE, Syracuse, NY: Modern Approaches in Applied Intelligence (Editors: G. Mehrotra, C. Mohan, J. C. Oh, and P. K. Varshney), (2011).

[9] Nagy, G. Seth, S., Embley, D. W., Krishnamoorthy, M., Jin, D., and Machado, S., "Data Extraction from Web Tables: the Devil is in the Details," Procs. ICDAR 11, Beijing, (2011).

[10] Jha, P. and Nagy, G., "Wang Notation Tool: Layout Independent Representation of Tables," Proceedings of International Conference on Pattern *Recognition XIX*, Tampa, FL (2008).

[11] Padmanabhan, R.K., Jandhyala, R.C., Krishnamoorthy, M., Nagy, G., Seth, S. and Silversmith, W. "Interactive Conversion of Web Tables," J.-M. Ogier, W. Liu, and J. Lladós (Eds.): GREC 2009, LNCS 6020, pp. 25–36, (2010).

[12] Wang, X., "Tabular Abstraction, Editing, and Formatting," Ph.D Dissertation, University of Waterloo, Waterloo, ON, Canada, (1996).

[13] Wang, Y. and Hu, J., "Automatic Table Detection in HTML Documents, in Web Document Analysis: Challenges and Opportunities," pp. 135-154, (2003).

[14] Zou, J. and Nagy G., "Human-computer interaction for complex pattern recognition problems," in Data Complexity in Pattern Recognition, pp. 271-286, M. Basu and T. K. Ho, Eds., Springer, (2006).

[15] Zou, J. and Nagy G., "Visible models for interactive pattern recognition," Pattern Recognition Letters Vol. 28, pp 2335-2342, (2007).

[16] Esposito, F., Ferilli, S., Di Mauro, N., and Basile, T.M.A. , "Incremental Learning of First Order Logic Theories for the Automatic Annotations of Web Documents," Procs. ICDAR-2007, Curitiba, Brazil, September 23-26, IEEE Computer Society, Los Alamitos, CA, (2007).

[17] Ferilli, S., Biba, M., Basile, T.M.A., and Esposito, F., "Incremental Machine Learning Techniques for Document Layout Understanding," Procs ICPR 19. Tampa, FL, IEEE Computer Society (2008).