G. Nagy, Learning the characteristics of critical cells form web tables, Procs. ICPR 2012.

# LEARNING THE CHARACTERISTICS OF CRITICAL CELLS FROM WEB TABLES

George Nagy
*Rensselaer Polytechnic Institute, Troy, NY, 12180 USA*
*nagy@ecse.rpi.edu*

## Abstract

*Critical Cells (CCs) are identified to partition a web table into mutually exclusive regions of stub, column header, row header, data, and neutral cells. Every table cell (including titles and footnotes outside the table proper but usually within the HTML table tags) is classified into one of six classes based on cell-features extracted from the target cell and its eight neighbors. Changing the domain of maximization over posteriors results in the assignment of exactly four CCs to each table. The average number of interactions required for error-free table data extraction can be reduced more than 75% by alternating between graphic interaction and auto-assignment.*

## 1. Introduction

Most published tables meant for human access (i.e., *visual tables* as distinguished from *relational tables*) are well-formed tables (WFTs) with a rectangular layout and clearly defined header and data regions [1]. The location, extent and structure of any WFT can be specified by four Critical Cells (CCs) that demarcate the boundaries between data cells, row and column headers, and surrounding matter like the table title and footnotes (Fig. 1). We show below how to modify conventional statistical pattern recognition algorithms for mapping table layouts (as opposed to the dominant practice of syntactic and structural table analysis).

We have demonstrated earlier that header paths representing the relationship between column/row headers and data cells be determined from the locations of the CCs. This relationship suffices for converting tables meant for human visual access to relational tables and for extracting *facts* (i.e., RDF triples).

The CC recognizer that we describe below is a significant improvement of our end-to-end system for web-table interpretation. The CC subsystem consists of cell-feature extractors, a table-cell classifier, and a graphic user interface for user entry or corrections to teach the classifier the characteristics of the CCs of already seen tables. Our VeriClick user interface accepts correct tables with a single click, which also prompts the display of the next table. Wrong CCs can be corrected in 6 seconds per table on the average. Over 75% of new tables processed with the learned CC characteristics require no user correction, only rapid validation.

.

| 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. 1. 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 column header roots like “Age” and “Gender”.

In Section 2 we give an overview of prior developments in table analysis. Section 3 describes the cell features used for classification. Section 4 explains the learning algorithm. Section 5 presents our experimental design and experimental results.

## 2. Prior Work

The earliest published research on table processing sought to find the underlying grid structure of scanned tables from rulings or text alignments [2,[3] and of ASCII tables in email [4]. Medium-independent table detection was also proposed [5]. Subsequently attention shifted to the detection, location and analysis of HTML web tables [6,7 8,9,10]. The first attempt that we are aware of to detect handwritten tables is [ 11 ]. Comprehensive but no longer up-to-date reviews can be found in [12] and [13]. Many table analysis procedures are similar to those developed for commercially far more important *forms processing* (although most bureaucratic forms are not WFTs).

Recently a large and innovative Google-based team has harvested millions of web tables [14,15,16]. They deem volume more important than accuracy and are therefore developing wholly automated methods of integrating table data into a form suitable for queries. Unlike table pioneer X. Wang [17] and us, they base their analysis on an essentially asymmetric view of the row and column structures of visual tables. Like us, they can validate their experimental results only on a small fraction of their table corpus.

None of the work cited above provides trainable algorithms to partition tables into *stub, row-header, column-header* and *data cell* regions, or addresses explicitly the relationship of possibly hierarchical row and column headers to data cells through *complete header paths*.

Our own recent research focused on the extraction of the relations of header cells to data cells and the representation of these relations by appropriate data structures. Such multi-dimensional indexing is necessary for interpreting individual tables and for querying their combined contents from a database or a populated ontology, as proposed in TANGO [18].

We introduced the methods of extracting and factoring header paths and creating RDF representations in [1] and [19], We showed that the erroneous location of Critical Cells can be corrected efficiently by VeriClick interaction in [ 20 ]. Our contribution here is a trainable table CC location algorithm for the above system.

## 3. Cell Features

The design of cell features is based on the following observations. Each row of a table's column headers and each column of its row headers tend to share the same format. Data cells often share format across both rows and columns. Table titles and footnotes usually have only one non-blank cell that contains several words. Cell-classification features should therefore reflect the formats that differentiate various regions of the table.

Most of the features should be mutually exclusive. Their probabilities of occurrence should vary widely among the cell regions and types of Critical Cells. The features should accommodate anomalies that may occur in the conversion of HTML to CSV (Comma Separated Variables) format and its downstream interpretations. For example, empty cells can appear as *NaN,* as *null* ("″"), or as one or more *space* characters.

Each cell of every table, in both training and test sets, is represented as a row vector of 9M elements, where $M$ is the number of features ($M$=7 in the experiments reported below). The first $M$ elements are the features of the cell, and the remaining 8M are the features extracted from its eight neighbors, clockwise from the North (i.e., the base features of the neighbors of cell $(r,c)$ in table $t$ are replicated in the feature vector for that cell). We chose the following base features:

$x_{1,r,c,t}$ empty or blank;
$x_{2,r,c,t}$ number of characters in the cell
$x_{3,r,c,t}$ non-integer numbers (string or numeric)
$x_{4,r,c,t}$ only letters
$x_{5,r,c,t}$ mixed alphanumeric
$x_{6,r,c,t}$ integer (string or numeric)
$x_{7,r,c,t}$ original format numeric

## 4. Critical Cell Assignment

For experiments on training and classification it is convenient to transform all information for an entire set of trebles into a single array. For one particular set of 200 tables, the VeriClick ("ground truth") input CSV table and the features extracted from all 30795 table cells are transformed into a single 30795×(9$M$+4) element array. The first four columns contain the table ID, cell row and column, and CC label.

The possible values of the CC labels are 0 to 5. Zero is used for non-critical cells (the vast majority of all cells), 1 to 4 for CC1 to CC4, and 5 means that CC1 and CC2 are collocated (single-cell stubs where that happens are common). The CC labels from VeriClick are used as *class labels* for training and as *ground truth* for determining the error rate on test set.

The cells are assigned to one of the six possible classes by a linear Bayesian classifier with *pooled covariance matrix* because earlier studies suggest that the class-conditional covariance matrices are determined mainly by the nature of the features rather than by the interclass differences between patterns [21]. For example, in every CC class, integers and letters are negatively correlated, while integers and numerics are positively correlated. Furthermore, one or

more of these matrices may be singular when there are few samples. Training sets have thousands of non-CCs, but only a few dozen samples of CCs. Classification priors are chosen accordingly. The classification formula and output are described more formally below.

*Tables and cells; features; training, and test sets:*

$t = t_1, t_2, \ldots, t_{Ntables}$ (list of tables)
$r = 1,2, \ldots, r(t)$ (r is the row index of table *t*)
$c = 1,2, \ldots, c(t)$ (c is the column index of table *t*)
$m = 1,2, \ldots, M$ (index of elementary features)
$u = 0, 1, \ldots, 5$ (cell labels and types of cells)

$[s(1), s(2), \ldots, s(Ntables)]$ is a random permutation of *t*

$D_{train} = \{t_{s(1)}, t_{s(2)}, \ldots, t_{s(Ntrain)}\}$ is a permuted training set
$D_{test} = \{t_{s(1)}, t_{s(2)}, \ldots, t_{s(Ntest)}\}$ is a permuted test set

$$\mathbf{x}_{r,c,t} = ( x_{1,r,c,t}, x_{2,r,c,t}, \ldots, x_{m,r,c,t}, \ldots, x_{M,r,c,t};\ x_{1,r-1,c,t}, x_{2,r-1,c,t}, \ldots, x_{M,r-1,c,t};\ x_{1,r-1,c-1,t}, x_{2,r-1,c-1,t}, \ldots, x_{M,r-1,c-1,t};\ \ldots\ x_{1,r+1,c+1,t}, x_{2,r+1,c+1,t}, \ldots, x_{M,r+1,c+1,t} )$$

is the feature vector for cell (r,c) of table *t*,

*Classification:*

*Input: Ntrain CC-labeled tables, Ntest unlabeled tables*

$P_{r,c,t,u} = P[\mathbf{x}_{r,c,t} \mid u, D] \bullet p_u$ ($p_u$ = prior for CC type *u*)

*Decision*:

$$(r_{u,t}, c_{u,t}) = \underset{(r,c)}{\text{argmax}}\ P_{r,c,t,u} \text{ (CC of type } u \text{ at } (r,c) \text{ of } t)$$

$$(r_{1,t}, c_{1,t}) = (r_{2,t}, c_{2,t}) = (r_{5,t}, c_{5,t}) \text{ (single-cell stub)} \quad [1]$$
$$\text{if } P_{r,c,t,5} > \max(P_{r,c,t,1}, P_{r,c,t,2})$$

*Output:*

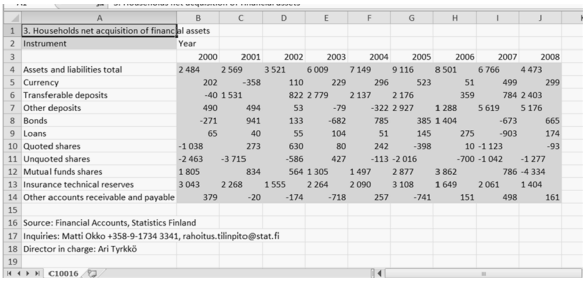
VeriClick representation of CCs of table *t*:

$t, (r_{1,t}, c_{1,t}), (r_{2,t}, c_{2,t}), (r_{3,t}, c_{3,t}), (r_{4,t}, c_{4,t})$

e.g. *145, A3, A4, B6, G18* in Excel A1 format

The classifier assigns to each table the CC of each of the four types with the highest posterior probability in that table. This is a twist from *local* cell classification because maximization here is over cells (*r,c*) of the table, with class *u* held constant. Equation [1] shows the special provision for *u*=5, collocated CC1 and CC2 (as in Fig. 2). The search for the most probable CC of each type (except *u*=0, i.e., non-CCs) is constrained by its expected location. The search region for CC1, CC2, CC3, and CC5 is limited to the top-left part of the table, while the search for CC4 is limited to the right edge.

The last step is exporting the new-labels for test tables to a CSV file in VeriClick format for interactive error correction. The test tables and labels can then be added to the previous training set in order to retrain the classifier for a new batch of tables. Processing 200 tables takes about 10 seconds on a 2Ghz laptop.



| | 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 | | | | | | | | | | |

Fig. 2. Partial VeriClick display of a table with a single-cell stub ("Instrument" in cell A2). The output for this table in VeriClick format is "C10016 A2 A2 B4 J14".

## 5. Experiments

200 tables were drawn from a set of 1000 tables collected earlier from large statistical websites in the US and abroad [19]. For each experiment, the order of the tables was randomly permuted. The first *Ntrain* tables were taken as the training set, and the last *Ntest* tables as the test set. We used different seeds for the random permutation for five-fold cross-validation.

Table I. Classification results for 3705 table cells.

| | | Assigned labels | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **Total** |
| | **0** | 3590 | 2 | 7 | 18 | 7 | 12 | 46 |
| | **1** | 1 | 8 | 0 | 0 | 0 | 0 | 1 |
| | **2** | 0 | 0 | 8 | 0 | 0 | 1 | 1 |
| GT | **3** | 1 | 0 | 0 | 19 | 0 | 0 | 1 |
| | **4** | 1 | 0 | 0 | 0 | 19 | 0 | 1 |
| | **5** | 1 | 0 | 0 | 0 | 0 | 10 | 1 |
| | **Total** | 4 | 2 | 7 | 18 | 7 | 13 | 51 |

Table I shows the results of table-independent (local) *classification* of the 3,705 cells of 20 test tables after training on 27,090 cells of 180 CSV web tables. The diagonal corresponds to correctly classified cells. The rightmost column and bottom row show the errors for each cell type and for each classification category.

The overall error rate is only 51/3705 = 1.4%. Nevertheless the number of data cells mislabeled as CCs is more than the actual number of CCs. The amount of VeriClick interaction necessary to correct the tables is directly proportional to the fraction of incorrectly assigned CCs.

Table II reports the number (out of 4×20) of correctly *assigned* CCs in the same test set of 20 tables as a function of the size of the training set. Over 180 test tables trained on only 20 tables, on average 80.5% of the CCs were labeled correctly (STD = 3.5%). The averages were taken over five randomly selected and mutually exclusive training and test tables, which resulted in an acceptably low STD.

Table II. Effect of size of training set.

| Ntrain | CC1 | CC2 | CC3 | CC4 | total |
|---|---|---|---|---|---|
| **20** | 18 | 15 | 17 | 19 | 69 |
| **50** | 17 | 17 | 17 | 19 | 70 |
| **80** | 18 | 18 | 17 | 19 | 72 |
| **140** | 19 | 20 | 17 | 19 | 75 |
| **180** | 19 | 20 | 17 | 19 | 75 |

## 6. Conclusion

We reported experimental results on a critical component of an end-to-end web table data retrieval system. The posterior probability matrices are generated by a standard classifier to avoid having to tune parameters. CC assignments are based on column instead of row maxima. We used "obvious" features but believe that including base features of neighboring cells for classification is novel, simple, and effective.

Earlier we applied static, syntactic cell-analysis to accomplish the same objective. The proposed trainable method benefits from interaction to train or retrain the classifier. Therefore results improve with use.

Further reduction of interactive corrections could be achieved with semi-supervised (adaptive) learning algorithms on sets of same-source (*isogenous*) tables.

The timing logs indicate that the proposed method is already four to five times faster than interactive labeling using VeriClick [20] on unprocessed tables (c. 15s/table), achieving a saving of 75%-80%. Entirely automated methods like those sought by the Google team are of course preferable in error-tolerant applications, but given the variety of table data conventions (and the number of formatting mistakes in the original tables) nearly error-free automatic table data extraction may still be many years away.

## References


[1 G. Nagy, S. Seth, D. Embley, M. Krishnamoorthy, D. Jin, S. Machado, Data Extraction from Web Tables: the Devil is in the Details, Procs. ICDAR 11, Beijing, 2011.

[2] T.A. Bayer, Understanding structured text documents by a model based document analysis system,: Procs. ICDAR'93, pp. 448–453, Tsukuba 1993.

[3] J.C. Handley, Table analysis for multiline cell identification. In:Kantor, P.B., Lopresti, D.P., Zhou, J. (eds.) Procs. DRR VIII (IS&T/SPIE ElectronicImaging), vol. 4307. San Jose, CA, 2001.

[4] D. Pinto, A. McCallum, Wei, W.B. Croft, Table extraction using conditional random fields, Procs. ACM SIGIR Conference on Research and Development in Informationn Retrieval, pp. 235–242, 2003.

[5] J. Hu, R. Kashi, D. Lopresti and G. Wilfong, Medium-independent table detection, Procs. SPIE DRR VII , San Jose, California, January 2000.

[6] M. Yoshida and K. Torisawa and J. Tsujii, A method to integrate tables of the World Wide Web, Procs. Int'l Workshop on Web Doc't Analysis (WDA 2001), 31-34.

[7] Y. Wang and J. Hu, Detecting Tables in HTML Documents, 5th IAPR Int'l Workshop on Document Analysis (DAS'02), Princeton, USA, August 2002.

[8] B. Krüpl, M. Herzog, W. Gatterbauer, 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.

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

[10] `Y. Liu, K. Bai, P. Mitra, C.L. Giles, TableSeer: Automatic Table metadata Extraction and Searching in Digital Libraries, Procs. ICDI, Vancou ver, June 2007.

[11] J. Chen and D. Lopresti, Table Detection in Noisy Off-line Handwritten Documents, Procs. ICDAR, Beijing 2011.,

[12] R. Zanibbi, D. Blostein, J.R. Cordy, A survey of table recognition: Models, observations, transformations, and inferences, J. Doc. Anal. Recognit. 7(1), 1–16, 2004.

[13] D.W. Embley, M. Hurst, M. Lopresti, G. Nagy, Table processing paradigms: A research survey, J. Doc. Anal. Recognit. 8(2-3), 66-86, 2006.

14 W.J. Cafarella, A. Halevy, D.Z. Wang, E. Wu, Y. Zhang, WebTables: Exploring the Power of Tables on the Web, Procs. VLDB '08 Auckland, New Zealand.

[15] A. Halevy, P. Norvig, and F. Pereira, The Unreasonable Effectiveness of Data. IEEE Intelligent Systems, March/April 2009.

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

[17] X. Wang, Tabular Abstraction, Editing, and Formatting, Ph.D Dissertation, University of Waterloo, 1996.

[18] Y. A. Tijerino, D.W. Embley, D. W. Lonsdale, and G. Nagy, Towards ontology generation from tables, World Wide Web Journal, vol. 6(3), 261-285, 2005.

[19] D. W. Embley, M. Krishnamoorthy, G.. Nagy, S. Seth, Factoring Web Tables, LNAI 6703, Springer, 253-263, 2011.

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

[21] G. Nagy and X. Zhang, Simple statistics for complex features spaces, Data Complexity in Pattern Recognition, pp. 173-195, M. Basu & T. K. Ho, Eds., Springer, 2006.