Notes on Contemporary Table Recognition

David W. Embley[1], Daniel Lopresti[2], and George Nagy[3]

[1] Computer Science Department,
Brigham Young University, Provo, UT 84602
embley@cs.byu.edu
[2] Department of Computer Science and Engineering,
Lehigh University, Bethlehem, PA 18015
lopresti@cse.lehigh.edu
[3] Department of Electrical, Computer, and Systems Engineering,
Rensselaer Polytechnic Institute, Troy, NY 12180
nagy@ecse.rpi.edu

Abstract. The shift of interest to web tables in HTML and PDF files, coupled with the incorporation of table analysis and conversion routines in commercial desktop document processing software, are likely to turn table recognition into more of a systems than an algorithmic issue. We illustrate the transition by some actual examples of web table conversion. We then suggest that the appropriate target format for table analysis, whether performed by conventional customized programs or by off-the-shelf software, is a representation based on the abstract table introduced by X. Wang in 1996. We show that the Wang model is adequate for some useful tasks that prove elusive for less explicit representations, and outline our plans to develop a semi-automated table processing system to demonstrate this approach. Screen-snaphots of a prototype tool to allow table mark-up in the style of Wang are also presented.

1 Introduction

Tables have long been widely used for *presenting* structured information just as printed forms are widely used for *collecting* structured information. Few scientific papers are considered complete without a table or two. There are several government agencies whose main product is tables. Because most of us prefer to "point-and-click" instead of trudging over to the library, tables available on the web are of most use and interest.

We note parenthetically that a successful solution to the table recognition problem will hasten the disappearance of tables, which are in any case an endangered species. Traditional railroad and airplane schedules have already been replaced by on-line Q/A forms. Five-place trigonometric tables were supplanted by ten-digit hand-held calculators. The fat volume of values of the binomial distribution for various n's, k's, and p's is also gone. The British Royal Commission on Mathematical Tables disbanded almost fifty years ago.

Old tables may appear on the web in scanned bitmap form, as in back issues of the *Transactions* in the IEEE Digital Library. Current information, however,

is more likely to be posted as PDF or HTML files, and XML tagging is making headway. Archival material is being gradually converted, most often by manual data entry. Aside from the shift from raw bitmaps to coded forms, an important development is the rapid incorporation of table analysis routines into Microsoft and Adobe software.

Earlier researchers were handicapped by having to build complete low-level table processing software, including OCR, before they could even think about putting table recognition to some use. Most of their hard work yielded only intermediate results. Our purpose here is to initiate discussion on whether we are now in a position to incorporate table recognition in operational applications.

The paper is organized as follows. After describing our view of low-level table processing, we will suggest that this step can now be circumvented by judicious us of common "office" software.[4] The inter-conversion between tables in HTML, PDF, XLM, XLS and DOC files is demonstrated. We then discuss possible table representations for bridging the semantic gap that has so far kept table recognition from being incorporated into routine information retrieval and data extraction applications. The foundations of a *table ontology* for organizing our current understanding of table processing techniques and tools are then sketched. We conclude by briefly outlining our plans to develop a semi-automated table processing system to demonstrate the approach we are proposing, and by describing a prototype tool we have developed for capturing table mark-up in the style of Wang.

2 Low-level table recognition

Low level table recognition implies table representations that allow the formal manipulation of tables without any real understanding of their contents. The "intelligence" comes entirely from the user, who is able to interpret the table in the context of previously acquired knowledge. In Table 1, it is easy to tell that the average "hepth" of "fleck" is approximately 233 "gd," but this does not readily connect with any other piece of generally known information.

Table 1.

fleck	gonsity (ld/gg)	hepth (gd)
burlam	1.2	120
falder	2.3	230
multon	2.5	350

Table 2.

goldam	1.3 ld/gg	320 gd
falder	2.3 ld/gg	230 gd
elmer	2.9 ld/gg	350 gd

[4] The functionality we will be illustrating is by no means limited to Microsoft Office, although we use applications from that software suite to illustrate our discussion.

Even low-level table recognition requires some analysis. Operations such as computing the average of a row or column of values require, at the minimum, a *geometric model* (i.e., a table frame, grid, array model).

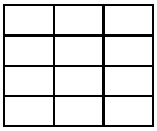

Fig. 1. Array models for Tables 1 and 2.

For Tables 1 and 2, the model is simple, as shown in Fig. 1. Other tables, with spanning top or side headers, may require merging cells. While all WFT (well-formed table) layouts are topologically equivalent to an array model derived by merging sets of adjacent cells in a regular grid, some models, like the one on the far right in Fig. 2, seem implausible because larger cells are generally above or to the left of smaller cells. We note, however, that a table author may indeed choose to use such a layout to emphasize that a set of columns share a common value in a given row.

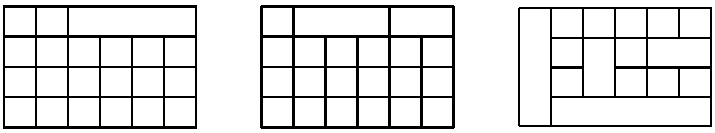

Fig. 2. Array models for more complex tables.

Deriving the appropriate array model for a table with multi-line cells without complete rulings has proved difficult. In the last 15 years, over 200 research papers have been published on table recognition [1–5]. Most published algorithms for cell alignment treat the table as a 2-D array of cells, and attempt to identify the coordinates and contents of each cell. Methods vary depending on whether the table is in scanned bitmap or coded (ASCII, HTML, RTF, PDF) form, ruled, partially ruled or unruled, and on the amount of prior information (including both "external" knowledge base and training data) available. Furthermore, some methods make use only of the layout geometry, while others bring in font, style, lexical and syntactic information extracted from the textual cell contents.

3 Commercial software

We will demonstrate some of the content-preserving table conversion routines that have been built into Microsoft and Adobe desk-top software. Consider the

table appearing in Fig. 3, *Country Data Codes*, rendered by Microsoft Explorer. (FIPS 10-4 codes are intended for general use throughout the US Government. ISO 3166 codes are activities involving exchange of data with international organizations. The Internet country code is the two-letter digraph used by the Internet Assigned Numbers Authority, IANA.) Recent versions of Microsoft Word are able to copy this table it into a .doc format file without evident loss of information, as seen in Fig. 4. Word is more accurate in this respect than Wordpad, we have found.

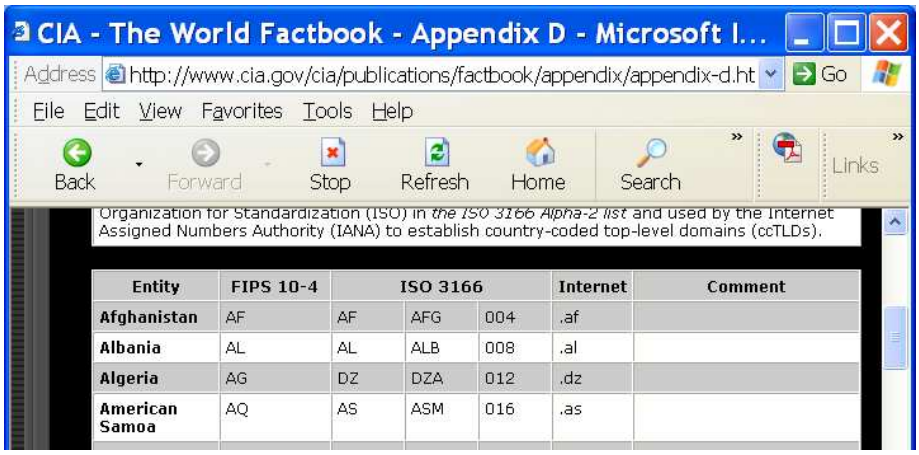

Fig. 3. Table as rendered by Microsoft Internet Explorer 6.0.

Entity	FIPS 10- 4	ISO 3166			Internet	Comment
Afghanistan	AF	AF	AFG	004	.af	
Albania	AL	AL	ALB	008	.al	
Algeria	AG	DZ	DZA	012	.dz	
American Samoa	AQ	AS	ASM	016	.as	

Fig. 4. Table copied into Microsoft Word 10.2.

From MS-Word, any table can be loaded into MS-Excel, which is a reasonable choice for either automated or interactive cell-level manipulation (cf. Fig. 5). Furthermore, Excel has built-in export routines for transferring the table to a database management system (Microsoft Access).

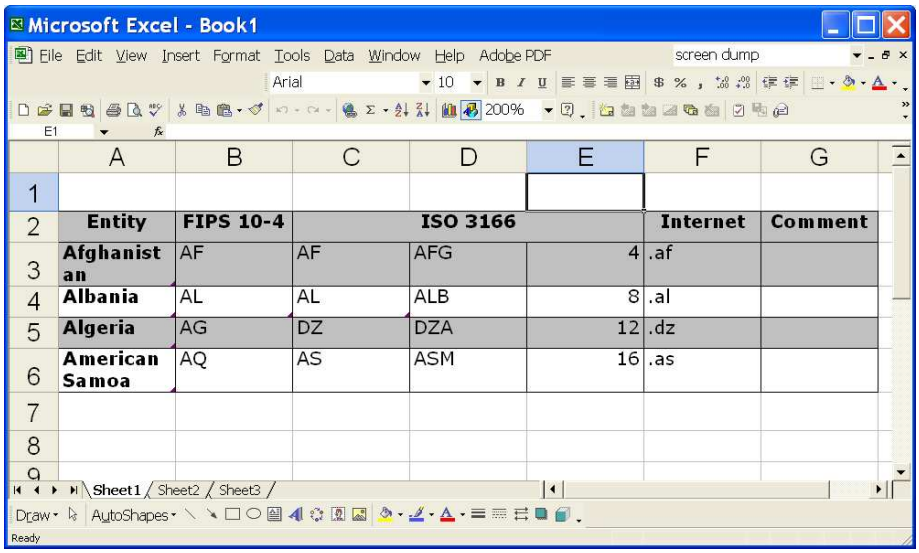

Fig. 5. Table copied from Microsoft Word into Microsoft Excel 10.2.

If the table appeared originally as ASCII text, as it might in an email, it would look like Fig. 6. This tab-separated text file can also be loaded into the spreadsheet program without loss of any layout information.

```
Entity          FIPS 10-4    ISO 3166     Internet   Comment
Afghanistan     AF    AF     AFG   004    .af
Albania         AL    AL     ALB   008    .al
Algeria         AG    DZ     DZA   012    .dz
American Samoa  AQ    AS     ASM   016    .as
```

Fig. 6. ASCII version of the table.

Another popular file format is Adobe's Portable Document Format, which is essentially enhanced PostScript. Fig. 7 was extracted from a PDF document posted on the website of the Bureau of Labor Statistics. It was then saved using the Adobe Acrobat "table picker" function with the XML tags added. We have not yet been able to automate the recovery of table structure from either PDF or XML, however the prototype tool we shall describe later provides support to facilitate human mark-up of the structure of tables encoded in HTML. Some researchers claim that recovering layout from PDF is done most easily from a rendered pixel map!

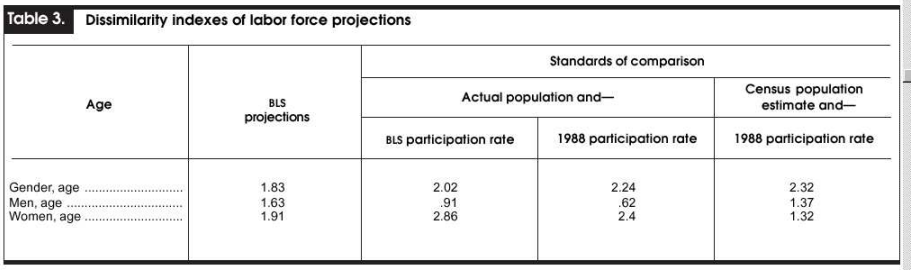

Table 3. Dissimilarity indexes of labor force projections

Age	BLS projections	Standards of comparison		
		Actual population and—		Census population estimate and—
		BLS participation rate	1988 participation rate	1988 participation rate
Gender, age	1.83	2.02	2.24	2.32
Men, age	1.63	.91	.62	1.37
Women, age	1.91	2.86	2.4	1.32

Fig. 7. Table rendered from a PDF file.

4 High-level table recognition

Some of the important applications of high-level table recognition are:

- Recreate an *equivalent* table for human reading by changing the spacing or the font, or by interchanging the rows and columns.
- Enter the attribute-value pairs (e.g., {(*fleck*, *burlam*), (*gonsity,1.2*), (*hepth*, 120)}) into a relational database, and re-frame queries in SQL.
- Combine several tables, or extend a selected table by adding information from another table. For example, we could discover that the gonsity of fleck oltan is 2.7 ld/gg, and its hepth is 100 gd, and add this row to Table 1.
- Compare several tables to determine if all or some entries are identical.
- Create conceptual models or ontologies from a large number of tables with interlocking entries. Having seen Table 1, clever software could derive a new average hepth of fleck (300 gd) from Table 2, or an average for all five fleck (274 gd) by combining the two tables. We will discuss below the distant but tantalizing possibility of *machine understanding* of tables.

An issue that keeps recurring in our on-going surveys of table recognition [1, 3, 4] is what should be the output of a table interpretation program. Specific output formats differ widely, but many are equivalent to a spreadsheet where cell contents can be addressed by coordinates. The more advanced methods allow addressing each cell by its row or column header.

The low-level array model is inadequate for answering some queries. The array model is identical for Tables 3 and 4 below. In order to take advantage of the headers in Table 3, it is necessary to first recognize that they are headers and to incorporate them in a higher-level table representation. We therefore attempt to go beyond the current output conventions, along the lines laid out by X. Wang [6] (who was interested in the description and manipulation of tables as *abstract data types*, not in table recognition). In the case of spanning cells, the row and headers form multiple trees. This is captured neatly by the Wang notation. The number of tree-paths necessary to specify a content cell is called the *dimensionality* of the table. (The number of content cells, in contrast, is the *size* of the table.) We stop short of claiming "semantic" interpretation, which is a murky concept in our view.

Table 3.

fleck	gonsity (ld/gg)	hepth (gd)
burlam	1.2	120
falder	2.3	230
multon	2.5	350

Table 4.

burlam	1.2 ld/gg	120 gd
falder	2.3 ld/gg	230 gd
multon	2.5 ld/gg	350 gd
goldam	2.9 ld/gg	350 gd

Very informally, the Wang Model consists of two components (C, δ), where C is a finite set of labeled domains (or categories), and δ is a mapping from the tree paths labels (or headers) to the possible values. We give the flavor with two examples.

The Wang Model for Table 3 is:

$$C = \begin{cases} (fleck,\ \{(bulram, \phi),\ (falder, \phi),\ (multon, \phi)\}) \\ (characteristic,\ \{(gonsity, \phi),\ (hepth, \phi)\}) \end{cases} \tag{1}$$

$$\delta = \begin{cases} (\{fleck.burlam,\ characteristic.gonsity\}) \rightarrow 1.2 \\ (\{fleck.falder,\ characteristic.gonsity\}) \rightarrow 2.3 \\ \ldots \\ (\{fleck.multon,\ characteristic.hepth\}) \rightarrow 350 \end{cases} \tag{2}$$

Note that "characteristic" does not appear in the table at all: we had to invent it in order to provide a root (i.e., a spanning label) for the column-header tree paths to the content cell entries! We call such imputed headers *implicit headers*. The absence of explicit headers is the major difference between high-level and low-level table interpretation. The experienced reader is able to infer them, but it is not easy to devise a robust algorithm that can infer them. The situation is even worse in Table 4, where both implicit headers are missing.

Consider now the more complex table shown in Fig. 7. Here we need a spanning label for "BLS projections" and "Standards of comparison." Let us call this "BLS/STD." Then there are three categories:

$$C = \begin{cases} (Age,\ \{(Gender\_age, \phi),\ (Men\_age, \phi),\ (Women\_age, \phi)\}) \\ (BLS\ projections, \phi) \\ (BLS/STD\ (Standards\ of\ comparison\ (Actual\ population\ and- \\ \quad \{(BLS\ participation\ rate, \phi),\ (1988\ participation\ rate, \phi)\}))) \\ (BLS/STD\ (Standards\ of\ comparison\ (Census\ populations \\ \quad estimate\ and-,\ \{(1988\ participation\ rate, \phi)\}))) \end{cases} \tag{3}$$

Note that there are two identical column headers called "1988 participation rate." Therefore we need to differentiate the "actual population" and the "1988 participation rates" from the "census population estimates" and the "1988 participation rates." Not an easy table to understand for human or machine! Once the correct label structure is derived, however, the tree-path specifications of the content values are straightforward.

If the logical structure is described completely, i.e., all the headers are present and have been clearly identified, and the geometric structure is specified (say in Excel, with merged cells of the fundamental grid), then it is not difficult to devise an algorithm to obtain a Wang Model of the target table. Such a model fills most of the needs listed at the beginning of this section. It does not, however, imply any true understanding. For instance, even if we know that burlam has only about half the gonsity and hepth of falder, we may not know which is better suited for some purpose, or whether one is likely to be more expensive than the other.

5 External information

Useful information can often be derived from the text, graphics, or other tables in the same or related documents as the target table. Currently, this is considered outside the domain of table processing, though several research papers have began to explore the topic [7, 8]. Table captions are similar to nearby ancillary information. They do not directly affect the physical or logical structure of the table. Some of the tasks listed above can be readily performed without considering captions.

Footnotes are also often neglected. Wang considers this the greatest shortcoming of her model, because nearly half of the tables she collected had footnotes. From our perspective, however, a footnote is simply cell content that exceeds the physical size of the cell. Footnotes can refer to either header or content cells.

The really important external information is that which does not appear anywhere in the vicinity of the table, but forms part of the users knowledge base. It is possible, however, that such a knowledge base can be assembled from studying a large collection of diverse but related tables. This, in fact, is one of our long term research objectives [9–11].

6 Components of a table ontology

Table recognition is a fast-moving target. To keep up with current and future developments, we believe that it is time to assemble the various ideas and tools so that they can be utilized and updated effectively. We believe that the appropriate conceptual and organizational framework for this purpose is an ontology, because an ontology is capable of representing a very broad class of relationships and is essentially open to new constituents and relationships. Some of the components that should be included in any table ontology are listed below.

- Table spotting, location, isolation, demarcation and classification
- Recognition of within-document and external references, including titles, captions, footnotes and citations
- Frame or border detection (box surrounding table)
  - Vertical rules
  - Horizontal rules

  - Line thickness, style, color
- Layout analysis to recover underlying array model
  - Ruled, partially ruled, unruled tables
  - White-space analysis
  - Horizontal text-line segmentation and alignment (single and multiple lines)
  - Vertical text alignment (justification, indentation, centering)
- Font analysis
  - Type size
  - Typeface (at least font family), color
  - Style (bold, italic, sub/superscript)
  - Case (capitalization)
  - Code translation or OCR
- Determination of cell-content similarities and affinities according to
  - Geometry (alignment, size)
  - Typography (type size and face)
  - Lexical category (words, phrases, commensurable decimal/integer values, abbreviations, units)
  - Grammatical construct (part of speech)
  - Semantics (to the extent possible)
- Extraction and codification of cell contents
  - Common and proper nouns (dictionaries and directories)
  - Phrases
  - Numeric fields (cardinal, ordinal, interval, integer, decimal)
  - Common data types (date, time, address, telephone number, email)
  - Punctuation (ellipsis, parenthesis, hyphen, comma)
  - Special symbols ($, ditto marks, leaders)
- Construction of logical table interpretation as a Wang Model or equivalent data structure

7 Work in progress

In this paper we have laid the foundation for work that is still very much in progress. Our goal — which we shall demonstrate at least in part at the DAS workshop — is to explore an area of table understanding that has hitherto remained unexamined: the transformation of table data presented in a simple array model to its full Wang representation. This is perhaps best illustrated in Fig. 8, where we show the most common formats for encoding tables, transitions between formats that are adequately supported by current commercial software solutions (solid arrows), transitions that have been the subject of much past research (thin dashed arrows), and the research we are targeting (the thick dashed arrow in the lower center of the figure).

We are currently in the process of implementing a graphical mark-up tool to semi-automate data capture from web tables, transforming a table in cell

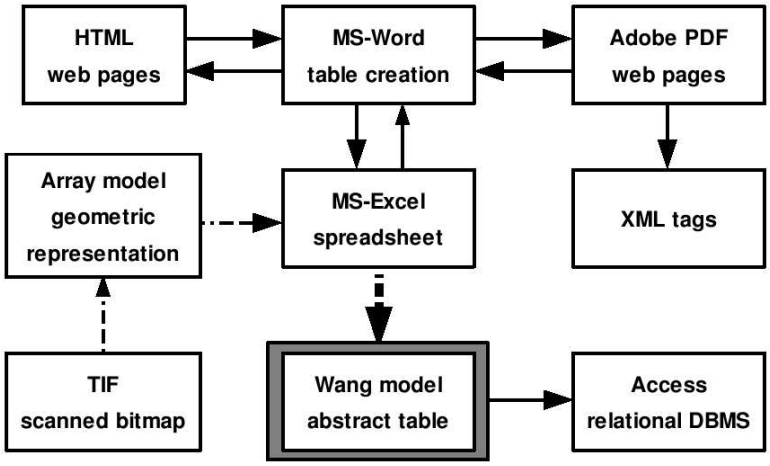

Fig. 8. Existing table conversion software. Solid arrows indicate available commercial software. Thin dashed arrows show focus of past research systems. Thick dashed arrow shows planned research.

array format into its Wang Model. Our operating hypothesis is that this approach, like past experience with Computer Assisted Visual Interactive Recognition (CAVIAR) [12, 13], will be both more accurate than a fully automated system while at the same time faster than an unaided human. This work is intended, of course, to be a first step towards the ultimate goal of autonomous table understanding.

Figures 9 and 10 show screen snapshots of our prototype which is written in Tcl/Tk, a popular scripting language for developing user interfaces. The table in Figure 9 should look familiar as it is simply the table from Figure 7 re-rendered in HTML, the input format used by our tool. The table in Figure 10 is the canonical example of a 3-D table used by Wang in her thesis [6].

Given an input table, the user first annotates the category structure, which consists of a set of trees specifying the row and column headers. Each logical dimension of the table corresponds to a single tree: the table in Figures 9 is 2-D, while Figure 10 shows a 3-D table. Note that every data cell in the table is uniquely specified (indexed) by a set of paths from the root of a tree to a leaf: one such path for each tree. After specifying the category trees, the user then populates the nodes by first clicking on the appropriate cell in the input table and then on the tree node that represents it. Once the category trees are completely populated, the mapping from sets of paths to individual data cells can be stepped through, one by one, with the user selecting at each step the appropriate cell in the input table. The tool then outputs the marked-up table in HTML format, with additional tags encoding the Wang structure.

While still just a prototype, this approach appears to be quite effective: the editing time for an expert user to create the annotation shown in Figure 9

was less than 30 seconds, requiring a total of 18 mouse-clicks. The larger, more complicated table in Figure 10, on the other hand, needed two minutes and 40 seconds and 116 mouse-clicks. We continue to work, of course, on refining our tool and plan more extensive user studies in the near future.

We close by citing some other recent developments that appear promising. The work by Pivk, et al. presents an approach for generating F-Logic frames from web tables which can then be used to populate ontologies [14]. They develop not only the methodology, but also an unusually thorough evaluation paradigm. Zanibbi, et al. present a language for representing table recognition strategies which exposes previously hidden assumptions buried within an implementation and offers to shed new light on the decision making process during table processing [15]. Both of these efforts relate to our own.

8 Acknowledgments

We wish to thank the anonymous reviewers whose comments helped make this a better paper. David W. Embley and George Nagy gratefully acknowledge the support of NSF grants #0414644 and #0414854.

References

1. Embley, D.W., Hurst, M., Lopresti, D., Nagy, G.: Table processing paradigms: A research survey. (2005) In submission.
2. Hurst, M.: The Interpretation of Tables in Texts. PhD thesis, University of Edinburgh (2000)
3. Lopresti, D., Nagy, G.: Automated table processing: An (opinionated) survey. In: Proceedings of the Third IAPR International Workshop on Graphics Recognition, Jaipur, India (1999) 109–134
4. Lopresti, D., Nagy, G.: A tabular survey of automated table processing. In Chhabra, A.K., Dori, D., eds.: Graphics Recognition: Recent Advances. Volume 1941 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany (2000) 93–120
5. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition: Models, observations, transformations, and inferences. International Journal on Document Analysis and Recognition **7** (2004) 1–16
6. Wang, X.: Tabular abstraction, editing, and formatting. PhD thesis, University of Waterloo (1996)
7. Douglas, S., Hurst, M., Quinn, D.: Using natural language processing for identifying and interpreting tables in plain text. In: Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR'95), Las Vegas, NV (1995) 535–545
8. Hurst, M., Douglas, S.: Layout and language: Preliminary investigations in recognizing the structure of tables. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR'97). (1997) 1043–1047
9. Embley, D., Tao, C., Liddle, S.: Automatically extracting ontologically specified data from HTML tables with unknown structure. In: Proceedings of the 21st International Conference on Conceptual Modeling (ER'02), Tampere, Finland (2002) 322–327

10. Embley, D., Tao, C., Liddle, S.: Automating the extraction of data from HTML tables with unknown structure. Data and Knowledge Engineering (2005) (in press).
11. Tijerino, Y.A., Embley, D.W., Lonsdale, D.W., Nagy, G.: Towards ontology generation from tables. World Wide Web Journal **8** (2005) 261–285
12. Zou, J.: Computer Assisted Visual InterActive Recognition. PhD thesis, Rensselaer Polytechnic Institute (2004)
13. Zou, J., Nagy, G.: Evaluation of model-based interactive flower recognition. In: Proceedings of the 17th International Conference on Pattern Recognition. Volume 2. (2004) 311–314
14. Pivk, A., Cimiano, P., Sure, Y.: From tables to frames. In: Proceedings of the Third International Semantic Web Conference (ISWC 2004). Volume 3298 of Lecture Notes in Computer Science. Springer Verlag, Hiroshima, Japan (2004) 166–181
15. Zanibbi, R., Blostein, D., Cordy, J.R.: The recognition strategy language. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, Seoul, South Korea (2005) 565–569

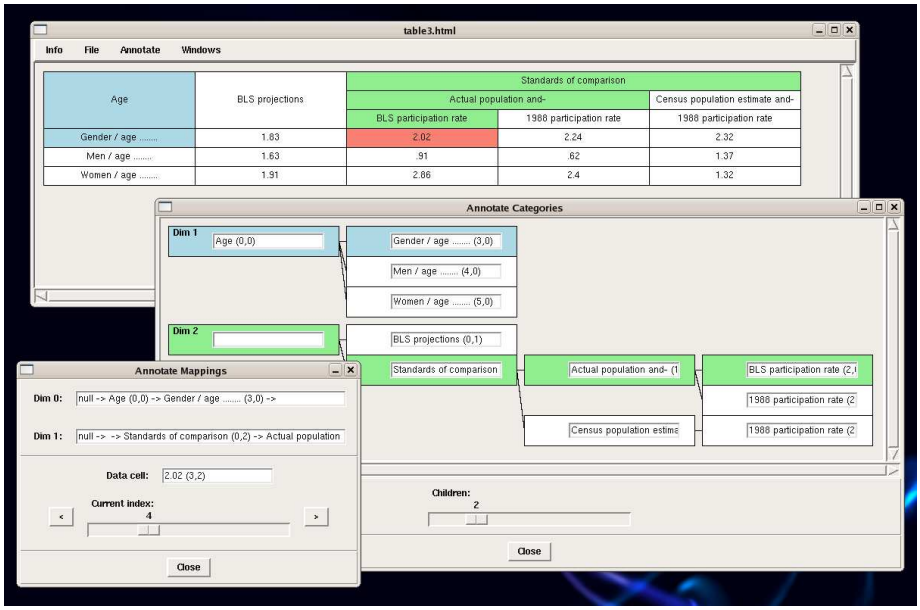

Fig. 9. Screen snapshot of a tool for supporting Wang-style table mark-up.

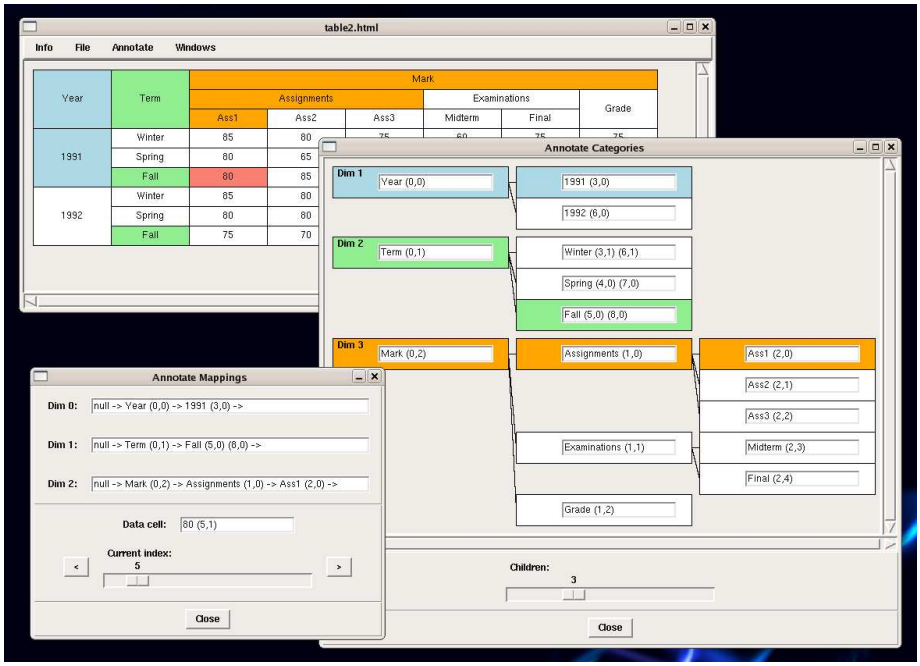

Fig. 10. Another example of Wang-style mark-up created using our prototype tool.