Learning Table Extraction from Examples

Ashwin Tengli, Yiming Yang and Nian Li Ma
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA - 15213
{tengli,yiming,manianli}@cs.cmu.edu

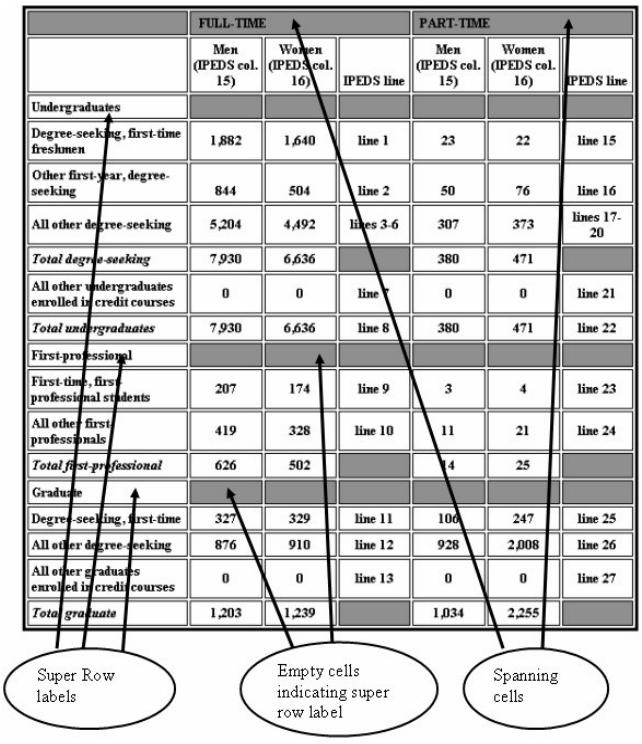
Abstract

Information extraction from tables in web pages is a challenging problem due to the diverse nature of table formats and the vocabulary variants in attribute names. This paper presents a new approach to automated table extraction that exploits formatting cues in semi-structured HTML tables, learns lexical variants from training examples and uses a vector space model to deal with non-exact matches among labels. We conducted experiments with this method on a set of tables collected from 157 university web sites, and obtained the information extraction performance of 91.4% in the F1-measure, showing the effectiveness of the combined use of structural table parsing and example-based label learning.

1. Introduction

Tables are an important means of presenting information and are widely used on the web for the reason that they show relational data in a form concise and easy to read by human eyes. Automatic extraction of tables has applications in information retrieval, knowledge acquisition, summarization and web mining. A table is made of cells. Each cell is either a label cell (attribute name) or a data cell (attribute value). The task of table extraction involves differentiating between the two types of cells and identifying the associations between labels and data.

Automatic information extraction from tables is a challenging problem since tables are designed for viewing by humans and not for understanding by computers. A table which is unambiguous for humans may still be difficult to parse by a computer. Figure 1 shows such an example: a table of student enrollment statistics (in a particular year) provided by a university. The layout of the table, the positions of the labels and the data, the meaning of the words, the shaded areas, and the contrasts between empty and non-empty cells together make the table readable and unambiguous for human understanding. However, it is not so obvious how a system can identify which cells are the attribute labels and which cells are the attribute values. The nested structure of labels makes the problem further complicated. For example, "Undergraduates" and "Degree-seeking, first-time freshmen" together define the row coordinate of a data cell, while "Full-Time" and "Men" together define the column coordinate of the data cell. Clearly, for tables as in the given example, relying on HTML tags along will not be sufficient to parse a table and to interpret the meanings of its cells. A better approach is to jointly use layout information (including HTML tags) and lexical constraints in the parsing of a table, and to learn those constraints automatically from training examples.



	FULL-TIME			PART-TIME		
	Men (IPEDS col. 15)	Women (IPEDS col. 16)	IPEDS line	Men (IPEDS col. 15)	Women (IPEDS col. 16)	IPEDS line
Undergraduates						
Degree-seeking, first-time freshmen	1,882	1,640	line 1	23	22	line 15
Other first-year, degree-seeking	844	504	line 2	50	76	line 16
All other degree-seeking	5,204	4,492	lines 3-6	307	373	lines 17-20
Total degree-seeking	7,930	6,636		380	471	
All other undergraduates enrolled in credit courses	0	0	line 7	0	0	line 21
Total undergraduates	7,930	6,636	line 8	380	471	line 22
First-professional						
First-time, first-professional students	207	174	line 9	3	4	line 23
All other first-professionals	419	328	line 10	11	21	line 24
Total first-professional	626	502		14	25	
Graduate						
Degree-seeking, first-time	327	329	line 11	100	247	line 25
All other degree-seeking	876	910	line 12	928	2,008	line 26
All other graduates enrolled in credit courses	0	0	line 13	0	0	line 27
Total graduate	1,203	1,239		1,034	2,255	

Figure 1. A table of student enrollment statistics in a university

Another challenging issue in table extraction is dealing with the formatting differences and lexical variants in the tables that have the same semantic structures. The Common Data Set (CDS) tables available on different university web sites are a good example. The Common Data Set (CDS) initiative is a collaborative effort among data providers in the higher education community and publishers to define a standard format for university data. However, the tables that appear on the university web sites differ greatly in terms of HTML formatting, lexical variants of labels, extra or missing portions in those tables, etc. Mapping the various forms of those tables in distributed web sites to the unified underlying structures (predefined) is indeed a non-trivial problem. We propose a simple and yet effective learning algorithm that learns lexical variants of attribute names from training examples, and that employs a vector space model to support "fuzzy" match between lexical variants and canonical forms of attribute names.

In our view this is the first work which provides a complete solution for extraction from HTML tables. We have not encountered any experimental evaluation of table extraction in the literature that can be used as a reference of comparison with our work. Therefore we have defined our own evaluation methodology. The research questions we address here for table extraction are: 1) how to use the layout information provided by the HTML tags, 2) how to learn from existing tables and predefined structures, 3) how to deal with lexical constraints of labels to improve the efficiency, and 4) how to evaluate the effectiveness of table extraction.

The rest of the paper has been organized as follows: In section 2 we discuss the related work done in this area. Section 3 describes our algorithm and Section 4 describes our experimental setup and evaluation results. In Section 5 we present the conclusions.

2. Related Work

Previous work in table extraction has addressed open domain tasks such as table extraction from ASCII, OCR and HTML documents plus closed domain tasks such as table detection, table merging and classifying rows in tables. Approaches such as deep understanding of the table structure and shallow parsing of the content have been used. Knowledge engineering techniques like heuristics based on the formatting cues and machine learning techniques like decision trees, SVM and conditional random fields have been explored previously. Most work was concentrated on extraction from ASCII tables with some work on extraction from images like OCR texts and from semi-structured formats like HTML.

The challenge in plain text ASCII tables is in interpreting structural information from spaces, tabs and ASCII character sequences like the '-' or '#' used for creating lines. While in the HTML tables the cells are already demarcated by the <td> tags. (Hurst and Nasukawa, 2000; Hurst and Douglas, 1997; Hurst, 2000; Pyreddy and Croft, 1997) describe methods for table extraction from plain and OCR scanned texts. The features used for extracting from tables in HTML are different and features such as spaces or tabs need not be used to identify structure.

There has been work done in the past on the task of table detection (Chen et al., 2000; Wang and Hu, 2002; Hu et al., 2000). The table detection task involves separating tables that contain logical and relational information from those that are used for formatting the layout. The input to a table detection system is documents containing both real tables and tables used for layout formatting. The output is a classification of the data into real tables and non-real tables. The task of table detection is sometimes the first step for performing table extraction because it needs real tables. However it not a necessary step as detection could be combined in the extraction algorithm. Heuristics and machine learning based approaches have been generally used to perform table detection.

(Chen et al., 2000) presents work on table detection and extraction on HTML tables. The table detection algorithm uses string, named entity and number category similarity to decide if it is a real or non-real table. Based on cell similarity the table extraction algorithm identifies whether the table is to be read row wise or column wise. They split the cells which span over multiple cells into individual cells. The table extraction algorithm presented in this work is simple and works only if spanning cells are used for nested labels. The paper did not provide evaluation results for their table extraction algorithm.

The problem of merging different tables which are about the same type of information has been addressed in (Yoshida et al., 2001). The merging task as defined in the paper considers combining different tables into one large table. They define different structures for tables based on the arrangement of the labels and use Expectation Maximization to classify the tables to one of the defined structures. The structure recognition task is similar to the classification task. However structure recognition or merging does not solve the table extraction problem.

Table extraction by wrapper learning has been explored in (Cohen et al., 2002). Wrappers learn rules based on examples. The rules are composed of tokens made of HTML tags or the content itself. The rules tend to be specific and can be applied only to those documents whose structure is similar to the training documents. The use of tokens to compose rules makes it difficult to generalize across distributed websites. Hence wrappers learned for one website cannot be used on another website. No clear evaluation for table extraction has been described this work.

Conditional random fields for table extraction from ASCII tables have been described in (Pinto et al., 2003). However the system described does not perform a complete table extraction task – it only classifies the rows of the table into a type such as "datarow", "sectionheader" or "superheader". They used a set of 12 types (classes) and achieve a precision of 93.5% for the classification task. Work presented in (Pyreddy and Croft, 1997; Pinto et al., 2002) also focused on the task of classifying table rows.

3. Methodology

The HTML format is a hierarchical markup language where the markup tags define the layout of the information. The <TABLE> tag is used in HTML documents to construct tables. The following tags are used in HTML to define tables:

- <table>….</table> - the tag enclosing the whole table.
- <tr>….</tr> - the tag that defines a row in the table.
- <td>…</td> - the tag that defines a column in each row. It identifies a cell in the table.
- <th>…</th> - the tag used instead of <td> to define that the information in the cell is a header information.
- <caption>…<caption> - the tag used to put caption for the table.

Often the <TABLE> tag is used for formatting the content in the web pages too. Therefore we use a table detection system to filter out the real tables from those used for formatting the layout. The program is developed by our group and uses decision trees to learn rules using features like cell similarity, number of cells and type of cells. It is a simplified version of the table detection algorithm using decision trees described in (Wang and Hu, 2002).

In spite of tags demarcating a cell boundary in HTML tables, it is still difficult to automatically identify the labels for each data cell, as the tables are designed for visual interpretation and not for automatic extraction. Figure 1 shows example of a table with hierarchical labels for the rows and spanning cells for column labels. Such tables though easy to read, are difficult to automatically assign correct labels for each data cell. Our algorithm processes the tables and brings it to a form where it is in a uniform two dimensional structure. There is no nesting of labels in the processed structure. In this form the data cells are separated from the label cells and are associated only with labels in the same row or columns. One of the important contributions of our work is learning labels and using them for structure recognition. In Section 3.1 we describe the method for learning the labels and in the next section we present our algorithm.

3.1 Learning Labels

Tables have two types of information: the label information and the data. The labels are the attributes of the data. If the some of the labels of these tables are known, then this information can be used to identify the structure of the tables, which in turn will help assign the right labels to the data cells. Since tables are a semi-structured form of representing data, the labels are present in a regular format where consecutive label cells appear either in a row or a column. Locating the known labels in a table will help differentiating the label rows and columns from the data rows and columns.

Our system is provided with tables as examples with labels in them marked. The label learning algorithm proceeds as follows:

1. The labels from the example tables are extracted and indexed.
2. Labels whose relative string edit distance is less than 0.09 are merged together. Relative edit distance is the ratio of number of edit operations (Levenstein, 1966) and the length of the string. Relative edit distance measure is not symmetric; therefore we consider the larger of the two relative edit distances.
3. A ranked list of these labels, obtained by thresholding on term frequency, is used by the extraction algorithm.

Use of relative edit distance limits the merging of lexical variants in labels to those whose edit distance to string length ratio is less than the threshold. The threshold for the relative edit distance is empirically set to a value that maximizes the performance. Identifying labels in a table will help separating the data cells from the label cells. For example, if 50% of the cells in a column are identified as labels then the rest of the cells in the column are also labels. This heuristic is used because there is a regularity in appearance of labels in tables, which implies that if most of the cells in a column are labels, then rest of the cells are labels too.

3.2 Algorithm for Table Extraction

The algorithm starts by parsing the content in the <TABLE> tag. It uses a set of heuristics in conjunction with the labels learnt from the examples, to recognize the structure. The examples of some of these heuristics are shown in the Figure 1. The heuristics used to recognize the structure are given below:

- Span tag: The <td> tags in HTML can contain the attribute '*span*' which specifies the number of row cells or column cells that the current cell spans. We recognize the span attribute in the tags and use this information to assign the spanning cell to multiple rows and columns.
- Rows with empty data cells: After separating the label cells from the data cells, we identify rows with empty data cells. If the previous data row is not empty and the next data row is also not empty, then the label of the current row is a super row label. A super-row label is one which spans over the row labels below it. Examples of super-row labels are shown in Figure 1.
- Single row cell in a row: If there is only one row cell in a row and that cell contains a label, then naturally the label is a super row label.
- Single empty data cell: If there is only one data cell in a row and the cell is empty then the label for that row is a super row label.

The algorithm proceeds as below:

1. Parse the table and read the contents into a array. Also store the attributes of the cell like '*span*' etc into another array. The cell *i,j* of the array corresponds to the cell *i,j* in the table. Use HTML Tidy or any other similar tool to handle errors in HTML. We use modules built in the Perl HTML parser to handle this.
2. For each cell process the HTML attributes like the '*span*' attribute. Split the spanning cell into as many cells as it spans.
3. Use the labels learnt to identify data cells and label cells. The vector space model is used for matching non-exact labels by considering each label as a vector of terms. To identify column labels parse each row. Match the contents of the cells in the row with the column labels learnt. If the similarity is greater than the threshold then set that row as containing column labels. Perform similar operation on the columns of the table to identify the columns containing the row labels. A threshold of 0.45 was used in our experiments. Separate the data cells from the labels cells and split the label cells into row label cells and column label cells.
4. Identify super-rows using the heuristics described previously. Perform partial matching with the super-row labels learnt. A relative string edit distance of less than or equal to 0.25 was considered as the threshold. Concatenate the super label with the label

cells below it until another super label is found or the end of the table is reached. Delete the row which contained the super label.

5. For each data cell extract the corresponding column labels and row labels. Output the results in XML format.

The extraction algorithm first separates the data cells from the label cells. It then expands the spanning cells. The super labels are identified and expanded appropriately. Then the result is outputted in XML format. Figure 2 shows extraction results from a table similar to the one in Figure 1.

```
<catlist filename="b1-evaluation/www.uiowa.edu#~provost#irm#cds#cds00.htm.3.html">
<Table category="newlabels-corr" id="1" year="2000">
<level0 >
<element rowlabel0="undergraduates" rowlabel1="degree-seeking, first-time freshmen" columnlabel0="full-time"
columnlabel1="men" >1,584</element>
<element rowlabel0="undergraduates" rowlabel1="degree-seeking, first-time freshmen" columnlabel0="full-time"
columnlabel1="women" >2,105</element>
<element rowlabel0="undergraduates" rowlabel1="degree-seeking, first-time freshmen" columnlabel0="part-time"
columnlabel1="men" >13</element>
<element rowlabel0="undergraduates" rowlabel1="degree-seeking, first-time freshmen" columnlabel0="part-time"
columnlabel1="women" >31</element>
</level0>
<level1 >
<element rowlabel0="undergraduates" rowlabel1="other first-year, degree-seeking" columnlabel0="full-time"
columnlabel1="men" >695</element>
<element rowlabel0="undergraduates" rowlabel1="other first-year, degree-seeking" columnlabel0="full-time"
columnlabel1="women" >630</element>
<element rowlabel0="undergraduates" rowlabel1="other first-year, degree-seeking" columnlabel0="part-time"
columnlabel1="men" >47</element>
<element rowlabel0="undergraduates" rowlabel1="other first-year, degree-seeking" columnlabel0="part-time"
columnlabel1="women" >52</element>
</level1>
```

Figure 2. Extraction results

4. Experiments

Since there has not been significant work in the area of table extraction from HTML, there have been no clear baselines. We define the baseline for our experiments as an algorithm which does not learn from examples and does not use the heuristics defined. For our experiments we collected from 157 university web sites HTML pages that contained tables. These tables are part of the Common Data Set and are semantically similar across websites. The Common Data Set organization defines specific format for data from universities in Unites States of America. The table detection program was used to identify the real tables from the non-real tables in the webpages. We used eight different types of tables and had two evaluation sets consisting of 193 and 55 tables. In Section 4.1 we describe the evaluation methodology. The experiments conducted and the results achieved are described in section 4.2.

4.1 Evaluation Methodology

The evaluation set consisted of 55 tables of 7 different types and 193 tables of the type B1 from the Common data Set. A user interface was created to annotate the tables manually. All the data cells in the tables were identified and each data cell was annotated with corresponding labels. The unit for evaluation is a data cell. The evaluation methodology is described in Figure 3.

In Figure 3, a 'yes' by the system or the human annotation means that all the labels for the data cell were identified correctly. The cell "a" is incremented by one if the human annotation and the system result agree completely on all the labels for a data cell. The cell "b" is added a score of one if the system extracted data cell is incorrect or its labels are incorrect. The cell "c" is incremented by one if the system misses a data cell identified by the human annotator. The cell "d" is ignored in the evaluation measure. The formulas for Precision, Recall and F1-measure are given in the Figure 3.

	Human Annotated Results		
Table Extraction Results		Yes	No
	Yes	a	b
	No	c	d

Precision P = a / (a+b)
Recall R = a / (a+c)
F1-measure = (2 * P * R) / (P + R)

Figure 3. Evaluation Methodology

The baseline used in our experiments does not use the heuristics and the labels learnt. This system identifies the first row and the first column of a table as label rows and columns respectively. The rest of the cells are assigned as data cells. For each of the data cells the cells in the beginning of the same row and column are selected as label cells. For example if the data cell is given by the index *i,j* then column label is the cell *1,j* and the row label is the cell *i,1*. If the cells *1,j* or *i,1* is empty then no label is assigned to the data cell.

4.2 Experiments Conducted

The first experiment was conducted on seven types of tables from the Common Data Set. The types of tables

selected were B2, C1, C5, C9-1, C9-2, C9-3 and H1. The tables of type C and B are simple tables with just the first row being labels or both the first row and first column being labels. The tables of type H1 were the most complicated type which had super labels for rows. These tables at times had lot of empty data cells which made recognition of super labels for the rows difficult. There were also unwanted labels that were not part of CDS and hence were not annotated. In our experiments we removed such labels. For this experiment the labels were learnt from the CDS definition.

Table Type	Baseline			Our System		
	Precision	Recall	F1	Precision	Recall	F1
B2	59.22	64.55	61.77	76.18	83.59	79.71
C1	0	0	0	100	100	100
C5	32.33	76.05	45.37	86.95	84.50	85.70
C9-1	68.57	100	81.35	100	100	100
C9-2	100	100	100	100	100	100
C9-3	66.6	100	79.95	100	100	100
H1	0	0	0	95.74	79.88	87.09

Table 1. Precision, Recall and F1 values on 7 types of tables

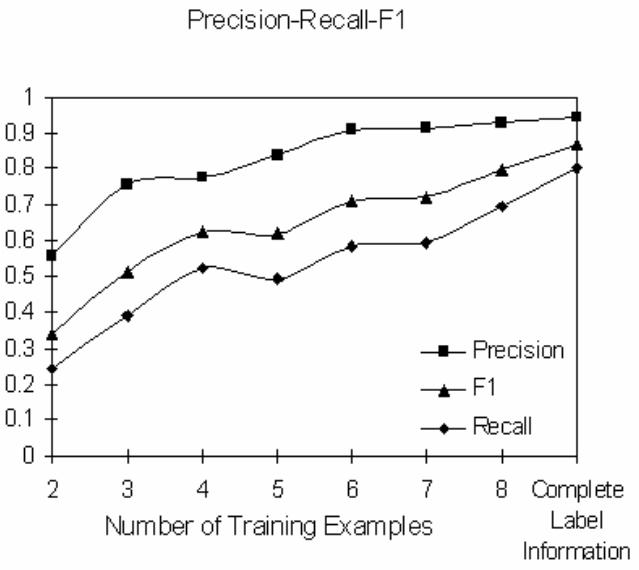


Figure 4. Precision, Recall and F1 values by selecting different number of examples

Table 1 shows the results of performance of the baseline system and our table extraction system. It can be seen that though the baseline system performs well on simple tables, the performance on complicated tables is poor. On the tables of type C1 the performance of the baseline was poor because there were no column labels but it assumed the first row to contain column labels. Our system achieved a combined recall of 87.84%, precision of 95.31% and F1-measure of 91.42%, based on the evaluation measure described in the previous section. Our system performed significantly better than the baseline which had a combined F1 score of 48.0%.

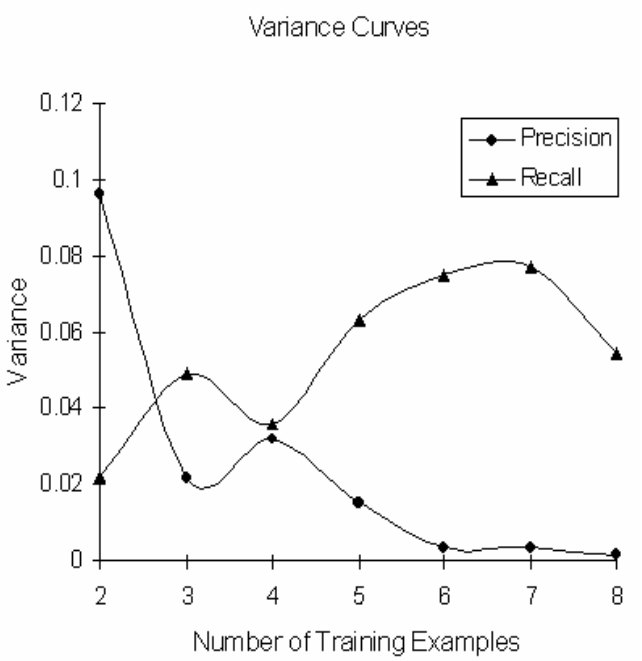


Figure 5. Variance of the Precision and Recall Values with the number of examples

The second experiment was conducted on the 193 B1 tables to evaluate the advantage of learning from examples. The B1 tables (example shown in Figure 1) have a complicated structure. They have nested column labels and super-row labels. In this experiment the precision, recall and F1 values were measured while increasing the number of examples. A pool of 10 examples was used for the experiments. The results of the experiments are shown in Figure 4. Each point on the x-axis corresponds to the number of training examples used for learning. The y-value is the mean of the scores of ten trails of randomly chosen examples from the set of $^{10}C_x$ possible combinations. Almost all the tables in our dataset had the complete label information. Therefore we induced incomplete label information into the ten examples by randomly removing some labels. This represents a

real world example where a single example table may not have complete label information and complete information is available by increasing the number of examples. Figure 4 shows the values of recall, precision and F1-measure for this experiment. The variance of the precision and recall has also been plotted separately in Figure 5. The performance of the system improves as the number of examples are increased. The performance saturates to the best performance. The best performance has a precision of 94.29%, recall of 80.04% and F1 measure of 86.76% when complete label information has been used. An extraction performance with high precision and good recall can be used to input the results of the extraction into a relational database. The results also show that the precision of our algorithm converges quickly and has low variance. Such a system is well suited to extract data that can be used as input to relational databases. Our system was used to extract data from the web pages of the 157 universities and store it into a relational database.

5. Conclusion

The work presented in this paper provides a simple and yet effective algorithm for performing the complete task of automatic table extraction, including the differentiation between labels and data, and the association of each data cell with the correct labels. Our approach leverages the formatting cues present in HTML tags, the knowledge about the semantic structures of tables, and the power of example-based learning. Our evaluation on a set of tables from 157 university web sites shows a performance level of 91.4% in the F1 measure. To our knowledge, this is the first evaluation on the complete task of table extraction, and the effectiveness of this approach is evident.

6. References

W. Cohen, M. Hurst and L. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In the Proc. of WWW2002, Honolulu, Hawaii, 2002.

D. Pinto, A McCallum, X. Wei and B. Croft. Table extraction using conditional random fields. In the Proceedings of SIGIR'03, Toronto, 2003.

M. Hurst. Language and Layout: Challenges for table understanding on the web. In web Document Analysis, In the Proc. of 1st International Workshop on Web Document Analysis, Seattle, 2001.

M. Yoshida, K. Torisawa and J. Tsujii. A method to integrate tables of the world wide web. In the Proc. of 1st International Workshop on Web Document Analysis, 2001.

H.-H. Chen, S.-C. Tsai, and J.-H. Tsai. Mining Tables from Large Scale HTML Texts. In the Proc. of 18th International Conference on Computational Linguistics, COLING , Saarbruecken, Germany, July 2000.

Yalin Wang and Jianying Hu. A Machine Learning Based Approach for Table Detection on the Web. In the Proceedings of WWW2002, Honolulu, Hawaii, 2002.

J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Medium-independent table detection. In SPIE Document Recognition and Retrieval VII, pages 291-302, San Jose, California, January 2000.

M. Hurst and T. Nasukawa. Layout and language: Integrating spatial and linguistic knowledge for layout understanding tasks. In the Proc. of 18th International Conference on Computational Linguistics, COLING, Saarbruecken, Germany, July 2000.

M. Hurst and S. Douglas. Layout Language: Preliminary experiments in assigning logical structure to table cells. In the Proc. of 5th Applied Natural Language Processing Conference, Washington, D.C., 1997.

M. Hurst. The Interpretation of Tables in Texts. PhD thesis, University of Edinburgh, School of Cognitive Science, Informatics, 2000.

P. Pyreddy and W. B. Croft. TINTIN: A System for Retrieval in Text Tables. In Proceedings of the Second ACM International Conference on Digital Libraries, 1997.

D. Pinto, W. Croft, M. Branstein, R. Coleman, M. King, W. Li, and X. Wei. Quasm: A system for question answering using semi-structured data. In Proceedings of the JCDL 2002.

Levenstein V.I. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and. Control Theory, 1966.*

The Common Data Set (http://www.commondataset.org)