

# Detecting Tables in HTML Documents

Yalin Wang<sup>1</sup> and Jianying Hu<sup>2</sup>

<sup>1</sup> Dept. of Electrical Engineering, Univ. of Washington,  
Seattle, WA 98195, US

ylwang@u.washington.edu

<sup>2</sup> Avaya Labs Research, 233 Mount Airy road,  
Basking Ridge, NJ 07920, US

jianhu@avaya.com

**Abstract.** Table is a commonly used presentation scheme for describing relational information. Table understanding on the web has many potential applications including web mining, knowledge management, and web content summarization and delivery to narrow-bandwidth devices. Although in HTML documents tables are generally marked as `<table>` elements, a `<table>` element does not necessarily indicate the presence of a *genuine* relational table. Thus the important first step in table understanding in the web domain is the detection of the genuine tables. In our earlier work we designed a basic rule-based algorithm to detect genuine tables in major news and corporate home pages as part of a web content filtering system. In this paper we investigate a machine learning based approach that is trainable and thus can be automatically generalized to including any domain. Various features reflecting the layout as well as content characteristics of tables are explored. The system is tested on a large database which consists of 1,393 HTML files collected from hundreds of different web sites from various domains and contains over 10,000 leaf `<table>` elements. Experiments were conducted using the cross validation method. The machine learning based approach outperformed the rule-based system and achieved an F-measure of 95.88%.

## 1 Introduction

The increasing ubiquity of the Internet has brought about a constantly increasing amount of online publications. As a compact and efficient way to present relational information, tables are used frequently in web documents. Since tables are inherently concise as well as information rich, the automatic understanding of tables has many applications including knowledge management, information retrieval, web mining, summarization, and content delivery to mobile devices. The processes of table understanding in web documents include table detection, functional and structural analysis and finally table interpretation [3].

In this paper, we concentrate on the problem of table detection. The web provides users with great possibilities to use their own style of communication and expressions. In particular, people use the `<table>` tag not only for relational information display but also to create any type of multiple-column layout to

facilitate easy viewing, thus the presence of the `<table>` tag does not necessarily indicate the presence of a true relational table. In this paper, we define *genuine* tables to be document entities where a two dimensional grid is semantically significant in conveying the logical relations among the cells [2]. Conversely, *Non-genuine* tables are document entities where `<table>` tags are used as a mechanism for grouping contents into clusters for easy viewing only. Examples of a genuine table and a non-genuine table can be found in Figure 1. While genuine tables in web documents could also be created without the use of `<table>` tags at all, we do not consider such cases in this article as they seem very rare from our experience. Thus, in this study, *Table detection* refers to the technique which classifies a document entity enclosed by the `<table></table>` tags as a genuine or non-genuine table.

Several researchers have reported their work on web table detection . Chen *et al.* used heuristic rules and cell similarities to identify tables and tested their algorithm on 918 tables form airline information web pages [1]. Yoshida *et al.* proposed a method to integrate WWW tables according to the category of objects presented in each table [4]. Their algorithm was evaluated on 175 tables.

In our earlier work, we proposed a rule-based algorithm for identifying genuinely tabular information as part of a web content filtering system for content delivery to mobile devices [2]. The algorithm was designed for major news and corporate web site home pages. It was tested on 75 web site front-pages and achieved an F-measure of 88.05%. While it worked reasonably well for the system it was designed for, it has the disadvantage that it is domain dependent and difficult to extend because of its reliance on hand-crafted rules.

To summarize, previous methods for web table detection all relied on heuristic rules and were only tested on a database that is either very small [2,4], or highly domain specific [1].

In this paper, we propose a new machine learning based approach for table detection from generic web documents. While many learning algorithms have been developed and tested for document analysis and information retrieval applications, there seems to be strong indication that good document representation including feature selection is more important than choosing a particular learning algorithm [12]. Thus in this work our emphasis is on identifying features that best capture the characteristics of a genuine table compared to a non-genuine one. In particular, we introduce a set of novel features which reflect the layout as well as content characteristics of tables. These features are then used in a tree classifier trained on thousands of examples. To facilitate the training and evaluation of the table classifier, we constructed a large web table ground truth database consisting of 1,393 HTML files containing 11,477 leaf `<table>` elements. Experiments on this database using the cross validation method demonstrate a significant performance improvement over the previously developed rule-based system.

The rest of the paper is organized as follows. We describe our feature set in Section 2, followed by a brief description of the decision tree classifier in Section 3. Section 4 explains the data collection process. Experimental results are then reported in Section 5 and we conclude with future directions in Section 6.

## 2 Features for Web Table Detection

Past research has clearly indicated that layout and content are two important aspects in table understanding [3]. Our features were designed to capture both of these aspects. In particular, we developed 16 features which can be categorized into three groups: seven layout features, eight content type features and one word group feature. In the first two groups, we attempt to capture the global composition of tables as well as the consistency within the whole table and across rows and columns. With the last feature, we investigate the discriminative power of words enclosed in tables using well developed text categorization techniques.

Before feature extraction, each HTML document is first parsed into a document hierarchy tree using Java Swing XML parser with W3C HTML 3.2 DTD [2]. A `<table>` node is said to be a *leaf table* if and only if there are no `<table>` nodes among its children [2]. Our experience indicates that almost all genuine tables are leaf tables. Thus in this study only leaf tables are considered candidates for genuine tables and are passed on to the feature extraction stage. In the following we describe each feature in detail.

### 2.1 Layout Features

In HTML documents, although tags like `<TR>` and `<TD>` (or `<TH>`) may be assumed to delimit table rows and table cells, they are not always reliable indicators of the number of rows and columns in a table. Variations can be caused by spanning cells created using `<ROWSPAN>` and `<COLSPAN>` tags. Other tags such as `<BR>` could be used to move content into the next row. To extract layout features reliably, we maintain a matrix to record all the cell spanning information and serve as a pseudo rendering of the table. Layout features based on row or column numbers are then computed from this matrix.

Given a table  $T$ , we compute the following four layout features:

- (1) and (2): Average number of columns, computed as the average number of cells per row, and the standard deviation.
- (3) and (4): Average number of rows, computed as the average number of cells per column, and the standard deviation.

Since the majority of tables in web documents contain characters, we compute three more layout features based on cell length in terms of number of characters:

- (5) and (6): Average overall cell length and the standard deviation.
- (7): Average *Cumulative length consistency*, *CLC*.

The last feature is designed to measure the cell length consistency along either row or column directions. It is inspired by the fact that most genuine tables demonstrate certain consistency either along the row or the column direction, but usually not both, while non-genuine tables often show no consistency in either direction. First, the average cumulative within-row length consistency,

$CLC_r$ , is computed as follows. Let the set of cell lengths of the cells from row  $i$  be  $\mathcal{R}_i$ ,  $i = 1, \dots, r$  (considering only non-spanning cells), and the the mean cell length for row  $\mathcal{R}_i$  be  $m_i$ :

1. Compute cumulative length consistency within each  $\mathcal{R}_i$ :  $CLC_i = \sum_{cl \in \mathcal{R}_i} LC_{cl}$ . Here  $LC_{cl}$  is defined as:  $LC_{cl} = 0.5 - D$ , where  $D = \min\{\frac{|cl - m_i|}{m_i}, 1.0\}$ . Intuitively,  $LC_{cl}$  measures the degree of consistency between  $cl$  and the mean cell length, with  $-0.5$  indicating extreme inconsistency and  $0.5$  indicating extreme consistency. When most cells within  $\mathcal{R}_i$  are consistent, the cumulative measure  $CLC_i$  is positive, indicating a more or less consistent row.
2. Take the average across all rows:  $CLC_r = \frac{1}{r} \sum_{i=1}^r CLC_i$ .

After the within-row length consistency  $CLC_r$  is computed, the within-column length consistency  $CLC_c$  is computed in a similar manner. Finally, the overall cumulative length consistency is computed as  $CLC = \max(CLC_r, CLC_c)$ .

## 2.2 Content Type Features

Web documents are inherently multi-media and have more types of content than any traditional document. For example, the content within a `<table>` element could include hyperlinks, images, forms, alphabetical or numerical strings, etc. Because of the relational information it needs to convey, a genuine table is more likely to contain alpha or numerical strings than, say, images. The content type feature was designed to reflect such characteristics.

We define the set of content types  $\mathcal{T} = \{\text{Image, Form, Hyperlink, Alphabetical, Digit, Empty, Others}\}$ . Our content type features include:

- (1) - (7): The histogram of content type for a given table. This contributes 7 features to the feature set;
- (8): Average *content type consistency*,  $CTC$ .

The last feature is similar to the cell length consistency feature. First, within-row content type consistency  $CTC_r$  is computed as follows. Let the set of cell type of the cells from row  $i$  as  $\mathcal{T}_i$ ,  $i = 1, \dots, r$  (again, considering only non-spanning cells), and the dominant type for  $\mathcal{T}_i$  be  $DT_i$ :

1. Compute the cumulative type consistency with each row  $\mathcal{R}_i$ ,  $i = 1, \dots, r$ :  $CTC_i = \sum_{ct \in \mathcal{R}_i} D$ , where  $D = 1$  if  $ct$  is equal to  $DT_i$  and  $D = -1$ , otherwise.
2. Take the average across all rows:  $CTC_r = \frac{1}{r} \sum_{i=1}^r CTC_i$ .

The within-column type consistency is then computed in a similar manner. Finally, the overall cumulative type consistency is computed as:  $CTC = \max(CTC_r, CTC_c)$ .

### 2.3 Word Group Feature

If we look at the enclosed text in a table and treat it as a “mini-document”, table classification could be viewed as a text categorization problem with two broad categories: genuine tables and non-genuine tables. In order to explore the the potential discriminative power of table text at the word level, we experimented with several text categorization techniques.

Text categorization is a well studied problem in the IR community and many algorithms have been developed over the years (e.g., [6,7]). For our application, we are particularly interested in algorithms with the following characteristics. First, it has to be able to handle documents with dramatically differing lengths (some tables are very short while others can be more than a page long). Second, it has to work well on collections with a very skewed distribution (there are many more non-genuine tables than genuine ones). Finally, since we are looking for a feature that can be incorporated along with other features, it should ideally produce a continuous confidence score rather than a binary decision. In particular, we experimented with three different approaches: vector space, naive Bayes and weighted kNN. The details regarding each approach are given below.

**Vector Space Approach.** After morphing [9] and removing the infrequent words, we obtain the set of words found in the training data,  $\mathcal{W}$ . We then construct weight vectors representing genuine and non-genuine tables and compare that against the frequency vector from each new incoming table.

Let  $\mathcal{Z}$  represent the non-negative integer set. The following functions are defined on set  $\mathcal{W}$ .

- $df^G : \mathcal{W} \rightarrow \mathcal{Z}$ , where  $df^G(w_i)$  is the number of genuine tables which include word  $w_i$ ,  $i = 1, \dots, |\mathcal{W}|$ ;
- $tf^G : \mathcal{W} \rightarrow \mathcal{Z}$ , where  $tf^G(w_i)$  is the number of times word  $w_i$ ,  $i = 1, \dots, |\mathcal{W}|$ , appears in genuine tables;
- $df^N : \mathcal{W} \rightarrow \mathcal{Z}$ , where  $df^N(w_i)$  is the number of non-genuine tables which include word  $w_i$ ,  $i = 1, \dots, |\mathcal{W}|$ ;
- $tf^N : \mathcal{W} \rightarrow \mathcal{Z}$ , where  $tf^N(w_i)$  is the number of times word  $w_i$ ,  $i = 1, \dots, |\mathcal{W}|$ , appears in non-genuine tables.
- $tf^T : \mathcal{W} \rightarrow \mathcal{Z}$ , where  $tf^T(w_i)$  is the number of times word  $w_i$ ,  $w_i \in \mathcal{W}$  appears in a new test table.

To simplify the notations, in the following discussion, we will use  $df_i^G$ ,  $tf_i^G$ ,  $df_i^N$  and  $tf_i^N$  to represent  $df^G(w_i)$ ,  $tf^G(w_i)$ ,  $df^N(w_i)$  and  $tf^N(w_i)$ , respectively.

Let  $N^G$ ,  $N^N$  be the number of genuine tables and non-genuine tables in the training collection, respectively and let  $C = \max(N^G, N^N)$ . Without loss of generality, we assume  $N^G \neq 0$  and  $N^N \neq 0$ . For each word  $w_i$  in  $\mathcal{W}$ ,  $i = 1, \dots, |\mathcal{W}|$ , two weights,  $p_i^G$  and  $p_i^N$  are computed:

$$p_i^G = \begin{cases} tf_i^G \log\left(\frac{df_i^G}{N^G} \frac{N^N}{df_i^N} + 1\right), & \text{when } df_i^N \neq 0 \\ tf_i^G \log\left(\frac{df_i^G}{N^G} C + 1\right), & \text{when } df_i^N = 0 \end{cases} \quad (1)$$

$$p_i^N = \begin{cases} tf_i^N \log\left(\frac{df_i^N}{N^N} \frac{N^G}{df_i^G} + 1\right), & \text{when } df_i^G \neq 0 \\ tf_i^N \log\left(\frac{df_i^N}{N^N} C + 1\right), & \text{when } df_i^G = 0 \end{cases} \quad (2)$$

As can be seen from the formulas, the definitions of these weights were derived from the traditional  $tf * idf$  measures used in informational retrieval [6], with some adjustments made for the particular problem at hand.

Given a new incoming table, let us denote the set including all the words in it as  $\mathcal{W}_n$ . Since we only need to consider the words that are present in both  $\mathcal{W}$  and  $\mathcal{W}_n$ , we first compute the *effective word set*:  $\mathcal{W}_e = \mathcal{W} \cap \mathcal{W}_n$ . Let the words in  $\mathcal{W}_e$  be represented as  $w_{m_k}$ , where  $m_k, k = 1, \dots, |\mathcal{W}_e|$ , are indexes to the words from set  $\mathcal{W} = \{w_1, w_2, \dots, w_{|\mathcal{W}|}\}$ . we define the following weight vectors:

- Vector representing the genuine table class:  $\vec{G}_S = \left(\frac{p_{m_1}^G}{U}, \frac{p_{m_2}^G}{U}, \dots, \frac{p_{m_{|\mathcal{W}_e|}}^G}{U}\right)$ ,  
where  $U$  is the cosine normalization term:  $U = \sqrt{\sum_{k=1}^{|\mathcal{W}_e|} p_{m_k}^G \times p_{m_k}^G}$ .
- Vector representing the non-genuine table class:  $\vec{N}_S = \left(\frac{p_{m_1}^N}{V}, \frac{p_{m_2}^N}{V}, \dots, \frac{p_{m_{|\mathcal{W}_e|}}^N}{V}\right)$ , where  $V$  is the cosine normalization term:  
 $V = \sqrt{\sum_{k=1}^{|\mathcal{W}_e|} p_{m_k}^N \times p_{m_k}^N}$ .
- Vector representing the new incoming table:  $\vec{I}_T = (tf_{m_1}^T, tf_{m_2}^T, \dots, tf_{m_{|\mathcal{W}_e|}}^T)$ .

Finally, the word group feature is defined as the ratio of the two dot products:

$$W_{vs} = \begin{cases} \frac{\vec{I}_T \cdot \vec{G}_S}{\vec{I}_T \cdot \vec{N}_S}, & \text{when } \vec{I}_T \cdot \vec{N}_S \neq 0 \\ 1, & \text{when } \vec{I}_T \cdot \vec{G}_S = 0 \text{ and } \vec{I}_T \cdot \vec{N}_S = 0 \\ 10, & \text{when } \vec{I}_T \cdot \vec{G}_S \neq 0 \text{ and } \vec{I}_T \cdot \vec{N}_S = 0 \end{cases} \quad (3)$$

**Naive Bayes Approach.** In the Bayesian learning framework, it is assumed that text data has been generated by a parametric model, and a set of training data is used to calculate Bayes optimal estimates of the model parameters. Then, using these estimates, Bayes rule is used to turn the generative model around and compute the probability of each class given an input document.

Word clustering is commonly used in a Bayes approach to achieve more reliable parameter estimation. For this purpose we implemented the distributional clustering method introduced by Baker and McCallum [8]. First stop words and words that only occur in less than 0.1% of the documents are removed. The resulting vocabulary has roughly 8000 words. Then distribution clustering is applied to group similar words together. Here the similarity between two words  $w_t$  and  $w_s$  is measured as the similarity between the class variable distributions they induce:  $P(C|w_t)$  and  $P(C|w_s)$ , and computed as the average KL divergence between the two distributions. (see [8] for more details).

Assume the whole vocabulary has been clustered into  $M$  clusters. Let  $w_s$  represent a word cluster, and  $C = \{g, n\}$  represent the set of class labels ( $g$

for for genuine,  $n$  for non-genuine), the class conditional probabilities are (using Laplacian prior for smoothing):

$$P(w_s|C = g) = \frac{tf^G(w_s) + 1}{M + \sum_{i=1}^M tf^G(w_i)}; \quad (4)$$

$$P(w_s|C = n) = \frac{tf^N(w_s) + 1}{M + \sum_{i=1}^M tf^N(w_i)}. \quad (5)$$

The prior probabilities for the two classes are:  $P(C = g) = \frac{N^G}{N^G + N^N}$  and  $P(C = n) = \frac{N^N}{N^G + N^N}$ .

Given a new table  $d_i$ , let  $d_{i,k}$  represent the  $k$ th word cluster. Based on the Bayes assumption, the posterior probabilities are computed as:

$$P(C = g|d_i) = \frac{P(C = g)P(d_i|C = g)}{P(d_i)} \quad (6)$$

$$\sim \frac{P(C = g) \prod_{k=1}^{|d_i|} P(w_{i,k}|C = g)}{P(d_i)}; \quad (7)$$

$$P(C = n|d_i) = \frac{P(C = n)P(d_i|C = n)}{P(d_i)} \quad (8)$$

$$\sim \frac{P(C = n) \prod_{k=1}^{|d_i|} P(w_{i,k}|C = n)}{P(d_i)}. \quad (9)$$

Finally, the word group feature is defined as the ratio between the two:

$$W_{nb} = \frac{P(C = g) \prod_{k=1}^{|d_i|} P(w_{i,k}|C = g)}{P(C = n) \prod_{k=1}^{|d_i|} P(w_{i,k}|C = n)} = \frac{N^G}{N^N} \prod_{k=1}^{|d_i|} \frac{P(w_{i,k}|C = g)}{P(w_{i,k}|C = n)}. \quad (10)$$

**Weighted kNN Approach.** kNN stands for k-nearest neighbor classification, a well known statistical approach. It has been applied extensively to text categorization and is one of the top-performing methods [7]. Its principle is quite simple: given a test document, the system finds the  $k$  nearest neighbors among the training documents, and uses the category labels of these neighbors to compute the likelihood score of each candidate category. The similarity score of each neighbor document to the test documents is used as the weight for the category it belongs to. The category receiving the highest score is then assigned to the test document.

In our application the above procedure is modified slightly to generate the word group feature. First, for efficiency purpose, the same preprocessing and word clustering operations as described in the previous section is applied, which results in  $M$  word clusters. Then each table is represented by an  $M$  dimensional vector composed of the term frequencies of the  $M$  word clusters. The similarity score between two tables is defined to be the cosine value ( $[0, 1]$ ) between the two corresponding vectors. For a new incoming table  $d_i$ , let the  $k$  training tables

that are most similar to  $d_i$  be represented by  $d_{i,j}, j = 1, \dots, k$ . Furthermore, let  $sim(d_i, d_{i,j})$  represent the similarity score between  $d_i$  and  $d_{i,j}$ , and  $C(d_{i,j})$  equals 1.0 if  $d_{i,j}$  is genuine and  $-1.0$  otherwise, the word group feature is defined as:

$$W_{knn} = \frac{\sum_{j=1}^k C(d_{i,j})sim(d_i, d_{i,j})}{\sum_{j=1}^k sim(d_i, d_{i,j})}. \quad (11)$$

### 3 Classification Scheme

Various classification schemes have been widely used in web document processing and proved to be promising for web information retrieval [11]. For the table detection task, we decided to use a decision tree classifier because of the highly non-homogeneous nature of our features. Another advantage of using a tree classifier is that no assumptions of feature independence are required.

An implementation of the decision tree allowing continuous feature values described by Haralick and Shapiro [5] was used for our experiments. The decision tree is constructed using a training set of feature vectors with true class labels. At each node, a discriminant threshold is chosen such that it minimizes an impurity value. The learned discriminant function splits the training subset into two subsets and generates two child nodes. The process is repeated at each newly generated child node until a stopping condition is satisfied, and the node is declared as a terminal node based on a majority vote. The maximum impurity reduction, the maximum depth of the tree, and minimum number of samples are used as stopping conditions.

### 4 Data Collection and Ground Truthing

Instead of working within a specific domain, our goal of data collection was to get tables of as many different varieties as possible from the web. At the same time, we also needed to insure that enough samples of genuine tables were collected for training purpose. Because of the latter practical constraint we biased the data collection process somewhat towards web pages that are more likely to contain genuine tables. A set of key words often associated with tables were composed and used to retrieve and download web pages using the Google search engine. Three directories on Google were searched: the business directory and news directory using key words: {`table`, `stock`, `bonds`, `figure`, `schedule`, `weather`, `score`, `service`, `results`, `value`}, and the science directory using key words {`table`, `results`, `value`}. A total of 2,851 web pages were downloaded in this manner and we ground truthed 1,393 HTML pages out of these (chosen randomly among all the HTML pages). The resulting database contains 14,609 `<table>` elements, out of which 11,477 are leaf `<table>` elements. Among the leaf `<table>` elements, 1,740 (15%) are genuine tables and the remaining 9,737 are non-genuine tables.



## 5 Experiments

A hold-out method is used to evaluate our table classifier. We randomly divided the data set into nine parts. The decision tree was trained on eight parts and then tested on the remaining one part. This procedure was repeated nine times, each time with a different choice for the test part. Then the combined nine part results are averaged to arrive at the overall performance measures [5].

The output of the classifier is compared with the ground truth and the standard performance measures precision (P), recall (R) and F-measure (F) are computed. Let  $N_{gg}$ ,  $N_{gn}$ ,  $N_{ng}$  represent the number of samples in the categories “genuine classified as genuine”, “genuine classified as non-genuine”, and “non-genuine classified as genuine”, respectively, the performance measures are defined as:

$$R = \frac{N_{gg}}{N_{gg} + N_{gn}} \quad P = \frac{N_{gg}}{N_{gg} + N_{ng}} \quad F = \frac{R + P}{2}.$$

For comparison among different features we report the performance measures when the best F-measure is achieved. The results of the table detection algorithm using various features and feature combinations are given in Table 1. For both the naive Bayes based and the kNN based word group features, 120 word clusters were used ( $M = 120$ ).

**Table 1.** Experimental results using various feature groups

	L	T	LT	LTW-VS	LTW-NB	LTW-KNN
R (%)	87.24	90.80	94.20	94.25	95.46	89.60
P (%)	88.15	95.70	97.27	97.50	94.64	95.94
F (%)	87.70	93.25	95.73	95.88	95.05	92.77

L: Layout features only.

T: Content type features only.

LT: Layout and content type features.

LTW-VS: Layout, content type and vector space based word group features.

LTW-NB: Layout, content type and naive Bayes based word group features.

LTW-KNN: Layout, content type and kNN based word group features.

As seen from the table, content type features performed better than layout features as a single group, achieving an F-measure of 93.25%. However, when the two groups were combined the F-measure was improved substantially to 95.73%, reconfirming the importance of combining layout and content features in table detection.

Among the different approaches for the word group feature, the vector space based approach gave the best performance when combined with layout and content features. However even in this case the addition of the word group feature brought about only a very small improvement. This indicates that the text enclosed in tables is not very discriminative, at least not at the word level. One possible reason is that the categories “genuine” and “non-genuine” are too broad for traditional text categorization techniques to be highly effective.

Overall, the best results were produced with the combination of layout, content type and vector space based word group features, achieving an F-measure of 95.88%. Figure 1 shows two examples of correctly classified tables, where Figure 1(a) is a genuine table and Figure 1(b) is a non-genuine table.

1961 (4-9-1)			
Date	Opponent	W/L	Score
Sept. 17	PITTSBURGH	W	27-24
Sept. 24	MINNESOTA	W	21-7
Oct. 1	Cleveland	L	25-7
Oct. 8	Minnesota	W	28-0
Oct. 15	N.Y. GIANTS	L	31-10
Oct. 22	PHILADELPHIA	L	43-7
Oct. 29	N.Y. Giants	W	17-16
Nov. 5	ST. LOUIS	L	31-17
Nov. 12	Pittsburgh	L	37-7
Nov. 19	WASHINGTON	T	28-28
Nov. 26	Philadelphia	L	35-13
Dec. 3	CLEVELAND	L	38-17
Dec. 10	St. Louis	L	31-13
Dec. 17	Washington	L	34-24

(a)

Worldwide Sugar Sites	
<a href="#">Links by Country</a>	<a href="#">Prices, Reports &amp; Subscriptions</a>
<a href="#">Agriculture</a>	<a href="#">Equipment &amp; Machinery</a>
<a href="#">Processing &amp; Refining</a>	<a href="#">Financial Services</a>
<a href="#">Traders &amp; Brokers</a>	<a href="#">Associations &amp; Organizations</a>
<a href="#">Logistics &amp; Packing</a>	<a href="#">Industrial Sugar Users</a>
<a href="#">Government &amp; Policy</a>	<a href="#">Research &amp; Technical</a>

(b)

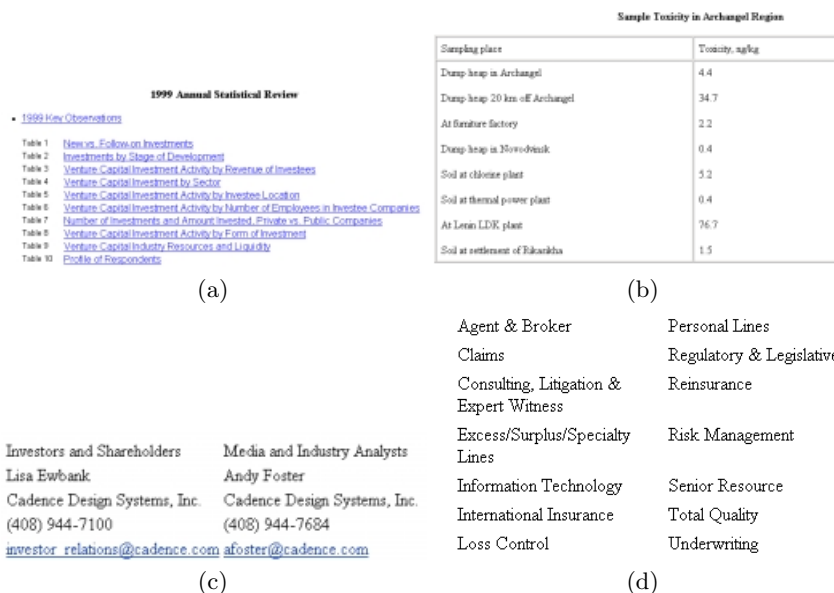
**Fig. 1.** Examples of correctly classified tables: (a) a genuine table; (b) a non-genuine table

Figure 2 shows a few examples where our algorithm failed. Figure 2(a) was misclassified as a non-genuine table, likely because its cell lengths are highly inconsistent and it has many hyperlinks which is unusual for genuine tables. Figure 2(b) was misclassified as non-genuine because its HTML source code contains only two `<tr>` tags. Instead of the `<tr>` tag, the author used `<p>` tags to place the multiple table rows in separate lines. This points to the need for a more carefully designed pseudo-rendering process.

Figure 2(c) shows a non-genuine table misclassified as genuine. A close examination reveals that it indeed has good consistency along the row direction. In fact, one could even argue that this is indeed a genuine table, with implicit row headers of *Title*, *Name*, *Company Affiliation* and *Phone Number*. This example demonstrates one of the most difficult challenges in table understanding, namely the ambiguous nature of many table instances (see [10] for a more detailed analysis on that).

Figure 2(d) was also misclassified as a genuine table. This is a case where layout features and the kind of shallow content features we used are not enough – deeper semantic analysis would be needed in order to identify the lack of logical coherence which makes it a non-genuine table.

For comparison, we tested the previously developed rule-based system [2] on the same database. The initial results (shown in Table 2 under “Original Rule Based”) were very poor. After carefully studying the results from the initial experiment we realized that most of the errors were caused by a rule imposing a hard limit on cell lengths in genuine tables. After deleting that rule the rule-based



**Fig. 2.** Examples of misclassified tables: (a), (b) genuine tables misclassified as non-genuine; (c), (d) non-genuine tables misclassified as genuine

system achieved much improved results (shown in Table 2 under “Modified Rule Based”). However, the proposed machine learning based method still performs considerably better in comparison. This demonstrates that systems based on hand-crafted rules tend to be brittle and do not generalize well. In this case, even after careful manual adjustment in a new database, it still does not work as well as an automatically trained classifier.

**Table 2.** Experimental results of the rule based system

	Original Rule Based	Modified Rule Based
R (%)	48.16	95.80
P (%)	75.70	79.46
F (%)	61.93	87.63

A direct comparison to other previous results [1,4] is not possible currently because of the lack of access to their system. However, our test database is clearly more general and far larger than the ones used in [1] and [4], while our precision and recall rates are both higher.

## 6 Conclusion and Future Work

We present a machine learning based table detection algorithm for HTML documents. Layout features, content type features and word group features were used

to construct a feature set and a tree classifier was built using these features. For the most complex word group feature, we investigated three alternatives: vector space based, naive Bayes based, and weighted K nearest neighbor based. We also constructed a large web table ground truth database for training and testing. Experiments on this large database yielded very promising results and reconfirmed the importance of combining layout and content features for table detection.

Our future work includes handling more different HTML styles in pseudo-rendering and developing a machine learning based table interpretation algorithm. We would also like to investigate ways to incorporate deeper language analysis for both table detection and interpretation.

**Acknowledgment.** We would like to thank Kathie Shipley for her help in collecting the web pages, and Amit Bagga for discussions on vector space models.

## References

1. H.-H. Chen, S.-C. Tsai, and J.-H. Tsai: Mining Tables from Large Scale HTML Texts. In: The 18th Int. Conference on Computational Linguistics, Saarbrücken, Germany, July 2000.
2. G. Penn, J. Hu, H. Luo, and R. McDonald: Flexible Web Document Analysis for Delivery to Narrow-Bandwidth Devices. In: ICDAR2001, Seattle, WA, USA, September 2001.
3. M. Hurst: Layout and Language: Challenges for Table Understanding on the Web. In: First International Workshop on Web Document Analysis, Seattle, WA, USA, September 2001, <http://www.csc.liv.ac.uk/wda2001>.
4. M. Yoshida, K. Torisawa, and J. Tsujii: A Method to Integrate Tables of the World Wide Web. In: First International Workshop on Web Document Analysis, Seattle, WA, USA, September 2001, <http://www.csc.liv.ac.uk/wda2001/>.
5. R. Haralick and L. Shapiro: Computer and Robot Vision. Addison Wesley, 1992.
6. T. Joachims: A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In: The 14th International Conference on Machine Learning, Nashville, Tennessee, 1997.
7. Y. Yang and X. Liu: A Re-Examination of Text Categorization Methods, In: SIGIR'99, Berkeley, California, 1999.
8. D. Baker and A.K. McCallum: Distributional Clustering of Words for Text Classification, In: SIGIR'98, Melbourne, Australia, 1998.
9. M. F. Porter: An Algorithm for Suffix Stripping. In: Program, Vol. 14, no.3, 1980.
10. J. Hu, R. Kashi, D. Lopresti, G. Nagy, and G. Wilfong: Why Table Ground-Truthing is Hard. In: ICDAR2001, Seattle, WA, September 2001.
11. A. McCallum, K. Nigam, J. Rennie, and K. Seymore: Automating the Construction of Internet Portals with Machine Learning. In: Information Retrieval Journal, vol. 3, 2000.
12. D. Mladenic: Text-learning and related intelligent agents. In: IEEE Expert, July-August 1999.