A Machine Learning Based Approach for Table Detection on The Web

Yalin Wang
Intelligent Systems Laboratory
Dept. of Electrical Engineering
Univ. of Washington
Seattle, WA 98195 US
ylwang@u.washington.edu

Jianying Hu
Avaya Labs Research
233, Mount Airy road
Basking Ridge, NJ 07920 US
jianhu@avaya.com

ABSTRACT

Table is a commonly used presentation scheme, especially for describing relational information. However, table understanding remains an open problem. In this paper, we consider the problem of table detection in web documents. Its potential applications include web mining, knowledge management, and web content summarization and delivery to narrow-bandwidth devices. We describe a machine learning based approach to classify each given table entity as either *genuine* or *non-genuine*. Various features reflecting the layout as well as content characteristics of tables are studied.

In order to facilitate the training and evaluation of our table classifier, we designed a novel web document table ground truthing protocol and used it to build a large table ground truth database. The database consists of 1,393 HTML files collected from hundreds of different web sites and contains 11,477 leaf `<TABLE>` elements, out of which 1,740 are genuine tables. Experiments were conducted using the cross validation method and an F-measure of 95.89% was achieved.

Keywords

Table Detection, Layout Analysis, Machine Learning, Decision tree, Support Vector Machine, Information Retrieval

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 [6]. 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 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 [10]. Conversely, *Non-genuine* tables are document entities where `<TABLE>` tags are used as a mechanism for grouping contents into clusters for easy viewing only. Figure 1 gives a few examples of genuine and non-genuine tables. 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 genuine or non-genuine tables.

Several researchers have reported their work on web table detection [2, 10, 6, 14]. In [2], Chen *et. al.* used heuristic rules and cell similarities to identify tables. They tested their table detection algorithm on 918 tables from airline information web pages and achieved an F-measure of 86.50%. Penn *et. al.* proposed a set of rules for identifying genuinely tabular information and news links in HTML documents [10]. They tested their algorithm on 75 web site front-pages and achieved an F-measure of 88.05%. Yoshida *et. al.* proposed a method to integrate WWW tables according to the category of objects presented in each table [14]. Their data set contains 35,232 table tags gathered from the web. They estimated their algorithm parameters using all of table data and then evaluated algorithm accuracy on 175 of the tables. The average F-measure reported in their paper is 82.65%. These previous methods all relied on heuristic rules and were only tested on a database that is either very small [10], or highly domain specific [2]. Hurst mentioned that a Naive Bayes classifier algorithm produced adequate results but no detailed algorithm and experimental information was provided [6].

We propose a new machine learning based approach for table detection from generic web documents. In particular, we introduce a set of novel features which reflect the layout as well as content characteristics of tables. These features are used in classifiers trained on thousands of examples. To facilitate the training and evaluation of the table classifiers, we designed a novel web document table ground truthing protocol and used it to build a large table ground truth database. The database consists of 1,393 HTML files collected from hundreds of different web sites and contains

Copyright is held by the author/owner(s).
WWW2002, May 7–11, 2002, Honolulu, Hawaii, USA.
ACM 1-58113-449-5/02/0005.

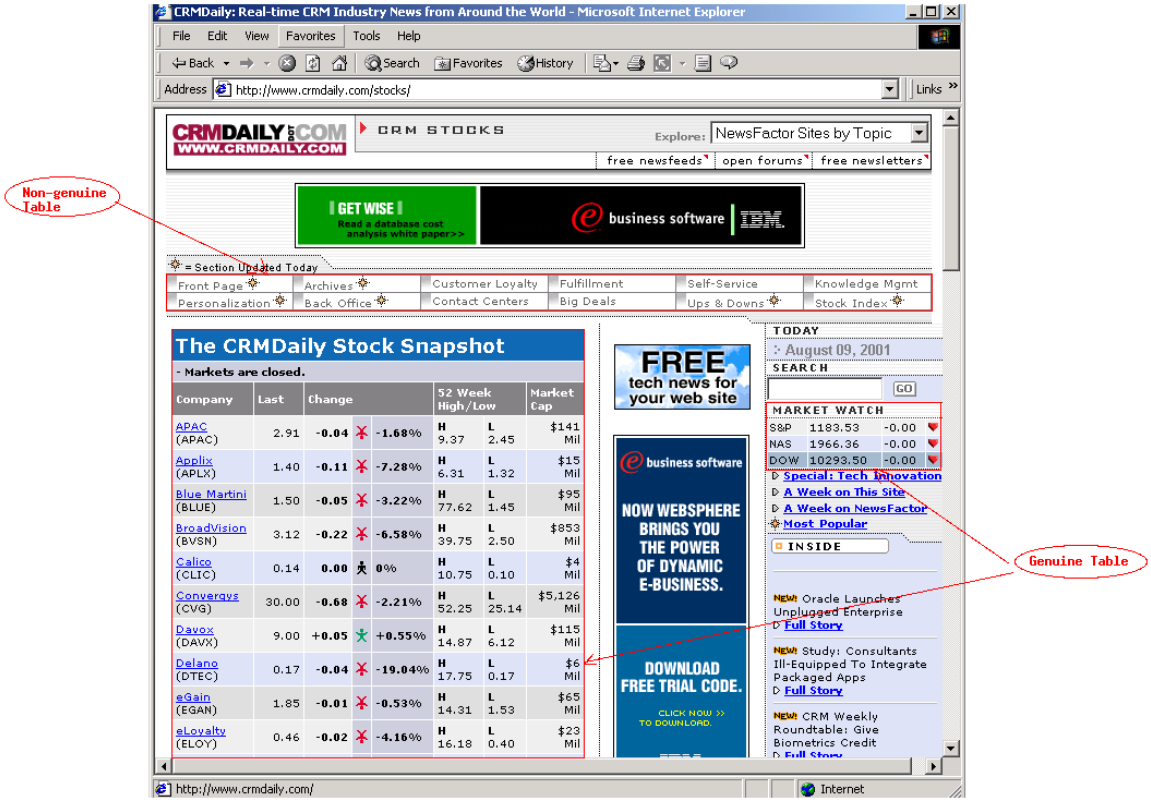

Figure 1: Examples of genuine and non-genuine tables.

11,477 leaf `<TABLE>` elements, out of which 1,740 are genuine tables. Experiments on this database using the cross validation method demonstrate significant performance improvements over previous methods.

The rest of the paper is organized as follows. We describe our feature set in Section 2, followed by a brief discussion of the classifiers we experimented with in Section 3. In Section 4, we present a novel table ground truthing protocol and explain how we built our database. Experimental results are then reported in Section 5 and we conclude with future directions in Section 6.

2. FEATURES FOR WEB TABLE DETECTION

Feature selection is a crucial step in any machine learning based methods. In our case, we need to find a combination of features that together provide significant separation between genuine and non-genuine tables while at the same time constrain the total number of features to avoid the curse of dimensionality. Past research has clearly indicated that layout and content are two important aspects in table understanding [6]. 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. The last feature looks at words used in tables and is derived directly from the vector space model commonly used in Information Retrieval.

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 [10]. A `<TABLE>` node is said to be a *leaf table* if and only if there are no `<TABLE>` nodes among its children [10]. 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. Therefore to extract layout features reliably one can not simply count the number of `<TR>`'s and `<TD>`'s. For this purpose, 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, assuming its numbers of rows and columns are rn and cn respectively, we compute the following layout features:

- Average number of columns, computed as the average

number of cells per row:

$$c = \frac{1}{rn}\sum_{i=1}^{rn} c_i,$$

where c_i is the number of cells in row i, $i = 1, ..., rn$;

- Standard deviation of number of columns:

$$dC = \sqrt{\frac{1}{rn}\sum_{i=1}^{rn}(c_i - c) \times (c_i - c)};$$

- Average number of rows, computed as the average number of cells per column:

$$r = \frac{1}{rn}\sum_{i=1}^{cn} r_i,$$

where r_i is the number of cells in column i, $i = 1, ..., cn$;

- Standard deviation of number of rows:

$$dR = \sqrt{\frac{1}{cn}\sum_{i=1}^{cn}(r_i - r) \times (r_i - r)}.$$

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:

- Average overall cell length: $cl = \frac{1}{en}\sum_{i=1}^{en} cl_i$, where en is the total number of cells in a given table and cl_i is the length of cell i, $i = 1, \ldots, en$;
- Standard deviation of cell length:

$$dCL = \sqrt{\frac{1}{en}\sum_{i=1}^{en}(cl_i - cl) \times (cl_i - cl)};$$

- 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 $\mathcal{R}_i$, $i = 1, \ldots, r$ (considering only non-spanning cells):

1. Compute the mean cell length, m_i, for row $\mathcal{R}_i$.
2. 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 $D = min\{|cl - m_i|/m_i, 1.0\}$. 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.

3. 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 has more types of content than any traditional documents. 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 $\mathcal{T} = \{\text{Image}, \text{Form}, \text{Hyperlink}, \text{Alphabetical}, \text{Digit}, \text{Empty}, \text{Others}\}$. Our content type features include:

- The histogram of content type for a given table. This contributes 7 features to the feature set;
- 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 $\mathcal{T}_i$, $i = 1, \ldots, r$ (again, considering only non-spanning cells):

1. Find the dominant type, DT_i, for $\mathcal{T}_i$.
2. Compute the cumulative type consistency with each row $\mathcal{R}_i$, $i = 1, \ldots, r$:

$$CTC_i = \sum_{ct \in \mathcal{R}_i} D,$$

where $D = 1$ if ct is equal to DT_i and $D = -1$, otherwise.

3. 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 treat each table as a "mini-document" by itself, table classification can be viewed as a document categorization problem with two broad categories: genuine tables and non-genuine tables. We designed the word group feature to incorporate word content for table classification based on techniques developed in information retrieval [7, 13].

After morphing [11] 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, $i = 1, ..., |\mathcal{W}|$;
- $tf^G : \mathcal{W} \rightarrow \mathcal{Z}$, where $tf^G(w_i)$ is the number of times word w_i, $i = 1, ..., |\mathcal{W}|$, 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, $i = 1, ..., |\mathcal{W}|$;
- $tf^N : \mathcal{W} \rightarrow \mathcal{Z}$, where $tf^N(w_i)$ is the number of times word w_i, $i = 1, ..., |\mathcal{W}|$, 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 $\mathcal{W}$, $i = 1, ..., |\mathcal{W}|$, two weights, p_i^G and p_i^N are computed:

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

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

As can be seen from the formulas, the definitions of these weights were derived from the traditional $tf * idf$ measures used in informational retrieval, 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 $\mathcal{W}$ is constructed using thousands of tables, the words that are present in both $\mathcal{W}$ and $\mathcal{W}_n$ are only a small subset of $\mathcal{W}$. Based on the vector space model, we define the similarity between weight vectors representing genuine and non-genuine tables and the frequency vector representing the incoming table as the corresponding dot products. 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 $m_k, k = 1, ..., |\mathcal{W}_e|$, are indexes to the words from set $\mathcal{W} = \{w_1, w_2, ..., w_{|\mathcal{W}|}\}$. we define the following vectors:

- Weight vector representing the genuine table group:

$$\vec{G_S} = \left(\frac{p_{m_1}^G}{U}, \frac{p_{m_2}^G}{U}, \cdots, \frac{p_{m_{|\mathcal{W}_e|}}^G}{U}\right),$$

  where U is the cosine normalization term:

$$U = \sqrt{\sum_{k=1}^{|\mathcal{W}_e|} p_{m_k}^G \times p_{m_k}^G}.$$

- Weight vector representing the non-genuine table group:

$$\vec{N_S} = \left(\frac{p_{m_1}^N}{V}, \frac{p_{m_2}^N}{V}, \cdots, \frac{p_{m_{|\mathcal{W}_e|}}^N}{V}\right),$$

  where V is the cosine normalization term:

$$V = \sqrt{\sum_{k=1}^{|\mathcal{W}_e|} p_{m_k}^N \times p_{m_k}^N}.$$

- Frequency vector representing the new incoming table:

$$\vec{I_T} = \left(tf_{m_1}^T, tf_{m_2}^T, \cdots, tf_{m_{|\mathcal{W}_e|}}^T\right).$$

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

$$wg = \begin{cases} \frac{\vec{I_T} \cdot \vec{G_S}}{\vec{I_T} \cdot \vec{N_S}}, & \text{when } \vec{I_T} \cdot \vec{N_S} \neq 0 \\ 1, & \text{when } \vec{I_T} \cdot \vec{G_S} = 0 \text{ and } \vec{I_T} \cdot \vec{N_S} = 0 \\ 10, & \text{when } \vec{I_T} \cdot \vec{G_S} \neq 0 \text{ and } \vec{I_T} \cdot \vec{N_S} = 0 \end{cases}$$

3. CLASSIFICATION SCHEMES

Various classification schemes have been widely used in document categorization as well as web information retrieval [13, 8]. For the table detection task, the decision tree classifier is particularly attractive as our features are highly non-homogeneous. We also experimented with Support Vector Machines (SVM), a relatively new learning approach which has achieved one of the best performances in text categorization [13].

3.1 Decision Tree

Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued functions that is robust to noisy data.

Decision trees classify an instance by sorting it down the tree from the root to some leaf node, which provides the classification of the instance. Each node in a discrete-valued decision tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. Continuous-valued decision attributes can be incorporated by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals [9].

An implementation of the continuous-valued decision tree described in [4] 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.

3.2 SVM

Support Vector Machines (SVM) are based on the *Structural Risk Management* principle from computational learning theory [12]. The idea of structural risk minimization is to find a hypothesis h for which the lowest true error is guaranteed. The true error of h is the probability that h will make an error on an unseen and randomly selected test example.

The SVM method is defined over a vector space where the goal is to find a decision surface that best separates the data points in two classes. More precisely, the decision surface by SVM for linearly separable space is a hyperplane which can be written as

$$\vec{w} \cdot \vec{x} - b = 0$$

where $\vec{x}$ is an arbitrary data point and the vector $\vec{w}$ and the constant b are learned from training data. Let $D = (y_i, \vec{x_i})$ denote the training set, and $y_i \in \{+1, -1\}$ be the classification for $\vec{x_i}$, the SVM problem is to find $\vec{w}$ and b that satisfies the following constraints:

$$\vec{w} \cdot \vec{x_i} - b \geq +1 \; for \; y_i = +1$$

$$\vec{w} \cdot \vec{x_i} - b \leq -1 \; for \; y_i = -1$$

while minimizing the vector 2-norm of $\vec{w}$.

The SVM problem in linearly separable cases can be efficiently solved using quadratic programming techniques, while the non-linearly separable cases can be solved by either introducing soft margin hyperplanes, or by mapping the original data vectors to a higher dimensional space where the data points become linearly separable [12, 3].

One reason why SVMs are very powerful is that they are very universal learners. In their basic form, SVMs learn linear threshold functions. Nevertheless, by a simple "plug-in" of an appropriate kernel function, they can be used to learn polynomial classifiers, radial basis function (RBF) networks, three-layer sigmoid neural nets, etc. [3].

For our experiments, we used the SVM^{light} system implemented by Thorsten Joachims [1].

4. DATA COLLECTION AND TRUTHING

Since there are no publicly available web table ground truth database, researchers tested their algorithms in different data sets in the past [2, 10, 14]. However, their data sets either had limited manually annotated table data (*e.g.*, 918 table tags in [2], 75 HTML pages in [10], 175 manually annotated table tags in [14]), or were collected from some specific domains, (*e.g.*, a set of tables selected from airline information pages were used in [2]). To develop our machine learning based table detection algorithm, we needed to build a general web table ground truth database of significant size.

4.1 Data Collection

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. To accomplish this, we composed a set of key words likely to indicate documents containing tables and used those key words 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). These 1,393 HTML pages from around 200 web sites comprise our database.

[1] http://svmlight.joachims.org

4.2 Ground Truthing

There has been no previous report on how to systematically generate web table ground truth data. To build a large web table ground truth database, a simple, flexible and complete ground truth protocol is required. Figure 2(a) shows the diagram of our ground truthing procedure. We created a new Document Type Definition(DTD) which is a superset of W3C HTML 3.2 DTD. We added three attributes for `<TABLE>` element, which are "tabid", "genuine table" and "table title". The possible value of the second attribute is *yes* or *no* and the value of the first and third attributes is a string. We used these three attributes to record the ground truth of each leaf `<TABLE>` node. The benefit of this design is that the ground truth data is inside HTML file format. We can use exactly the same parser to process the ground truth data.

We developed a graphical user interface for web table ground truthing using the Java [1] language. Figure 2(b) is a snapshot of the interface. There are two windows. After reading an HTML file, the hierarchy of the HTML file is shown in the left window. When an item is selected in the hierarchy, the HTML source for the selected item is shown in the right window. There is a panel below the menu bar. The user can use the radio button to select either genuine table or non-genuine table. The text window is used to input table title.

4.3 Database Description

Our final table ground truth database contains 1,393 HTML pages collected from around 200 web sites. There are a total of 14,609 `<TABLE>` nodes, including 11,477 leaf `<TABLE>` nodes. Out of the 11,477 leaf `<TABLE>` nodes, 1,740 are genuine tables and 9,737 are non-genuine tables. Not every genuine table has its title and only 1,308 genuine tables have table titles. We also found at least 253 HTML files have unmatched `<TABLE>`, `</TABLE>` pairs or wrong hierarchy, which demonstrates the noisy nature of web documents.

5. EXPERIMENTS

A hold-out method is used to evaluate our table classifier. We randomly divided the data set into nine parts. Each classifier 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 [4].

For the layout and content type features, this procedure is straightforward. However it is more complicated for the word group feature training. To compute w_g for training samples, we need to further divide the training set into two groups, a larger one (7 parts) for the computation of the weights p_i^G and p_i^N, $i = 1, ..., |\mathcal{W}|$, and a smaller one (1 part) for the computation of the vectors $\vec{G_S}$, $\vec{N_S}$, and $\vec{I_T}$. This partition is again rotated to compute w_g for each table in the training set.

The output of each classifier is compared with the ground truth and a contingency table is computed to indicate the number of a particular class label that are identified as members of one of two classes. The rows of the contingency table represent the true classes and the columns represent the assigned classes. The cell at row r and column c is the number of tables whose true class is r while its assigned class is c.

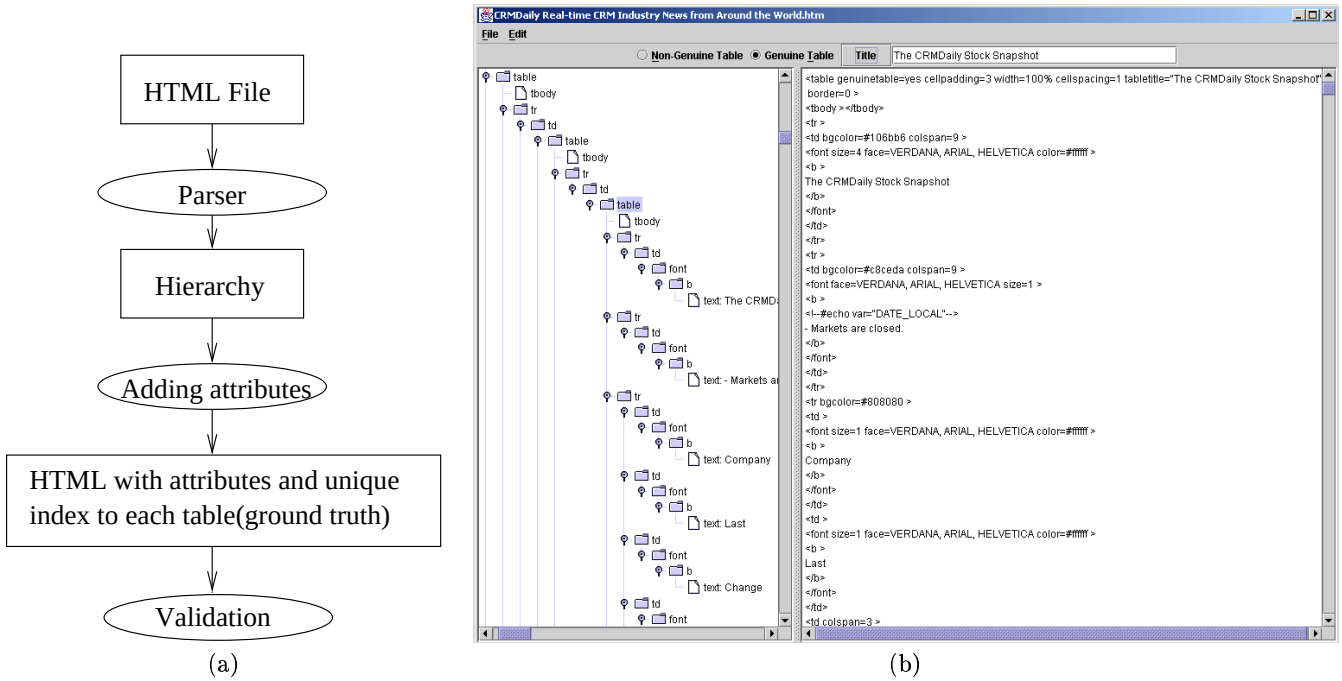

Figure 2: (a) The diagram of ground truthing procedure; (b) A snapshot of the ground truthing software.

The possible true- and detected-state combination is shown in Table 1. Three performance measures *Recall Rate(R)*, *Precision Rate(P)* and *F-measure(F)* are computed as follows:

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

True Class	Assigned Class	
	genuine table	non-genuine table
genuine table	N_{gg}	N_{gn}
non-genuine table	N_{ng}	N_{nn}

Table 1: Possible true- and detected-state combinations for two classes.

For comparison among different features and learning algorithms we report the performance measures when the best F-measure is achieved. First, the performance of various feature groups and their combinations were evaluated using the decision tree classifier. The results are given in Table 2.

	L	T	LT	LTW
R (%)	87.24	90.80	94.20	94.25
P (%)	88.15	95.70	97.27	97.50
F (%)	87.70	93.25	95.73	95.88

L: Layout only.
T: Content type only.
LT: Layout and content type.
LTW: Layout, content type and word group.

Table 2: Experimental results using various feature groups and the decision tree classifier.

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. The addition of the word group feature improved the F-measure slightly more to 95.88%.

Table 3 compares the performances of different learning algorithms using the full feature set. The leaning algorithms tested include the decision tree classifier and the SVM algorithm with two different kernels – linear and radial basis function (RBF).

	Tree	SVM (linear)	SVM (RBF)
R (%)	94.25	93.91	95.98
P (%)	97.50	91.39	95.81
F (%)	95.88	92.65	95.89

Table 3: Experimental results using different learning algorithms.

As seen from the table, for this application the SVM with radial basis function kernel performed much better than the one with linear kernel. It achieved an F measure of 95.89%, comparable to the 95.88% achieved by the decision tree classifier.

Figure 3 shows two examples of correctly classified tables, where Figure 3(a) is a genuine table and Figure 3(b) is a non-genuine table.

Figure 4 shows a few examples where our algorithm failed. Figure 4(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. The reason why Figure 4(b) was misclassified as non-genuine is more interesting. When we looked at its HTML source code,

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)

(b)

Figure 3: Examples of correctly classified tables. (a): a genuine table; (b): a Non-genuine table.

we found it contains only two `<TR>` tags. All text strings in one rectangular box are within one `<TD>` tag. Its author used `<p>` tags to put them in different rows. This points to the need for a more carefully designed pseudo-rendering process. Figure 4(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 [5] for a more detailed analysis on that). Figure 4(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 [10] on the same database. The initial results (shown in Table 4 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 system achieved much improved results (shown in Table 4 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.

A direct comparison to other previous results [2, 14] 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 [2] and [14], while our precision and recall rates are both higher.

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

Table 4: Experimental results of a previously developed rule based system.

6. CONCLUSION AND FUTURE WORK

Table detection in web documents is an interesting and challenging problem with many applications. 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 novel feature set. Decision tree and SVM classifiers were then implemented and tested in this feature space. We also designed a novel table ground truthing protocol and used it to construct a large web table ground truth database for training and testing. Experiments on this large database yielded very promising results.

Our future work includes handling more different HTML styles in pseudo-rendering, detecting table titles of the recognized genuine tables 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.

7. 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.

8. REFERENCES

1999 Annual Statistical Review

- 1999 Key Observations

Table 1	New vs. Follow-on Investments
Table 2	Investments by Stage of Development
Table 3	Venture Capital Investment Activity by Revenue of Investees
Table 4	Venture Capital Investment by Sector
Table 5	Venture Capital Investment Activity by Investee Location
Table 6	Venture Capital Investment Activity by Number of Employees in Investee Companies
Table 7	Number of Investments and Amount Invested, Private vs. Public Companies
Table 8	Venture Capital Investment Activity by Form of Investment
Table 9	Venture Capital Industry Resources and Liquidity
Table 10	Profile of Respondents

(a)

Sample Toxicity in Archangel Region

Sampling place	Toxicity, ng/kg
Dump heap in Archangel	4.4
Dump heap 20 km off Archangel	34.7
At furniture factory	2.2
Dump heap in Novodvinsk	0.4
Soil at chlorine plant	5.2
Soil at thermal power plant	0.4
At Lenin LDK plant	76.7
Soil at settlement of Rikasikha	1.5

(b)

Investors and Shareholders	Media and Industry Analysts
Lisa Ewbank	Andy Foster
Cadence Design Systems, Inc.	Cadence Design Systems, Inc.
(408) 944-7100	(408) 944-7684
investor_relations@cadence.com	afoster@cadence.com

(c)

Agent & Broker	Personal Lines
Claims	Regulatory & Legislative
Consulting, Litigation & Expert Witness	Reinsurance
Excess/Surplus/Specialty Lines	Risk Management
Information Technology	Senior Resource
International Insurance	Total Quality
Loss Control	Underwriting

(d)

Figure 4: Examples of misclassified tables. (a) and (b): Genuine tables misclassified as non-genuine; (c) and (d): Non-genuine tables misclassified as genuine.

[1] M. Campione, K. Walrath, and A. Huml. The java(tm) tutorial: A short course on the basics (the java(tm) series).

[2] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai. Mining tables from large scale html texts. In *Proc. 18th International Conference on Computational Linguistics*, Saabrucken, Germany, July 2000.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–296, August 1995.

[4] R. Haralick and L. Shapiro. *Computer and Robot Vision*, volume 1. Addison Wesley, 1992.

[5] J. Hu, R. Kashi, D. Lopresti, G. Nagy, and G. Wilfong. Why table ground-truthing is hard. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR01)*, pages 129–133, Seattle, WA, USA, September 2001.

[6] M. Hurst. Layout and language: Challenges for table understanding on the web. In *Proc. 1st International Workshop on Web Document Analysis*, pages 27–30, Seattle, WA, USA, September 2001.

[7] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proc. 14th International Conference on Machine Learning*, pages 143–151, Morgan Kaufmann, 1997.

[8] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. In *Information Retrieval Journal*, volume 3, pages 127–163, Kluwer, 2000.

[9] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[10] G. Penn, J. Hu, H. Luo, and R. McDonald. Flexible web document analysis for delivery to narrow-bandwidth devices. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR01)*, pages 1074–1078, Seattle, WA, USA, September 2001.

[11] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[12] V. N. Vapnik. *The Nature of Statistical Learning Theory*, volume 1. Springer, New York, 1995.

[13] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. SIGIR'99*, pages 42–49, Berkeley, California, USA, August 1999.

[14] M. Yoshida, K. Torisawa, and J. Tsujii. A method to integrate tables of the world wide web. In *Proc. 1st International Workshop on Web Document Analysis*, pages 31–34, Seattle, WA, USA, September 2001.