International Journal on Document Analysis and Recognition manuscript No.
(will be inserted by the editor)

Evaluating the Performance of Table Processing Algorithms

J. Hu[1]**, R.S. Kashi**[1]**, D. Lopresti**[2]**, G.T. Wilfong**[2]

[1] Avaya Labs Research, 233 Mt. Airy Road, Basking Ridge, NJ 07920
[2] Bell Laboratories, Lucent Technologies, Inc., 600 Mountain Avenue, Murray Hill, NJ 07974

Abstract. While techniques for evaluating the performance of lower-level document analysis tasks such as optical character recognition have gained acceptance in the literature, attempts to formalize the problem for higher-level algorithms, while receiving a fair amount of attention in terms of theory, have generally been less successful in practice, perhaps owing to their complexity. In this paper, we introduce intuitive, easy-to-implement evaluation schemes for the related problems of table detection and table structure recognition. We also present the results of several small experiments, demonstrating how well the methodologies work and the useful sorts of feedback they provide.

We first consider the table detection problem. Here algorithms can yield various classes of errors, including non-table regions improperly labeled as tables (insertion errors), tables missed completely (deletion errors), larger tables broken into a number of smaller ones (splitting errors), and groups of smaller tables combined to form larger ones (merging errors). This leads naturally to the use of an edit distance approach for assessing the results of table detection.

Next we address the problem of evaluating table structure recognition. Our model is based on a directed acyclic attribute graph, or table DAG. We describe a new paradigm, "graph probing," for comparing the results returned by the recognition system and the representation created during ground-truthing. Probing is in fact a general concept that could be applied to other document recognition tasks as well.

Key words: Table detection – Table recognition – Document Layout Analysis – Document Understanding – Edit Distance – Graph Matching – Performance Evaluation

1 Introduction

As the sophistication of document analysis systems grows, it becomes increasingly important to be able to evaluate and compare their performance. With a few notable exceptions, however, little has been achieved along these lines beyond the informal sorts of assertions that accompany most work published in the literature. A thoughtful overview of the subject of automated performance evaluation can be found in [12]. In this paper, we present general methods for evaluating a class of higher-level document analysis algorithms as embodied by the table understanding problem.

Many document recognition tasks can be viewed as comprising two basic steps: detecting a particular type of region on the page (*e.g.*, tables, figures, photographs, text blocks), and interpreting the structure within a detected region (*e.g.*, rows and columns within a table). We describe separate approaches for each of these cases.

We first consider the problem of evaluating algorithms for detecting tables in scanned pages and ASCII text documents. While a number of researchers have studied table detection, evaluation is often accomplished via a low-level, "local" measure of correctness based on whether individual text lines are labeled as belonging to a table or not (*e.g.*, [15]). Note, though, that this approach fails to discern important kinds of errors, including splitting and merging of adjacent tables. On the other hand, another, more conservative view holds that a result is correct if and only if the table has been perfectly delimited (*i.e.*, if no lines of the table are missed, and no extraneous lines are included). However, one disadvantage of this policy is that it over-penalizes small mistakes (*e.g.*, adding an extra blank line to the end of the table). In this paper, we propose an edit distance measure that captures the full range of errors that might be made by a table detection algorithm. Unlike existing approaches for the seemingly similar problem of page segmentation that also employ distance-type measures, our approach is based on logical structure, not on pixel-level comparisons [8,21] or matching the text characters output from OCR [1,8].

For the problem of evaluating structure recognition, there is already a fair amount of theory (see, *e.g.*, [6, 11,16,17,19]). Nevertheless, the empirical literature has largely ignored this work, perhaps owing to its complexity, and usually defaults to a simple, manual approach to evaluation: counting by hand the number of structures that have been missed or added (*e.g.*, [14]).

In this paper, we offer a first step towards an intuitive, easy-to-implement evaluation methodology for table structure recognition that may be extensible to other applications as well. As is common in the theory world, our approach is based on a directed attribute graph representation, which we call a *table DAG*. One way to evaluate such results would be to compare the DAG returned by the recognition system with the DAG produced by ground-truthing via some sort of edit distance criterion. There are two obvious drawbacks to this scheme, however. First, it is quite possible that the representation for a given table will not be unique; different-looking DAG's may be functionally equivalent. Secondly, any procedure for solving this problem could also be used to solve the directed graph isomorphism problem, which is widely regarded to be hard [2]; hence, it seems unlikely that an efficient optimal algorithm exists. Another possibility would be to evaluate table recognition by measuring the overall performance of an application in which it is embedded [18]. The difficulty here is that errors in other subsystems (*e.g.*, layout analysis, OCR, user interface) will affect the results and make it impossible to distinguish which errors are due to which subsystem.

The method we propose can be tailored to lie anywhere between these two extremes. Our approach uses "random probes" of a DAG to assess the agreement of the result returned by the recognition system and the representation created during ground-truthing. Since different types of probes are possible, ranging from very low- to very high-level, this paradigm can be viewed as subsuming existing techniques that try to measure the structural similarity of the graph representations on the one hand, and the effectiveness of recognition results when incorporated in a particular application on the other. While graph probing is presented as a means of evaluating the performance of table recognition systems, it is in fact a more general concept that could be applied to other computer vision tasks employing representations for which probes can be generated and their responses compared.

We begin by considering the problem of evaluating table detection results in Section 2. After describing our methodology, we apply it in Section 3 to a system we have designed for locating tables in scanned and ASCII text documents, showing how well it captures and explains the performance of the detection algorithm. We then turn to the table structure recognition problem and its evaluation. Section 4 presents our model for graph probing, which we exercise in Section 5 using an algorithm for parsing table structure we have developed. Finally, we offer our conclusions and topics for future research in Section 6.

2 A Methodology for Evaluating the Results of Table Detection

In this section we describe a method for evaluating the performance of table detection algorithms building on the well-known concept of edit distance [3,4]. For the purposes of this evaluation, we assume the input is in single column format. A table T is considered to be a region consisting of one or more contiguous text lines ranging from line i to line j inclusive, $T = [i, j]$. A document contains some number of non-overlapping (but possibly adjacent) tables listed consecutively in order of occurrence. Let $R = T_1^R T_2^R \dots T_m^R$ be the recognized document and $G = T_1^G T_2^G \dots T_n^G$ be the ground truth. Note that m need not equal n in general, and that T_i^R need not necessarily be the table corresponding to T_i^G for any particular i.

In considering the output from table detection, it becomes evident that certain classes of errors may arise as depicted in Figure 1. These include non-table regions improperly labeled as tables (insertion errors), tables missed completely (deletion errors), larger tables broken into a number of smaller ones (splitting errors), and groups of smaller tables combined to form larger ones (merging errors). This leads naturally to the use of an edit distance model for assessing the results of table detection.

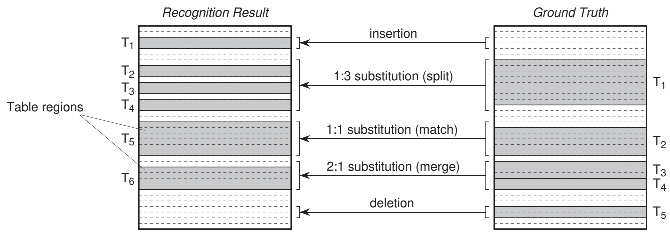


Fig. 1. Various possible errors in table detection.

We define *teval* (for *table evaluation*) recursively, where $teval_{i,j}$ is the distance computed between the first i tables of R and the first j tables of G. The costs functions are: $c_{del}()$, the cost of deleting a particular table; $c_{ins}()$, the cost of inserting a particular table; and $c_{sub_{k:l}}()$, a generalized substitution cost that maps a series of k tables from one document to l tables from the other. The initial conditions are:

$$\begin{aligned} teval_{0,0} &= 0 \\ teval_{i,0} &= teval_{i-1,0} + c_{del}(T_i^R) \quad 1 \leq i \leq m \\ teval_{0,j} &= teval_{0,j-1} + c_{ins}(T_j^G) \quad 1 \leq j \leq n \end{aligned} \tag{1}$$

and the main dynamic programming recurrence is:

$$teval_{i,j} = \min \begin{cases} teval_{i-1,j} \ + \ c_{del}(T_i^R) \\ teval_{i,j-1} \ + \ c_{ins}(T_j^G) \\ \displaystyle\min_{1 \leq k \leq i,\ 1 \leq l \leq j} \begin{bmatrix} teval_{i-k,j-l} \ + \\ c_{sub_{k:l}}(T_{i-k+1}^R \dots T_i^R, \\ T_{j-l+1}^G \dots T_j^G) \end{bmatrix} \end{cases} \tag{2}$$

for $1 \leq i \leq m$, $1 \leq j \leq n$. Once the computation has completed, $teval_{m,n}$ is a measure of the similarity of the two documents in terms of their table structure. The smaller the value, the more closely the recognition result matches the ground truth. Moreover, by tracing back the sequence of optimal decisions made when evaluating the recurrence, it becomes possible to recover a detailed interpretation of the errors that were determined to have arisen.

Note that the cost functions in the above formulation are completely general. For the tests that follow, we calculate a cost by aligning the tables in question on a line-by-line basis and tallying the number of lines that match. These are charged -1, while mismatches are charged 1. Although quite simple, this scheme appears to work well in practice. This value is then normalized to the target interval $[-1, 1]$, where -1 represents an exact correspondence (*i.e.*, perfect recognition relative to the ground truth) and 1 represents the opposite:

$$norm\_teval = 2\left(\frac{teval_{m,n} - min\_teval}{max\_teval - min\_teval}\right) - 1 \qquad (3)$$

where *min_teval* and *max_teval* are, respectively, the minimum and maximum possible distances for the comparison in question.

3 Case Study: Evaluation of an Algorithm for Table Detection

In this section, we apply the evaluation methodology described in the previous section to the problem of detecting tables. We begin by briefly reviewing an algorithm we have developed for locating tables in ASCII and scanned documents [3]. We then present the results of several small experiments, demonstrating how well the methodology works and the useful kinds of feedback it provides.

3.1 A Table Detection Algorithm

The goal of table detection is to identify the presence of one or more tables in a document and delineate their boundaries. We assume that the input is a single column document segmentable into individual, non-overlapping text lines (referred to simply as "lines" henceforth).

While it may be tempting to assume some form of delimiter (*e.g.*, that tables will always be separated from the rest of the text by at least one blank line or some minimum amount of white space), to preserve generality we do not want to make any *a priori* assumptions about where table(s) might begin or end in the input. Instead, we compute a value for all possible starting and ending positions and then choose the best possible way to partition the input into some number of tables.

Say there are a total of n lines in the input, and let $tab[i, j]$ be a measure of our confidence when lines i through j are interpreted as a single table. Let $merit_{pre}(i, [i+1, j])$ be the merit of prepending line i to the table extending from line $i + 1$ to line j, and $merit_{app}([i, j - 1], j)$ be the merit of appending line j to the table extending from line i to line $j - 1$. We have chosen white space correlation and vertical connected component analysis to model specific functions for $merit_{pre}$ and $merit_{app}$ [3]. As a rule, they return larger values for more compatible combinations and can be tuned for specific applications and/or input media. Then we define:

$$tab[i, i] = 0 \qquad 1 \le i \le n \qquad (4)$$

and

$$tab[i, j] = \max \begin{cases} merit_{pre}(i, [i+1, j]) \ + \ tab[i+1, j] \\ tab[i, j-1] \ + \ merit_{app}([i, j-1], j) \end{cases} \qquad (5)$$

for $1 \le i \le n$. This computation builds an upper triangular matrix holding the values for all possible table starting and ending positions.

We then formulate the partitioning of the input into tables as an optimization problem. Let $score[i, j]$ correspond to the best way to interpret lines i through j as some number of (*i.e.*, zero or more) tables. The computation is defined as follows:

$$score[i, i] = tab[i, i] \qquad 1 \le i \le n \qquad (6)$$

and

$$score[i, j] = \max \begin{cases} tab[i, j] \\ \max_{i \le k < j} \begin{Bmatrix} score[i, k] + \\ score[k+1, j] \end{Bmatrix} \end{cases} \qquad (7)$$

for $1 \le i < j \le n$. One way to interpret Equation 7 is that the best way to partition lines i through j into some number of tables is to: (1) treat lines i through j as coming from a single table, or (2) break the region into two subregions between lines k and $k + 1$ and consider each separately. Note that if region $[i, j]$ contains exactly one table (or a portion thereof), case (1) will apply. Otherwise, there must surely exist a k such that the region can be broken into two independent subregions (without splitting a table) and then case (2) applies. The precise decomposition can be obtained by backtracking the sequence of decisions made in evaluating Equation 7. Any region on the optimal path whose *tab* value is higher than a predetermined threshold is considered a table region. The final output of the table detection stage are the boundaries of the detected tables.

3.2 Experimental Results

In this section, we present experimental results that demonstrate the usefulness of our methodology for evaluating table detection algorithms. The test database was composed of 26 Wall Street Journal articles in text format (WSJ database) and 25 email messages. The WSJ database represented a more homogeneous collection of documents with a few classes of table structures. In sharp contrast, the email documents were a heterogeneous collection with varied layouts. Each test sample was in a single column format and contained one or more tables.

We used the edit distance measure described in Section 2 to study the performance of the table detection algorithm. A one-to-one matching was done on the test sample and its corresponding ground truth. The ground truth was generated by manually delineating the boundaries of the tables using a graphical tool we have developed, to be described later (Section 4.2). One useful feature of this evaluation scheme is that it performs error analysis by quantifying the types of detection errors. To illustrate this point, the sum of the errors due to insertion, deletion, substitution, merges and splits for the documents of each class was plotted against the threshold. The threshold refers to the value above which a region in a document is considered a table. This is shown for both the WSJ and email documents in Figure 2. As seen in Figure 2, for both sets of documents, errors at lower threshold values (left portions of the individual plots) are likely to be insertion errors, which decrease as the threshold is increased. On the other hand, at higher threshold values, deletion errors increase significantly. Substitution errors are errors in tables which have been detected but not accurately delineated. This is partly due to the header lines which have been missed by the table detection algorithm. The drop-offs in the substitution errors in both the classes of documents at higher thresholds are due to the fact that entire tables were missed at these thresholds and these were classified as deletion errors.

The evaluation scheme developed for detection, apart from providing error analysis is also helpful in improving and tuning the underlying algorithm. This is illustrated for both WSJ and email documents in Figure 3. The plot shows the average, over the number of documents in their respective class, of the structural similarity measure, $norm\_teval$ (Equation 3) across a range of threshold values. This plot helps us choose a range of threshold values to obtain acceptable detection performance. The ordinates which represent the structural similarity measure, $norm\_teval$, range from -1 to 1 with -1 being a perfect match between the test sample and its corresponding ground truth and 1 being the worst possible match.

In order to evaluate the edit distance methodology, we conducted an informal experiment with three subjects (different from the authors) who were not involved in the design of the underlying algorithms. The subjects were given six test documents along with their associated ground-truths. The test documents had the tables delineated by the detection algorithm. The ground truth, which had the tables clearly delineated, was manually generated by one of the authors. The task was to rank order the set of six documents (labeled A to F for email and a to f for WSJ) with rank 1 corresponding to the best match between the test and its associated ground truth in the group of 6. The ranking was based on the subjects' interpretation as to how good a job the algorithm did at detecting the table regions, as defined by the ground truth. The subjects were asked to use their own interpretations in deciding how similar a given recognition result was to its corresponding ground truth. These rankings were then compared to the rankings obtained by our edit distance evaluation method. The rankings from the three subjects are shown in Table 1 for the email dataset and in Table 2 for the WSJ dataset. The fifth column represents the average ranking obtained from the three subjects. The last column in each of the tables represents the ranking obtained by using our edit distance evaluation methodology. Comparing both the individual rankings and the average rankings reveals a close match to the ranking obtained using the edit distance evaluation.

Doc Index	Rankings from subjects			Average Ranking	Ranking based on Edit distance
	Sub 1	Sub 2	Sub 3		
A	1	1	1	1	1
B	2	2	2	2	2
C	3	3	3	3	3
D	5	6	5	5	4
E	6	5	6	6	5
F	4	4	4	4	6

Table 1. Ranking from the subjects and our evaluation procedure for the email dataset

Doc Index	Rankings from subjects			Average Ranking	Ranking based on Edit distance
	Sub 1	Sub 2	Sub 3		
a	2	1	1	1	1
b	3	4	2	3	2
c	1	2	4	2	3
d	4	3	3	4	4
e	5	5	5	5	5
f	6	6	6	6	6

Table 2. Ranking from the subjects and our evaluation procedure for the WSJ dataset

As part of the experiment, we also attempted to assess the relative difficulty of performing this task. The subjects felt it easier to rank-order the email documents rather than the WSJ documents. We believe that is due to the non-homogeneous nature of the email database which results in a larger range of performance variation. Based on the experimental results it is believed that similar strategies were employed by the three observers in ranking the similarities between recognition results and their corresponding ground truths. This was corroborated by the verbal feedback given by the subjects.

An important issue in performance evaluation is obtaining the ground truth. Not only is the manual process of ground truthing laborious and time consuming, it can also be non-unique. We conducted a simple experiment by using our edit distance evaluation scheme to compare the ground truth results amongst the four authors. All possible pairwise combination of four individuals gives six such correlations. Figure 4 shows the plot of the average correlation for 25 documents (email dataset). This data shows that while there is fairly close agreement in most cases, several of the documents in our test set cause difficulties even for human interpreters (deciding what counts as tables and the location of their boundaries), evidence of the challenging nature of the problem both in detection and evaluation.

4 A Methodology for Evaluating the Results of Table Structure Recognition

In this section we present our approach to evaluating algorithms that extract structure from tables [4]. We de-

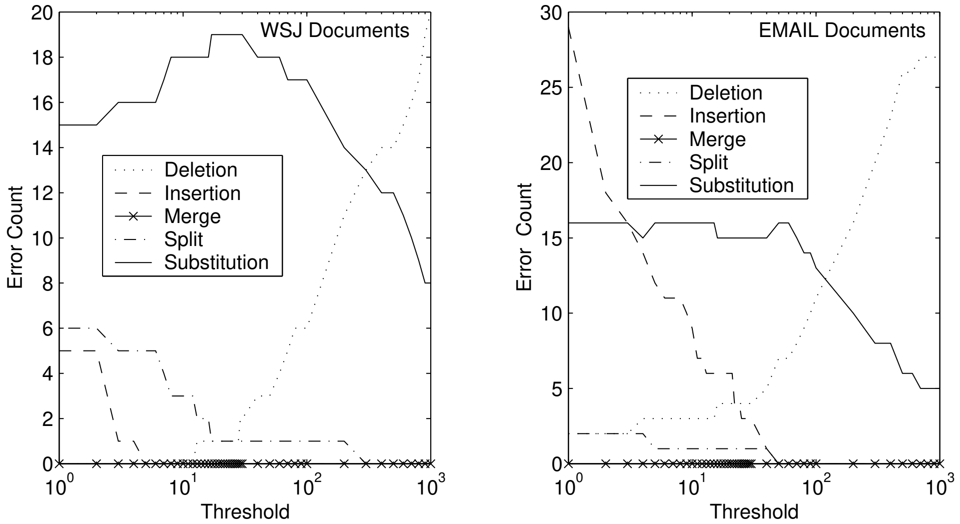


Fig. 2. Count of detection errors for the WSJ and ASCII document databases.

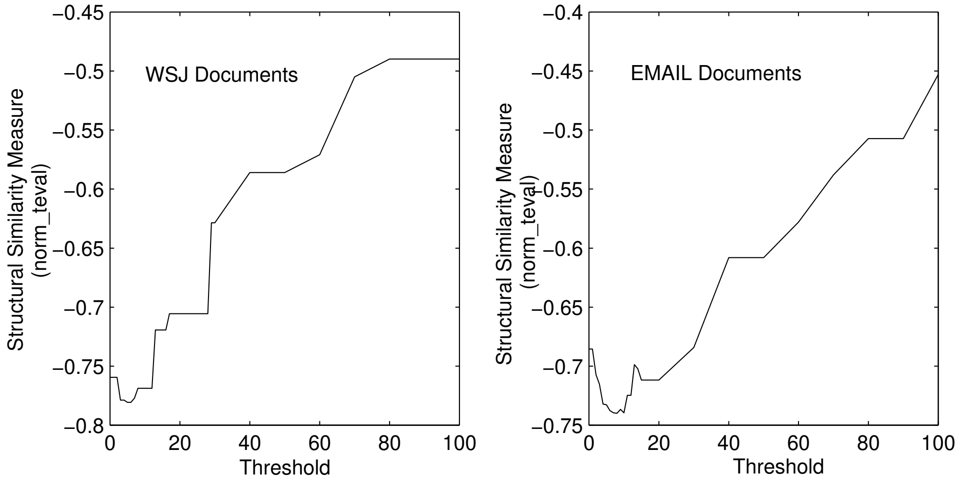


Fig. 3. Table detection evaluation measure as function of threshold for the WSJ and email document databases.

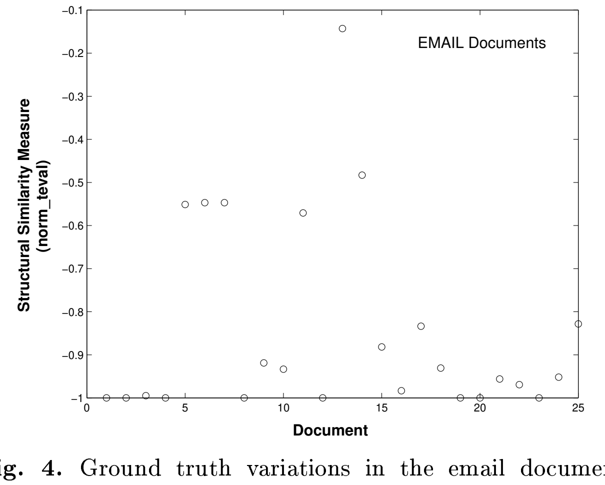


Fig. 4. Ground truth variations in the email document database.

scribe in turn the graph model we have adopted, a system we have created for ground-truthing table structure, and our procedure for comparing the output from table recognition to the corresponding truth. The paradigm we have developed is a general one, and could be applied to other algorithms that attempt to extract structure from documents.

4.1 Attribute Graph Model

The goal of the recognition step is to determine the structure of a given table and identify functional elements such as columns, rows, headers, etc. Defining a table model is itself a difficult issue, since both logical and layout conventions vary depending on the document type, the subject domain, and the medium (see [9]). Clearly, no one model will be able to represent all possible tables. We chose to base our model on Wang's formalism [20] because it provides a clean separation of content (logical model) from form (physical/presentational model), and offers a rigorous mathematical representation as the logical model.

Figure 5 illustrates the terminology we use for table structure. At the lowest level, a table contains two types of cells: *Dcells* for data cells and *Acells* for access cells. These cells are organized into columns and rows. The column headers are grouped into a region named the *box*, and the row headers are grouped into a region called the *stub*. The header for the box/stub is called the *box head* or *stub head*. The collection of all the *Dcells* comprises the *body*. The body is the only required region of a table; *Acells* and all header regions are optional.

While it is traditional to regard document analysis results, and segmentation results in particular, as tree-structured, we have adopted a slightly more general representation, a directed acyclic attribute graph (DAG) as in [17]. This flexibility is important both because there are real-life tables that fall outside the Wang model, and because the output from an imperfect recognition process may not necessarily correspond to a legal instance of a table.

There are two basic classes of nodes in our table DAG: *leaf nodes* which have no children and which contain content corresponding to a specific region on the page (*i.e.*, one or more text strings), and *composite nodes* which are simply unordered collections (sets) of previously-defined leaf and composite nodes. Every node has an optional label. There is, however, no rigid policy enforcing how nodes must be labeled relative to one another. Instead, conventions can be developed on a per-application basis. Indeed, our plan is to apply this same general formalism to other document analysis and computer vision tasks in the future.

For the complete graph corresponding to the table depicted in Figure 6, there are 28 leaf nodes labeled *Dcell* and 14 leaf nodes labeled *ACell*, while *Row* and *Column* are composite nodes. Note that the lower rightmost *DCell* (with content "$-2\ 3/8$") is a child both of a node labeled *Row* (headed by "PINK LTD") and of another node labeled *Column* (headed by "YESTERDAY'S CHANGE"); hence, this graph is not a tree.

4.2 Ground-Truthing

To enable the viewing of document analysis results and to support the ground-truthing process, we have developed an interactive tool we call *Daffy* for browsing and editing table DAG's. The user interface portions of Daffy are written in Tcl/Tk, a powerful and well-known scripting language developed by Ousterhout [13].

Daffy makes it possible to:

1. display and edit graphical mark-up,
2. define new mark-up types,
3. examine hierarchical structure,
4. print and save PostScript page images,
5. run algorithm animation scripts for visualizing the effects of document analysis.

Input is accepted in both image (TIF) and text (ASCII) formats.

Figure 7 presents a screen snapshot of Daffy running on an SGI O2 workstation. The main window, on the right, shows the same sample table document as shown in Figure 5. Several layers of mark-up are visible – on-screen these are displayed in color and are much more legible. Structure corresponding to the leftmost table column which the user has selected is displayed in the child window on the left in the snapshot.

Daffy supports the full generality of the graph model described in the preceding section. In particular, it manages and updates the table DAG across operations that include adding and deleting leaf nodes, grouping and ungrouping to create composite nodes, moving and resizing nodes, copy-and-pasting, editing type definitions, etc. Consistency of the graph is maintained automatically without placing burdensome restrictions on the user.

4.3 Graph Probing

Given the table DAG's for a recognition result and its corresponding ground truth, it is natural to consider comparing the two as a way of determining how well the algorithm has done. Attempting to compare the graphs directly, however, gives rise to two dilemmas. The first is that a solution would imply a solution to the graph isomorphism problem which is not likely to have an efficient algorithm [2]. While heuristics exist that are sometimes fast, their worst-case behavior is still exponential (see, *e.g.*, [11]). Hence, the problem remains a difficult one.

The other obstacle is that there may be several different ways to represent the same table as a graph, all equally applicable. Minor discrepancies in labeling and/or structure could create the appearance that two graphs are dissimilar when in fact they are functionally equivalent from the standpoint of the intended application. Forcing one graph to correspond to the other through a series of rigidly defined editing operations obscures this important point.

At the other end of the spectrum, we could embed our table recognition algorithm in a query-based table processing system ([4]), and measure the performance of the complete system on a specific task from a user's perspective: Does it provide the desired information? (this is "goal-directed evaluation" as discussed by Nagy in [12]). This approach has its own shortcomings, however, as it limits the generality of the results and makes it difficult to identify the precise source of errors that arise when complex processes interact.

We have developed a third methodology that lies midway between these two. We work directly with the graph representation. However, instead of trying to match the graphs under a formal editing model, we probe their structure and content by asking relatively simple queries that mimic, perhaps, the sorts of operations that might arise in a real application.

Conceptually, the idea is to place each of the two graphs under study inside a "black box" capable of evaluating a set of graph-oriented operations (*e.g.*, returning a list of all the leaf nodes, or all nodes labeled in a certain way). We then pose a series of probes and correlate the responses of the two systems. A measure of their similarity is the number of times their outputs agree. Note

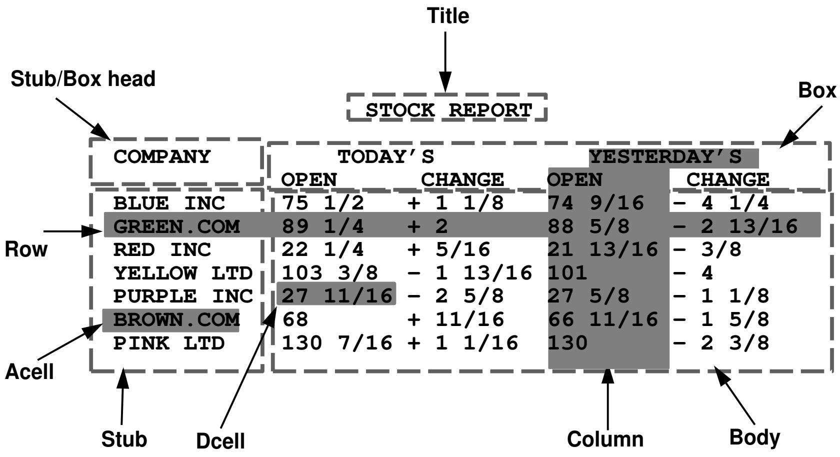


Fig. 5. Table terminology (adapted from Wang's Ph.D. thesis [20]).

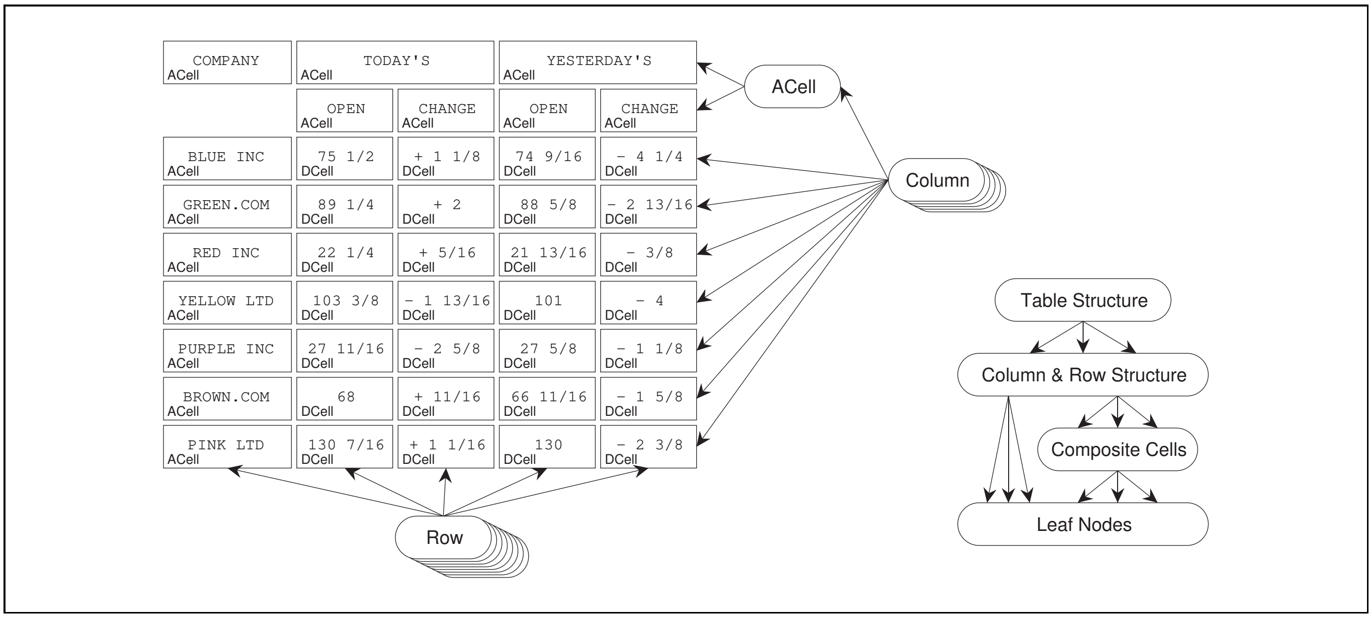


Fig. 6. Graph representation for table recognition results.

that it is essential the probes themselves have simple answers that are easily compared. They might return, for example, a count of the number of nodes satisfying a certain property (*e.g.*, possessing a particular label), or the content of a designated leaf node. The probing becomes recursive if the target of a probe is a graph itself (*i.e.*, a composite node). The intention is that this probing process abstracts the access of content away from the specific details of the graph's structural representation. We define a *discriminating probe* to be a probe that demonstrates a difference between two graphs. Two fundamental questions are of interest: (1) For two graphs that are different, does there exist at least one discriminating probe? and (2) Over the entire set of probes, how many are discriminating? The first of these reflects the graph isomorphism problem. The second can serve as a measure of how similar the two graphs are. To make this more explicit, we define the *agreement* between two probe sets to be:

$$agreement \equiv 1.0 \frac{\#\ of\ discriminating\ probes}{total\ \#\ of\ probes} \tag{8}$$

If the agreement is 1.0, then the two graphs are indistinguishable with respect to the probe set in question. Values less than 1.0 indicate some degree of similarity falling short of a perfect match. Our aim is to equate "agreement" with the traditional concept of "accuracy".

While the paradigm is open-ended, for our table recognition work we have defined three categories of probes for the table recognition problem:

Class 0 These probes count the number of occurrences of a given type of node in the graph. Referring again to Figure 6, a typical Class 0 probe might be paraphrased as: "How many nodes labeled 'Column' does the graph have?" The answer in this case is "5."

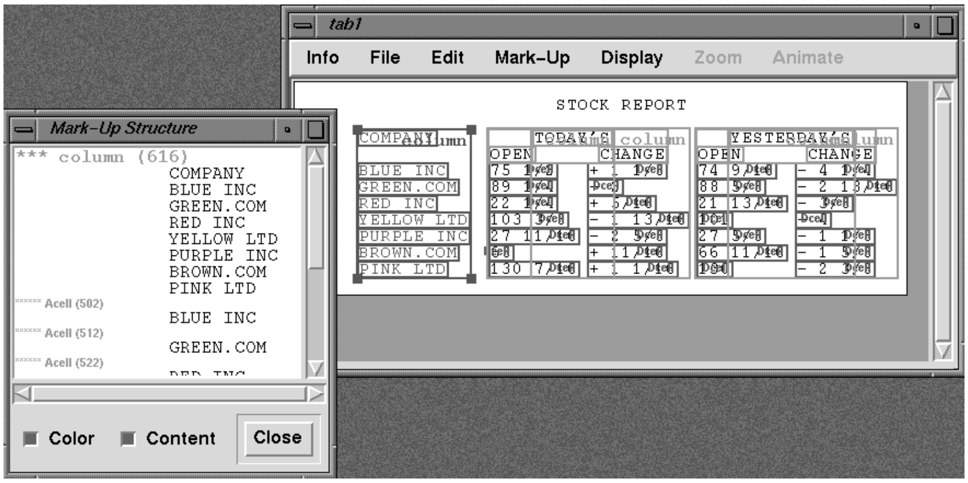


Fig. 7. Daffy screen snapshot.

Class 1 These probes combine content and label specifications. Currently they apply only to leaf nodes. A representative Class 1 probe might be: "How many leaf nodes labeled 'Acell' with content 'OPEN' does the graph have?" The reply here is "2."

Class 2 These are the most sophisticated probes we have implemented to date. Class 2 probes mimic simple database-type queries, although phrased entirely in terms of graph manipulations. For a given target node, keys that uniquely determine its row and column are identified. These are used to index into the graph, retrieving the content of the node (if any) that lies at their intersection. An example of a Class 2 probe for the graph in Figure 6 is: "What is the value of 'TODAY'S OPEN' for 'RED INC'?" The response would be "22 1/4."

The generation of a probe set is based on one or the other of the graphs in question. That graph will obviously return the definitive responses for all of the probes in the set, while the other graph will do more or less well depending on how closely it matches the first. We then repeat the process from the other direction, generating the probe set from the second graph and tallying the responses for both. The probes are synthesized automatically, working from the table DAG that is output from the recognition and ground-truthing processes. For specifying probes, we have implemented a graph-oriented query language embedded in a general-purpose programming language; this offers a great deal of power and flexibility.

Our experience so far, although brief, suggests that this could be an effective paradigm for evaluating table processing results (and perhaps even document analysis algorithms in general).

5 Case Study: Evaluation of an Algorithm for Table Structure Recognition

In this section, we apply the evaluation methodology described in the previous section to the problem of recognizing table structure. We begin by briefly describing an algorithm we have developed for parsing tables [5]. We then present experimental results that illustrate how the methodology works in practice.

5.1 A Table Structure Recognition Algorithm

The goal of the recognition step is to determine the structure of a given table and identify functional elements such as columns, rows, headers, etc.

Column Segmentation The input to the column segmentation step is the boundaries of a detected table region. It is assumed that this region contains all or nearly all the lines occupied by the body of the table. Depending on the layout of the particular table, it could also contain lines from column headers. If the table has row headers, they are considered included in the region as well. At this point, no distinction is made between the column of row headers (stub) and a "normal" column.

Hierarchical clustering is applied to all words in this region to identify their likely groupings. Such groupings are represented as a binary tree, where the leaves represent words, the root represents the whole body, and the intermediate nodes represent nested groupings at different levels. This cluster tree is constructed in a bottom-up manner [7] – first the leaf level clusters are generated where each word belongs to a unique cluster, then the two clusters with the minimum inter-cluster distance are merged into a new cluster. The merging process is repeated recursively until there is only one cluster left. In this clustering process, the distance between two words is defined to be the Euclidean distance applied to the horizontal coordinates of the beginning and the end of the words; the distance between two clusters is defined to be the maximum inter-cluster word distance.

The cluster tree generated in the above manner represents the hierarchical structure of the table body in terms of vertical groupings of words. Each cut across the tree provides one way of clustering these words. We need to find the cut such that each resulting cluster corresponds to a column. Such a cut is called the *column cut*.

The column cut is found using a breadth-first traversal of the cluster tree starting from the root, and by applying heuristics such as the spacing between columns tends to be evenly distributed within a table.

Note that even though the distance between words is currently defined only in terms of horizontal distance, the nature of the hierarchical clustering algorithm insures that the proposed column segmentation algorithm can handle imperfect vertical alignment very well. This ability is demonstrated in Figure 8 with a table containing ragged columns as well as columns with a small gap inside. As shown in the figure, all columns are clustered properly.

Header Detection and Row Segmentation As mentioned before, headers including box (containing column headers), stub (containing row headers) and stub/box head are all considered optional. The potential headers are identified using a lexical distance measure and assuming typical layout rules for headers used in most tables. To capture the potential hierarchical structure, headers are represented by a tree structure which is initialized with the root representing the box, and k leaf nodes corresponding to the k columns. We define the *joint span* of a list of n spans $p_i = (s_i, e_i), i = 1 \ldots n$ as $p_{1 \cdot n} = [min(s_i, i = 1 \ldots n), max(e_i, i = 1 \ldots n)]$. Once a higher level header is found, the corresponding intermediary node is generated, and the joint span of its subsidiary nodes is used to analyze the next line. Figure 9 shows the box of the table in Figure 5 represented as a tree. This tree is then traversed to assign headers to each column (higher lever headers are shared by more than one column).

Row segmentation is carried out after header detection. The difficulty here is that some or all of the cells in a table row could contain more than one line and there is often no obvious separator between rows. The following heuristics have been adopted by our algorithm: 1) A blank line is always a row separator; 2) If a line contains non-empty strings for the stub (if it exists) and at least one other column, or if it contains non-empty strings for a majority of columns, then it is considered a *core line*, otherwise it is considered a *partial line*. Each table row contains one and only one core line and a partial line is always grouped with the core line above it. Occasionally there are tables where partial lines are grouped with the core line below. Such cases could be detected using statistical syntax analysis, such as N-grams.

5.2 Experimental Evaluation

In this section, we present experimental results that demonstrate the usefulness of our method for evaluating table recognition algorithms. As before, the test database was composed of 26 Wall Street Journal articles in text format (WSJ database) and 25 email messages.

We applied the graph probing based evaluation method to study the performance of the table recognition algorithm. Tables along with the detected boundaries were input to the table recognition algorithm. Tables split or merged as a result of our table spotting stage were removed for the recognition evaluation. A DAG representation was used for functional elements like headers, rows and columns obtained by table recognition. The test documents used for table recognition were ground-truthed manually using the Daffy interface (Section 4.2). Tables 3 and 5 give the counts of leaf nodes, Acells, Dcells, rows and columns for each table in the WSJ and email datasets respectively. Shown are the counts both for the manually labeled ground truth and the recognition result.

As explained in Section 4, probing was done bidirectionally, *i.e.*, the recognition result was probed based on its corresponding ground truth and vice-versa. This involved generating a set of queries to probe the various nodes of the graph which represent the table structure. Three classes of queries were generated in the probing experiment. The total number of queries generated for each document is tabulated for both the WSJ and email documents in Tables 4 and 6. The agreement of the probes for each of the three classes is plotted in Figure 10 for the WSJ documents and Figure 11 for the email documents. Also superimposed on the plot is the total agreement (combining all the classes).

As seen from Figure 10, the class agreement are highest for Documents 8 and 9. Looking at the corresponding rows in Table 3, it can be seen that the recognition results match well with the corresponding ground truth in terms of the number of nodes, Acells, Dcells, rows and columns. On the other hand Document 7 has one of the lowest scores in this data set. The corresponding document is shown in Figure 12. One plausible interpretation of the table is that it has three columns, with the leftmost column indicating the rank, the center column corresponding to names and the third column corresponding to the vote count. The table is complicated by the fact that a few names share the same rank (the 8th, 11th and 14th). This can be deduced only from a semantic analysis of the text. The recognized result, however, has merged the first two columns as indicated in Table 3, and Figure 12 shows only two columns. This difference in the layout structure between the recognition result and the ground truth leads to its poor performance. It must be noted that several plausible interpretations of a single table (ground truths) can be made and this makes the evaluation task extremely challenging.

Figures 10 and 11 contain documents which have either zero or no Class 2 scores. One reason was that the table in such documents had only two rows (a row containing headers and a row containing data) and no Class 2 queries were generated. Another reason is that, currently, our algorithm does not capture hierarchical row headers and this leads to incorrect recognition of table structure in such tables.

As a first step towards correlating our evaluation method based on graph probing with a user's perception of the quality of the recognition results, we conducted an informal experiment. The experiment was performed with three subjects who were not involved in the design of the underlying algorithms. The subjects were shown

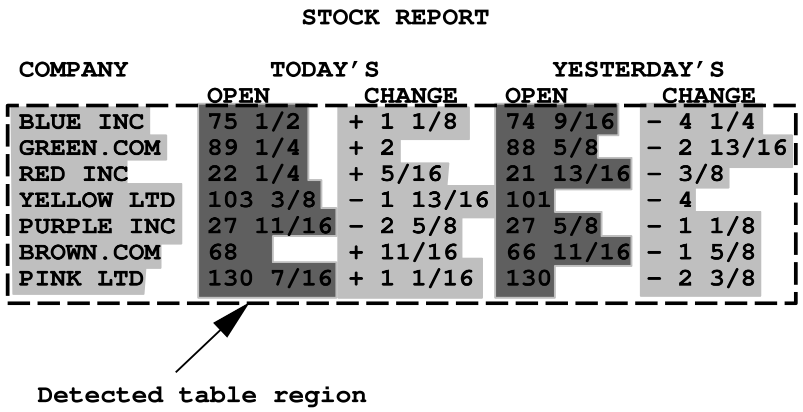


Fig. 8. A detected table region and its column segmentation using hierarchical clustering.

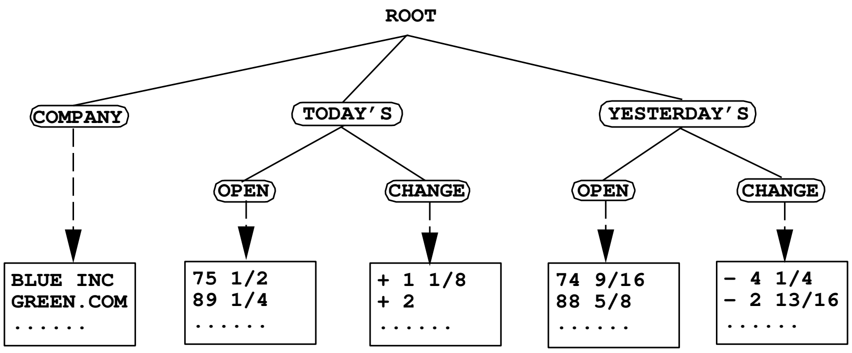


Fig. 9. The tree representation of the box of the table in Figure 5.

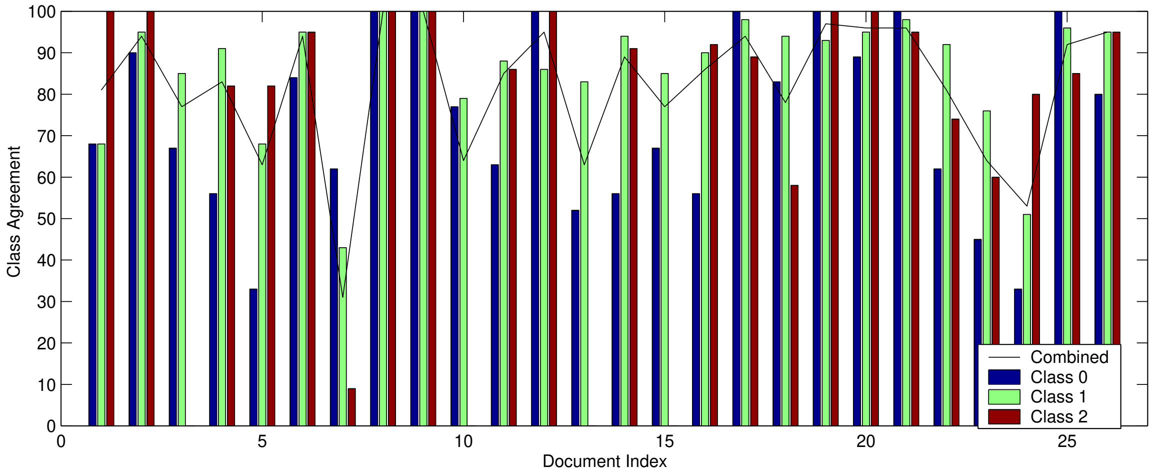


Fig. 10. Class agreement for documents in the WSJ database.

six test documents (chosen from each of the datasets, labeled A through F for email documents and a through f for WSJ documents), along with their associated ground-truths using the Daffy interface. The six test documents were shown with the table structure marked-up by the recognition algorithm. The corresponding ground-truth was the same document with the table structure marked-up by one of the authors. The task was to rank-order the six documents based on how good a job the algorithm performed at recognizing the table structure regions, as defined by the ground-truth, with rank one corresponding to the best match in the group of six. The subjects were asked to look at row structure, column structure, Acells and Dcells only, and to use their own interpretations in ranking the similarities between recognition results and their corresponding ground truths.

The rankings from the three subjects are shown in Table 7 for the email dataset and in Table 8 for the Wall Street dataset. The fourth column represents the computed average ranking obtained by the three subjects. Also shown in the last columns for both the tables are the rankings obtained by using our graph probing

Doc	Ground Truth					Recognition Result				
Index	Leaf Nodes	Acells	Dcells	Rows	Cols	Leaf Nodes	Acells	Dcells	Rows	Cols
1	24	3	21	8	3	24	10	14	8	3
2	17	8	9	4	4	17	8	9	4	4
3	16	12	4	2	4	16	13	3	2	4
4	33	15	18	7	4	36	16	20	8	4
5	67	43	24	14	3	64	40	24	13	3
6	75	36	39	22	3	77	38	39	22	3
7	44	12	32	16	3	32	16	16	16	2
8	33	15	18	10	3	33	15	18	10	3
9	45	15	30	11	4	45	15	30	11	4
10	85	61	24	24	2	58	34	24	24	2
11	80	26	54	10	7	74	26	48	9	7
12	6	3	3	3	2	6	3	3	3	2
13	56	39	17	17	2	44	26	18	18	2
14	56	28	28	28	2	58	31	27	27	2
15	16	12	4	2	4	16	13	3	2	4
16	33	15	18	7	4	37	17	20	8	4
17	45	15	30	11	4	45	15	30	11	4
18	38	22	16	9	3	32	16	16	9	3
19	14	6	8	5	3	14	6	8	5	3
20	19	7	12	4	5	19	7	12	4	5
21	45	15	30	11	4	45	15	30	11	4
22	39	19	20	11	3	39	17	22	11	4
23	72	42	30	16	3	61	31	30	16	3
24	15	5	10	6	2	16	12	4	5	2
25	22	6	16	5	5	22	6	16	5	5
26	44	18	26	14	3	43	17	26	14	3

Table 3. Node counts for the WSJ database.

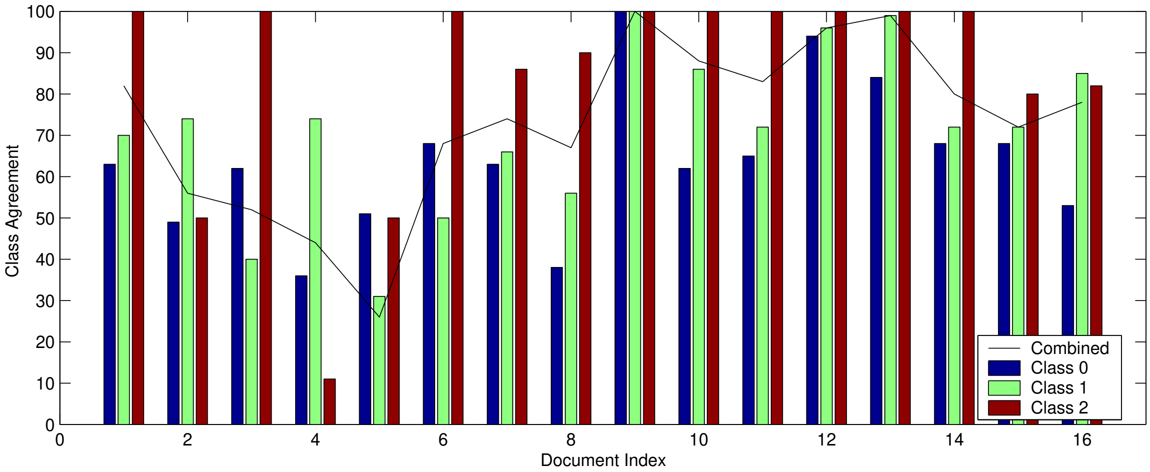


Fig. 11. Class agreement for documents in the email database.

technique to compare the results from the recognition algorithm to the ground-truth. As seen from the tables, the subjects' average ranking compares well with that obtained by our evaluation measure. Individual differences in ranking among subjects are due to the fact that the subjects were asked to use their own interpretations in deciding how similar a given structure recognition result was to its corresponding ground truth. The inconsistency between subjects suggests that it is difficult to judge structure recognition performance.

6 Conclusions

This paper has presented evaluation methods for both the table detection and recognition tasks. The detection task was evaluated based on the concept of edit-distance. Random graph probing was introduced as a new paradigm for evaluating the performance of a table recognition system.

In the area of evaluation, there remain many directions for future work. For example, there is a need for more rigorous studies of the correlation of our evaluation measures with a users' perceptions of the quality of the recognition results. Another direction for further study is concerned with the generation of more sophisticated

Doc Index	Ground Truth			Recognition Result		
	Class 0	Class 1	Class 2	Class 0	Class 1	Class2
1	10	25	24	8	25	24
2	10	19	14	10	20	14
3	10	13	0	8	13	0
4	9	34	27	9	32	24
5	18	70	12	19	75	10
6	16	86	54	15	83	54
7	8	33	25	8	45	37
8	8	31	27	8	31	27
9	9	46	42	9	46	42
10	12	65	32	8	56	0
11	8	68	56	12	82	62
12	8	7	6	8	7	6
13	12	49	26	8	40	0
14	10	61	38	8	57	44
15	10	13	0	8	13	0
16	9	35	27	9	32	24
17	9	46	42	9	46	42
18	9	31	23	8	29	16
19	9	15	14	9	15	14
20	9	20	19	9	21	19
21	9	46	42	9	46	42
22	8	37	32	8	35	30
23	12	63	41	22	84	30
24	9	14	8	9	18	10
25	9	23	22	9	23	22
26	9	44	31	10	45	33

Table 4. Probe counts for the WSJ database.

Doc Index	Ground Truth					Recognition Result				
	Leaf Nodes	Acells	Dcells	Rows	Cols	Leaf Nodes	Acells	Dcells	Rows	Cols
1	64	4	60	13	5	64	16	48	13	5
2	14	0	14	7	2	23	7	16	7	4
3	50	0	50	10	5	50	10	40	10	5
4	70	15	55	12	6	88	14	74	11	8
5	54	0	54	6	9	24	6	18	6	4
6	20	0	20	10	2	20	10	10	10	2
7	24	3	21	7	3	21	9	12	7	3
8	25	5	20	5	5	21	6	15	6	4
9	45	18	27	10	4	45	18	27	10	4
10	70	0	70	7	10	70	7	63	7	10
11	27	0	27	7	5	27	7	20	7	5
12	14	2	12	2	7	14	2	12	2	7
13	49	13	36	7	7	49	13	36	7	7
14	36	0	36	9	4	36	9	27	9	4
15	50	0	50	5	10	52	5	47	5	10
16	27	12	15	6	4	25	8	17	7	4

Table 5. Node counts for the email database.

probes for the evaluation of table structure and for extending this paradigm to evaluate other algorithms that attempt to extract structure from documents.

Another line of future work is concerned with the problem of determining the ground-truth used for performance evaluation [10]. One of the difficulties encountered in dealing with complex table layouts is that human experts are not always in agreement over what constituted ground-truth. Thus, it is not always clear that a single "ground-truth" exists.

7 Acknowledgments

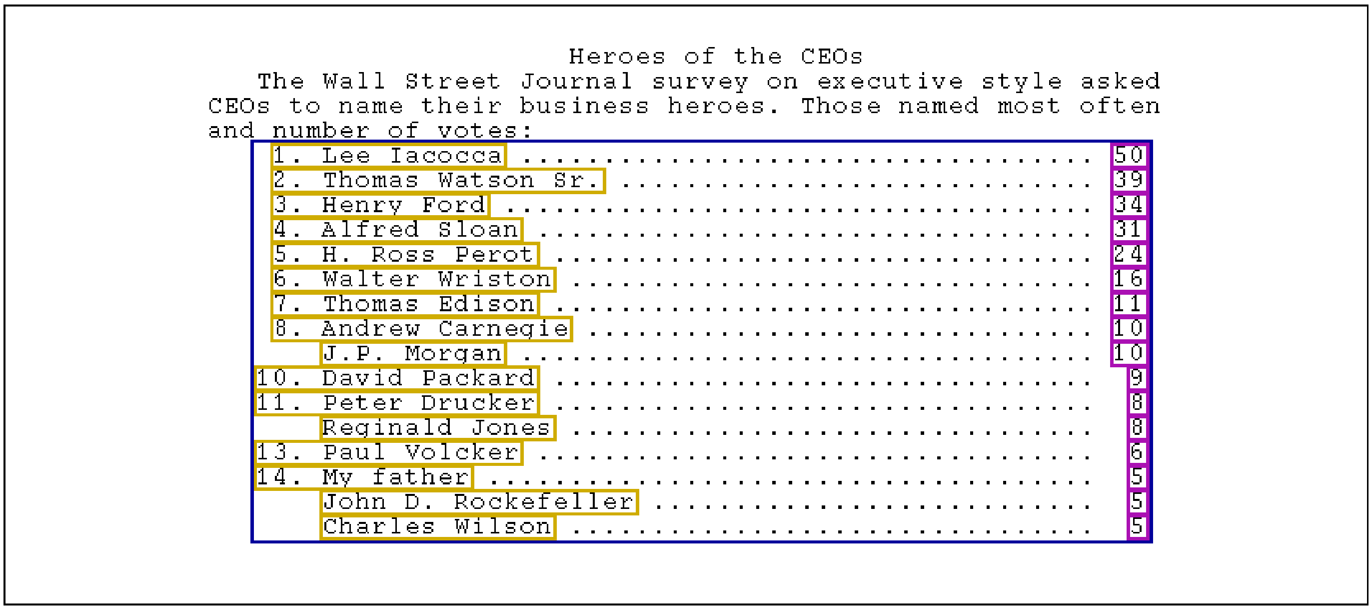
The authors would like to thank Hengbin Luo, April Rasala, and Ann Vilasuso for their help in evaluating the techniques described in this paper.

References

1. S. Agne, M. Rogger, and J. Rohrschneider. Benchmarking of document page segmentation. In *Proceedings of Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging)*, volume 3967, pages 165–171, San Jose, CA, January 2000.

Doc	Ground Truth			Recognition Result		
Index	Class 0	Class 1	Class 2	Class 0	Class 1	Class2
1	9	64	49	7	64	49
2	8	24	23	5	14	14
3	7	50	10	6	51	10
4	14	94	70	9	71	60
5	7	24	18	5	54	54
6	7	20	12	5	20	12
7	9	21	21	7	24	21
8	9	22	20	7	25	25
9	9	43	36	9	43	36
10	7	70	35	6	71	35
11	8	28	25	5	27	25
12	7	14	6	8	15	6
13	9	49	47	10	51	47
14	7	36	19	5	36	19
15	7	51	13	5	50	20
16	8	25	25	11	29	22

Table 6. Probe counts for the email database.



Heroes of the CEOs
The Wall Street Journal survey on executive style asked
CEOs to name their business heroes. Those named most often
and number of votes:
1. Lee Iacocca 50
2. Thomas Watson Sr. 39
3. Henry Ford 34
4. Alfred Sloan 31
5. H. Ross Perot 24
6. Walter Wriston 16
7. Thomas Edison 11
8. Andrew Carnegie 10
J.P. Morgan 10
10. David Packard 9
11. Peter Drucker 8
Reginald Jones 8
13. Paul Volcker 6
14. My father 5
John D. Rockefeller 5
Charles Wilson 5

Fig. 12. Daffy screen snapshot of Document 7 from the WSJ database.

Doc	Rankings from subjects			Average	Ranking based on
Index	Sub 1	Sub 2	Sub 3	Ranking	Graph Probing
A	1	1	1	1	1
B	4	2	2	2	2
C	2	5	5	4	3
D	3	3	6	3	4
E	6	6	3	6	5
F	5	4	4	5	6

Table 7. Ranking from the subjects and our evaluation procedure for the email dataset.

2. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, CA, 1979.
3. J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Medium-independent table detection. In *Proceedings of Document Recognition and Retrieval VII (IS&T/SPIE Electronic Imaging)*, pages 291–302, San Jose, CA, January 2000.
4. J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. A system for understanding and reformulating tables. In *Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems*, pages 361–372, Rio de Janeiro, Brazil, December 2000.
5. J. Hu, R. Kashi, D. Lopresti, and G. Wilfong. Table structure recognition and its evaluation. In *Proceedings of Document Recognition and Retrieval VIII*, volume 4307, pages 44–55, San Jose, CA, January 2001.
6. Y. Ishitani. Model matching based on association graph for form image understanding. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 287–292, Montréal, Canada, Au-

Doc Index	Rankings from subjects			Average Ranking	Ranking based on Graph Probing
	Sub 1	Sub 2	Sub 3		
a	1	1	1	1	1
b	2	6	2	3	2
c	3	2	3	2	3
d	6	5	4	6	4
e	4	4	6	5	5
f	5	3	5	4	6

Table 8. Ranking from the subjects and our evaluation procedure for the WSJ dataset.

gust 1995.
7. A. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.
8. J. Kanai. Automated performance evaluation of document image analysis systems: Issues and practice. *International Journal of Imaging Science and Technology*, 7:363–369, 1996.
9. D. Lopresti and G. Nagy. Automated table processing: An (opinionated) survey. In *Proceedings of the Third IAPR International Workshop on Graphics Recognition*, pages 109–134, Jaipur, India, September 1999.
10. D. Lopresti and G. Nagy. Issues in ground-truthing graphic documents. In *Proceedings of the Fourth IAPR International Workshop on Graphics Recognition*, pages 59–72, Kingston, Ontario, Canada, September 2001.
11. B. T. Messmer and H. Bunke. Efficient error-tolerant subgraph isomorphism detection. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition*, pages 231–240. World Scientific, Singapore, 1995.
12. G. Nagy. Document image analysis: Automated performance evaluation. In A. L. Spitz and A. Dengel, editors, *Document Analysis Systems*, pages 137–156. World Scientific, Singapore, 1995.
13. J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, MA, 1994.
14. C. Peterman, C. H. Chang, and H. Alam. A system for table understanding. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 55–62, Annapolis, MD, 1997.
15. P. Pyreddy and W. B. Croft. TINTIN: A system for retrieval in text tables. Technical Report UM-CS-1997-002, University of Massachusetts, Amherst, January 1997.
16. A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, May/June 1983.
17. R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*, chapter 8. John Wiley & Sons, New York, NY, 1992.
18. J. H. Shamalian, H. S. Baird, and T. L. Wood. A retargetable table reader. In *Proceedings of 4th International Conference on Document Analysis and Recognition*, pages 158–163, Ulm, Germany, August 1997.
19. L. Shapiro and R. M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):90–94, January 1985.
20. X. Wang. *Tabular abstraction, editing, and formatting.* PhD thesis, University of Waterloo, 1996.
21. B. A. Yanikoglu and L. Vincent. Ground-truthing and benchmarking document page segmentation. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 601–604, Montréal, Canada, August 1995.