Table Detection Across Multiple Media

Jianying Hu *Ram Kashi* *Daniel Lopresti* *Gordon Wilfong*

Bell Laboratories Lucent Technologies, 700 Mountain Ave., NJ 07974
Phone: 908-582-5660 Fax: 908-582-7308
{jianhu,ramanuja,dpl,gtw}@research.bell-labs.com

1 Introduction

Tables are an important means for communicating information in written media, and understanding such tables is a challenging problem in document layout analysis. Table understanding can be broken into two logical steps: table detection and table recognition. Much existing work on tables described in the literature addresses the latter step and assumes that the table has already been identified and segmented out from the input (or that identifying the table is trivial – *e.g.*, the whole document is the table).

Most prior research on the problem of table detection has concentrated on detecting tables in scanned images, and the vast majority depends on the presence of at least some ruling lines (*e.g.*, [LV92, Hir95]). Symbolic tables, however, are becoming increasingly important as well. These may originate either in ASCII form (*e.g.*, as part of an e-mail message), or as the result of saving a "richer" document (*e.g.*, an HTML page) in "text-only" format. More often than not, ASCII tables contain no ruling lines whatsoever, depending only on the 2-D layout of the cell contents to convey the table's structure. Kieninger [Kie98] describes a system for parsing tables in ASCII or paper documents based on a straightforward clustering technique. A method based on $LR(k)$ parsing is mentioned in [KW98] for a specific class of tables (financial tables). However, neither of these works explicitly considers the detection of tables.

In this paper, we describe a technique for detecting tables that does not rely on ruling lines and has the desirable property that an identical high-level approach can be applied to tables expressed as ASCII text and those in image format. We also present the results of a preliminary experimental evaluation.

2 Algorithms

In this section we describe our approach to solving the table detection problem. It consists of: (1) a high-level framework that determines the optimization problem and algorithms for its solution, and (2) table quality measures that can be tuned for specific applications and/or input media. We assume that the input is a single column document segmentable into individual, non-overlapping text lines (referred to simply as "lines" henceforth). Our general framework is independent of the table quality measures and, in particular, independent of the input medium.

2.1 High-Level Framework

While it may be tempting to assume some form of delimiter, 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 line 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 j, and $merit_{app}([i, j-1], j)$ be the merit of appending line j to the table extending from line i to line $j-1$. Specific functions for $merit_{pre}$ and $merit_{app}$ will be described in the next subsection, but as a rule they return larger values for more compatible matchings. Then we define the initial conditions to be $tab[i, i] = 0$ for $1 \leq i \leq n$, and the main recurrence to be:

$$tab[i, j] = \max \begin{cases} tab[i+1, j] \ + \ merit_{pre}(i, [i+1, j]) \\ tab[i, j-1] \ + \ merit_{app}([i, j-1], j) \end{cases} \quad 1 \leq i \leq n, \ i \leq j \leq n \tag{1}$$

This computation builds an upper triangular matrix holding the values for all possible table starting and ending positions.

The best (*i.e.*, highest quality) table in the input can then be found by searching the $tab[i, j]$ matrix for its maximum value. The second-best table can then be located by excluding the region $[a, b]$ occupied by the best table and searching the regions $[1, a-1]$ and $[b+1, n]$ for the next highest value, etc.

This greedy approach could, however, lead to an unsatisfactory solution when the document contains more than one table. This situation can be remedied by formulating 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 initial conditions are defined to be $score[i, i] = 0$ for $1 \leq i \leq n$, and the recursive computation is then defined as:

$$score[i, j] = \max \begin{cases} tab[i, j] \\ \max_{i \leq k < j} \{score[i, k] \ + \ score[k+1, j]\} \end{cases} \quad 1 \leq i < j \leq n \tag{2}$$

Whereas $tab[i, j]$ represents the quality of the region covering lines i through j when interpreted as a *single table*, $score[i, j]$ represents the best way to decompose this region into *some number of tables* (*i.e.*, zero or more) along with separate, non-table text lines. The precise decomposition can be obtained by backtracking the sequence of decisions made in evaluating Equation 2; this gives us the globally optimal strategy for partitioning the input into however many tables it may have contained.

2.2 Table Quality Measures

One of the simplest quality measures imaginable is the degree to which the white space in the line to be added correlates to the spacing in the presumed table. On a given line, we distinguish between *inside spaces* and *outside spaces*; the former have at least one non-space character to the left and to the right, while the latter have no non-space character on one or both sides.

In the case of ASCII input, let α and β correspond to specific character positions on two lines. We define:

$$acorr(\alpha, \beta) = \begin{cases} 1 & \text{if } \alpha \text{ and } \beta \text{ are both inside space characters} \\ -1 & \text{if exactly one of } \alpha \text{ or } \beta \text{ is an inside space} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Let $atext[i, k]$ be the input text where the first index designates the line and the second the character on that line. For two lines i and j, we pad the shorter line with (outside) spaces to match the length of the longer line. Let this length be m. Now we define:

$$lncorr(i, j) = \sum_{k=0}^{m} acorr(atext[i, k], atext[j, k]) \tag{4}$$

This is a measure of the correlation between the white space "streams" on the two lines in question. A positive value indicates the lines correlate well (*i.e.*, they may potentially belong to the same table).

We now define our first set of merit functions:

$$merit1_{pre}(i, [i+1, j]) = \sum_{k=i+1}^{j} \frac{1}{e^{\gamma(k-i-1)}} \cdot lncorr(i, k) \tag{5}$$

and

$$merit1_{app}([i,j-1],j) = \sum_{k=i}^{j-1} \frac{1}{e^{\gamma(j-1-k)}} \cdot lncorr(k,j) \tag{6}$$

where γ is a constant that determines the exponential decay. This gives one possible criterion for judging the quality of ASCII tables.

In the case of image input, we can define an analogous measure computed from word or character bounding boxes, obtained from low level layout analysis [Bai92]. The details are straightforward and we omit them here.

Another possible way of formulating merit functions is through vertical connected component analysis (VCCA). The basic units in VCCA are words. Words in ASCII text are sequences of characters delineated by spaces. In printed text, word bounding boxes can be derived from low-level layout analysis as mentioned above. For this application, the vertical "connectedness" is defined such that two words on adjacent lines are considered connected if they overlap significantly and have similar widths. The transitive closures of all pairs of vertically connected words define the vertical connected components. These can be efficiently computed using an equivalence class algorithm [HSAF93]. Kieninger used a similar operation for "block" expansion, but with no constraints on the widths of the overlapping words and the overlap itself [Kie98]. In order to overcome occasional gaps in a table caused by missing items or double-line row headings, simple morphological operations of horizontal dilation followed by horizontal erosion [Dav97] are carried out before VCCA is applied.

After VCCA, each line is assigned a list of records containing the labels (and any other necessary attributes) of connected components intersecting this line. For any pair of lines i and j, let $S(i,j)$ denote the set of connected components intersecting both lines, called *shared components*; and $D(i,j)$ denote the set of connected components intersecting only one of the lines, called *unique components*. We now define a second set of table quality measures:

$$merit2_{pre}(i,[i+1,j]) = \sum_{k\in S(i,i+1)} w_s(k) - \sum_{k\in D(i,i+1)} w_d(k) \tag{7}$$

and

$$merit2_{app}([i,j-1],j) = \sum_{k\in S(j-1,j)} w_s(k) - \sum_{k\in D(j-1,j)} w_d(k) \tag{8}$$

where $w_s(k)$ and $w_d(k)$ are weights assigned to connected component k depending on whether it is a shared or unique component. These weights could in general be functions of the attributes (*e.g.*, height) of the connected component, but in our initial investigations they are assigned constant values.

The overall quality measure is defined as a weighted sum of the white space based and the connected component based measures.

3 Experimental Evaluation

In this section, we present preliminary experimental results demonstrating that our algorithms can do a reasonably good job at detecting tables in ASCII text and scanned image documents. The test database was composed of 20 ASCII documents (mostly e-mail messages) and 20 scanned journal pages. Each test sample was in single column format and contained one or more tables. None of the ASCII tables had ruling lines, while most scanned tables had some ruling lines. Any ruling lines detected by low level page segmentation were ignored in these experiments. The weights for combining the two table quality measures were set to 1.0 for the white space based measure and 3.0 for the connected component based one. The execution time for a typical page was around 2 seconds on an SGI O2 station (200 MHZ).

We report both recall (the percentage of true tables that were found) and precision (the percentage of tables found that were in fact true) in detecting tables. A threshold of 100 on the table scores was imposed for this experiment. In other words, a region is considered a table only if its corresponding score $tab[i, j]$ exceeds 100. A table can be considered to consist of two parts: table headings and the body of the table. Due to the simplicity of our current quality measures, table headings are not always detected as part of the table. For this experiment we considered a table to be correctly detected if at least the entire body of the table was correctly detected; *i.e.*, every line in the body of the table is identified as being part of the same table, and no neighboring non-table line is included. As shown in Table 1, the accuracies for both the image and the ASCII tables are reasonably high. One of the most common errors made by our algorithm was interpreting one table as multiple short tables.

Media	Recall	Precision
Image	82.67 %	72.09 %
ASCII	80.76 %	74.18 %

Table 1: Precision and recall for detecting ASCII and printed tables.

4 Conclusion

This paper describes a new approach for detecting tables across multiple media. The approach consists of a high-level framework and a set of table quality measures. The general framework is independent of the table quality measure or input format. The set of table quality measures presented can be applied to both ASCII and printed documents. The detection is entirely based on spatial layout information and does not rely on the presence of ruling lines. Preliminary experiments on ASCII and printed documents demonstrate the effectiveness of this approach. Future work includes exploring more sophisticated table quality measures, associating table heading with the body of the table, and expanding the algorithm to handle multi-column documents.

References

[Bai92] H. Baird. Anatomy of a versatile page reader. *Proceedings of the IEEE*, 80(7):1059–1065, 1992.

[Dav97] E. R. Davis. *Machine Vision, Theory Algorithms Practicalities.* Academic Press, 1997.

[Hir95] Y. Hirayama. A method for table structure analysis using DP matching. In *Proc. ICDAR'95*, pages 583–586, Montréal, Canada, August 1995.

[HSAF93] E. Horowitz, S. Sahni, and S. Anderson-Freed. *Fundamentals of Data Structures in C.* Computer Science Press, 1993.

[Kie98] T. G. Kieninger. Table structure recognition based on robust block segmentation. In *Proc. Document Recognition V, SPIE*, volume 3305, pages 22–32, San Jose, CA, January 1998.

[KW98] W. Kornfeld and J. Wattecamps. Automatically locating, extracting and analyzing tabular data. In *Proc. SIGIR*, pages 347–349, 1998.

[LV92] A. Laurentini and P. Viada. Indentifying and understanding tabular material in compound documents. In *Proc. 11th ICPR*, pages 405–409, 1992.