[19] G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. Information Processing and Management, Vol. 24, No. 5, pages 513-523, 1998.

[20] S. Sekine, J. Carroll, A. Ananiadou, and J. Tsujii. Automatic Learning for Semantic Collocation. Proceedings of the Third Conference on Applied Natural Language Processing, ACL. pages 104-110, 1992.

[21] Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. Journal of Information Retrieval, Vol. 1, No. 1/2, pages 67-68, 1999.

[22] T. Yavuz and A. Guvenir. Application of k-Nearest Neighbor on Feature Projections Classifier to Text Categorization. In Proceedings of the 13th International Symposium on Computer and Information Sciences - ISCIS'98, U. Gudukbay, T. Dayar, A. Gursoy, E. Gelenbe (eds.), Antalya, Turkey, pages 135-142, Oct. 26-28, 1998.

[23] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic Query Expansion Using SMART: TREC 3. The Third Text Retrieval Conference (TREC-3). National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, MD, 1995.

The TREATS Approach to Reuse of Tables in Plain Text Documents.

L.E.Hodge[1], N.J. Fiddian and W.A.Gray

Department of Computer Science,
Cardiff University,
Cardiff, UK.

{scmleh|njf|wag}@cs.cf.ac.uk

Abstract.

In this paper we present the table processing approach employed by the TREATS (Table Recognition, Extraction, Analysis and Tranformation System) software toolkit.

In order to support the large variety of table layouts that appear in plain text documents, our system aims to identify the layout of cells that exist within a table and to tailor processing accordingly. This results in more effective processing than preexisting approaches that apply one general technique to all types of table.

The classification process is the key to processing and exploits a cellular automaton (CA) based approach to the identification of cell structure within a table. The input to the CA is a simple representation of the content of the source table. This is evolved via the application of intelligent transformation rules, resulting in a representation of the cell structure that exists within the table. Based on the combination of cell types that appear in this representation, the layout of the table can be determined and appropriate processing can be performed. During this processing, the content of the source table is transformed into a relational form suitable for reuse in other applications.

Keywords: Information retrieval, resource discovery.

1. Introduction.

The TREATS (Table Recognition, Extraction, Analysis and Transfonnation System) software toolkit [1,2] offers support for reuse of tabular content from a wide variety of source documents[2]. Of the supported types, tables in plain text form are by far the most difficult to reuse. Whilst other document types such as HTML and Latex contain table definitions that enable their content to be extracted via a parsing approach, with plain text the only guide to the structure of the table (the key to correct reuse) is the layout of its content. Combined with the variety of table layouts that are possible, this makes tables in plain text documents difficult to effectively and conectly reuse.

Although methods for the processing of such tables exist, we feel that in general a specific approach can only support reuse of a limited subset of the possible table layouts. In this paper, we describe the approach used by the TREATS toolkit that provides more wide ranging support by identifying and classifying table layout such that processing can be tailored accordingly.

During this discussion, we concentrate on processing the content of a table i.e. everything below the level of the column labels[3]. Whilst these labels are important to the understanding of table content and will need to be extracted and in some cases transformed[4] to enable reuse, they do not offer any indication of the cell structure of the table.

The remainder of a table is concerned with the presentation of data and consists of two components, the stub and the body [3]. In general, the stub is the leftmost column of a table and may contain either a simple column of entries or in some cases where grouping exists, nested entries that indicate the hierarchy within the data. This hierarchy may also be indicated through the use of spanning cells and where this occurs, we extend the definition of the stub to cover all columns that contain spanning cells involved in defining the hierarchy. Spanning and nested cells are discussed in more detail in section 3.1. The body

[1] With support from EPSRC and BT (Case Studentship).
[2] Currently, plain text, HTML and Latex are supported.
[3] The area where labels are displayed is often known as the boxhead.
[4] Processing of label structures is addressed elsewhere [1].

Maker	Model	Esize
Ford	Fiesta	1.1
Ford	Escort	1.3
Vauxhall	Astra	1.2
Vauhall	Carlton	1.6

Type I

Maker	Model	Esize
Ford	Fiesta	1.1
		1.3
	Escort	1.3
		1.6
Vauxhall	Astra	1.2
		1.3
	Carlton	1.6
		2.0

Type II

	Price	Insurance Group
Ford		
Fiesta		
1.1	8000	38
1.3	8650	35
Escort		
1.3	9000	35
1.6	9750	32

Type III

Model	Axle Type
Fast Eddy Shooter	Fixed wooden axle
Henrys Viper	Ball bearing transaxle

Type IV

Figure 1: Illustration of Table Layouts.

of the table consists of all columns not involved in the stub.

2. Complexities of Processing Tables in Plain Text Documents.

When designing a table in plain text, an author has few constraints on how its content should be laid out, beyond accepted styles for table content [3,4]. The result of this is that a variety of table layouts are commonly encountered within plain text documents. Because of this flexibility, existing table processing approaches based on white streams [5,6] and bounding boxes [7.8] may only be effective for certain types of table layout. The reason for this is that these techniques employ a single overall approach that handles all tables in the same manner. Unfortunately, this is not always effective since tables with different types of layout may need to be handled in different ways to conectly extract their content.

Thus, we feel that in order to effectively process an arbitrary plain text table, tailored processing will be required. To achieve this, the layout of the table must be identified whereupon appropriate processing can be employed. To enable this type of approach, through a survey of a large range of tables from various sources[5] we have identified five classifications of table layout. These are based on the different types of layout that may appear within the stub and the body of a table:

- Type I - tables with simple cells that span only a single row and column.
- Type II - tables where stubs contain spanning cells.
- Type III - tables where stubs contain nested entries.
- Type IV – tables whose entries span multiple lines.
- Type V - hybrid layouts that combine elements of the other classifications but don't fit into any particular one.

Stub layout		Body Layout	Classification
Single line onirics		Single line entries	Type I
Spanning cell	with single line entries	Single line entries	Type Ha
	with multi-line entries	Single line entries	Type IIb
	with single-line entries	Multi-line entries	Type IIc
	with multi-line entries	Multi-line entries	Type IId
Nested cell	with single line entries	Single line entries	Type IIIa
	with multi-line entries	Single line entries	Type IIIb
	with single line entries	Multi-line entries	Type IIIc
	with multi-line entries	Multi-line entries	Type IIId
Single line entries		Multi-line entries	Type IVa
Multi-line entries		Multi-line entries	Type IVb
Multi-line entries		Single-line entries	Type IVc

Figure 2: Layout Classifications.

Examples of these layouts can be seen in Figure 1. Within each classification, a number of alternative layouts can result if entries that span more than one line appear in the stub, the body or both. The table in Figure 2 shows all possible layouts that are available within Type I-IV classifications.

[5] Numerous engineering and medical texts were consulted during the survey.

3. The TREATS Approach to Table Structure Processing.

The approach employed in the TREATS toolkit is to firstly classify the source table, then apply appropriate processing to enable the transformation of its content into a form suitable for reuse.

Tables are located within source documents using a method that exploits the patterns of whitespace and data types that appear within tables [1]. The pattern of whitespace between columns of entries is generally consistent for each line of the table body, as are data types for each entry within a column. By examining the patterns of whitespace and data types that appear for each line of the source document, it is possible to locate and extract areas of text that contain tables.

3.1 Classifying Table Layout.

In order to classify table layout, TREATS attempts to locate any of four basic 'table elements' within a source table. The combination of such elements within a column and the position of the column can be used to identify the layout of the table based on the classifications presented in Figure 2. Within a table, the following table elements may appear.

3.1.1 Simple (single-line) Cells.

The most basic type of cell, this consists of a single line of text and can appear anywhere in a table.

3.1.2 Multi-line Cells.

Multi-line cells contain entries that span over a number of lines (Figure 3). By their nature, multi-line cells will only contain string type entries. In order to locate multi-line cells our approach relies on the fact that either the start line or all continuation lines will be indented[6].

```
This is a Multi-line      This is not!
 Cell
```

Figure 3: A Multi-line Cell.

3.1.3 Spanning Cells.

Spanning cells appear where grouping relationships occur in a table, as in the first two columns in Figure 4. They are used to indicate groupings of related data, where the relationship is indicated by the number of rows over which the cell spans. Spanning cells can be nested to form hierarchies. In most cases, these appear on the left of a table and are considered to represent the stub, although in some hybrid layouts they may appear within the body of the table. They may contain entries that span one or more lines. As with multi-line cells, entries that span more than one line are generally indicated through the use of indents.

```
Animals   Cats    Persian
                  British Blue
          Dogs    Collie
                  Alsatian
```

Figure 4: Spanning Cells.

3.1.4 Nested Cells.

Nested cells contain entries that are nested to form a classification hierarchy as illustrated in Figure 5. These cells generally appear in the stub[7] and may contain single or multi-line entries. Nested cells are characterized by multiple levels of indenting, where the level of indent indicates the level at which an element belongs in the hierarchy. Notice that there is a relationship between nested cells and spanning cells and that they can be used interchangeably (e.g. Figure 5 is equivalent to Figure 4).

```
Animals
 Cats
  Persian
  British Blue
 Dogs
  Collie
  Alsatian
```

Figure 5: Nested Cells.

3.2 The Classification Process.

The table classification process involves the following steps:

i) Determine the alignment and data types of the entries in each column.
ii) Identify the table elements in each column - indicating the cell structure,
iii) Classify layout.

Before any classification processing can take place, the column structure of the table is deduced using an extension to a projection profile approach that exploits vertical streams of white space between columns [1].

[6] If they are not, it is impossible to determine the existence of multi-line entries automatically.

[7] In rare cases, where hybrid tables occur, they may appear elsewhere in the table body.

3.2.1 Determining Alignment and Data Types.

The alignment and data types of entries in columns directs processing in a number of ways. The alignment of entries within a column provides clues to logical structure embedded within the content and is an essential guide to the recognition of nested and multi-line entries within a table. As a number of alignments are possible it is essentia] to be able to uniquely identify alignment to ensure that table elements are correctly recognised. A number of different types of alignment for column entries can be identified:

- Left aligned - the most common type of alignment.
- Right aligned - often seen where integers are present within a table
- Centred.
- Nested.
- Multi-line – both indenting of continuation lines (the most common layout) and indenting of the first line of an entry (paragraph style) are supported[8] by TREATS.

Data types are exploited to overcome layout conflicts that may occur. For example, it is possible that a column of right aligned integers may have a layout that is identified as possibly being a column with multi-line entries. Since only string type entries can form multi-line entries, inconect processing of ambiguous layouts can be avoided by data type checking.

3.2.2 Identifying Table Elements.

Identifying table elements is achieved by examining the layout of content within columns and matching this to the layout that would be expected for these elements. In tables with a relatively simple structure, this can be done by examining the patterns of layout within a single column. In tables with more complex layouts however, due to ambiguity that may occur within a single column, the layout of entries in neighbouring columns must also be considered to ensure that elements are correctly identified. For this reason, TREATS employs a Cellular Automaton (CA) [9,10] based approach to the identification of table elements.

A Cellular Automaton is an array of identically programmed automata or 'cells' which interact with each other. At regular intervals, the content of these cells is evolved. Evolution is directed by a number of transformation rules, where the evolution of a cell is controlled by its state and those of its neighbouring cells.

To determine cell stiucture, a simple representation of the table is evolved via the application of intelligent transformation rules, resulting in a representation that clearly indicates the elements that exist within the table. This process has three steps:

i) Generate initial table representation.
ii) Evolve cell structure through repeated application of transformation rules.
iii) Extract table element/cell stiucture information from the evolved representation.

Generating the Initial Representation.

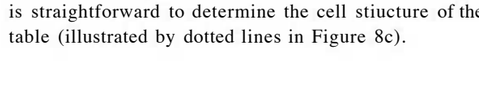
The first stage of the process is to generate a representation of the content of the table within the CA. This is modelled in a simple 2-D array which contains tokens that represent the layout of the entries within the table. Each line of text within each column is represented as its own cell. To illustrate this, consider Figure 8b which shows the initial representation for the table in Figure 8a. The choice of token is guided by the layout and content of a cell and the alignment information deduced in the previous phase. The tokens used in the representation are summarised in Figure 6.

Token	Description
s	A simple cell with no formatting[9]
e	An empty cell
i	A cell with an indented entry - indicating inclusion in a multi-line entry
n?	A cell involved in a nesting hierarchy. ? indicates the level of nesting, starting at 0

Figure 6: Tokens in Original CA Representation.

Evolution of Table Representation.

The initial representation is evolved by the repeated application of a number of transformation rules. These rules are applied to every cell in the CA until no further transformations occur (i.e. the CA becomes stable). An example of two rules used to evolve multi-line cells can be seen below:



The table in Figure 7 summarises the tokens that are used in the transformation rules and consequently appear in the evolved table representation.

Token	Description
es	Empty cells that are involved in a spanning cell
c	Continuation sections of multi-line entries
nc	Continuation sections of multi-line entries appearing within nested stubs
v	Void cells - empty cells that appear opposite classifications of a nested stub and are not involved in spanning cells

Figure 7: Tokens Involved in Transformation Rules and Resulting CA Representation.

Extracting Table Element/Cell Structure from the Evolved Representation.

Figure 8c shows the result of transforming the initial representation of the table in Figure 8b. From this, the cell structure is clearly defined by the tokens defined in Figures 6 and 7. Determining which table elements appear in each column is achieved using a pattern matching approach e.g. s followed by one or more c's indicates the existence of a multi-line cell. From the positions and sizes of these CA elements, it is straightforward to determine the cell stiucture of the table (illustrated by dotted lines in Figure 8c).

3.2.3 Classifying Layout.

From the information about column alignment and cell structure the layout of a table is classified through the application of niles based on the table characteristics presented in Figure 2. An example of one of these rules is:

```
If stub contains nesting and exhibits multi-
line entries
  if the body contains multi-line entries
   classification is Type IIId
  else
   classification is Type IIIb
```

3.3 Processing and Transformation of Table Content.

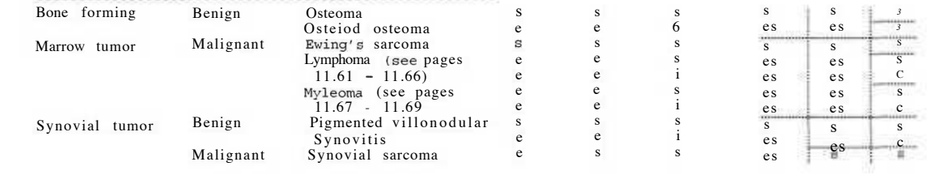
Once the cell structure of a table has been determined, the content of the source table is transformed into a relational fonn to enable its reuse. This processing requires two actions:

i) Multi-line entries are replaced by an equivalent single line entry.
ii) For Type II and III layouts, nested and spanning cells must be transformed into an appropriate form.

Multi-line entries must be processed first as they may appear within nested and spanning cells. Multi-line entries require only simple transformation involving the concatenation of the content of all line segments over which the cell concerned spans.

Since nested and spanning cells cannot be supported in a relational table, where such layout occurs content must be transformed. For nested entries, the nesting must first be expanded so that a column exists for each level of nesting (as with the equivalent content where spanning cells are used).

Type	Benign/ Malignant	Tumor
Bone forming	Benign	Osteoma
		Osteiod osteoma
Marrow tumor	Malignant	Ewing's sarcoma
		Lymphoma (see pages 11.61 - 11.66)
		Myleoma (see pages 11.67 - 11.69
Synovial tumor	Benign	Pigmented villonodular Synovitis
	Malignant	Synovial sarcoma



(a) The Source Table (b) Initial CA Representation (c)1Evolved CA

Figure 8: A Table and its Associated CA Representations.

[8] In some cases it is impossible to determine which layout exists. Where this is the case, the user is required to determine which type of layout is used.

[9] Note that non-left aligned entries are considered simple entries in this representation, as they are not involved in multi-line entries.

Finally, repeated entries are introduced. Only this final stage is required during the processing of spanning cells. This process is illustrated in Figure 9, where Figure 9a shows an example of a nested stub (see Figure 4 for the equivalent with spanning cells) and Figure 9b shows the equivalent relational fonn, with reintroduced repeated entries indicated by italics.

(a) Nested stub. (b) Transformed stub.

Figure 9: Transformation of a Nested Stub.

4. Conclusion.

By classifying table layout and tailoring table processing accordingly, the TREATS approach to processing plain text tables can effectively support a large variety of table layouts. It has been tested on many of the examples used in the aforementioned layout classification survey and has proved to be highly effective.

We have identified a small number of limitations of our approach, where processing is affected by ambiguous layouts due to empty cells resulting from missing entries and lack of indentation to indicate where multi-line entries appear. Missing entries result in the incorrect formation of spanning cells and lack of indentation means that multi-line entries cannot always be formed. Unfortunately, there is no way to overcome these limitations automatically, due to conflicts with other types of layout. These problems can only be overcome through user interaction, but are exceptional.

Finally, whilst we have not considered hybrid tables in this paper, support for the most commonly appearing forms is available in the TREATS toolkit through suitable combination of the techniques we have described.

5. References.

[1] L. E. Hodge. W. A. Gray and N. J. Fiddian. Effective Reuse of Textual Documents Containing Tabular Information. *Proceedings of the 4th Australasian Document Computing Symposium (ADCS 99)*, Coffs Harbour, NSW. Australia, December 1999, pp 47 - 53.

[2] L. E. Hodge, W. A. Gray and N. J. Fiddian. A Toolkit to Facilitate the Integration of Tabular Information in Textual Documents with Database Applications. *Proceedings of the 4th Multiconference on Systemics, Cybernetics and Informatics (SCI 2000)*, Orlando. Florida, USA, **July** 2000.

[3] *The Chicago Manual of Style*, Thirteenth Edition, The University of Chicago Press, 1982.

[4] R. J. Beach. Tabular Typography. *Proceedings of the International Conference on Text Processing and Document Manipulation*, University of Nottingham. April 1986. pp 18- 33.

[5] T. Pavalidis and J. Zhou. Page Segmentation by White Streams. *Proceedings of the International Conference on Document Processing (ICDAR '91)*. Saint Malo, Fiance. 1991. pp 945-953.

[6] S. **Chandran** and R. Kasturi. Structural Recognition of Tabulated Data. In *Proceedings of the International Conference on Document Processing (ICDAR 93)*. 1993.

[7] K. Itonori. Table Structure Recognition Based on Textblock Arrangement and Ruled Line Position. *Proceedings of the International Conference on Document Processing (ICDAR 93)*, 1993.

[8] T. G. Kieninger. Table Structure Recognition Based on Robust Block Segmentation. *Proceedings of Electronic Imaging 98 (SPIE), Document Recognition*, 1998.

[9] J. D. Farmer. T. Toffoli and S. Wolfram, editors. *Cellular Automata: Proceedings of an Interdisciplinary Workshop*, Los Alamos, New Mexico, March 7-11, 1983.

[10] Stephan Wolfram. *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley, 1994.

Recovering Structure from Unstructured Web-accessible Classified Advertisements

Richard Cole, Peter Eklund and Age Strand
School of Information Technology, Griffith University
PMB 50 Gold Coast MC
QLD 9726. Australia
{r.cole,p.eklund}@gu.edu.au, mstrand@hotmail.com

Abstract

This paper describes a research prototype system called RFCA for structuring Web-accessible rental classified advertisements based on semantic content. A hand crafted parser is used to extract various facets of the rental property being advertised including amongst others; member of room, type of garage, dwelling type (unit, house, or high rise apartment), price and contact details. The performance of the parser is measured in terms precision and recall by comparing its output to that of human expert.

The structured information once extracted is stored in a relational database and users searching for rental properties are presented with a graphical organisation of rental properties according to predefined themes. The overall result is a suite of tools for extracting, cleaning, structuring, and visually querying/browsing collection of web-accessible rental advertisements.

The mathematical and, methodological foundation for the graphical organisation of the structured information is provided by formal concept analysis. Using formal concept analysis each property is understood to be, an object possessing attributes with attribute values. The data is then conceptually organised via concept lattices dynamically according to a pre-definedconceptualscales. The, concept lattice organises rental properties into conceptual groupings. The, user then has the opportunity to view the attributes of all properties in a grouping as well as navigate back to the source advertisements.

The, interface is delivered over the web using a CGI interface and dynamic creation of image and image maps. The, ideas presented are general enough to be relevant to other web-accessible unstructured text sources.

1 Rental Formal Concept Analysis

Many newspapers hold large keyword indexed free-text collections of classified advertising on the Web and these can be searched on-line. The intention with this work is to demonstrate how such data can be value-added by extraction, cleaning and structuring. A structured database derived from free-text classifieds can then be browsed effectively. We argue that a browsing interface that structures the presentation of classifieds, related to a particular purpose (such as rental classifieds), can facilitate retrieval. More specifically, we maintain that a browsing interface using formal concept analysis gives a sense of the way that attributes within the free-text sources are distributed across the text collection, something that a keyword-based search interface cannot do.

Formal Concept Analysis (FCA) [13] has been developed during the last twenty years and successfully applied to data analysis and knowledge processing [15]. The Mathematics of FCA has been carefully described in Ganter and Wille [9] and the basic details of the theory are omitted here for brevity. There have been a number of examples of FCA applied to information retrieval and filtering [10, 1] including our own work [3. 4]. In these systems the main difficulty is attribute identification front texts. In the medical and email domains in which have worked [8. 2. 5], objects are easily identified since they correspond to documents: typical stored as a single file. In the case of Web-accessible rental classifieds the extraction task is complicated by object recognition: several rental classifieds often appear in the same advertisement and are grouped by location. (trice or the number of bedrooms. An example of such a problematic advertisement is illustrated in Fig. 1.

2 Object and Attribute Identification

For these reasons one of the first tasks of a unstructured text parser for RFCA is object recognition. disambiguating a single rental property from an advert that may list several or many properties for rent. For example, in Fig. 1, lines 3, 4, 5, 7 and 8 of the advert are individual properties which require representation as objects. In addition, there are cases where attribute recognition can also be com-

Proceedings of the **Fifth Australasian** Document Computing Symposium, Sunshine Coast, Australia, December 1, 2000.