Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text

Shona Douglas, Matthew Hurst, David Quinn

Construction Industry Specification Analysis and Understanding System (CISAU)

Project No: IED*4/1/5818*

9 December 1994

Preface

This document records as a *CISAU* project discussion document the paper presented as

> Douglas, S., Hurst, M., & Quinn, D. (1995). Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, (pp.535-546)., Desert Inn Hotel, Las Vegas, Nevada. University of Nevada.

This document is publically available as

> `ftp://ftp.cogsci.ed.ac.uk/pub/shona/Tables.ps.gz`

1 Introduction

Documents in many technical domains express much of their content in tabular form. Natural language processing applications in such domains need table interpretation functions to make this information available. Our work on information extraction in plain text 'legacy documents' in the domain of construction industry specifications shows the usefulness of NLP techniques in the adequate interpretation of tables into logical structures.[1]

In this paper, we present an analysis of table layout and linguistic characteristics, based on the tables encountered in the *CISAU* project (Construction Industry Specification and Understanding), and show how such an analysis supports a two-phase processing regime for the interpretation of plain text tables (cf. [WW93]).

2 Abstract Characteristics of Tables

2.1 Table Denotations

A simple table in some abstract sense denotes a relation among sets of values.[2] (We can think of this as corresponding to relations or views in relational database terms. See, for example, [Ull88]) In these terms, the underlying representation of a table is a set of n-tuples, where n is the number of domains or value sets in the table. The n-tuples of the table is that subset of the cross-product $D_1 \times D_2 \times \ldots \times D_n$ for which the relation holds. This is the **canonical form** of the table.

For our NLP application, we need a relatively detailed ontology or world model of the objects of the sublanguage of construction. The classes of this world model we call **world domains**; they include

[1]We thank Stephen McCarron of BICC. We acknowledge the support of the Department of Trade and Industry, the Engineering and Physical Sciences Research Council, and the BICC Group on the CISAU project (IED4/1/5818), and the Human Communication Research Centre at the University of Edinburgh.

[2]All the tables we deal with in this paper are simple tables in this sense; we briefly discuss our definition of complex tables, and their treatment, in Section 6.

things like the types of object or entity, types of entity attributes and so on they may have, units in which attributes may be expressed, and values units can take. The set of values occurring in a **table domain** is in general a subset of those in one of these world domains. In interpreting the table, we can say that the relation it embodies defines a relation in the world model.

The ontological model of the domain is a **hierarchy** of concepts which we can use to distinguish between domain *labels* and domain *values*. A domain consists of a set of values with a common ancestor or supertype, to which all the values bear the same relation in the hierarchy. Such a supertype is the interpretation of the label of a domain. If the node `MATERIAL` in the world model has below it in the hierarchy `WOOD`, `CONCRETE`, etc., then `MATERIAL` or its cognates will be likely to be a label for the subordinate classes.

2.2 A Functional View of Tables

From a functional perspective, a table not only embodies information about a relation (what we have called the table's denotation) but **organises** that information according to the purpose the table is intended to fulfill in its text. The domains that compose a table can be arranged in **groups** and **orders**, corresponding to a **decision structure** for the table ([Wri82]). This decision structure reflects the groups and orders in which domains are to be used as **keys** in choosing and reading (constructing) a tuple that contains the information required. While in relational database terms a key domain is one whose values uniquely identify tuples, in our functional terms we say that a table has as part of its functional aspect a **key precedence list** of domains, reflecting the decision structure the table supports.

We can represent the grouping and ordering of domains and their values in relations of type $D_1 \times D_2 \times \ldots \times D_n$ by viewing one or more of the domains of the relation as the domain of a function, and one or more of the domains of the relation as the range of the function. For example, the relations $(D_1 \times D_2) \rightarrow D_3$ and $D_1 \rightarrow (D_2 \times D_3)$ are the same in terms of denotation, but in functional terms we say that in the first expression the pair of domains D_1 and D_2 are used as key domains to find or determine values in the range domain D_3, whereas in the second, values in D_1 are used to pick out pairs of values taken from D_2 and D_3.

2.3 Representing the Abstract

We have distinguished two types of abstract property of tables, denotational and functional. We now go on to investigate how these properties are embodied in

- the layout of domain values and labels as cells in a 2-D table, and
- the linguistic properties of text fragments that form the cell contents.

3 Layout Characteristics of Tables

Our analysis of table layout characteristics relates the 2-D arrangement of tables to the underlying relation, and shows how pragmatic and functional considerations affect the way such a relation may be embodied in a table. We introduce the repertoire of layout transformations that form the basis for the processing described in Section 5.

3.1 Canonical Layout of Tables

A relation, expressed as a table in two dimensions, has a unique **canonical layout**:

- An ordering is defined on the domains.
- For each domain an ordering is defined on its values.
- Each domain in the relation corresponds to a single column in the table, ordered by the domain order from left to right.
- Each column is headed by a label for the domain.
- Tuples are composed in domain order, and sorted by value order, with left-most domains being most significant.

Figure 1 provides an example of a 3-domain relation in the canonical layout.

Cement	Standard Mixes	BS
Ordinary	ST1	BS 12
Ordinary	ST2	BS 12
Ordinary	ST3	BS 12
Ordinary	ST4	BS 146
Ordinary	ST5	BS 146
Blastfurnace	ST1	BS 6588
Blastfurnace	ST2	BS 6588
Blastfurnace	ST3	BS 6588
Blastfurnace	ST4	BS 146
Blastfurnace	ST5	BS 146

Figure 1: A 3-domain relation in canonical layout

Where a value in a given domain occurs in more than one tuple, the sorting defined for the canonical representation will result, as in the table in Figure 1, in the patterned repetition of the values of a domain, with a periodicity depending on the domain's position in the relation. We call these **recapitulation** patterns.

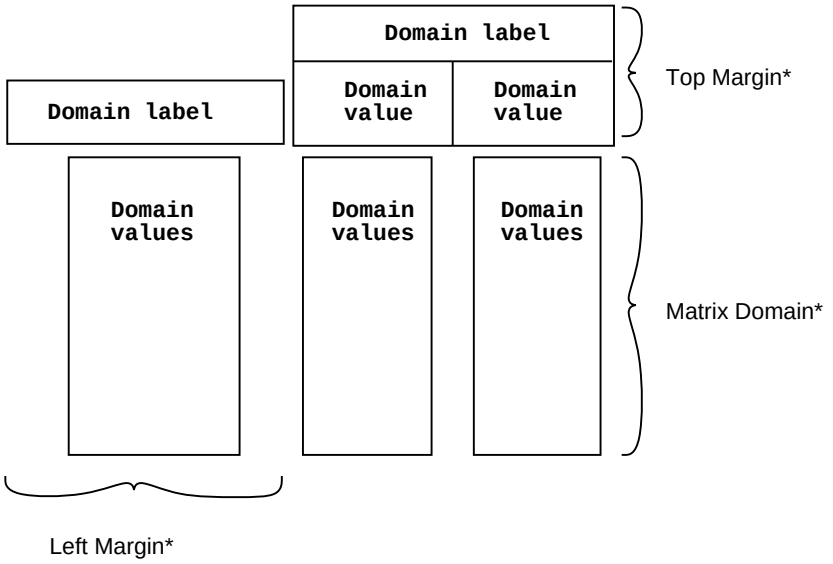


Figure 2: A generic table with terminology ('*' represents optionality)

3.2 Variations on Table Layout

Different layout arrangements can be thought of as expressing functional aspects of the relation, using some simple heuristics about the way grouping and ordering may be expressed in two dimensions. Because we conventionally read tables from the left and from the top, we distinguish the **left margin** and **top margin** of tables as areas of the table in which high-precedence domains are placed (see Figure 2). A given layout supports a certain **reading order**. This reading order reflects the way in which domains are organised and specifically the **groups** and **orders** in which domains can be easily used as **keys** in choosing and reading/constructing a tuple from the table; the reading order is thus the embodiment of the decision structure we identified as the functional part of the table.

Thus, while a single canonical form may have many layouts, a given canonical form plus functional information will have a much reduced range of felicitous layouts. These typical constraints on layout will be used later in our processing heuristics. First, we present a repertoire of transformations of simple tables in terms of which we can analyse the variations in layout that occur.

Rotation This transformation is best described by example:

Standard	slump	Value
ST1	75mm	v1
	125mm	v2
ST2	75mm	v3
	125mm	v4

→

Standard	slump	
	75mm	125mm
	Value	Value
ST1	v1	v2
ST2	v3	v4

Note how this transformation alters the layout of other domains (e.g., the repetition of the `Value` label, altering the recapitulation patterns) and often results in a **matrix domain** (see Figure 2). Such a pattern results in the rotated domain being promoted to a key domain, by the use of the second dimension; we then have a double key, and the domain 'enfolded' to be at the end of the reading paths through key domains in both dimensions is therefore the result or range domain.

Value Groupings Domain values are grouped where they are invariant for corresponding values in other domains. Where that domain is a key domain, the grouping provides a simpler decision structure by reducing the number of cell choices in that domain.

Domain Label Omission Self-explanatory. It is often associated with a domain having been placed **in focus** by its label being mentioned in the title. Once in focus, such information is used by default to complete underspecified elements such as this missing label in the table.

Domain Reordering Adjacent domains are swapped, resulting in an increase in the number of recapitulations for the original right-most domain and a decrease in the number of recapitulations in the original left most domain. This corresponds functionally to a promotion in key precedence for the new left-most domain; recapitulation is most common in secondary keys.

Value Distribution Values are distributed by repetition, producing a one to one alignment between the cells expressing the value in the distributed domain and the corresponding cells in the domain distributed over.

Value Factoring Repetitions of a value in a domain are collapsed into a single cell (the opposite of **distribution**), providing a simpler decision structure in that domain.

Similar Value Listing Values whose corresponding entries in another domain are equal are grouped to form a list in a single cell.

To demonstrate the use of these transformations, we show how a table from our domain can be derived by the application of successive transformations to the canonical layout. First, the canonical layout:

`Cement`	`Standard Mixes`	`BS`
Ordinary	ST1	BS 12
Ordinary	ST2	BS 12
Ordinary	ST3	BS 12
Ordinary	ST4	BS 146
Ordinary	ST5	BS 146
Blastfurnace	ST1	BS 6588
Blastfurnace	ST2	BS 6588
Blastfurnace	ST3	BS 6588
Blastfurnace	ST4	BS 146
Blastfurnace	ST5	BS 146

Omitting the domain label `BS`, *grouping* the `Cement` domain values and *listing* the `Standard Mixes` gives

Cement	Standard Mixes	
Ordinary	ST1, ST2, ST3	BS 12
	ST4, ST5	BS 146
Blastfurnace	ST1, ST2, ST3	BS 6588
	ST4, ST5	BS 146

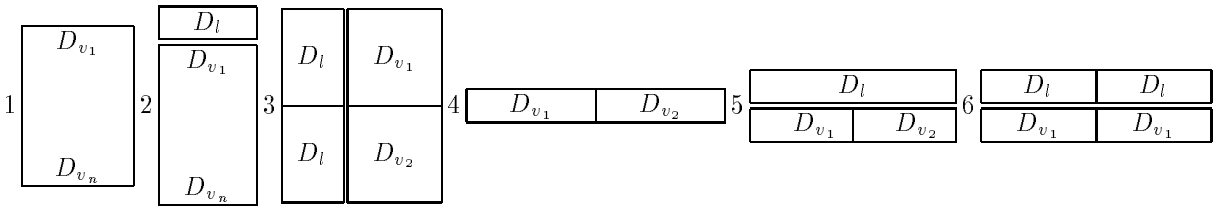
Finally, *rotating* results in the final table:

Cement	Standard Mixes	
	ST1, ST2, ST3	ST4, ST5
Ordinary	BS 12	BS 146
Blastfurnace	BS 6588	BS 146

Transformation operations are considered to operate over table domains. Consequently, the operations described above must respect the consistency of the table as a representation of the underlying relation. This requirement results in the repetition of operations for recapitulation.

3.3 Domain Syntax

A domain will have certain syntactic characteristics when rendered as a component in a table. These patterns are results of the transformations described in Section 3.2.



1. Values in a column without a label.
2. Values in a column headed by the label.
3. Values to the right of the repeated domain label. This case is entered for completeness, though hasn't been observed.
4. Values appear horizontally.
5. Values horizontal below the domain label.
6. Values horizontal below the repeated domain label.

These simple cases can be recapitulated (as described in Sections 2.1 and 3.2). In broad terms, this recapitulation results in syntactic repetition either horizontally (4, 5, 6) or vertically (1, 2, 3). Labels may be omitted in some cases (2, 5). Note that in the case of 5, the original label may be extended across the values, or repeated. The later case seems to occur most frequently.

Recapitulation can be thought of as an iterative feature of the syntax for the domain in consideration. The effect of this recapitulation on the syntax of the table as a whole is more generally thought of as a recursive feature of the syntax of the table, the recapitulation resulting in further nested syntactic repetitions.

4 Linguistic Characteristics of Tables

Tables exhibit a number of characteristic natural language properties, which reflect their denotational and functional properties and hence provide clues for their interpretation. This section introduces phenomena found in the *CISAU* domain.

We group our observations under the following headings:

- focus;
- ellipsis and movement;
- lexical and morphosyntactic cohesion.

4.1 Focus

In natural language, the phenomenon of **focus** is used to allow the use of underspecified forms in referring to discourse elements in focus. Interpreting **underspecified** elements, such as pronouns, thus requires knowledge of what is in focus (see, for example, [GS86]). A similar phenomenon occurs in our tables, in the relationship between mention of domain labels in the title and their omission in the actual table. This is particularly common for matrix domains, of course, where it is difficult to find a place to put a label domain. The correspondence between matrix domains being the range or result domain, and the statement of the intended result domain in the title, is thus established only by the recognition of a focus-based interpretation of the matrix domain.

4.2 Ellipsis and Movement

In expressing functional properties such as the contrast between given and new information in a sentence, natural language uses a number of tricks of which movement in the form of topicalisation is one: the new or emphasised information is **moved** out of its normal syntactic position, and typically **fronted**, possibly requiring syntactic rearrangements like passivisation or clefting (see, e.g., [HH76]) or relying on intonation contours. The movement may leave a gap, or ellipsis. Ellipsis often occurs in parallel constructions, where a process of **abstracting** occurs, with a common element taken out of subsequent parallel constructions, to be interpreted as redistributed from the first occurrence, e.g.,

Pat took the plants and Bill the books, where the elliptical expression *Bill the books* is to be interpreted as if the verb *took*, abstracted out, is redistributed to give *Pat took the plants and Bill took the books*.

In our table domain, the set of tuples that make up a relation is a large parallel construction, and abstraction and ellipsis of cell contents are often used to render tables more concise. In particular, it is common for syntactically and semantically **unsaturated** elements to occur in domain labels, linguistic expressions that are missing some information necessary for their interpretation, information which is found in the value cells for the domain. For example, we may see a label heading like `Slump mm` over a set of cells containing integer values.

The combinatorial properties of these linguistic elements tell us that `Slump mm` is unsaturated or incomplete, that the type of element that would complete the expression is a number of some kind (from the `mm` part), and probably tell us something about the range of integer values plausible for `Slump` expressed in `mm`. These constraints help us to identify links between label cells and the value cells of the same domain, in the face of the rearrangements caused by layout transformations. Alternatively, the ellipsis may occur in the value cell, where for example a domain label which is a noun like `cement` may dominate a set of cells with names like `Ordinary Portland`, `Portland Blastfurnace`, and so on, which are adjectival phrases requiring a noun of appropriate type, like `concrete`, to saturate them. The parallel elliptical forms, each displaying the same kind of abstraction and ellipsis, are powerful indicators of domain structure.

4.3 Lexical and Morposyntactic Cohesion

Cohesion ([HH76]) is a semantic property of related elements in a text, which may be expressed in certain kinds of systematic similarity in their linguistic realisation. Many of the underlying ontological properties of tables, domain labels and domain values find expression in distinguishable forms of lexical cohesion (at the level of word-forms) or morphosyntactic cohesion (at the level of more structured analyses of text).

At its simplest, lexical cohesion is instanced by **repetition**, which certainly confirms that the cells in question may be in the same domain, particularly if the repeated elements are laid out in a plausible recapitulation pattern for the domain's apparent precedence. Terms appearing in the title that are repeated in the table are often domain labels.

Superclass/subclass relations, such as hold between domain labels and their values, have characteristic lexical counterparts. There are also cases where there are structural similarities between label and value sets, and among the value cells, that help to identify their place in terms of domain and relation. For example, noun phrases in the label may generalise (semantically) over those found as the values, while the syntactic structure of the label and value is the same. Thus, a domain labeled `Vertical formwork to columns, walls and beams` is populated has value cells `Soffit forms to slabs`, `Props to slabs` ....

5 Processing Tables

The aim of this work is to process a given table into its canonical layout, and hence, the relation it represents. As has been shown above, the production of a table can be thought of as a series of transformations on the canonical form. We propose a table interpretation method which uses a combination of iterative processing and heuristic matching to obtain the primary information (i.e., the location and type of domain labels, domain values and the composite domains which they represent). Once this information is recovered, reconstituting the canonical layout, and hence the relation, is possible. The method employs the techniques and observations discussed in this paper:

1. Table segmentation.
2. Semantic analysis with respect to the sublanguage ontology.
3. Syntactic analysis of the table layout.

5.1 Abstraction and Interpretation Considerations

So far, we have presented a rationalisation of the table with a particular goal in mind: automated information extraction. This approach results in a view of the table as a *functional* realisation of a *relational* structure (see Sections 2.2 and 2). Other levels of abstraction exist, from those concerned with low level formatting (see, for example, [McG91]) to more process independent, mathematical descriptions (see [WW93]).

We claim that both of these view points are in some way inadequate, both have a certain information poverty which prevents their use in an information extraction process. The low level representation contains no information concerning the true relationship between cells other than their physical layout, the high level representation offers no account of incomplete, or assumed (i.e., world domain) information. In other words, the content of a cell is not some passive component in providing a mapping between table layout and some logical form, but is, in our view, key to establishing such a mapping.

As we suggest in Section 6, a more appropriate abstraction is some declarative formalism, complete with operators which allow, through domain knowledge, the correct manipulation of tables generatively as well as the reverse functionality interpretively. Models like that proposed in [WW93] ignore such considerations by allowing the abstraction to accommodate the result of transformations with a functional purpose (e.g., the omission of labels), while providing no hint as to how to achieve the abstraction from real tables, and hence losing information in their representation of tables.

5.2 Table Recognition and Cell Identification

This section outlines methods for extracting and utilising text layout for recognition of tables, and identification of the rows, columns, and cells of each table.

First, we need to process the whole text, to distinguish non-table text such as Figure 3(a), from table text, such as Figure 3(b).

```
133    CONCRETE AT ATRIUH COLUHNS (CASING TO STEELWORK)
-      Designed mix to BS 5328, Grade C30
-      Cement(s): as BS 5328
-      Aggregate(s):
       Coarse:  as BS 882
       Fine:    as BS 882
-      Nominal maximum size of aggregate: 10 mm
-      Hinimum cement content: 275 kg/cu  m
-      Haximum free-water/cement ratio:  0.65
-      Haximum cement content; 325 kg/cu  m
-      Admixture:  Not permitted unless approved in writing by CA
-      Rate of sampling for compressive strength testing:  one sample per 10
       cu m but not less than one for each day of use.
```

(a)

```
-      Standard Hixes and related strengths.

       Standard          Characteristic compressive strength at
       Hix               28 days assumed for structural design
                                  n/mm2 (= Hpa)

       ST1                                 7.5
       ST2                                10.0
       ST3                                15.0
       ST4                                20.0
       ST5                                25.0
```

(b)

Figure 3: (a) Non-table text (b) Table text

We proceed as follows:

Segments of Contiguous Spaces We record the positions of all the **segments** of contiguous spaces in the whole document. Each of these segments, except the first in each new line, immediately follows a sequence of continuous printable characters (a token). We also record whether or not each of these character sequences contains at least one alphanumeric character. This information is used later to help identify non content-bearing sequences (no alphanumeric characters) that function purely as layout markers.

Areas and Plumb Lines Next we group pairs of adjacent lines of text into an area, and pairs of adjacent areas into a larger area. An **intersection** algorithm produces a new list of segments for the area, containing all the spaces common to both the original lists. Each segment produced for an area represents the maximal width solid block of spaces that can be traced through all the lines of the area, without any intervening printable characters. This may be thought of as dropping a weight vertically from the top to the bottom of an area, hence the name **plumb lines**.

First Division into Areas Next the document is divided into areas of text separated by one or more empty lines — a first attempt at a division of the text into sections. For each area, the intersection algorithm is used to find the list of segments for the area as a whole.

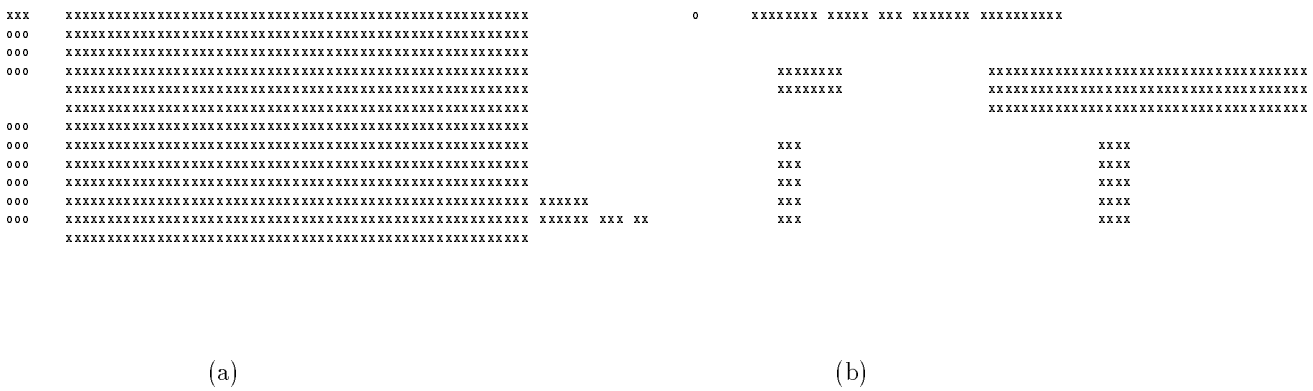
Once we have a first division into areas, we go on to identify columns within those areas:

Columns A **column** algorithm converts a list of segments of contiguous spaces into a list of columns of characters, whose width is determined by the segments. While the segments represent the negative image of the lack of text, the columns are their inverse — the positive image of the text layout.

Weightings A **weighting** heuristic provides an indication of how typical a column appears to be of a column in a table.

A typical table column is rather narrow, and usually has a high density. A table column typically is demarcated by adjacent segments that are not very narrow. We encode these properties as a heuristic.

Figure 4 (a) shows the column layout for the area of non-table text in Figure 3 (a). In layout figures '`x`'s and '`o`'s are used to represent columns. 'x's represent a column line containing at least one alphanumeric character, while 'o's represent a column line with no such character. The width of each column is determined by the adjacent segments of spaces — each non-empty line in the column is represented as being of maximal width.



(a) (b)

Figure 4: (a) Layout figure for text in Figure 3; (b) Layout figure for table in Figure 3

Figures 3 (a) and 4 (a) show an area of normal text that is recognisable at this stage as not being a table even though, having five columns, it superficially resembles one.

Figure 4 (b) shows the column layout for the table in Figure 3.

We can now proceed with turning columns into consolidated tables, and cells:

Similarity of Areas A **correspondence** heuristic indicates the degree of similarity between two lists of segments from two areas. The intersection algorithm is used to produce the list of segments common to both areas. If two areas are similar, then most or all segments in each area should have corresponding segments in the intersection (indicating that there is a corresponding segment in the other area), and they should be of approximately the same size (indicating a significant overlap with the corresponding segment in the other area).

Merging Areas Areas that have not already been rejected include headings, other short sections of about 3 lines or less, assorted page header and footer lines, and, naturally, parts of tables. Groups of adjacent areas occur between the gaps formed by rejected areas. Application of the correspondence heuristic to successive pairs of adjacent areas results in the merging of areas with similar columns into larger areas. The weighting heuristic is then applied to the columns of these new areas, some of which will be accepted as parts of tables.

Subdividing Areas An area accepted as part of a table, on the basis of its division into columns, may contain empty lines which subdivide it horizontally into rows. In this case the table cells

are already determined, by the intersection of the columns and rows. In the absence of such an explicit indication, the area must be subdivided into its component rows by the grouping of lines.

Now we have identified the cells in the table, we can go on to apply table layout heuristics and NLP techniques to interpret the cells as a relation.

5.3 Cell Analysis

The cells are analysed one by one to provide a number of informative tags:

1. The cell's unique coordinates.
2. The semantic type of the cell with respect to the sublanguage domain model. E.g., `Portland Blastfurnace` would be assigned the type `_CEMENT_INSTANCE`.
3. It relationship with the title. This is a boolean indicator of whether or no some reference can be computed between the term in the cell (e.g., `Use`) and a term in the title (e.g., `general purposes`).

In parallel to this process, indexes are built up: the `X` index and the `Y` index, keyed by tuples representing a variety of the linguistic properties discussed, together with either a `X` or `Y` coordinate, e.g., `(SEMANTIC_TYPE, X-VALUE)` and `(SEMANTIC_TYPE, Y-VALUE)`. It is these indexes that form the basis of the heuristic approach. Below, we use the example of the above index form, however, any of the linguistic properties discussed may be used.

5.4 Heuristic Analysis

This final phase uses the pragmatic knowledge of the syntactic layout and the semantic clustering (embodied in the indexes) to identify domains in the table. We use the index system to filter candidate domains by semantic relatedness (i.e., by identifying either semantic equality or a hierarchical relationship) and characteristic layout patterns:

- Semantic consistency is observed through inspection of a particular `X` index for a semantic key. This represents a possible column of domain values.
- This `X` value is used to index on other `X` indices to find a possible domain label.
- A semantic type superior to the possible domain value type is found.
- Reference to the title supports the hypothesis that this is a domain label.

This process is a heuristic encoding of both the syntactic and semantic knowledge about the layout of tables we have discussed previously (see Section 3.3, layout 2).

6 Conclusions and Further Work

In this paper we have presented a number of issues related to developing a computational model of tables making use of linguistic and layout cues:

A canonical form and layout was given based on the **relation** as found in DataBase literature.

Functional aspect of the layout.

A set of transformations for altering the structure of the tables presentation and content.

Linguistic requirements for the analysis of cell contents.

A simple characteristic syntax for the layout of tables as presented.

Algorithms for recognizing and segmenting tables as raw text based on the notion of **plumb lines**.

A heuristic approach to analysis based on the NLP components.

Combined, these demonstrate the need for a broad selection of NLP functionality for a system which intends to carry out any sort of table analysis for information extraction.

We see further work in this area continuing in a number of directions.

- The paper has restricted discussion to the class of **simple tables**. **Complex tables** are formed out of combinations of tables with multiple purposes, and with embellished domain labelling methods. Dealing with these will require a more complex model of table syntax as well as extensions to the classification tags for cells.
- The analyses presented suggests an underlying declarative treatment of table generation and analysis. Rationalising the methods presented in this manner would have many advantages not least of which is a more general model which would be able to support a greater range of processes (e.g., generation).
- The processes described for table analysis rely strongly on fixed **Knowledge Representation** and NLP components. It is clear that a practical system will have to cope in situations where these components are incomplete with respect to the text. How the approach suggested will generalize to a more robust system requires investigation.
- The complexity of table generation, i.e., the number of table layouts possible given a relation, discourse purpose etc., is high, however; a rigorous analysis is required to ascertain how this affects out approach.
- Little on the relationship between **table domains** presented in the table was discussed. This factor is important to the suggested declarative approach, and relates to the knowledge representation and **world domains**.

References

[GS86] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[HH76] M. A. K. Halliday and Ruqaiya Hasan. *Cohesion in English*. English Language Series. Longman, 1976.

[McG91] Robert W. McGaffey. Sgml versus/and tex. In Barbara Beeton, editor, *TUG*BOAT, volume 12, pages 406–408, December 1991.

[Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

[Wri82] Patricia Wright. A user-oriented approach to the design of tables and flowcharts. In David H. Jonassen, editor, *The Technology of Text*, pages 317–340. Educational Technology Publications, 1982.

[WW93] Xinxin Wang and Derick Wood. An abstract model for tables. University of Waterloo Department of Computer Science Research Paper, 1993.