Finding Critical Cells in Web Tables with SRL: Trying to Uncover the Devil's Tease

Nicola Di Mauro and Floriana Esposito and Stefano Ferilli
Dipartimento di Informatica, LACAM laboratory
University of Bari
Bari, Italy
Email: firstname.lastname@uniba.it

***Abstract*—Tables are extremely important components of documents, because they bear very informative content in a compact and structured way. Being able to understand a table's internal organization would allow to extract and reuse the data they contain. This can be reduced to recognizing critical cells only. Since purely algorithmic approaches are unable to deal with the many different table layouts designed to represent particular kinds of information and/or particular perspectives on them, Machine Learning may represent an effective solution. On one hand, the spatial organization of tables puts a strong emphasis on the relationships among cells; on the other, the extreme variability in style, size, and aims of tables requires flexible approaches. This paper proposes the exploitation of a Statistical Relational Learning approach, that is able to model the complex spatial relationships involved in a table structure, by mixing the power of a relational representation formalism with the flexibility of a statistical learning tool. Experiments on a real-world dataset are reported both for single cell classification and for overall table structure recognition, whose results prove the validity of the proposed approach.**

I. Introduction

The huge amount of electronic documents available nowadays poses serious problems for the efficient and effective exploitation of the information content they bear. In some cases, this situation may tamper the very role of documents, which is supporting the users in their activities by satisfying their information needs. In fact, full management of the documents and of their content goes beyond human capabilities. Hence, the need for automatic techniques that can suitably index, understand and relate different documents in a collection. In turn, understanding documents require to go beyond 'surface cues' that can be determined syntactically, up to the semantic level underlying the appearance.

This objective has been so far mainly aimed at based on the textual content of documents, as the most close component to the semantic level. Indeed, although words suffer from linguistic tricks such as synonymy and polysemy, they are human artifacts purposely developed to explicitly indicate concepts. Hence, the focus of the Information Retrieval (IR) and Information Extraction branches of Computer Science has typically concerned natural language, often limited to the purely lexical aspects of the text. However, there are other components of documents that may be very important towards its full understanding. For instance, pictures and graphics are often exploited to immediately hit perception and communicate contents that many words could not satisfactorily express.

Another extremely important component of documents are tables. Authors use tables to compactly represent many important data in a small space, to attract the reader's attention, or for information comparison [1]. Thus, the availability of automatic components that can identify tables in documents, and that are able to understand the table structure, would be a precious support to extract the knowledge they contain, represent it formally (e.g., using a relational Database) and make it available to people and/or other software (e.g., using semantic technologies that are being developed nowadays).

This paper focuses on the process of understanding table structures by identifying their key cells, and proposes the use of Statistical Relational Learning [2] (SRL) techniques to do this automatically. In fact, spacial (relational) information is fundamental in tables, but the large variety of possible table structures also requires the support of sufficiently flexible (statistical) approaches.

Our approach was embedded in the DOMINUS framework [3], that brings to cooperation a set of intelligent techniques aimed at covering the whole set of steps going from the submission of a document in digital format, through its processing to extract the relevant knowledge it contains, up to its delivery to interested users. Specifically, the layout analysis engine of DOMINUS can extract tabular items, that are provided to a SRL system in order learn and later recognize their structure. This allows to extract the table data, and use them for better indexing the document and relating it to other items in the collection.

This paper is organized as follows. After introducing the problem setting in the next section, the proposed approach is described in Section III. This approach is then evaluated in Section IV, before concluding the paper.

II. Preliminaries and Problem Setting

A table contains a rectangular configuration of data cells, each of which can be uniquely referred by a row and a column index. In general, six kinds of table-related elements can be distinguished:

- **Caption**: explanatory text placed above the table;
- **Data**: The set of cells containing the actual information carried by the table;

- **Column Heading**: The cells placed above the table data, aimed at explaining part of the dimensions according to which the data are organized;
- **Row Heading**: The cells placed to the left of the table data, aimed at explaining the remaining part of the dimensions according to which the data are organized;
- **Stub**: the cells placed at the intersection between the horizontal projection of the row heading and the vertical projection of the column heading;
- **Notes**: One or more optional text lines following the table, aimed at explaining portions of its content.

Row and column headings may be quite complex, when the table is intended to represent data that are conceptually distributed along more than two dimensions. In such a case, the row and/or column header must accommodate more than one dimensions, which typically causes some cell to span over many rows or columns as a side effect. *content-cells* are identified by a *column-header path* and a *row-header-path* [4]. The stub can be made up of just one cell (when there are just two dimensions, one reported on the column header and the other on the row header) or of many cells (when the column and/or row header accommodate many dimensions). It may be empty, but it often contains a meta-header aimed at explaining the row and/or column headings.

The Document Analysis literature has faced several kinds of table-related tasks. Some concern the distinction between genuine tables (aimed at representing and organizing meaningful information) from those just aimed at obtaining a spatial partition of the page (as in most Web documents) [5]. Others face table boundary identification [6] and table structure decomposition [7], or the classification of tables according to their type of content and intended exploitation [1]. Table search [8] or table classification [5] are also investigated. [4] exploits the table structure for a formal manipulation aimed at transposing the content into a standard relational database representation.

Here we are interested in table structure identification, which is preliminary to many high-level processing steps. High accuracy is required if the table data are to be extensively and precisely extracted. Although the mutual position of the elements that make up a table is known and fixed, identifying their specific boundaries may be very complex. Nagy et al. adopted an algorithmic approach [9] leveraging typical patterns, and later tried to improve its accuracy using propositional Machine Learning based on information on the 8-neighbor cells of the cell to be classified [10]. While obtaining interesting results, this approach does not fully exploit the relational structure of the whole table. Hence, we believe that a fully relational approach can further improve recognition performance. Moreover, the candidate solution must also be flexible enough to capture all the different kinds of tables that can be found in documents, and the ways in which information can be organized therein.

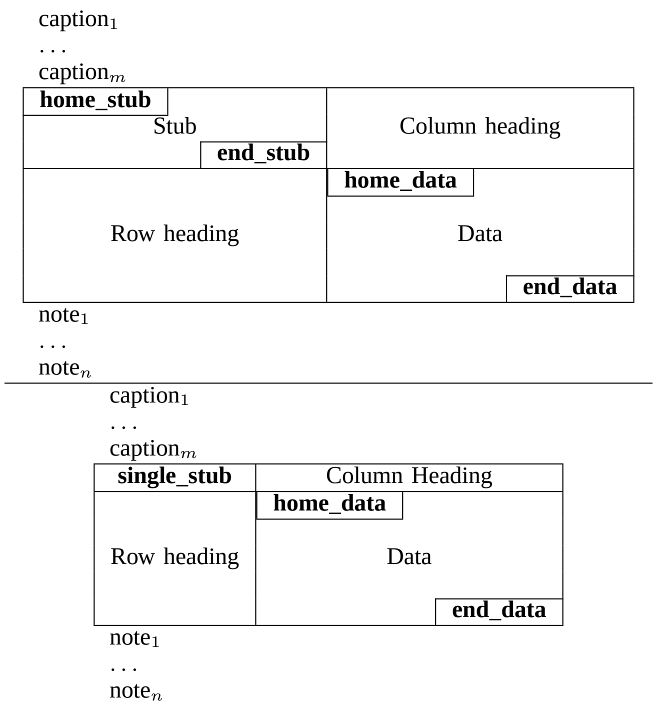

Figure 1. Classes for the table structure learning problem

Following [4], we assume that the input table is represented in a Comma Separated Values (CSV) file including the whole set of table-related elements (stub, table cells and headings, caption, notes). This file provides no structural hint to distinguish different kinds of elements (not even caption and notes, if any). In particular, the content of multi-row or multi-column cells (including caption and notes) is assumed to be placed in the (top-left)-most cell of the group.

The table segmentation process aims at identifying four *critical cells* useful to partition the table into *stub*, *row header*, *column header*, and *data* regions (see Figure 1). Learning, for each cell, the type of table component to which it belongs, would cause a significant growth in the number of examples, which would affect computational costs. It would also be more difficult to handle in the subsequent classification phase, because each cell would be classified independently of the others (so that, for example, a data cell might be identified in the heading area). To solve the former problem, and to reduce the impact of the latter, a different solution was adopted. Four classes were defined as shown in Figure 1, that are non-redundant and are sufficient, alone, to univoquely determine the whole table structure:

- **home_stub**: The top-left cell in the stub;
- **end_stub**: The bottom-right cell in the stub;
- **home_data**: The top-left cell in the data;
- **end_data**: The bottom-right cell in the data.

Indeed, the captions can be identified as the content cell above the home_stub row, and the notes as the content cells below the end_data row; the column heading cells are those in the columns to the right of the end_stub column and in the rows between the home_stub row and the end_stub row; the row heading cells are those in the rows below the end_stub row and in the columns between the home_stub column and the end_stub column. Assuming that the home_data cell is always placed just one column to the right, and one row below, the end_stub, either of the two is redundant. Conversely, if classes are considered as mutually exclusive, an additional class must be included for the case of overlapping home_stub and end_stub:

- **single_stub**: The stub cell, in the case of a single-cell stub.

Humans understand tables and their components based on spatial structure and content regularities among cells. Propositional techniques are not sufficiently powerful to handle this kind of complexity. Switching to the first-order logic setting, the following features might be suitable for table description:

- Table boundaries
- Columns and Rows, and adjacency between them
- Cells and their belonging to a given row and column
- Cell content type

However, being the problem very multi-faceted, and due to the lack of stable criteria to identify the table components, the contribution of statistical approaches is also advised to improve flexibility.

III. Statistical Relational Learning Approach

To learn to recognize critical cells we used the SRL algorithm Lynx [11], [12]. It is a probabilistic query-based classifier that uses first-order logic as a representation language. A first-order *alphabet* consists of a set of *constants*, a set of *variables*, a set of *function symbols*, and a non-empty set of *predicate symbols*. An atom $p(t_1, \ldots, t_n)$ is a predicate symbol p of arity n applied to n terms t_i (constant or variable symbols).

Lynx adopts a *selective propositionalization (featurization)* approach to feature engineering, that combines a feature construction phase with a feature selection process. The feature construction process is carried out by mining frequent queries with an approach similar to that in [13], based on the same idea as the generic level-wise search method [14]. The algorithm starts with the most general queries, and then, at each step tries to specialize all the candidate frequent queries, storing those whose length is equal to the user specified input parameter. The algorithm uses a background knowledge $\mathcal{B}$ containing both the training examples and a set of constraints that must be satisfied by the generated queries.

Let $\mathcal{X}$ be the input space of relational examples, and $\mathcal{Y} = \{1, 2, \ldots, Q\}$ denote the finite set of possible class labels. Given a training set $D = \{(X_i, Y_i)\}_{i=1,\ldots,m}$, where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$, the goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ that predicts the label for each unseen example. Let $\mathcal{P}$, with $|\mathcal{P}| = d$, be the set of features constructed in the first phase. For each example X_k we can build a d-component vector-valued random variable $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ where each x_i is 1 if the query $p_i \in \mathcal{P}$ subsumes example X_k, or 0 otherwise.

The posterior probability $p(Y_j|\mathbf{x})$ can be computed as

$$p(Y_j|\mathbf{x}) = \frac{p(\mathbf{x}|Y_j)p(Y_j)}{\sum_{i=1}^{Q} p(\mathbf{x}|Y_i)p(Y_i)}.$$

Using a maximum a posteriori probability (MAP) approach we classify examples by considering the the maximum discriminant function $g_i(\mathbf{x}) = P(Y_i|\mathbf{x})$. Considering the components of vector $\mathbf{x}$ as conditionally independent, we can write the class-conditional probability as $P(\mathbf{x}|Y_j) = \prod_{i=1}^{d}(p_{ij})^{x_i}(1 - p_{ij})^{1-x_i}$, being $p_{ij} = \text{Prob}(x_i = 1|Y_j)_{\substack{i=1,\ldots,d \\ j=1,\ldots,Q}}$.

Given these methods to construct a set of features and to use them to classify unseen examples, the problem is how to find a subset of these features that optimizes the prediction accuracy. Let $\mathcal{P}$ be the constructed original set of queries, and let $f : 2^{|\mathcal{P}|} \to \mathbb{R}$ be a function scoring a selected subset $X \subseteq \mathcal{P}$. The problem of feature selection is to find a subset $\widehat{X} \subseteq \mathcal{P}$ such that $f(\widehat{X}) = \max_{Z \subseteq \mathcal{P}} f(Z)$. An exhaustive approach to this problem would require examining all $2^{|\mathcal{P}|}$ possible subsets of the feature set $\mathcal{P}$, which is impractical even for small values of $|\mathcal{P}|$. The use of a stochastic local search procedure [15] allows to obtain *good* solutions without having to explore the whole solution space.

Given a subset $P \subseteq \mathcal{P}$, for each example $X_j \in \mathcal{X}$ we let the classifier find the MAP hypothesis $\widehat{h}_P(X_j) = \arg\max_i g_i(\mathbf{x}_j)$, where $\mathbf{x}_j$ is the feature based representation of example X_j obtained using queries in P. The optimization problem corresponds to minimize the expectation $\mathrm{E}[\mathbf{1}_{\widehat{h}_P(X_j) \neq Y_j}]$ where $\mathbf{1}_{\widehat{h}_P(X_j) \neq Y_j}$ is the characteristic function of training example X_j returning 1 if $\widehat{h}_P(X_j) \neq Y_j$, or 0 otherwise.

Consider a *combinatorial optimization* problem, where one is given a discrete set X of solutions and an objective function $f : X \to \mathbb{R}$ to be minimized, and seek a solution $x^* \in X$ such that $\forall x \in X : f(x^*) \leq f(x)$. High-quality solutions for a combinatorial problem can be found using a two-step approach made up of a greedy construction phase followed by a perturbative local search [15]. GRASP [16] solves the problem of the limited number of different candidate solutions generated by a greedy construction search method by randomizing the construction method. GRASP is an iterative process combining at each iteration a construction and a local search phase. In the construction phase a feasible solution is built, and then its neighborhood is explored by the local search. Lynx includes an implementation of GRASP to perform the feature selection task [11].

```
label(c_2_1, single_stub).
label(c_1_1, caption).
label(c_3_2, home_data).
...
cell(c_1_1, alphanumeric).
cell(c_2_1, empty).
cell(c_2_2, integer).
cell(c_3_2, numericSymbol).
...
right(c_2_1,c_2_2).
left(c_2_2,c_2_1).
bottom(c_1_1, c_2_1).
top(c_2_1, c_1_1).
...
```

Figure 2. An example of a relational table description.

	HS	ES	SS	HD	ED	N	Total	#errors
HS	14.6	0.2	1.0	0.0	0.0	0.4	16.2	**1.6**
ES	0.0	14.6	0.4	0.0	0.0	1.2	16.2	**1.6**
SS	0.8	0.0	22.2	0.0	0.0	0.2	23.2	**1.0**
HD	0.0	0.0	0.0	37.8	0.0	1.6	39.4	**1.6**
ED	0.0	0.0	0.0	0.0	38.4	1.0	39.4	**1.0**
N	0.2	8.0	2.8	36.2	20.4	4214.0	4281.6	**67.6**
							4416	**220.2**

Table I
LOCAL CLASSIFICATION

IV. EXPERIMENTAL RESULTS

We ran several experiments on a dataset consisting of 200 Comma Separated Value (CSV) files, each containing the description of an HTML table randomly selected from 10 large statistical Web sites [17]. In a previous work [18] we exploited `Lynx` to learn a classification model for the critical cells using an independent classification approach on a subset of this dataset. Here, we aim at correctly predicting the label of the critical cells belonging to each table in a collective way. In particular, given the set C of cells in a table T, we want to find the critical cells of T among all the cells in C. Furthermore, we changed the relational description language for the examples.

In our new relational representation, each table cell has an identifier. Given a table, a `cell/2` atom is introduced for each cell in the CSV file, reporting as arguments the corresponding identifier and the type of content. `right/2`, `left/2`, `bottom/2` and `top/2` atoms express the spatial relationships between pairs of adjacent cells. Each cell is also associated with a `label/2` atom reporting its class, among: `home_data` (HD), `end_data` (ED), `home_stub` (HS), `end_stub` (ES), `single_stub` (SS), `data` (D), and `none` (N). Cells labeled as `data` are those belonging to the data region, while cells labeled as `none` are those outside the stub and the data region, and being neither captions nor notes. Notes and captions are simply identified based on the location of HS/SS and ED. Figure 2 reports a sample table description expressed in our relational language.

	HS	ES	SS	HD	ED	N	Total	#errors
HS	14.6	0.0	1.0	0.0	0.0	0.0	15.6	**1.0**
ES	0.0	13.0	0.4	0.0	0.0	0.0	13.4	**0.4**
SS	0.2	0.0	22.8	0.0	0.0	0.0	23.0	**0.2**
HD	0.0	0.0	0.0	36.4	0.0	0.0	36.4	**0.0**
ED	0.0	0.0	0.0	0.0	38.2	0.0	38.2	**0.0**
N	0.2	2.0	0.2	3.0	1.2	4282.8	4289.4	**6.6**
								8.2/4416

Table II
GLOBAL CLASSIFICATION

Given a training set, `Lynx` was applied to the relational descriptions of critical cells belonging to each table in order to construct the relevant relational features maximizing the likelihood on the training data, as reported in Section III. Then, the system builds models made up of probabilistic queries such as:

$$q = \{\ \texttt{label(A), type(A,alphabetic),}\ \texttt{top(A,B), type(B,numeric)}\ \}$$

with corresponding class probabilities, e.g. $p(q|HD) = 0.0$, $p(q|ED) = 0.006$, $p(q|SS) = 0.205$, $p(q|HS) = 0.271$ $p(q|ES) = 0.029$, $p(q|D) = 0.005$, and $p(q|N) = 0.066$. These probabilistic queries are then used to predict critical cells belonging to test tables.

The first experiment aimed at validating the proposed approach by comparing it to [18] on classifying the cells independently from the table they belong. The dataset, consisting of 23076 cells, was randomly split into 5 folds having 4416 cells each on average. Table I reports the results obtained by `Lynx` with a 5-fold cross-validation in terms of accuracy (values averaged over the 5 folds). The system made 220.2 errors per fold on average (4.98%). It performed best on the two labels regarding the data region, while it had some difficulties in correctly classifying labels `HS` and `ES`.

In the next experiment we collectively classified all the cells belonging to the same table in order to assess how many tables were correctly segmented. In order to solve this problem the following approach was adopted. For each table T in the test set, let S be the set of cells in T; then:

1) the cell with label $C \in \{$ HS, ES, SS, ED $\}$ is identified solving the maximization $\arg\max_{s \in S} P(C|\mathbf{x}_s)$; i.e., fixed class C, we find the cell $s \in S$ with MAP for C;
2) however, we label a cell s as SS (=HS=ES) if $P(SS|\mathbf{x}_s) > P(HS|\mathbf{x}_s)$ or $P(SS|\mathbf{x}_s) > P(ES|\mathbf{x}_s)$;
3) finally, we label as HD the bottom-right 8-neighbor of the cell labeled as ES.

As Table II shows, the number of classification errors was thus reduced from 220.2 to just 8.2 (0.18%) per fold on average. While the data region is now correctly identified, the problem persists on classes HS and ES. For the sake of comparison, in [10] the same problem was tackled, obtaining a 1.38% error rate. Finally, Table III reports the number of

#errors	#tables
0	35.0 (88.9%)
1	1.2 (3.0%)
2	2.6 (6.6%)
3	0.6 (1.5%)

Table III
CORRECTLY CLASSIFIED TABLES

errors per table in each fold on average. We were able to correctly identify the structure of 88.9% tables.

V. CONCLUSIONS

Tables are very informative components of documents, that compactly represent many inter-related data. It would be desirable to extract these data in order to make them available also outside the document. This requires to understand a table structure. Machine learning solutions may help to deal with the extreme variability in table styles and structures.

In this paper we proposed the exploitation of a Statistical Relational Learning approach, that is able to model the complex spatial relationships involved in a table structure, by mixing the power of a relational representation formalism with the flexibility of a statistical learning tool.

Experiments on a real-world dataset are reported both for single cell classification and for overall table structure recognition, whose results prove the validity of the proposed approach.

REFERENCES

[1] S. Kim and Y. Liu, "Functional-based table category identification in digital library," in *International Conference on Document Analysis and Recognition*, 2011, pp. 1364–1368.

[2] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[3] F. Esposito, S. Ferilli, T. M. Basile, and N. Di Mauro, "Machine learning for digital document processing: From layout analysis to metadata extraction," in *Machine Learning in Document Analysis and Recognition*, ser. Studies in Computational Intelligence, S. Marinai and H. Fujisawa, Eds. Springer, 2008, vol. 90, pp. 105–138.

[4] G. Nagy, S. C. Seth, D. Jin, D. W. Embley, S. Machado, and M. S. Krishnamoorthy, "Data extraction from web tables: The devil is in the details," in *International Conference on Document Analysis and Recognition*, 2011, pp. 242–246.

[5] Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," in *Proceedings of WWW*, 2002, pp. 242–250.

[6] Y. Liu, P. Mitra, and C. Giles, "Identifying table boundaries in digital documents via sparse line detection," in *Proceedings of CIKM-08*, 2008.

[7] T. Kieninger, "Table structure recognition based on robust block segmentation," in *Proc. Document Recognition V*, vol. 3305. SPIE, 1998, pp. 22–32.

[8] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Tableseer: Automatic table metadata extraction and searching in digital libraries categories and subject descriptors," in *Proceedings of JCDL-07*, 2007, pp. 91–100.

[9] G. Nagy and M. Tamhankar, "Vericlick: an efficient tool for table format verification," in *19th Document Recognition and Retrieval Conference (DRR)*, ser. SPIE Proceedings, C. Viard-Gaudin and R. Zanibbi, Eds., vol. 8297. SPIE, 2012.

[10] G. Nagy, "Learning the characteristics of critical cells from web tables," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, 2012, p. 4.

[11] N. Di Mauro, T. M. Basile, S. Ferilli, and F. Esposito, "Optimizing probabilistic models for relational sequence learning," in *19th International Symposium on Methodologies for Intelligent Systems*, ser. LNCS, M. Kryszkiewicz, H. Rybinski, A. Skowron, and Z. W. Ras, Eds. Springer, 2011, pp. 240–249.

[12] F. Esposito, N. Di Mauro, T. Basile, and S. Ferilli, "Multidimensional relational sequence mining," *Fundamenta Informaticae*, vol. 89, no. 1, pp. 23–43, 2008.

[13] S. Kramer and L. De Raedt, "Feature construction with version spaces for biochemical applications," in *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2001, pp. 258–265.

[14] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the International Conference on Data Engineering*, 1995, pp. 3–14.

[15] H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[16] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–133, 1995.

[17] G. Nagy, R. Padmanabhan, R. C. Jandhyala, W. Silversmith, and M. Krishnamoorthy, "Table metadata: Headers, augmentations and aggregates," in *Ninth IAPR International Workshop on Document Analysis Systems*, 2010.

[18] N. Di Mauro, S. Ferilli, and F. Esposito, "Learning to recognize critical cells in document tables," in *8th Italian Research Conference on Digital Libraries and Archives*, ser. CCIS, M. Agosti, F. Esposito, S. Ferilli, and N. Ferro, Eds., vol. 354. Springer, 2012, pp. 105–116.