# Learning to Recognize Critical Cells in Document Tables

Nicola Di Mauro[1,2], Stefano Ferilli[1,2], and Floriana Esposito[1,2]

[1] Dipartimento di Informatica, LACAM laboratory
Università degli Studi di Bari "Aldo Moro"
{ndm,ferilli,esposito}@di.uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni
Università degli Studi di Bari "Aldo Moro"

**Abstract.** Tables are among the most informative components of documents, because they are exploited to compactly and intuitively represent data, typically for understandability purposes. The needs are to identify and extract tables from documents, and, on the other hand, to be able to extract the data they contain. The latter task involves the understanding of a table structure. Due to the variability in style, size, and aims of tables, algorithmic approaches to this task can be insufficient, and the exploitation of machine learning systems may represent an effective solution. This paper proposes the exploitation of a first-order logic representation, that is able to capture the complex spatial relationships involved in a table structure, and of a learning system that can mix the power of this representation with the flexibility of statistical approaches. The obtained encouraging results suggest further investigation and refinement of the proposal.

## 1 Introduction

The main motivation underlying the birth and spread of libraries consisted in collecting large quantities of documents, usually in paper format, with preservation and access objectives. Each library was typically characterized by a specific focus-of-interest, that established the direction and limits according to which the collections were developed, thus helping users to have in one place a complete landscape of the information they were interested in. As a technological counterpart, digital libraries have the same aims and objectives, but dealing with documents in digital format. This change of medium has a dramatic impact on the management of the collections, and on their exploitation by end-users. Huge quantities of documents can be easily collected and spread all over the world using the Internet, however, the users may experience difficulties in properly retrieving the data they are interested in. Information Retrieval (IR) and Machine Learning (ML) technologies can provide automatic tools to support such activities.

The identification of relevant documents that can satisfy the users' query is the subject of the IR research field. Of course, to provide more effective results

the automatic techniques should move from the purely lexical aspects of document to those concerning their semantics, which is still an open research issue. But having a thorough understanding of the document content is not only useful to support search and retrieval. It is also a fundamental requirement to be able to correlate the documents, and in particular the data and information they carry. In particular, a component of documents that is usually very informative and information-dense are tables. Authors use tables to compactly represent many important data in a small space, to draw more attention from readers, or for information comparison [10]. 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 proposes a set of intelligent techniques that cover the steps going from a document in digital format up to the identification of tables and the understanding of their structure, and particularly focuses on the exploitation of Machine Learning methodologies and systems for understanding the table structure.

## 2 Preliminaries

Many works are present in the literature concerning tables and their analysis, focusing on different objectives, aspects and problems. Some concern theoretical contributions, such as the distinction between genuine tables (tables aimed at representing and organizing meaningful information) and non-genuine ones (tables just aimed at obtaining a spatial partition of the page, as in most Web documents) [16]. This is a relevant issue, since, according to [2], only 1% Web tables are genuine. Others face more practical problems, such as table boundary identification [13] and table structure decomposition [9], or the classification of tables according to their type of content and intended exploitation. In addition to table data extraction, table functionality analysis (aimed at understanding table types, functions, and purposes) is another crucial task for table understanding, table data sharing and reuse [10]. Yet other researchers focus on applications such as table search [12] or table classification [16]. Indeed, accurately extracting tables from document repositories is a challenging problem, but also selecting interesting tables from the set of collected tables is an open issue.

As concerns the table identification step, we considered the DOMINUS framework [5] for document processing and management, and extended it with suitable techniques for table recognition. DOMINUS provides an integrated and general framework to manage digital libraries in which most knowledge-intensive tasks are carried out using intelligent techniques, among which Expert System and symbolic Machine Learning technologies play a predominant role. After submission, documents in different digital formats are processed to identify their layout structure, then to identify the kind of document and its relevant components, to extract the content from selected components and to exploit such a content

for indexing and information extraction purposes. Hence, while the layout analysis phase is involved in table recognition, the information extraction step is concerned with table structure identification and subsequent content extraction.

As to table recognition, DOMINUS deals with two kinds of digital documents: born-digital ones and digitized ones (typically obtained by scanning legacy paper sources). This distinction is relevant because the basic document components (text, images, geometric shapes—and specifically lines) are explicitly represented in born-digital documents (such as PDF ones), but not in digitized ones (usually coming in the form of raster images). Thus, in the latter case, suitable image processing techniques must be applied to identify them. In particular, horizontal and vertical lines are fundamental components for table recognition, although not sufficient (some tables do not show a perfect grid for visually highlighting their cell organization). Thus, in the case of document images, a variation of the Hough transform, specifically focused on horizontal/vertical lines, and on the identification of line segments, was developed and integrated in the DOMINUS framework. Then, the set of lines and other content blocks in a page were fed as an input to an expert module in charge of identifying and collecting the subsets of elements that together make up a table. Expert Systems technology was exploited because there is no standard representation for tables, and the many different styles used by different authors can vary significantly as regards the alignment of the content of rows and columns, the use of horizontal/vertical lines to separate portions of the table, and the position of the table in the page. Moreover, some tables are particularly tricky due to the presence of blank cells, or of cells that span several rows and/or columns. The expert component, whose detailed description is outside the scope of this paper, was able to identify and extract most tables in different kinds of documents, with some difficulties on very small tables and on multi-column documents.

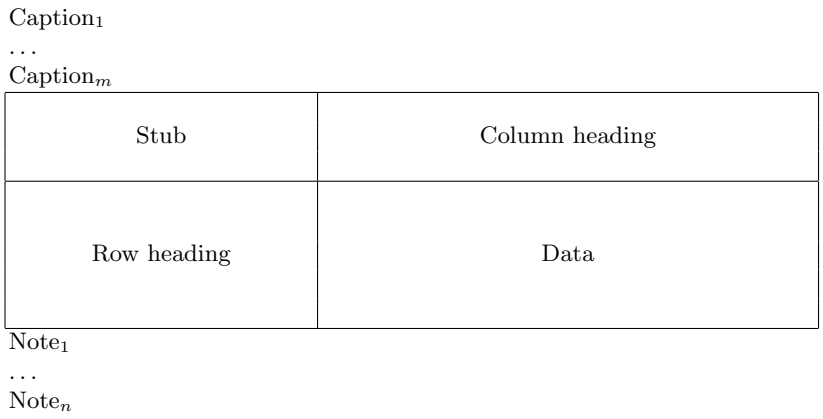


**Fig. 1.** Table structure according to Nagy et al.

As to table structure identification, our work specifically stems from a research stream carried on by Nagy et al. [15], specifically concerned with the extraction of the table structure and with its formal manipulation aimed at transposing the content into a relational representation that can be integrated in a typical database. They presented [15] a method based on *header paths* for efficient and complete extraction of labeled data from tables meant for humans. Header paths are a purely syntactic representation of visual tables that can be transformed (*factored*) into existing representations of structured data such as category trees, relational tables, and RDF triples.

A table contains a rectangular configuration of data cells, each of which can be uniquely referred by a row and a column index. As reported in [15], a table contains a set of *content-cells* that can be identified by a *column-header path* and a *row-header-path*. The table segmentation process aims at identifying four *critical cells* useful to partition the table into *stub*, *row header*, *column header*, and *delta* regions. In particular, this setting is concerned with six kinds of table-related elements, as shown in Figure 1:

**Caption** A text placed above the table, aimed at explaining it;
**Data** The set of cells containing the actual information carried by the table;
**Column Heading** The table cells placed above the table data, aimed at explaining part of the dimensions according to which the data are organized;
**Row Heading** The table 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** One or more cells that correspond to the intersection between the horizontal projection of the row heading and the vertical projection of the column heading;
**Notes** One or more text lines following the table, aimed at explaining portions of its content.

Some details should be pointed out. First of all, the notes are optional, and hence might be missing in some tables. The 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 (as a side effect, this event typically causes the presence of cell content that spans over many rows or columns). The stub can be made up of just one cell (if both the row headings consist of a single column, and the column headings consist of a single row) or of many cells (in case of composite row and/or column headings); it may be empty, but it often contains a meta-header aimed at explaining the row and/or column headings.

Thus, although the mutual position of these elements is known and fixed, identifying the specific boundaries of each of them may become very complex. To do this Nagy et al. [15] adopt an algorithmic approach, leveraging typical patterns. High accuracy should be required if the table data in the available documents are to be extensively and precisely extracted. Due to the many different kinds of tables that can be found in documents, and to the many different ways in which information can be organized in tables, we believe that a significant high accuracy cannot be reached by hand-written rules, but the characterizing

essence of the above elements can be captured only using automatic techniques provided by Machine Learning.

## 3 Proposed Approach

The first question to answer for applying Machine Learning to table structure recognition is the type of approach to be used. To answer this question, several aspects must be considered. A fundamental one concerns the kind of representation to be exploited. In this respect, it is quite clear that the most important feature to understand a table lies in its spacial structure, which in turn is made up of several relationships among the cells (both spacial and content ones). Indeed, it is self-evident that, when trying to understand a table, and specifically its components as described above, these are the parameters that any human considers. As a consequence, propositional techniques don't have a sufficient representational power to handle this kind of complexity, and first-order logic approaches must be considered. In particular, the following features/predicates were deemed as profitable 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

It should be noted that, in the proposed setting, the whole set of elements (stub, table cells and headings, caption, notes) associated to a table is represented in a Comma Separated Values (CSV) file, and hence in this file not only the actual table elements, but also caption and notes (if any) are represented as cells. Thus, there is no structural hint in the CSV file to distinguish different kinds of elements. In particular, caption and notes are considered as belonging to a single cell (typically in the first column), and the content of multi-row or multi-column cells is assumed to be placed in the (top-left)-most cell.

Another issue is the choice of classes to be learned. A straightforward possibility would be learning, for each cell, the type of table component to which it belongs. However, this would cause a significant growth in the number of examples, which would affect computational costs, and would 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 section). 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 2, 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.

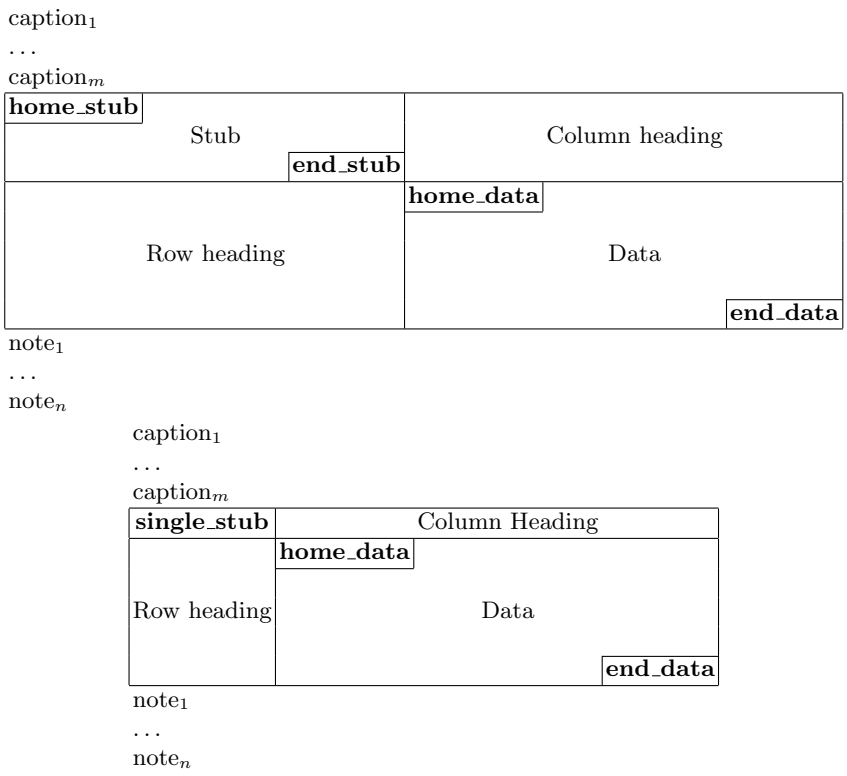


**Fig. 2.** Classes for the table structure learning problem

In fact, either end_stub or home_data is redundant, because the home_data cell is always assumed to be placed just one column to the right, and one row below, the end_stub. Conversely, if classes are to be considered mutually exclusive, an additional class must be included, to specifically identify the case of a stub in which home_stub and end_stub coincide:

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

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.

The last question concerns how rigid the learned models should be. Due to the problem being very multi-faceted, and to the lack of stable criteria to identify the table components, it is desirable that the learned models are quite flexible, with a preference for statistical approaches over purely logical ones.

### 3.1 Lynx: A Statistical Relational Learning Approach

The SRL approach `Lynx` [3] was used here to tackle the specific problem of critical cells identification in tables. `Lynx` implements a probabilistic query-based classifier, using 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*. Both function symbols and predicate symbols have a natural number (its *arity*) assigned. A *term* is a constant symbol, a variable symbol, or an $n$-ary function symbol $f$ applied to $n$ terms $t_1, t_2, \ldots, t_n$. An atom $p(t_1, \ldots, t_n)$ is a predicate symbol $p$ of arity $n$ applied to $n$ terms $t_i$. An atom $l$ and its negation $\bar{l}$ are said to be (resp., positive and negative) *literals*. `Lynx` adopts the relational framework, and the corresponding query mining algorithm, reported in [4].

**Feature Construction via Query Mining** The first step of `Lynx` carries out a feature construction process by mining frequent queries with an approach similar to that reported in [11]. The algorithm for frequent relational query mining is based on the same idea as the generic level-wise search method, known in data mining from the Apriori algorithm [1]. The algorithm starts with the most general queries. Then, at each step it tries to specialize all the candidate frequent queries, discarding the non-frequent queries and storing those whose length is equal to the user specified input parameter `maxsize`. For each new refined query, semantically equivalent queries are detected (using the $\theta_{\text{OI}}$-subsumption relation [7]) and discarded. The algorithm uses a background knowledge $\mathcal{B}$ containing the examples and a set of constraints that must be satisfied by the generated queries, among which: `maxsize(M)`, maximal query length; `type(p)` and `mode(p)`, denote, respectively, the type and the input/output mode of the predicate's arguments `p`, used to specify a language bias; `key([`$p_1, p_2, \ldots, p_n$`])` specifies that each query must have one of the predicates $p_1, p_2, \ldots p_n$ as a starting literal. Given a set of relational examples $D$ defined over a set of classes $C$, the *frequency* of a query $p$, $\text{freq}(p, D)$, corresponds to the number of examples $s \in D$ such that $p$ subsumes $s$.

**Query-based Classification** After identifying the set of frequent queries, the next question is how to use them as features in order to correctly classify unseen examples. 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 $D = \{(X_i, Y_i) | 1 \leq i \leq m\}$, where $X_i \in \mathcal{X}$ is a single relational example and $Y_i \in \mathcal{Y}$ is the label associated to $X_i$, the goal is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from $D$ that predicts the label for each unseen instance. Let $\mathcal{P}$, with $|\mathcal{P}| = d$, be the set of constructed features obtained in the first step of the `Lynx` system (the queries mined from $D$), as previously reported. For each example $X_k \in \mathcal{X}$ we can build a $d$-component vector-valued random variable $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ where each $x_i \in \mathbf{x}$ is 1 if the query $p_i \in \mathcal{P}$ subsumes example $X_k$, and 0 otherwise, for each $1 \leq i \leq d$.

Using Bayes' theorem, if $p(Y_j)$ describes the prior probability of class $Y_j$, then the posterior probability $p(Y_j|\mathbf{x})$ can be computed from $p(\mathbf{x}|Y_j)$ as $p(Y_j|\mathbf{x}) = \frac{p(\mathbf{x}|Y_j)p(Y_j)}{\sum_{i=1}^{Q} p(\mathbf{x}|Y_i)p(Y_i)}$. Given a set of discriminant functions $g_i(\mathbf{x})$, $i = 1, \ldots, Q$, a classifier is said to assign vector $\mathbf{x}$ to class $Y_j$ if $g_j(\mathbf{x}) > g_i(\mathbf{x})$ for all $j \neq i$. Taking $g_i(\mathbf{x}) = P(Y_i|\mathbf{x})$, the maximum discriminant function corresponds to the *maximum a posteriori* (MAP) probability. For minimum error rate classification, the following discriminant function will be used

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|Y_i) + \ln P(Y_i). \tag{1}$$

We are considering a multi-class classification problem involving discrete features. In this problem the components of vector $\mathbf{x}$ are binary-valued and conditionally independent. In particular, let the component of vector $\mathbf{x} = (x_1, \ldots, x_d)$ be binary valued (0 or 1). We define $p_{ij} = \text{Prob}(x_i = 1|Y_j)_{\substack{i=1,\ldots,d \\ j=1,\ldots,Q}}$ with the components of $\mathbf{x}$ being statistically independent for all $x_i \in \mathbf{x}$. The factors $p_{ij}$ can be estimated by frequency counts on the training examples, as $p_{ij} = \text{support}_{Y_j}(p_i)$.

By assuming conditional independence we can write $P(\mathbf{x}|Y_i)$ as a product of the probabilities of the components of $\mathbf{x}$. Given this assumption, a particularly convenient way of writing the class-conditional probabilities is as follows: $P(\mathbf{x}|Y_j) = \prod_{i=1}^{d} (p_{ij})^{x_i} (1-p_{ij})^{1-x_i}$. Hence, Eq. 1 yields the discriminant function

$$g_j(\mathbf{x}) = \ln p(\mathbf{x}|Y_j) + \ln p(Y_j) = \sum_{i=1}^{d} x_i \ln \frac{p_{ij}}{1-p_{ij}} + \sum_{i=1}^{d} \ln(1-p_{ij}) + \ln p(Y_j). \tag{2}$$

The factor corresponding to the prior probability for class $Y_j$ can be estimated from the training set as $p(Y_i) = \frac{|\{(X,Y) \in D \text{ s.t. } Y = Y_i\}|}{|D|}, 1 \leq i \leq Q$. The minimum probability of error is achieved by the following decision rule: decide $Y_k$, $1 \leq k \leq Q$, if $\forall j, 1 \leq j \leq Q \wedge j \neq k : g_k(\mathbf{x}) \geq g_j(\mathbf{x})$, where $g_i(\cdot)$ is defined as in Eq. 2.

**Feature Selection with Stochastic Local Search** After constructing a set of features, and presenting a method to use those features to classify unseen examples, the problem is how to find a subset of these features that optimizes prediction accuracy. The optimization problem of selecting a subset of features with a superior classification performance may be formulated as follows. 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 $\mathcal{P}$, making it impractical for even small values of $|\mathcal{P}|$. The use of a stochastic local search procedure [8] 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 $\widehat{h}_P(X_j) = \arg\max_i g_i(\mathbf{x}_j)$ by adopting the discriminant

function reported in Eq. 1, where $\mathbf{x}_j$ is the feature based representation of example $X_j$ obtained using queries in $P$. Hence the initial 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 $\widehat{h}_P(X_j) \neq Y_j$, and 0 otherwise. Finally, given $D$ the training set with $|D| = m$ and $P$ a set of features, the number of classification errors made by the Bayesian model is $err_D(P) = m\mathrm{E}[\mathbf{1}_{\widehat{h}_P(X_j)\neq Y_j}]$.

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 $\forall x \in X : f(x^*) \leq f(x)$. A method to find high-quality solutions for a combinatorial problem consists of a two-step approach made up of a greedy construction phase followed by a perturbative local search [8]. `GRASP` [6] 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. The `Lynx` system includes an implementation of the `GRASP` procedure in order to perform the feature selection task, as reported in [3].

## 4 Problem Characterization and Validation

`Lynx` has been applied to a dataset consisting of 100 table descriptions. The dataset[3] is a collection of tables randomly selected from ten large statistical web sites [14]. HTML tables are represented in Comma Separated Value (CSV) files. Information about each table cell (its contained value and its absolute position in terms of row and column index) are used to provide its relational representation to be exploited by `Lynx`. The goal is to correctly predict the label of the critical cells belonging to each table. In particular, each table cell has been labeled to belong to one of the following classes: `caption`, `note`, `home_data`, `end_data`, `home_stub`, `end_stub`, and `single_stub`.

Figure 3 reports a sample table description adopting the relational language we used. In particular, predicate `label/2` indicates the corresponding class label of a cell; `row/3` (resp., `col/3`) define the position and the identifier of a row (resp., column) belonging to the table; `next_row/2` (resp., `next_col/2`) denote the spatial relationship between two adjacent rows (resp., columns); and finally, `cell/4` specifies the type of a cell. Given a table (also including caption and notes, if any), a `row/3` (resp., `col/3`) atom is introduced for each row (resp., column) of the CSV file, reporting as arguments the table identifier, the row (resp., column) identifier, and its index. Then, suitable `next_row/2` (resp., `next_col/2`) atoms are introduced to link adjacent rows (resp., columns) to each other in the

[3] The DocLab Dataset for Evaluating Table Interpretation Methods available at `http://www.iapr-tc11.org/mediawiki/index.php/Datasets_List`.

```
doc_table(c10076).
label(c10076_2_1, single_stub).
label(c10076_1_1, caption).
label(c10076_3_2, home_data).
...
row(c10076, c10076_r_1, 1).
next_row(c10076_r_1, c10076_r_2).
row(c10076, c10076_r_2, 2).
next_row(c10076_r_2, c10076_r_3).
row(c10076, c10076_r_3, 3).
...
col(c10076, c10076_c_1, 1).
next_col(c10076_c_1, c10076_c_2).
col(c10076, c10076_c_2, 2).
next_col(c10076_c_2, c10076_c_3).
col(c10076, c10076_c_3, 3).
...
cell(c10076_1_1, c10076_r_1, c10076_c_1, alphanumeric).
cell(c10076_2_1, c10076_r_2, c10076_c_1, empty).
cell(c10076_2_2, c10076_r_2, c10076_c_2, integer).
cell(c10076_3_2, c10076_r_3, c10076_c_2, numericSymbol).
...
```

**Fig. 3.** An example of a relational table description.

proper sequence. Finally, for each cell a `cell/4` atom is added, reporting as arguments the corresponding identifier, the associated row and column identifiers, and the type of content.

Given a training set made up of the relational descriptions of critical cells belonging to each table, `Lynx` is applied in order to construct the relevant relational features maximizing the likelihood on the training data, as reported in Section 3.1. After this first step the system build a model composed of probabilistic query such as `label(A),cell(B,A,C,D),next_row(C,E),cell(B,_,E,D)`, whose corresponding class probabilities are $p(q|\texttt{note}) = 0.507$, $p(q|home\_data) = 0.944$, $p(q|caption) = 0.497$, $p(q|single\_stub) = 0.628$, $p(q|end\_data) = 0.000$, $p(q|end\_stub) = 0.714$, and $p(q|home\_stub) = 0.548$. These probabilistic queries are then used to predict critical cells belonging to testing tables.

Table 1 reports the results obtained with `Lynx` with a 10-fold cross validation in terms of accuracy, Conditional Log Likelihood (CLL), and areas under the Receiver operating characteristic (ROC) and Precision Recall (PR) curve. As we can see from the table, the results are very promising. The two labels on which the system obtains best results are those regarding the data region. While, it has some difficulties in correctly classifying the labels `single_stub` and `end_stub`. The next step towards improving these results is to use some collective classification techniques.

| | Accuracy | CLL |
|---|---|---|
| Mean | 90,69 | -4,89 |
| Dev.St. | 0,017 | 2,38 |

| | AUC-ROC | AUC-PR |
|---|---|---|
| caption | 0,984 | 0,951 |
| note | 0,986 | 0,978 |
| home stub | 0,987 | 0,925 |
| end stub | 0,983 | 0,825 |
| single stub | 0,989 | 0,810 |
| home data | 0,991 | 0,968 |
| end data | 1,000 | 0,998 |
| **Mean** | 0,989 | 0,922 |
| **Dev.St.** | 0,006 | 0,075 |

**Table 1.** Accuracy, CLL, AUC of ROC and PR with a 10-fold cross validation.

## 5 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. We propose to exploit a first-order logic representation to capture the complex spatial relationships involved in a table structure, and a learning system that can mix the power of this representation with the flexibility of statistical approaches. On a dataset including different kinds of tables, encouraging results were obtained.

As a future work we are trying to combine the prediction of single critical cell labels in order to improve the accuracy of the segmentation process. We will adopt a collective classification procedure whose aim should be to improve the likelihood of a prediction for a given label knowing the probability of the prediction made on the neighbor labeled cells with an iterative approach. The iterative procedure will combine expectation steps, predicting labels on the known distribution, and maximization steps, improving the probability distribution.

### Acknowledgment

The research leading to this paper has been partially funded by the MIUR PRIN 2009 project "A multi-relational approach to spatial and spatio-temporal data mining".

## References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the International Conference on Data Engineering, pp. 3–14 (1995)
2. Cafarella, M., Halevy, A., Wang, Z., Wu, E., Zhang, Y.: Webtables: Exploring the power of tables on the web. In: Proceddings of VLDB (2008)

3. Di Mauro, N., Basile, T.M., Ferilli, S., Esposito, F.: Optimizing probabilistic models for relational sequence learning. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Ras, Z.W. (eds.) 19th International Symposium on Methodologies for Intelligent Systems. pp. 240–249. LNCS, Springer (2011)
4. Esposito, F., Di Mauro, N., Basile, T., Ferilli, S.: Multi-dimensional relational sequence mining. Fundamenta Informaticae 89(1), 23–43 (2008)
5. Esposito, F., Ferilli, S., Basile, T.M., Di Mauro, N.: Machine learning for digital document processing: From layout analysis to metadata extraction. In: Marinai, S., Fujisawa, H. (eds.) Machine Learning in Document Analysis and Recognition, Studies in Computational Intelligence, vol. 90, pp. 105–138. Springer (2008)
6. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
7. Ferilli, S., Di Mauro, N., Basile, T., Esposito, F.: $\theta$-subsumption and resolution: A new algorithm. In: Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E. (eds.) Foundations of Intelligent Systems. LNCS, vol. 2871, pp. 384–391. Springer Verlag (2003)
8. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)
9. Kieninger, T.: Table structure recognition based on robust block segmentation. In: Proc. Document Recognition V. vol. 3305, pp. 22–32. SPIE (1998)
10. Kim, S., Liu, Y.: Functional-based table category identification in digital library. In: International Conference on Document Analysis and Recognition. pp. 1364–1368 (2011)
11. Kramer, S., De Raedt, L.: Feature construction with version spaces for biochemical applications. In: Proceedings of the 18th International Conference on Machine Learning, pp. 258–265. Morgan Kaufmann Publishers Inc. (2001)
12. Liu, Y., Bai, K., Mitra, P., Giles, C.L.: Tableseer: Automatic table metadata extraction and searching in digital libraries categories and subject descriptors. In: Proceedings of JCDL-07. pp. 91–100 (2007)
13. Liu, Y., Mitra, P., Giles, C.: Identifying table boundaries in digital documents via sparse line detection. In: Proceedings of CIKM-08 (2008)
14. Nagy, G., Padmanabhan, R., Jandhyala, R.C., Silversmith, W., Krishnamoorthy, M.: Table metadata: Headers, augmentations and aggregates. In: Ninth IAPR International Workshop on Document Analysis Systems (2010)
15. Nagy, G., Seth, S.C., Jin, D., Embley, D.W., Machado, S., Krishnamoorthy, M.S.: Data extraction from web tables: The devil is in the details. In: International Conference on Document Analysis and Recognition. pp. 242–246 (2011)
16. Wang, Y., Hu, J.: A machine learning based approach for table detection on the web. In: Proceedings of WWW. pp. 242–250 (2002)