Yosveni Escalona Escalona

Algorithms for Table Structure Recognition

DISSERTAÇÃO DE MESTRADO

Dissertation presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Informática

Advisor: Prof. Marco Antonio Casanova

Rio de Janeiro
September 2019

Yosveni Escalona Escalona

Algorithms for Table Structure Recognition

Dissertation presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee.

Prof. Marco Antonio Casanova

Advisor

Departamento de Informática – PUC-Rio

Prof. Luiz André Portes Paes Leme

Universidade Federal Fluminense, Instituto de Computação

Prof. Antonio Luz Furtado

Departamento de Informática – PUC-Rio

Rio de Janeiro, September 17th, 2019

Yosveni Escalona Escalona

The author graduated in Computer Science Engineer from University of Informatics Sciences (UCI), Havana - Cuba in 2007. He joined the Graduate Program in Informatics at Pontifical Catholic University of Rio de Janeiro (PUC-Rio) in 2017.

Bibliographic data

Escalona Escalona, Yosveni

Algorithms for Table Structure Recognition / Yosveni Escalona Escalona; advisor: Marco Antonio Casanova. –2019

v. 61 f.: il.color; 30 cm

1. Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Dados tabulares. 3. Tabelas HTML. 4. Planilhas. 5. Conditional Random Fields. 6. Aprendizagem de Máquina. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

I would like to thank in the first place to God because without him I would not be here today, to my family and especially to my wife, Yialeynis Morales Diaz, for her support, dedication and effort during these years at my side which gave me a wonderful son, Marcos David Escalona Morales, for the also thanks for being my motivation, to sons of my wife, Leonardo and Alianis because are part of me too. To my parents, Juan and Oralina, they always taught for me the best ethical and moral values.

Thank you so much to Professor Marco Antonio Casanova. An excellent advisor which I will always be very grateful for his professionalism and dedication with his students. Thanks again.

To PUC-Rio, CNPq, for the financial support granted during the course.

To all my classmates, professors and staff from the Department of Informatics.

To all the friends I conquered at PUC-Rio and those still in Cuba.

To everyone, in general, thank you very much.

Abstract

Escalona, Yosveni Escalona; Casanova, Marco Antonio (Advisor). **Algorithms for Table Structure Recognition**. Rio de Janeiro, 2019. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Tables are widely adopted to organize and publish data. For example, the Web has an enormous number of tables, published in HTML, imbedded in PDF documents, or that can be simply downloaded from Web pages. However, tables are not always easy to interpret because of the variety of features and formats used. Indeed, a large number of methods and tools have been developed to interpret tables. This dissertation presents the implementation of an algorithm, based on Conditional Random Fields (CRFs), to classify the rows of a table as header rows, data rows or metadata rows. The implementation is complemented by two algorithms for table recognition in a spreadsheet document, respectively based on rules and on region detection. Finally, the dissertation describes the results and the benefits obtained by applying the implemented algorithms to HTML tables, obtained from the Web, and to spreadsheet tables, downloaded from the Brazilian National Petroleum Agency.

Keywords

Tabular Data; HTML Tables; Spreadsheets; Conditional Random Fields; Machine Learning.

Resumo

Escalona, Yosveni Escalona; Casanova, Marco Antonio. **Algoritmos para Reconhecimento de Estruturas de Tabelas**. Rio de Janeiro, 2019. 61p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Tabelas são uma forma bastante comum de organizar e publicar dados. Por exemplo, a Web possui um enorme número de tabelas publicadas em HTML, embutidas em documentos em PDF, ou que podem ser simplesmente baixadas de páginas Web. Porém, tabelas nem sempre são fáceis de interpretar pois possuem uma grande variedade de características e são organizadas de diversas formas. De fato, um grande número de métodos e ferramentas foram desenvolvidos para interpretação de tabelas. Esta dissertação apresenta a implementação de um algoritmo, baseado em Conditional Random Fields (CRFs), para classificar as linhas de uma tabela em linhas de cabeçalho, linhas de dados e linhas de metadados. A implementação é complementada por dois algoritmos para reconhecimento de tabelas em planilhas, respectivamente baseados em regras e detecção de regiões. Por fim, a dissertação descreve os resultados e os benefícios obtidos pela aplicação dos algoritmos a tabelas em formato HTML, obtidas da Web, e a tabelas em forma de planilhas, baixadas do Web site da Agência Nacional de Petróleo.

Palavras-chave

Dados Tabulares; Tabelas HTML; Planilhas; Conditional Random Fields; Aprendizado de Máquina.

Table of contents

List of figures

List of tables

1 Introduction

1.1 Motivation

The volume of data available on the Web has grown enourmously, which makes the Web a vast repository of data that describe our environment and our interactions. The wealth and strength of the Web contributes to the development of today's economy and society.

The Web can be described as an interlinked collection of documents[1] that cover topics that meet the user needs. The data found on the Web are related to product information, articles imparting encyclopedic knowledge, presentations of state-of-the-art scientific results or reports on current financial data. Such data allow us to make the right decisions at the right time. The tasks range from data harvesting to data analysis. For example, developers, journalists, and other professionals manipulate the data to create visualizations and to perform data analysis.

The required data are not always easy to find on the Web. People navigate the Web visually, viewing content, clicking on links and downloading files. Machines, by contrast, only browse the Web through logical rules embedded in the code. While humans look for engaging, interactive content, machines require structure, logic, and clarity.

Another essential aspect to consider is the way data are stored on the Web. Most of the data are typically included in unstructured documents. A large number of such unstructured documents contain critical data in the form of tables, such as financial statements. Often, businesses are required, by law, to provide these documents for public consumption. Thus, the data in these documents need to be extracted and structured.

[1] https://www.w3.org/standards/semanticweb/data

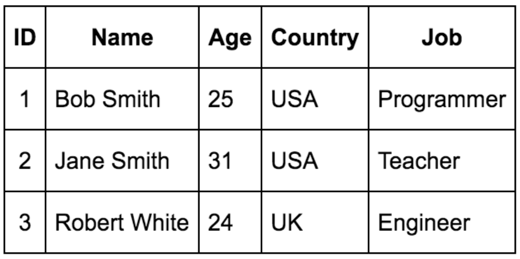
Specifically, tables found on the Web require a particular analysis since they may be expressed in HTML, imbedded in PDF documents or made available as downloadable spreadsheets, among other formats. Tables are usually organized merely and compactly as rows and columns, but they can be much more complex, with metadata and additional information. Other aspects to take into account about spreadsheet documents are the diversity of tables that found in a worksheet. Tables proved to be valuable sources, but their use can be very diversified, ranging from Web search to data discovery in spreadsheets and knowledge base augmentation [1].

Exploring a large set of tables has been a challenge because, in general, table semantics is not known. In [2], a corpus of over 100 million tables is presented, but the meaning of each table is rarely explicit in the table itself. Another challenge is the structure of the table. For example, the tasks described in [3,4,5] focus on recovering table semantics and linking table data with external sources for tables classified as *genuine*, with considerable data loss. These works do not consider fundamental aspects, such as the orientation of the table, and discarded those tables that are classified as *non-genuine*.

In recent years, numerous researchers focused on developing methods and tools that help work with tabular data. Several methods are based on machine learning algorithms, such as matching techniques to uncover the table scheme, and heuristic rules and cell similarities to identify tables [6,7,8]. However, such work presents certain limitations. In the first place, we can mention limitations linked to the loss information or limitations of research that is aimed at data extraction from a single kind of document.

1.2 Tables and Table Row Classification

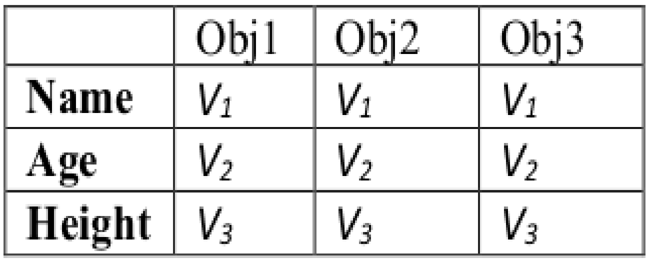
Tables may be distinguished according to their structure and orientation. A *relational* or *horizontal* [9] table, as illustrated in Figure 1, has rows, which provide data about specific objects, called *entities*, and columns, which represent *attributes* that describe the entities.



ID	Name	Age	Country	Job
1	Bob Smith	25	USA	Programmer
2	Jane Smith	31	USA	Teacher
3	Robert White	24	UK	Engineer

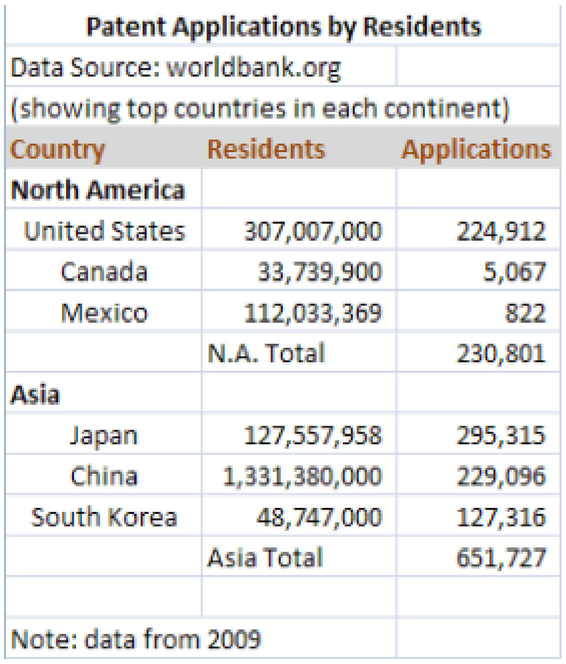
Figure 1: Example of a relational table

More complex types of tables exist, such as those where the attributes that describe the entities are laid vertically and the entities in a horizontal way, as in Figure 2. Tables may even follow a more complex structure with titles and subtotals, as in Figure 3, or have nested tables, i.e., have cells which themselves are tables.



	Obj1	Obj2	Obj3
Name	V_1	V_1	V_1
Age	V_2	V_2	V_2
Height	V_3	V_3	V_3

Figure 2: Example of a non-relational table



Patent Applications by Residents		
Data Source: worldbank.org		
(showing top countries in each continent)		
Country	Residents	Applications
North America		
United States	307,007,000	224,912
Canada	33,739,900	5,067
Mexico	112,033,369	822
	N.A. Total	230,801
Asia		
Japan	127,557,958	295,315
China	1,331,380,000	229,096
South Korea	48,747,000	127,316
	Asia Total	651,727
Note: data from 2009		

Figure 3: Example of a non-relational table with additional information

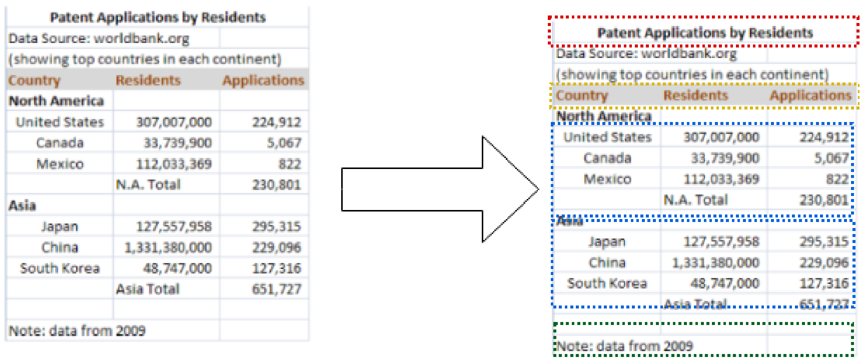
More precisely, a table is defined as follows:

Definition 1: A *table* is a pair $T = (H,D)$ consisting of an optional header H and data D, where:

- The *header* $H = \{h_1,h_2,\ldots,h_n\}$ is an n-tuple of header elements h_i; if the set of header elements exists, it may be represented either as a row or as a column.
- The *data nodes* are organized as an (n,m) matrix consisting of n rows and m columns:

$$D = \begin{bmatrix} C_{11} & \cdots & C_{1m} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nm} \end{bmatrix}$$

The *table row classification process* consists of identifying each of the elements of a table. The general idea is based on locating the header and data in the table. It is also relevant to identify the layout elements and metadata in a table. Figure 4 shows the table row classification process denoting in different colors some of the elements present in the table: red indicates the elements that represent the titles; yellow the row header; blue the data rows; and green the additional metadata.



Patent Applications by Residents
Data Source: worldbank.org
(showing top countries in each continent)

Country	Residents	Applications
North America		
United States	307,007,000	224,912
Canada	33,739,900	5,067
Mexico	112,033,369	822
	N.A. Total	230,801
Asia		
Japan	127,557,958	295,315
China	1,331,380,000	229,096
South Korea	48,747,000	127,316
	Asia Total	651,727
Note: data from 2009		

Patent Applications by Residents
Data Source: worldbank.org
(showing top countries in each continent)

Country	Residents	Applications
North America		
United States	307,007,000	224,912
Canada	33,739,900	5,067
Mexico	112,033,369	822
	N.A. Total	230,801
Asia		
Japan	127,557,958	295,315
China	1,331,380,000	229,096
South Korea	48,747,000	127,316
	Asia Total	651,727
Note: data from 2009		

Figure 4: Tabular extraction process

1.3 Contributions

The contributions of this dissertation are:

1. Implementation of an algorithm, based on Conditional Random Fields (CRFs), to classify the rows of a table as header rows, data rows or metadata rows.

2. Implementation of a rule-based algorithm and a graph-based algorithm to identifying table into spreadsheet documents.

3. Construction of two test datasets containing tables formatted in HTML and spreadsheet tables.

   - HTML tables downloaded from Web.

   - Spreadsheet tables downloaded from the Web Site of the National Oil Agency of Brazil.

1.4 Dissertation Structure

This dissertation is structured as follows. Chapter 1 provides an introduction and an overview of the main concepts related to this dissertation. Chapter 0 summarizes related work. Chapter 3 covers details of the implementations of the algorithms for table recognition and for table row classification. Chapter 4 describes experiments with the implementations. Finally, Chapter 5 presents the conclusions and proposes future work.

2 Related Work

In the literature, one finds research about methods and tools for tabular data extraction from spreadsheets, HTML table, tables embedded in PDF documents, etc. The vast majority of these methods and tools follows strategies based on heuristic rules and machine learning algorithms. The strategy for tabular data extraction and for table row classification also depends on the document format.

2.1 PDF Table Extraction

We can find multiple methods and tools for tables embedded in the PDF documents. Correa and Zander [10] analyzed a group of methods and tools focused on extracting tabular content from PDF, based on two main characteristics: ease of use and output results. Furthermore, the authors presented a categorization of the tools based on theoretical proposals, free of cost and commercial. The tools that were analyzed are ABBYY FineReader 12, Adobe Acrobat DC, Nuance Power PDF 2, Nuance Power PDF 2(v2.0), PDFTables, Tabula (v1.1.1). The authors concluded that Tabula was the best choice since it showed excellent output results and is open source.

2.2 Pdf2table

Pdf2table is a tool proposed in [11] that developed several heuristics, which together recognize and decompose tables in PDF files and store the extracted data in a structured data format (XML) for ease of use. These heuristics are divided into two groups: table recognition and table decomposition. Before applying the heuristics, it is necessary to do a preprocessing to return text chunks and their absolute coordinates in the PDF file in the same order as they were inserted into the original file. In the process of table recognition, information is gained from pre-processing to identify the tables in the document, and algorithms for text element fusion and for the classification of single and multi-line rows are applied. The second stage is related to the process table decomposition and applies algorithms for each multi-line block object that assigns text element for the column. The authors highlighted limitations of this method, mainly due to the complexity of the tasks and the heuristics used; for example, it cannot cover all possible table structures.

2.3 Identifying Table Boundaries

Other techniques for data tabular extraction from PDF documents are based on table boundary identification. The research presented in [12] describes a methodology that applies two machine learning algorithms, CRFs and Support Vector Machine (SVM). The idea proposed provides an effective method that locates the boundary of a table and identifies table lines. The authors introduced two important definitions – sparse and non-sparse line – derived from the same features as documents, that lead to efficient table boundary detection. In the process of detection, all labels that belong to a sparse line and do not correspond to the table boundary are removed. Other features were considered by the authors, such as orthographic features, lexical features, and the document layout to obtain better results.

2.4 PDF-TREX

PDF-TREX is an heuristic-based approach for table recognition and extraction from PDF documents [13]. The approach considers a PDF document as a Cartesian plane on which are placed content elements contained in 2-dimensional visualization areas. The system recognizes tables included in a report, aligns and groups, in a bottom-up way, content elements by exploiting only spatial relationships existing among them. The approach does not require: (i) scientific or domain knowledge; (ii) graphical metadata and ruling lines; (iii) predefined table layouts. The output obtained by PDF-TREX is an XML representation of the extracted cells that can be further processed for understanding table contents and improving information extraction from PDF documents.

2.5 Table Structure Recognition Based on Robust Block Segmentation

The table structure recognition based on Robust Block Segmentation was one of the pioneering works on table detection [14], which developed a table spotting and structure extraction system, called T-Recs. The system relies on word bounding boxes as input. These word boxes are clustered with a bottom-up approach into regions by building a "segmentation graph." These regions are then designated as candidate table regions if they satisfy a particular criterion. The critical limitation of the approach is that based only on word boxes, multi-column layouts cannot be handled very accurately, so, it works well for only single column pages.

2.6 Infogather

InfoGather [1] is a system designed for the semantic matching and annotation of numeric and time-varying attributes in Web tables. Furthermore, it proposes a new technique based on probabilistic graphical models to discover the semantic labels of columns and semantic matches between columns over a set of Web tables, collectively instead of individually. Their model elegantly combines several signals such as “local” extraction of labels from the column headers and values, semantic matches computed using traditional schema matching techniques and label propagation. Additionally, the authors developed efficient algorithms to solve the common discovery task and presented a new entity augmentation API suited for numeric and time-varying attributes. It allows users to specify the unit, scale and time information to unambiguously specify the argumentation task and offers a new query processing algorithm for the new entity augmentation operation. Their approach consists in finding the values of attributes of one or more entities and finding the relevant attributes of an entity type. The architecture is divided into two layers (Pre-processing and Query Time Processing) as showed in Figure 5.

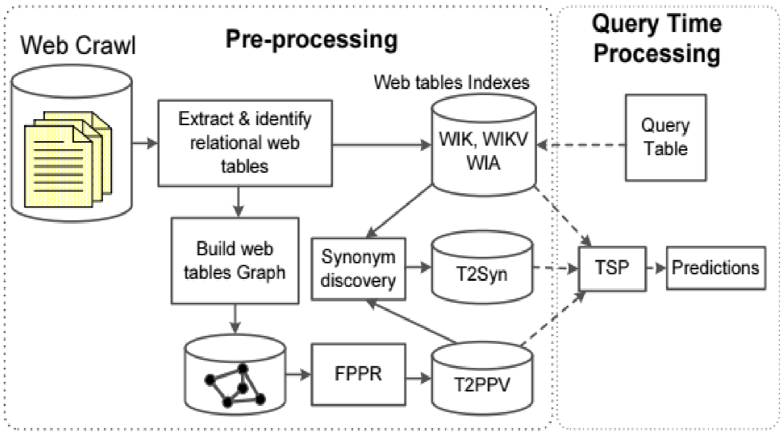


Figure 5: INFOGATHER System Architecture

:

2.7 Entity Discovery and Table Annotation

In the literature, one can also find research on tabular data extraction with the help of a knowledge base, ontology or information extracted from Web pages [2] . Most of this research aims at entity discovery and the detection of relationships between the entities and data row into tables.

2.7.1. TableMiner+

TableMiner+ is a semantic table interpretation method that annotates Web tables effectively and efficiently [15]. The method adopts an incremental, bootstrapping approach that starts by creating preliminary and partial annotations of a table using 'sample' data and then uses the outcome as 'seed' to guide the interpretation of the remaining contents. It has been implemented as open-source software (as part of the STI library). However, the system lacks an intuitive user interface, which makes it difficult to be used by an average person with limited technical knowledge.

2.7.2. TANGO

TANGO (Table Analysis for Generating Ontologies) is an approach based on conceptual modeling extraction techniques that: understand a table structure and conceptual content; discover the constraints that hold between concepts extracted from the table; match the recognized concepts with ones from a more general specification of related concepts; and merge the resulting structure with other similar knowledge representations. TANGO is thus a formalized method of processing the format and content of tables that may serve to incrementally build a relevant reusable conceptual ontology [16] . TANGO involves four main tasks: (1) Recognize and canonicalize table information; (2) Construct mini-ontologies from the canonicalized tables; (3) Discovery inter-ontology mappings; (4) Merge mini-ontologies into a growing application ontology.

2.7.3. WebTable

WebTables [2] is a system that develops new techniques for keyword search over a corpus of tables and shows that they can achieve substantially higher relevance than solutions based on a traditional search engine. It introduces a new object derived from the database corpus: the attribute correlation statistics database (AcsDB) that records corpus-wide statistics on occurrences of schema elements.

2.7.4. Annotating and Searching Web Tables Using Entities, Types, and Relationships

Another interesting work was proposed in [4] for annotating and searching Web Tables, using entities, types, and relationships through machine learning techniques to interpret table cells with entities that they likely mention, table columns with types from which entities are drawn for cells in the columns and relations. The authors proposed a new graphical model for making all these labeling decisions for each table simultaneously, rather than make separate local decisions for entities, types, and relations. Experiments used the YAGO catalog, DBPedia, tables from Wikipedia, and over 25 million HTML tables from a 500 million pages Web crawl uniformly show the superiority of their approach. They also evaluated the impact of better annotations on a prototype relational Web search tool. They demonstrated clear benefits of the annotations beyond indexing tables in a purely textual manner.

2.8 Recognition of HTML Table Structure

Another approach related to the recognition of HTML Table Structure can be found in [17], which proposed a system that can identify the boundary between attribute name rows (or columns) and its corresponding value data rows (or columns) in the table. The algorithm developed in this system extracts many linguistic features from each cell data of the table and uses them to identify table structures at first. This work also addresses the concept related to the similarity of cells based on the vector space model used in the table recognition process. Besides, it classifies tables according to the structure as tables for Web page layout, useful tables, tables for emphasis among others.

2.9 Table Header Detection and Classification

The table header detection and classification [18] suggested some techniques based on heuristics rules and used a learning classification algorithms for delineating kinds of tables existing into a document and detecting the structure and header types. The algorithms proposed automatically detect header row and header column applying a forward weighted average score strategy to calculate the similarity between consecutive table rows/columns, and then finds the first local minimum top-down/left-right to be the separation between header and data. They treated the header and data binary problem as a classification problem and applied three classifiers: support vector machine (SVM), logistic regression, and random forests. The research was based on an existing table extraction tool, which is part of the search engine system TableSeer [12], which automatically identifies tables in PDF digital documents, detects table boundaries and extracts the contents in the table cells. The main contributions of this work were: i) The investigation of document samples in a digital library to identify the frequency of different styles of headers and categorization; ii) The design of table header detection methods and an empirical comparison of their performance to demonstrate their efficacy.

2.10 Table Extraction with CRFS

The approach presented in [19] was based on machine learning techniques that cover two fundamental tasks of the table extraction process: localization of the table and identification of the row positions and types. These tasks employ a conditional probability Markov model for labeling the lines of a document. Each label indicates whether or not the line is part of a table and, if so, what role it plays in the table. Like all learning algorithm, it requires features set for good performance. In this case, it adopted several features based on the work proposed in [20].

2.11 Using Linked Data to Interpret Tables

Further research on the automatic interpretation and information extraction from tables that can be found in the literature uses common Linked Data knowledge bases. In [21], it was proposed an automatic framework for interpreting tables, in four steps: association of ontology classes with columns; linking cell values to instances of those classes; discover implicit relations between columns in the table; and generation annotation output. For each of these tasks, it was developed some algorithms. For example, the first algorithm is aimed at mapping each cell value to a ranked list of classes and selecting that which best characterizes the entire column. The second algorithm links table cell strings to entities from the Linked Open Data cloud. To identify relations between tables, it was developed another algorithm that generates a set of candidates relations from the relations that exist between the concepts associated with the string in each row of two columns. This process requires querying DBpedia through its public SPARQL endpoint.

2.12 Table recognition in Spreadsheets

Spreadsheets are used for a great variability of data management tasks from users with different expertise level. The abundance of usage led to an enormous wealth of structured data contained in spreadsheets. Table detection is the problem of determining the occurrence of a table model in the document. Table recognition is the problem of analyzing a detected table to determine its different elements according to the table model. A table into a spreadsheet document can be viewed as a collection functional component presented in a tree structure. The assembling of the table components by the aggregation of spreadsheet cells can provide a view of the table and later on it can be used to perform a non-visual navigation on the table. The objective of capturing the tabular data embedded in spreadsheets can be treated as a classification problem where the individual section of a table have to be identified, In the Section 1.2 were described several elements that form a table that allows the tables recognition process in spreadsheets documents. In [22] focuses on the task of table recognition for single-table and multi-table spreadsheets. The authors make use of Random Forest classifier to infer the layout role of individual cells. Also, they propose a graph representation where vertices correspond to the layout regions of sheet and edges encode the spatial interrelations between these regions.

3 Implementations of Algorithms for Table Recognition and for Table Row Classification

This chapter describes implementations of algorithms to recognize tables in spreadsheets and an algorithm, based on Conditional Random Fields, to classify table rows as header rows, data rows, and metadata rows. It also presents some heuristics based on the cell features that help table recognition in a document. Section 3.1 presents the implementation of a Rule-Based algorithm to detect tables in spreadsheet. Section 3.2 shows the implementation of a Graph-Based algorithm to detect table in spreadsheet. It describes the stages of the algorithm, as well as a graph-base representation. The algorithm removes edges at the left and right to form the tables that it found in a worksheet of the spreadsheet document. Section 3.3 presents the definition and implementation of a machine learning algorithm to identify table rows. Section 3.3.2 introduces the table row classes used in the classification process. Section 3.3.3 explains the main features of the solution proposed. Sections 3.3.4 3.3.4 show the system architecture. Finally, Section 3.3.6 describes each of the modules that compose the architecture.

3.1 A Rule-based Algorithm to Detect Tables in Spreadsheets

This section introduces a rule-based algorithm to detect tables in spreadsheets that uses cell attributes, such as *border*, *format* and *data type*. Each cell attribute in the spreadsheet has a specific value associated with that cell. In turn, the cell border has the attributes *direction*, *style*, and *color*. The border may surround the cell in 4 different directions: *top*, *bottom*, *left*, and *right*.

A cell format is the visual formatting applied to the data of the cell, such as *number format*, *font style name*, *font name*, *font size*, *font bold*, *font italic*, and *font color*. The detection of multiple tables in the same spreadsheet is performed by finding a separator between two tables (usually a set of empty rows), as explained in [23].

Given a table *T*, with *rn* rows and *cn* columns, the following layout features are computed:

- Average number of columns, computed as the average number of cells per row.

  $c = \frac{1}{rn}\sum_{i=1}^{rn} c_i$

  where c_i is the number of cells in row $i, i = 1, \dots rn$.

- Average number of rows, computed as the average number of cells per column.

  $r = \frac{1}{cn}\sum_{i=1}^{cn} r_i$

  where r_i is the number of cells in column $i, i = 1, \dots cn$.

Algorithm 1 identifies the number of tables into a document and captures the range of the rows that represent the tables.

```
Input: A spreadsheet document
Output: Number of tables into the document (ct)
1.  Compute the values of rn and cn
2.  U ← The threshold to empty rows
3.  H ← Set header cells
4.  D ← Set data cells
5.  T ← Set title cells
6.  E ← Set empty rows
7.  $R_i$ ← Numbers of cells tagged as header in row i
8.  X ← Numbers of Empty cells between a header row and title row
9.  foreach i = 0 to rn  //Number of rows in the  spreadsheet document
10.     foreach j = 0 to cn //Number of columns in the spreadsheet document
11.         if C(i-1 ,j) is Header and  C[i-1 ,j].type = String  then
12.             if  C[i,j].format = C[i,j-1]. format = C[i,j+1]. format and
                        C(i-1,j). font = C(i,j).font = C(i-1,j).font
13.                 if $R_i$ > 0 then
14.                     C[i,j] ∈ Header
15.                 if C[i-1 ,j] ≠ Empty and  C[i,j]. format ≠ C[i-1,j]. format
16.                     H ← H ∪ {c}    // Adds c to the set of header cells
17.         else if C(i ,j)  and C(i-1 ,j) and C(i+1,j) ∈ { type, format} then
18.             D ← D ∪ {c}    // Adds c to the set of data cells
19.         else if C[i ,j].type = String and E >= X
20.             if C(i ,j-1) and C(i ,j+1) ∈ E  and  j is first column
21.                 T ← T ∪ {c}    // Adds c to the set of title cells
22.         else
23.             E ← E ∪ {c}    // Adds c to the set of empty cells
24.     if len(E) = U // if the count of empty cells equals to threshold
25.         ct = ct + 1
26. return ct
```

Algorithm 1: Table Detection and Recognition

3.2 A Graph-based Algorithm to Detect Tables in Spreadsheets

This section presents a graph-based algorithm to detect tables in spreadsheets, called *Remove and Conquer* [22], that uses a comprehensive set of rules and heuristics based on a graph representation of a spreadsheet.

3.2.1. Region Detection

A spreadsheet file contains one or more sheets and each sheet consists of a collection of cells organized in rows and columns. The following definitions help the region detection process.

Definition 2: W is the set containing all the cells of a worksheet.

Region detection consists of scanning the spreadsheet from the first cell in the left top corner until the last non-empty cell in the right bottom corner to check cells with similar formatting and to detect separators, such as empty rows, different cells formatting or different kinds of borders, such as different cell value type. The regions of a spreadsheet are then detected based on these principles [22].

More precisely, a region is defined as follows.

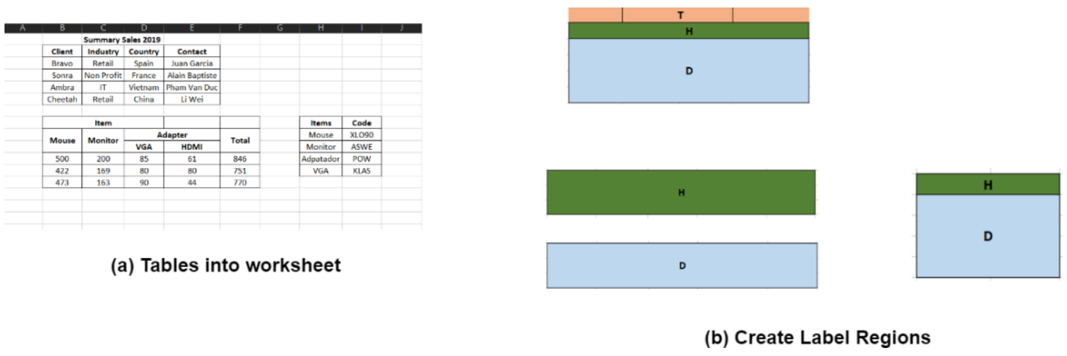
Definition 3: A *region* is a maximal collection $R \subseteq W$ of cells from a rectangular area of the worksheet.

Region detection also infers the layout role of non-empty cells in the worksheet, where each non-empty cell is assigned one of the following roles: *Header*(H), *Data*(D), *Title*(T), *Metadata* or *non-relational* (N). A cell role is defined as follows.

Definition 4: Let *label*: $W \rightarrow$ *Labels*, where *Labels* = {*Header*, *Data*, *Title*, *Metadata*}, be a function that maps cells to their assigned layout role.

For empty cells, label is undefined. We identify such cells using the function *empty*: $W \rightarrow \{0, 1\}$, which returns 1, if the cell is empty, and 0, otherwise.

The spreadsheet cells are grouped so that adjacent cells have the same layout role (*label*) or form larger structures. These groups are called *label regions* [24], as shown in Figure 6 .



A	B	C	D	E	F	G	H	I	J
		Summary Sales 2019							
	Client	Industry	Country	Contact					
	Bravo	Retail	Spain	Juan Garcia					
	Sonra	Non Profit	France	Alain Baptiste					
	Ambra	IT	Vietnam	Pham Van Duc					
	Cheetah	Retail	China	Li Wei					
	Item						Items	Code	
	Mouse	Monitor	Adapter		Total		Mouse	XLO90	
			VGA	HDMI			Monitor	ASWE	
	500	200	85	61	846		Adpatador	POW	
	422	169	80	80	751		VGA	KLAS	
	473	163	90	44	770				

Figure 6: Creation Label Regions Process

Formally, a label region is defined as follows:

Definition 5: A *label region* is a region *LR* of a spreadsheet such that, for any two cells c and c' in *LR*, $label(c) = label(c')$ and $empty(c) \neq 1$ and $empty(c') \neq 1$.

Figure 6(a) shows tables in a spreadsheet and Figure 6(b) indicates the regions corresponding to the table structures. The ***label region detection process*** clusters cells based on their label. It iterates over each row to create sequences of cells having the same label. These form the base ***LR***s. Subsequently, it merges ***LR***s from consecutive rows, if their labels, minimum column, and maximum column match.

3.2.2. Table Representation through Graphs

Regions allow constructing graphs that captures the spatial interrelations of label regions. Figure 7 shows the representation of tables as a graph.

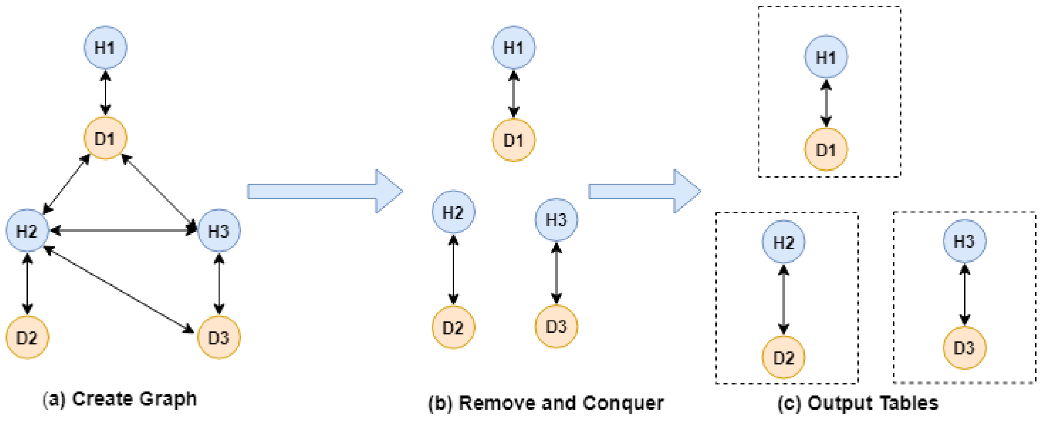


Figure 7: Table Representation through Graphs.

The graph construction process consists of identifying spatial relations, such as *top*, *bottom*, *left*, and *right*, based on locating the *nearest neighboring* regions for each direction and identifying all vertices whose maximum row is less than the minimum row of another vertex. For each direction, a distance function is defined where all the nearest vertices are identified:

$$ND_v = \left\{ n \in D_v \middle| ddist(v,n) = \min_{u \in Dv} ddist(u,v) \right\}$$

where D_v is the direction for the vertex v; a directed edge (v,n) is created for each $n \in ND_v$.

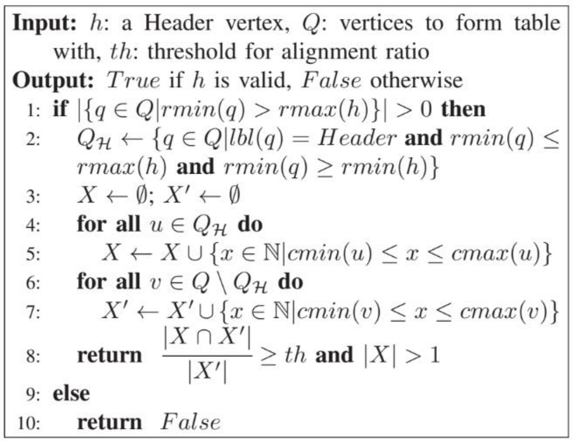
3.2.3. The Remove and Conquer Algorithm

Remove and Conquer (RAC) is a rule-based algorithm whose objective is to separate the edges that are farther to the left and to the right direction of the graph that was created from each worksheet in a spreadsheet. The algorithm detects the strong connected components of the graph to pair all groups and to detect valid tables.

```
Input: $G$ // Graph representation of a worksheet
Output: $P$ //Tables into document
1.  $P \leftarrow 0$
2.  El ← {e ∈E/dir(e) = Left and ldist(e) > 1}
3.  Er ← {e ∈E/dir(e) = Right and rdist(e) > 1}
4.  $E \leftarrow E \setminus (El \cup Er)$
5.  foreach $G^S \in$ getSCC(G) do
6.      $LQ \leftarrow Null$
7.      $S' \leftarrow \{v \in S' / lbl(v) = Header\}$
8.      if | $S'_H$ | > 0 then
9.          foreach $h \in S'_H$
10.             if $h \in LQ$ continue
11.                 $Q \leftarrow \{s \in S / rmin(s) \geq rmin(h)$ and $hasPath(s, h, ES)\}$
12.             if $isValid(h, Q, 0.5)$ then
13.                 $P \leftarrow P \cup \{LQ\}$
14.                 $S' \leftarrow S' \setminus Q$
15.             elif $LQ = Null$
16.                 if $|Q| = 1$ and $isAligned(h, LQ)$
17.                     $LQ \leftarrow LQ \cup \{h\}$
18.     $P \leftarrow P \cup \{LQ\}$
19. $U \leftarrow U \cup S$ // remaining unpaired
20. $P, U \leftarrow handleOverlapping(P, U)$
21. foreach $u \in U$ // find nearest table left or right
22.     $N, dist \leftarrow getNearestVertices(u, (El \cup Er))$
23.     $P' \leftarrow \{P \in P / 0 < |N \cap P|\}$
24.     if | $P'$| = 1 and $dist$ <= 3 then
25.         $P \leftarrow P \cup \{u\}$, where $P \in P'$
26. return $P, U$
```

Algorithm 2: Remove and Conquer Algorithm

The vertices are sorted in descending order of their maximum row, followed by the ascending order of their minimum row; thus, the tables are searched in order inverse, from the bottom to top. Each header *h* is individually processed to identify vertices with minimum row greater or equal to *h*. The algorithm that verifies the valid header is shown in Figure 8. All valid headers are stored in *Q*, which represents the vertices set, including *h*; this vertices set is called *potential tables*. The algorithm ensures that other vertices connected to *h* are not left isolated. Those vertices paired with a valid header are subtracted from the vertices set and ordered to create set $S^{'}$. The valid headers are appended to the set of valid headers, *LQ*. The vertices that represent potential tables, *Q*, are not appended directly to the tables set *P* because the algorithm needs to check that *h* is not connected to other vertices. The tables that cannot be formed are stored in *U*. Then, in the last step the algorithm, it attempts to pair the tables in *U* with the nearest table on their left or right.



```
Input: $h$: a Header vertex, $Q$: vertices to form table
    with, $th$: threshold for alignment ratio
Output: $True$ if $h$ is valid, $False$ otherwise
if $|\{q \in Q | rmin(q) > rmax(h)\}| > 0$ then
    $Q_{\mathcal{H}} \leftarrow \{q \in Q | lbl(q) = Header$ and $rmin(q) \leq$
    $rmax(h)$ and $rmin(q) \geq rmin(h)\}$
    $X \leftarrow \emptyset$; $X' \leftarrow \emptyset$
    for all $u \in Q_{\mathcal{H}}$ do
        $X \leftarrow X \cup \{x \in \mathbb{N} | cmin(u) \leq x \leq cmax(u)\}$
    for all $v \in Q \setminus Q_{\mathcal{H}}$ do
        $X' \leftarrow X' \cup \{x \in \mathbb{N} | cmin(v) \leq x \leq cmax(v)\}$
    return $\frac{|X \cap X'|}{|X'|} \geq th$ and $|X| > 1$
else
    return $False$
```

Figure 8: Header Validity Check.

3.3 A Machine Learning Algorithm for Table Row Classification

This section it describes the implementation of a machine learning algorithm that allows identifying the kinds of rows that compose a table. The section also presents the features set, based on the values of the cells, as well the classes that represent the table structure.

3.3.1. Conditional Random Fields

Conditional Random Fields, introduced by Lafferty et al. [25], offer several advantages over hidden Markov models and stochastic grammars for several tasks, including the ability to relax strong independence assumptions made in those models. Conditional Random Fields also avoid the fundamental limitation of maximum entropy Markov models (MEMMs) and other discriminative Markov models based on directed graph models, which can be biased towards states with few successor states.

More precisely, Conditional Random Fields (CRFs) are undirected graph models that can act as classifiers for sequence labeling tasks. They are frequently used for natural language processing, such as part-of-speech tagging. The CRF algorithm defines *X* as a random variable over data sequences to be labeled, and *Y* as a random variable over the corresponding label sequences. Figure 9 shows a structure of a linear Conditional Random Field.

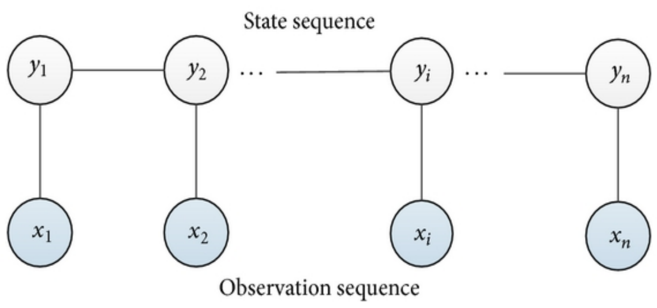


Figure 9: Structure of a linear chain Conditional Random Field

In a linear chain conditional random field, the label for a given frame depends jointly on the label of the previous frame, the label of the succeeding frame, and the observed data *x*. These dependencies are computed in terms of functions defined by pairs of labels and by label-observation pairs.

In the specific problem of classifying table rows, the input sequence ***x*** corresponds to a series of rows of a given table, while the label sequence ***y*** is the series of labels assigned to the observed rows. Each row in ***x*** is assigned exactly one label in ***y***.

Formally, Conditional Random Fields are defined as follows:

Definition 6: Let *G* = (*V*, *E*) be a graph and let $\boldsymbol{Y} = (\boldsymbol{Y}_v)_{v\in V}$ be a sequence of random variables indexed by the vertices of *G*. A *conditional random field* is a pair (***X***,***Y***) such that, when conditioned on ***X***, the random variables $\boldsymbol{Y}_v$ obey the Markov property with respect to the graph

$$P(\boldsymbol{Y}_v|\boldsymbol{X},\boldsymbol{Y}_w,w \neq v) = P(\boldsymbol{Y}_v \vee \boldsymbol{X},\boldsymbol{Y}_w,w \sim v)$$

where *w* ∼ *v* means that *w* and *v* are neighbors in *G*.

The probability $P(\boldsymbol{X}|\boldsymbol{Y})$ of a state sequence ***Y***, given an observation sequence ***X***, is:

$$P(\boldsymbol{X}|\boldsymbol{Y}) = \frac{1}{Z(\boldsymbol{X})}\, exp\left(\sum_j \lambda_j\, f_j(\boldsymbol{Y}_{i-1},\boldsymbol{Y}_i,\boldsymbol{X},i) + \sum_k \mu_k g_k(\boldsymbol{Y}_i,\boldsymbol{X},i)\right)$$

where $f_j(\boldsymbol{Y}_{i-1},\boldsymbol{Y}_i,\boldsymbol{X},i)$ is a *transition feature function* of the observation sequence and of the labels at positions *i* and *i-1* in the label sequence; $g_k(\boldsymbol{Y}_i,\boldsymbol{X},i)$ is a *state feature function* of the label at position *i* and the observation sequence; and λ_j and μ_k are parameters to be estimated from training data.

In the data table scenario, ***X*** represents the list of rows in the table, and ***Y*** represents the corresponding row classes. Before presenting in detail the method, we explain some important elements related to the row identification process. Each relational data table has a schema, which, in the context of data tables, consists of attribute names, values, and types, where attribute names are column titles, attribute types are the types of values in the column, and attribute values correspond to data values in the column's cells. Column names are stored in a special row or rows, usually near the head of the table, called *header rows*, while the data is stored in rows referred to as *data rows*. The data table may also contain descriptions of data, called *metadata*. These criteria help identify each type of row according to the properties of each cell in a data table. Then, we focus on the problem of assigning one label to each row, where each row consists of constituent cells, which can exhibit different sets of attributes. The feature selection process involves the extraction of a collection of attributes for individual cells and combining attributes from all cells in the row, in order to construct a set of row features. Consider the ideas addressed above and an example of a simple table with header and data as showed in

Figure 10, where the variables ***X*** and ***Y*** are defined according to the CRF model for the data table scenario. Then, ***X*** represents a vector with the rows of the table and ***Y*** represents another vector with the tags of each row ***x*** of the table. We observe that, for each row ***x*** in the sequence, we defined a label ***y***. The next section shows the features in the row classification process computed through Boolean functions (*True* or *False*) that determine the value and states of each row into the data table, respectively.

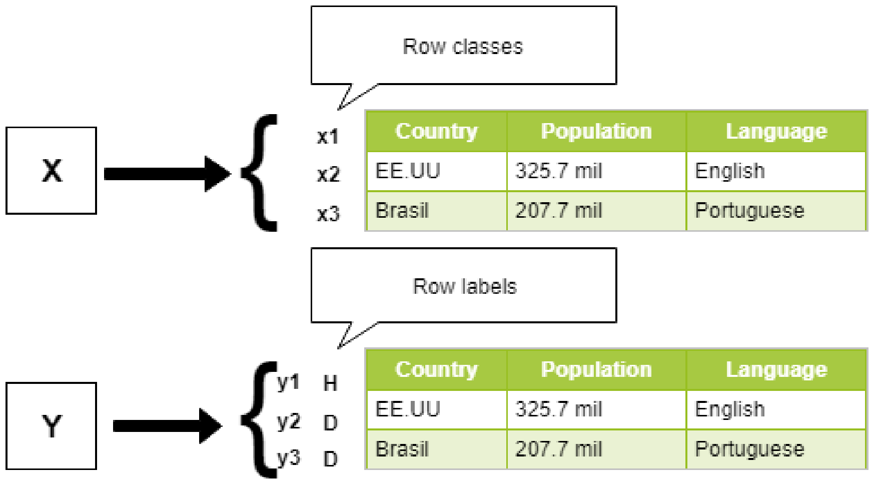


Figure 10: Example of a labelled table.

3.3.2. Row Classes

According to the structure of tables and the definitions presented in Section 3.3.1, the elements that compose a table can be divided into a small set of row classes, as already mentioned. Table 1 shows the rows classes considered.

Table 1: Row classes.

Label	Description
H	Header rows contain cell values that describe the values contained in the subsequent data rows of that column.
D	Data rows contain data records (or tuples in relational parlance).
T	Title rows represents the titles of tables generally contains string values or alphanumeric characters.
N	Non-relational metadata, such as a note, clarification, or any text that does not contribute to the data or structure of the data table.

3.3.3. Feature Set

In any machine learning algorithm, a *feature* is an individual measurable property or characteristic of the phenomenon being observed [26]. The fact that each row consists of a set of cells, where each cell presents different characteristics, induces other features that help identifying the class of a row as either a *header row* or a *data row*. The feature set is partitioned into three categories considering aspects related to the layout, styles, and values which we call layout attributes.

Layout attributes are the various properties of the cells that are commonly found in header rows, which usually contain merged table cells with centered text. In HTML data rows, other elements may be present, such as the <thead> and <th> tags on the alignment of text within cells.

Style attributes are various properties derived from stylesheets, such as font type, font color, font weight or underlined text, which are more common in header rows or title rows than in data rows.

Value attributes are t the various properties of that represent cells where the stored information is exclusively linked to the data rows.

Header rows often contain relatively short textual values, rather than numbers or dates. While data rows may contain many empty cells, header rows typically contain few to zero blank cells across the full width of the table.

Aggregate rows often indicate that the row contains a total or subtotal, so we include an attribute to match the case insensitive string "total". All of these features are extracted from data tables and used as cell attributes.

Taking into account these features, as well as the CRF model, each table row is classified according to the three categories defined (header, data, no-relational metadata). The transition feature function $f_j(\boldsymbol{Y}_{i-1}, \boldsymbol{Y}_i, \boldsymbol{X}, i)$ corresponds to the transition from a header row to a data row and the state feature function $g_k(\boldsymbol{Y}_i, \boldsymbol{X}, i)$ computes the value of each cell in a given row. For example, assume that:

- $\boldsymbol{x}$ is a row into the data table.
- j is a position-row in the table (each feature is associated with a position); more than one feature associated with the same position.
- y_{j-1}, y_j are the tags (classes) assigned to rows *j* and *j*-1 of $\boldsymbol{x}$

Then, the feature function and the state function are as follows:

$$f_1(\boldsymbol{Y}_{i-1}, \boldsymbol{Y}_i, \boldsymbol{X}, i) = \begin{cases} 1 \; if \; x_j \in header \; y_j = H \\ 0 \qquad\qquad otherwise \end{cases}$$

$$f_2(\boldsymbol{Y}_{i-1}, \boldsymbol{Y}_i, \boldsymbol{X}, i) = \begin{cases} 1 \; if \; x_j \in data \; y_j = D \\ 0 \qquad\qquad otherwise \end{cases}$$

$$g_1(\boldsymbol{Y}_i, \boldsymbol{X}, i) = \begin{cases} 1 \; if \; (x_j \; is \; a \; cell \; \in x) and \; (x_j \in \; row \; features) \; and \; y_j = H \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$

$$g_2(\boldsymbol{Y}_i, \boldsymbol{X}, i) = \begin{cases} 1 \; if \; (x_j \; is \; a \; cell \; \in x) and \; (x_j \in \; row \; features) \; and \; y_j = D \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$

The full list of individual cell attributes is given in

Table 2. The features are divided according to the kind of attributes that they represent.

Spatial Features describe the neighborhood of the cell such as the location and the features of its neighbors. Only the neighbors that share a border (edge) with the cell are inspected, i.e., the *top*, *down*, *left*, and *right*. When the neighboring cells are blank or outside of the bounds of the sheet, we set the value of the corresponding features to “Not Applicable” (n/a).

Based on the elements exposes above, a direction function is defined as follow.

Definition 4: *dir: E → {T op, Bottom, Left, Right}* is a function that maps edges to directions. For an edge (v, u) ∈ E the result of this function communicates the direction that u is nearest neighbor of v.

Table 2: Cell Attributes

Layout	Style	Value	Spatial
IsMerged	**IsBold**	**IsEmpty**	**Row_num**
Alignment	**IsItalic**	**IsText**	**Column_num**
	IsUnderlined	**IsNumeric**	**Number_of_ neighbors**
	IsColored	**IsDate**	**Matches_Style**
	Font	**IsAlphaNumeric**	**Matches_Type**
	Format	**Is_Capitalized**	
	Border	**Words_Like_Total**	

3.3.4. Similarity between rows

With the goal of finding attributes related to row similarity and generalize the training data, a unique feature is assigned to each unique combination (***c*, *r***), where ***c*** is the number of cells exhibiting an attribute and ***r*** is the number of cells in the row. Then, two rows $\boldsymbol{R_x}$ and $\boldsymbol{R_y}$ are considered similar with respect to a certain cell attribute *α* if the logarithm of their widths are equal and the logarithm of the number of cells exhibiting or lacking attribute *α* [27]. This aspect is very important because a vast majority of tables could exhibit different column distribution in each row. This approach is known as *feature binnig* and can be defined as follow.

Formally, for a row R_i of length *r* in which *c* cells exhibit a specific cell attribute α, the feature "$R\alpha = (a, b)$" to $R_i\ (a, b)$ is assigned, where a and b are the bin and computed as follows.

$$a = \begin{cases} 0, & if\ c = 0 \\ \lfloor \mathbf{log}_2(c) + 1 \rfloor, & if\ 0 < c \leq r/2 \\ \lfloor \mathbf{log}_2(r - c) + 1 \rfloor^{-}, & if\ r/2 < c < r \\ 0,^{-} & if\ c = r \end{cases}$$

$$b = \lfloor \mathbf{log_2}(r) \rfloor$$

The superscript minus sign in some values of *a* represents that *a* is computed based on *r* - *c* rather than *c*.

The aim of the bins is given by:

1. Differentiate Between Table Widths.
2. Aggregate Wide Tables.
3. Highlight Uniform Rows.

3.3.5. Architecture

Figure 11 depicts the architecture of the classification process proposed in this dissertation, whose main components are:

1) A module that labels HTML tables.
2) A module that labels spreadsheets tables.
3) A component that implements the CRFs algorithm, which receives a set of labeled rows, either from the HTML module or the spreadsheets module.
4) A module that enables the annotation process for HTML tables.
5) A module that allow table recognition into a spreadsheet document.

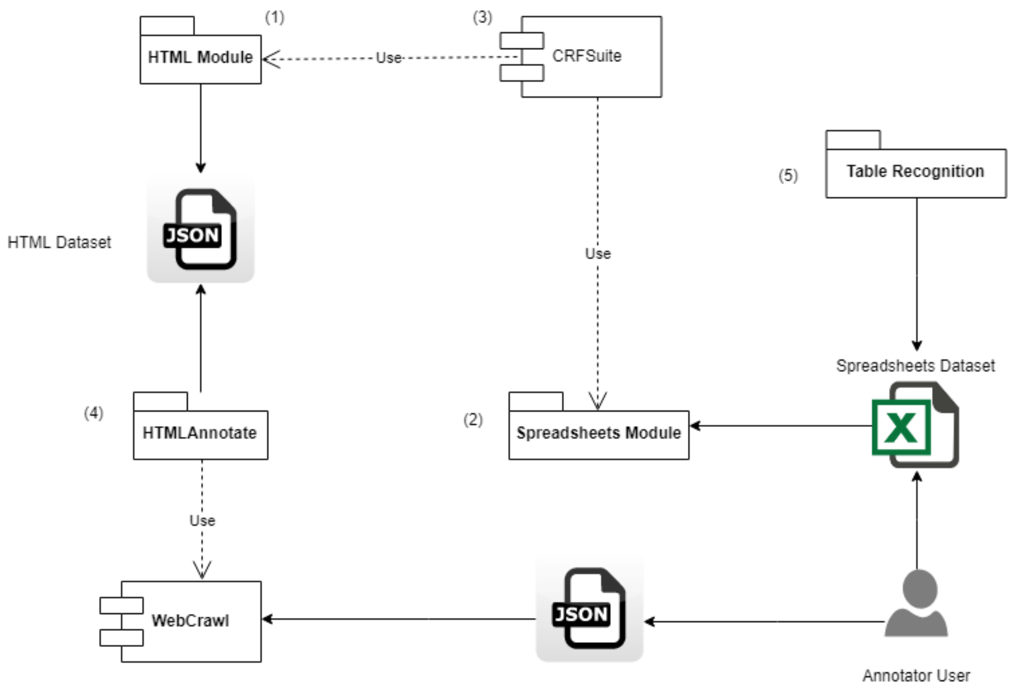


Figure 11: System Architecture

3.3.6. Description of the Modules

As introduced in Section 3.3.4, the table row classification process consists of four modules. This section details each of these modules, covering aspects related to the structure, composition of the classes, and functionality.

3.3.6.1. HTMLAnnotate Module

The HTMLAnnotate module allows the user to annotate HTML tables. This module uses a crawling tool to download tables from the Web. For each downloaded table, a JSON file is created with the corresponding annotations, and an attribute, called XPATH, which indicates where the table can be found in the Web page. We will explain this annotation process in more detail in Section 4.1.

3.3.6.2. HTML module

The HTML module implements the tagging process for the HTML tables that receives as input an entry in a JSON file with the annotations in tables and returns a sequence of correctly labeled rows. The module is composed of two main classes and a common package classes to both HTML and Spreadsheets and CRSuite module, this module creates the model for HTML tables. Figure 12 shows the class diagram, detailed as follows:

- *Header* is the class that represents the header elements in both HTML and Spreadsheets tables.

- *Data* is the class that represents the data in both HTML and Spreadsheets tables.

- *Row* is a generic class that represents rows in both HTML and Spreadsheets tables.

- *Cell* is the class that represents the row cells in a table. The cell characteristics allow the classifier to identify a row as either a data row or a header row.

- *TagHTMLTable* is the main class of the module that labels each row in an HTML table, using the cell features involved. It returns a correctly labeled table list, which is passed to the classifier to be trained and tested.

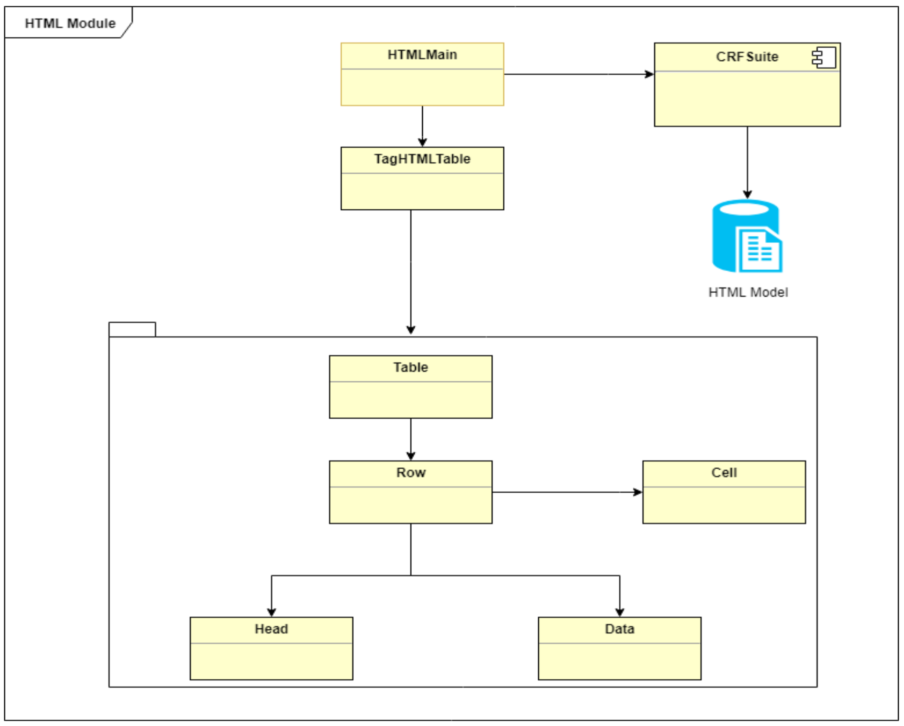


Figure 12: Class Diagram of the HTML module

The classes diagram of the HTML module represents the relationship among all the classes that intervene in the work with the HTML tables. The process starts with the input of a set of JSON files that were tagged through the HTML table annotation module, as previously explained in this section. Each JSON file represents an HTML table with its types or classes of corresponding rows. The TagHTMLTable class labels each row of a table as a data row or a header row, according to the cell characteristics defined in the cell class. After each row is correctly labeled, the TagHTMLTable class returns the entire set of tagged tables, which are passed to the main class where the model is created for the training and test processes.

3.3.6.3. Spreadsheets module

This module is structurally identical to the module explained above, but it treats tables in spreadsheets, which requires another type of processing. Among the aspects to consider, including the labeling process, since such tables already have an annotation suggestion. In fact, each spreadsheet document was reviewed, performing corrections in the annotations. Figure 13 shows the class diagram of this module; the module uses the common package classes that represent the tables and their structures as was explained above.

TagTableSpreadSheet is the main class of the module. It labels each of the rows in a spreadsheet table, starting with the cell features involved in each row. It returns a table list, correctly labeled, which is passed to the classifier to be trained and tested.

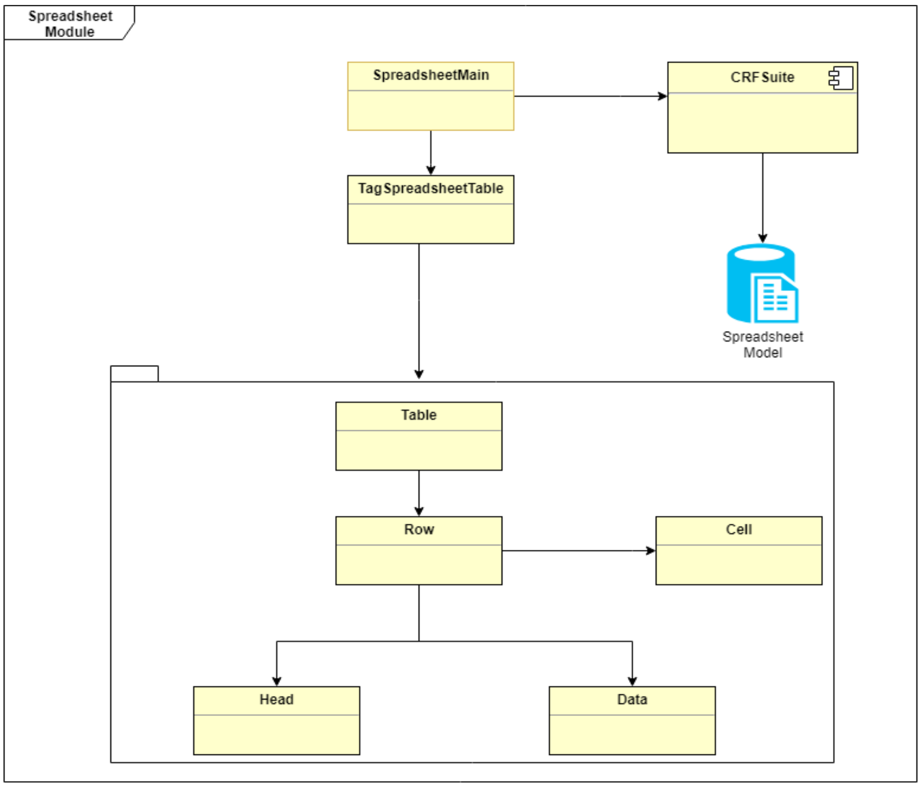


Figure 13: Class Diagram of the Spreadsheet module

The classes diagram of the Spreadsheets module represents the relationship among all the classes that intervene in the work with spreadsheet tables. The process starts in the TagTableSpreadSheet class, which gets the JSON file that represents the spreadsheets tables, manually annotated, as was explained above. Each position into the JSON file represents a spreadsheets table. Then, the TagTableSpreadSheet class labels each of the rows as a data row or a header row according to the cell features, defined in the cell class. After each row has been correctly labeled, the TagTableSpreadSheet class gets a set of labeled rows, which are passed to the main class, where the model for the trained and test process is created.

3.3.6.4. CRFsuite module

CRFsuite is a module that implements the CRF algorithm, using a library developed in python called sklear-crfsuite[2], with the following features:

[2] https://sklearn-crfsuite.readthedocs.io/en/latest/

- **Fast training and tagging.** The primary goal of this software is to train and use CRF models as fast as possible.
- **A simple data format for training and tagging**. The data format is similar to those used in other machine learning tools; each line consists of a label and attributes (features) of an item, continuous lines represent a sequence of items (an empty line denotes an end of item sequence). This means that users can design an arbitrary number of features for each item.
- **State-of-the-art training methods**. The CRFsuite implements:
    - Limited-memory BFGS (L-BFGS).
    - Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method.
    - Stochastic Gradient Descent (SGD)
    - Averaged Perceptron
    - Passive Aggressive

In the implementation of our table row classifier, we adopted L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno). This approximates the Broyden–Fletcher–Goldfarb–Shannon (BFGS) algorithm using a limited amount of computer memory and often gets a better solution than the other methods mentioned above, with less iterations. Another aspect that was considered in the settings of the algorithm was the regularizations parameters, since they reduce the error by fitting a function appropriately to the given training set, while avoiding overfitting. In this case, L1 and L2 were set to 0.1 and 0.01, respectively.

3.3.6.5. The Table Recognition Module

The table recognition module implements the table identification process in a spreadsheet document. Figure 14 shows the complete process. This process starts with the creation of the regions in the document, which are loaded and the table graph created, as was explained in Section 3.2. Then, the remove and conquer algorithm (RAC) is applied, as explained in the previous sections, obtaining the set of implicit tables within a document.

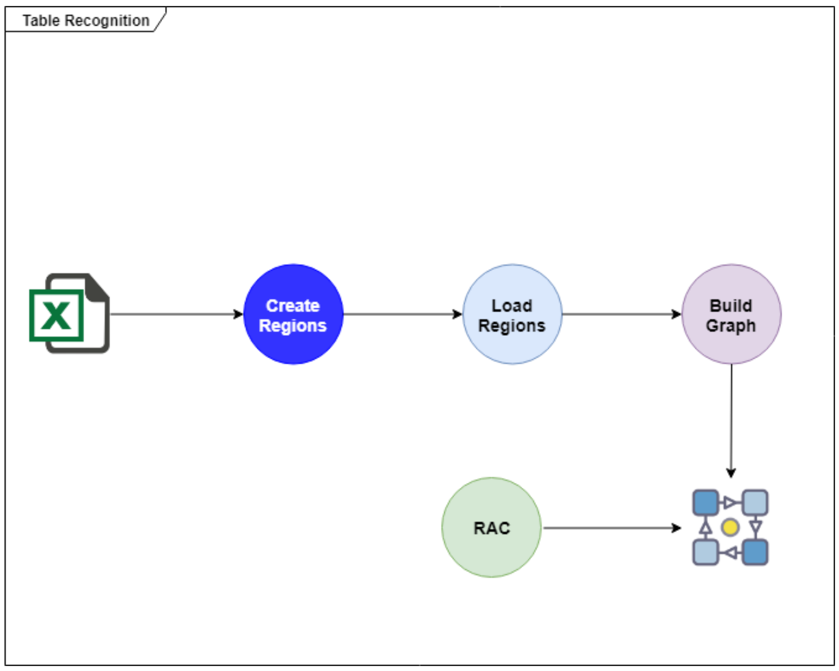


Figure 14: Table Recognition Module

4 Experiments and Results

This chapter presents the experiments performed to test the accuracy of the implementation of the table row classifier as well as the experiments with table recognition in spreadsheet documents. Section 4.1 describes the required preprocessing. Section 4.2 depicts the main characteristics of the datasets used. Section 4.3 describes the experiments with the model over the datasets. Section 4.3.2 and Section 4.3.3 show the results obtained with the model for the HTML and the spreadsheets tables, respectively. Finally, Section 4.3.4 presents the confusion matrix for the model for both HTML tables and Spreadsheets tables.

4.1 Preprocessing

The preprocessing task focused on two table scenarios, HTML tables and spreadsheets tables, with the goal of removing irrelevant content or content that will not provide information for our table row classifier. Other aspects that were considered were the structure of the tables and the information present in both kinds of tables. This section details the preprocessing of each kind of table.

4.1.1. Preprocessing HTML tables

Before going into the details of the preprocessing of HTML tables, we describe the characteristics of the HTML tables dataset. First, we remember that this dataset was downloaded from the Web and does not belong to any specific domain. Hence, the tables can vary; for example, we find tables describing films, countries, etc. The disadvantage is that the tables do not have the same structure. Another aspect to keep in mind is that ,quite often, the tables are no longer available on the Web. Figure 15 shows the preprocessing flow.

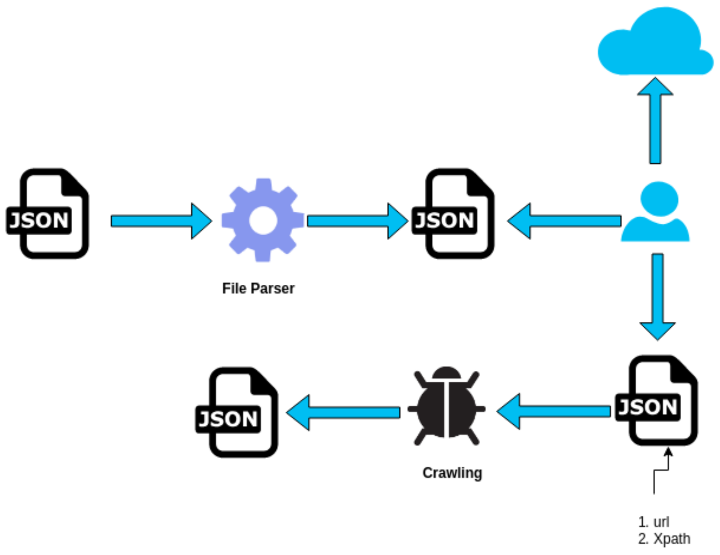
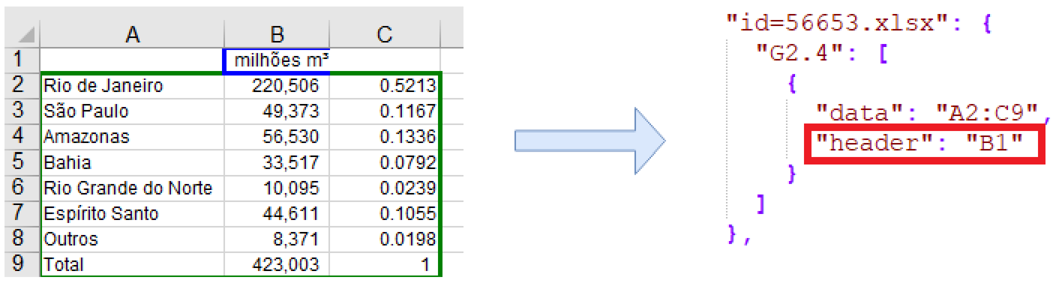


Figure 15: Preprocessing Flow on HTML Tables.

As shown in the figure above, the first JSON file represents each of the tables downloaded from the Web with a predefined annotation that does not cover our proposed solution and that present the problems mentioned above. Then, for this stage, a tool was implemented to parse each file and to get an URL list of all tables being manually analyzed. In this stage, we discarded the tables that were not available and that presented undefined structure (tables that do not contain data rows, heads or tables created only to design purpose). Here, we build a JSON file with an attribute indicating the URL of the page where the table can be found and a XPath attribute that represents the location of the table in the Web page, as already mentioned. In the last stage of the process, we used a crawling tool to crawl each table and we annotated the JSON files with the features related to the tables.

4.1.2. Preprocessing Spreadsheets Tables

The preprocessing stage for spreadsheets tables was much easier than that for HTML tables, but much heavier, since each file had to be reviewed, discarding those that did not present a suitable structure. This preliminary dataset had a predefined annotation, but with many errors related to the identification of ranges of data rows and head rows. Another aspect related to these tables had to do with the structure and the data that the vast majority of these tables presented. A total of approximately 1,000 tables were analyzed, and we kept those with a well-defined structure. Figure 16 shows an example of a table with a header row that is not defined and its representation in the annotated JSON file. This example shows that the table was wrongly interpreted with a header defined from cell B1. In this case, we removed the table from our dataset.



	A	B	C
1		milhões m³	
2	Rio de Janeiro	220,506	0.5213
3	São Paulo	49,373	0.1167
4	Amazonas	56,530	0.1336
5	Bahia	33,517	0.0792
6	Rio Grande do Norte	10,095	0.0239
7	Espírito Santo	44,611	0.1055
8	Outros	8,371	0.0198
9	Total	423,003	1

```
"id=56653.xlsx": {
  "G2.4": [
    {
      "data": "A2:C9",
      "header": "B1"
    }
  ]
},
```

Figure 16: Example of Table reviewed into the dataset

4.2 Main Characteristics of the Datasets used for Testing

After the pre-processing stage, we created a dataset that contained tabular data represented in HTML and as well as in spreadsheets. The HTML tables, as already explained in previous sections, were obtained from other published research work and do not belong to any specific domain. The spreadsheet tables were downloaded from the Web Site of the National Oil Agency of Brazil. Both datasets were preprocessed in agreement with the classification model used. Each row of each table was annotated with the label corresponding to its class: “H” for header, “D” for data, etc. The annotation process for HTML tables was done through the module presented in the Section 3.3.6.1, resulting in 105 annotated tables. The annotation process for spreadsheets tables was manual and resulted in a total of 252 annotated tables. Statistics about the annotations are listed in Table 3.

Table 3: Annotated tables.

	HTML	**Spreadsheets**
Annotated tables	105	252
Annotated rows	13,025	227,638
Header rows	105 (<1%)	252 (<1%)
Data rows	12,920 (99%)	227,254 (98%)
Other row classes	0 (0%)	132 (<1%)

Table 3 indicates that a critical aspect of both HTML and spreadsheets tables is that the percentage of header rows is very low, due to the fact that the tables obtained were simple tables with simple schemas (tables with a single header row followed by one or more data rows).

4.3 Table Classification Experiments

This section presents the experiments to evaluate the table classification solution proposed. In a first stage, the algorithm was trained with 80% of the data and tested with 20% of the data, randomly selected from both the spreadsheets and the HTML datasets. The algorithm used L-BFGS as the optimization method and the regularizations parameters L1 and L2 set to 0.1 and 0.01, as defined in Section 3.3.6.4. The experiments with HTML and spreadsheet tables were performed separately to expose the differences between the two table formats.

4.3.1. Performance Metrics

The performance metrics adopted were *precision*, *recall*, *f1-score*, and *support*, defined as follows:

- *Precision* is the ratio TP / (TP + FP), where TP is the number of true positives and FP the number of false positives.
- *Recall* is the ratio TP / (TP + FN), where TP is the number of true positives and FN the number of false negatives.
- The *F-beta score* weights recall more than precision by a factor of beta; beta = 1.0 means that recall and precision are equally important.
- *Support* is the number of occurrences of each class in true

4.3.2. Results

Table 4 shows the results obtained. We observe that the precision value for spreadsheet tables was higher than that for HTML tables due to two main factors: (1) the features of the spreadsheet tables have a better definition; (2) we guarantee a correct definition for data rows. The recall was similar for both kinds of tables, as well as the F1-score. An important point in this analysis relates to the number of rows classified as non-relational in the spreadsheet dataset, due to the fact that we annotated spreadsheet tables manually, as opposed to the HTML tables, where some rows could have been identified as data rows or header rows, being in fact non-relational rows.

Table 4: Results for HTML tables.

Row Class	Precision	Recall	F1-Score	Support
HTML				
D	0.966	0.982	0.970	2,496
H	0.955	0.992	0.970	17
N	0.980	0.980	0.970	92
Spreadsheets				
D	0.997	0.985	0.994	39,080
H	0.969	0.993	0.983	49
N	0.985	0.965	0.974	5

Note: row labels are as in Table 1:

D	Data rows
H	Header rows
N	Non-relational metadata (a note, clarification, etc.)

4.3.3. Cross Validation

Validation[3] is the process of deciding whether the numerical results quantifying hypothesized relationships between variables are acceptable as descriptions of the data. In most cases, there is not enough data to train a model and, therefore, removing a part of it for validation poses a problem of under-fitting. By reducing the training data, we risk losing important patterns/trends in a dataset, which in turn increases the error induced by bias. So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. Therefore, in a second experiment, a cross-validate technique was applied to improve the precision of the method. Figure 17 illustrates a 5-fold cross validation.

[3] https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

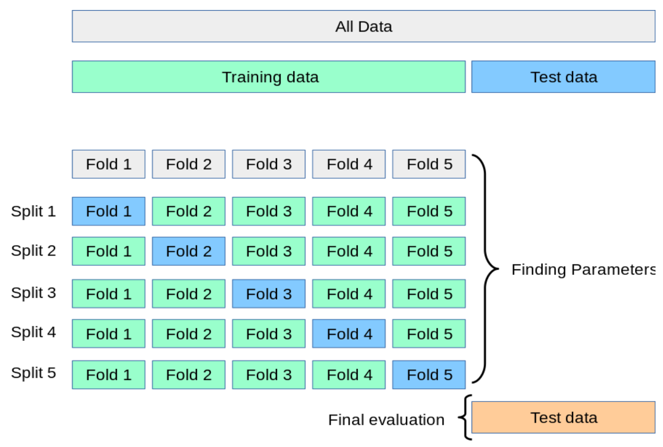


Figure 17: 5-fold Cross-Validation[4]

The performance measure reported by a k-fold cross-validation is then the average of the values computed. This approach can be computationally expensive, but it does not waste too much data.

Another aspect to be considered in the validation process is the fact that there is a large imbalance in the number of objects in each class. In fact, the number of data rows is much higher than the number of header rows. Then, we applied a k-fold strategy known as *stratified k-fold*, which is a slight variation in the k-fold cross-validation technique, such that each fold contains approximately the same percentage of samples of each target class as the complete set.

Table 5 shows the results obtained by applying the stratified k-fold strategy to the dataset containing HTML tables. We observe the best results were achieved for k=2 and k=3 and that the average accuracy was 0.958.

Table 5: Accuracy of cross-validation for HTML Tables

Stage	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
Testing	0.92	0.98	0.98	0.94	0.97

[4] https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Table 6 shows the results obtained by applying the stratified k-fold strategy to the dataset containing spreadsheets tables. We observe that the results obtained for k=1,…,5 are similar and that the average accuracy was 0.997.

Table 6: Accuracy of cross-validate method for spreadsheets tables.

Stage	k=1	k=2	k=3	k=4	k=5
Testing	0.997	0.998	0.996	0.998	0.996

4.3.4. Confusion matrix

As in any classification problem, there are aspects that may be improved. In our experiments, we have to examine the rows in each class that were confused with rows in another class. We then used a confusion matrix, as shown in Table 7 and Table 8. Each cell of the matrix shows the percentage of all classified rows that were actually of the class with the label shown in the first column, but which the classifier assigned the row label shown in the first row. The shaded cells along the diagonal show correct row classifications, while the surrounding cells show incorrect classifications.

Table 7: Confusion matrix for spreadsheets tables

	D	H	N
D	97.5%	0.0%	2.5%
H	1.6%	96.0%	3.3%
N	0.0%	5.5%	94.4%

Table 8: Confusion matrix for HTML tables

	D	H	N
D	91.6%	0.4%	7.9%
H	0%	93.3%	6,6%
N	3.3%	2.4%	94.2%

Ideally, our classifier would result in zeroes for the values off the diagonal. However, our model indeed misclassified rows. In the case of spreadsheets tables, we observed that, for both data rows and header rows, erroneous results were obtained with respect to the non-relational rows, that is, a considerable number of data rows and header rows were identified as non-relational rows. In the HTML tables, the erroneous results for non-relational rows was larger than for spreadsheet tables, being 7.9% and 6.6% for data rows and header rows, respectively. This deserves some explanation: (1) the difference between the average number of rows of the HTML tables and the average number of rows of spreadsheets tables; (2) in our classification process, a given row is classified as "non-relational metadata" when the row cannot be identified as data or header; (3) the spreadsheets tables have a better definition in term of features, for example, tables depend on properties encapsulated into CSS files.

4.4 Table Recognition Experiments

This section briefly describes the experiments with the table recognition process in a spreadsheet document, using Algorithm 1 and Algorithm 2 presented in Section 3.1 and Section 3.2, respectively. The dataset that contents spreadsheets table downloaded from the Web site of the National Oil Agency of Brazil was used to test both algorithms.

4.4.1. Experiments with the Rule-based Table Detection Algorithm

The rule-based table detection algorithm was applied to worksheets in a sample set that contained tables with different layouts and embedded charts. We then deleted the embedded charts since they do not represent relevant information to the table detection process.

Table 9 summarizes the results obtained by the rule-based algorithm, which analyzed a total of 1,000 spreadsheet documents, detected 1,481 tables and misclassified 141.

Table 9: Results for the table detection rule-based algorithm

Spreadsheet document	1,000
Tables	1,481
Tables misclassified	141
Single Table	700
Multi Table	158

The algorithm failed for multi-tables with internal separators that are less than the thresholds defined (i.e., the number of empty rows that separate two tables). In that case, the algorithm would consider the two tables as a single table. The algorithm would not recognize tables correctly when the table cells do not have attributes or separators (e.g., a table with no borders, no font formatting, no background colors, and no empty rows that separate headers and the table title). Also, the algorithm did not discover tables where the number of empty cells to the right and left is extremely large.

4.4.2. Experiments with the Remove and Conquer Table Detection Algorithm

The Remove and Conquer table detection algorithm were applied to the same dataset, but with the caveat that the dataset was manually annotated, knowing the cell layout roles and the location of tables. This algorithm detected tables that could not be recognized by the rule-based algorithm, which did not consider elements such as empty cells and columns with width less than 2 characters. The Remove and Conquer algorithm maximized the match between a proposed table *P* and a true table *T*, which is equivalent to maximizing the number of cells that they have in common and minimizing the number of cells by which they differ.

Table 10 shows the results for the Remove and Conquer algorithm (RAC). When compared with the rule-based algorithm, observe that the number of tables misclassified decreased and the number of multi-tables detected increased.

Table 10: Table recognized through of RAC

Spreadsheet document	1,000
Tables	1,481
Tables misclassified	100
Single Table	650
Multi Table	230

4.4.3. Results of the algorithms and environment description

This section describes the environment used in the execution of the table recognition algorithms starting with the hardware and ensuing detailing the software that aided the development and execution of the algorithms.

4.4.3.1. Hardware and Software

The characteristics main of the environment execution are: Portable Computer (PC) Lenovo 80YH model with 8 Gigabytes of Random Memory. Processor Intel(R) Core i7-7500 with 2.70 GHz. Board Graphic Intel(R) 620 with 128 of Memory. Operating System Windows 10 Home of 64 Bits.

4.4.3.2. Execution Time

Table 11 shows the execution times for each of the Tables Recognition Algorithms for the spreadsheet documents, as well as the Machine Learning Algorithm. The time is expressed in seconds. In the case of the Conditional Random Fields, the time shown is the sum of training and testing. The Rule-based Algorithm took less execution time, considering that it is a linear algorithm, whereas the Remove and Conquer is graph-based.

Table 11: Total Execution Time of the Tables Recognition Algorithms

Algorithm	Total Execution Time
Remove and Conquer Algorithm	114.28
Rule-based Algorithm	69.19
Conditional Random Fields	376.57

5 Conclusions

In this work, we described the implementation of an algorithm, based on Conditional Random Fields (CRFs), to classify the rows of a table as header rows, data rows or non-relational metadata rows. The implementation was complemented by two algorithms for table recognition in spreadsheet documents, respectively based on rules and on region detection. We detailed the system architecture proposed and described its components as well as the workflow of each module.

We performed experiments to test the performance of the implementations using: (1) HTML tables, obtained through of a crawl tool, and stored as manually annotated JSON files; (2) spreadsheet tables, downloaded from the Brazilian National Petroleum Agency. The experiments showed that the implementations obtained excellent results for both HTML and spreadsheets tables. We also applied a k-fold cross-validation and obtained results similar to experiments reported in [27].

To summarize, the contributions of this dissertation were:

- The implementation of a table row classifier, applicable to both HTML and spreadsheet tables.
- The implementation of two algorithms for table recognition in spreadsheet documents.
- The creation of two datasets containing annotated HTML and spreadsheets tables.
- Experiments to validate the implementations.

As future work, we propose to increment the number of instances and classes in our datasets and add more complex features. We expect that table row classifier, based on Conditional Random Fields, can also be applied to other non-tabular classification tasks involving content of various formatting and layouts.

6 Bibliography

[1] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, “InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables,” in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 97–108.

[2] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, “WebTables: Exploring the Power of Tables on the Web,” *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, 2008.

[3] H. Alani *et al.*, “Automatic ontology-based knowledge extraction from Web documents,” *IEEE Intell. Syst.*, vol. 18, no. 1, pp. 14–21, Jan. 2003.

[4] G. Limaye, S. Sarawagi, and S. Chakrabarti, “Annotating and Searching Web Tables Using Entities, Types and Relationships,” *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 1338–1347, 2010.

[5] P. Venetis *et al.*, “Recovering Semantics of Tables on the Web,” *Proc. VLDB Endow.*, vol. 4, no. 9, pp. 528–538, 2011.

[6] V. Mulwad, T. Finin, and A. Joshi, “Generating Linked Data by Inferring the Semantics of Tables,” in *Proceedings of the First International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search, Seattle, WA, USA, September 2, 2011*, 2011, vol. 880, pp. 17–22.

[7] M. Brambilla, F. Casati, and S. Ceri, Eds., “Proceedings of the First International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search, Seattle, WA, USA, September 2, 2011,” 2012, vol. 880.

[8] D. Jannach, K. Shchekotykhin, and G. Friedrich, “Automated Ontology Instantiation from Tabular Web sources-The AllRight System,” *Web Semant.*, vol. 7, no. 3, pp. 136–153, 2009.

[9] H. Sun, H. Ma, X. He, W. Yih, Y. Su, and X. Yan, “Table Cell Search for Question Answering,” in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 771–782.

[10] A. S. Corrêa and P.-O. Zander, “Unleashing Tabular Content to Open Data: A Survey on PDF Table Extraction Methods and Tools,” in *Proceedings of the 18th Annual International Conference on Digital Government Research*, 2017, pp. 54–63.

[11] B. Yildiz, K. Kaiser, and S. Miksch, “pdf2table: A Method to Extract Table Information from PDF Files.” .

[12] Y. Liu, P. Mitra, and C. L. Giles, “Identifying Table Boundaries in Digital Documents via Sparse Line Detection,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008, pp. 1311–1320.

[13] E. Oro and M. Ruffolo, “PDF-TREX: An Approach for Recognizing and Extracting Tables from PDF Documents,” in *2009 10th International Conference on Document Analysis and Recognition*, 2009, pp. 906–910.

[14] T. Kieninger, “Table Structure Recognition Based On Robust Block Segmentation,” 1998, pp. 22–32.

[15] Z. Zhang, “Towards Efficient and Effective Semantic Table Interpretation,” in *Proceedings of the 13th International Semantic Web Conference - Part I*, 2014, pp. 487–502.

[16] Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, Y. Ding, and G. Nagy, “Towards Ontology Generation from Tables,” *World Wide Web*, vol. 8, no. 3, pp. 261–285, 2005.

[17] H. Masuda, S. Tsukamoto, and H. Nakagawa, “Recognition of HTML table structure,” *Proc. First Int’l Jt. Conf. Nat. Lang. Process.*, 2004.

[18] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, “Table Header Detection and Classification,” in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012, pp. 599–605.

[19] D. Pinto, A. McCallum, X. Wei, and W. B. Croft, "Table Extraction Using Conditional Random Fields," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, 2003, pp. 235–242.

[20] D. Pinto *et al.*, "QuASM: A System for Question Answering Using Semi-structured Data," in *Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries*, 2002, pp. 46–55.

[21] V. Mulwad, T. Finin, Z. Syed, and A. Joshi, "Using Linked Data to Interpret Tables," in *Proceedings of the First International Conference on Consuming Linked Data - Volume 665*, 2010, pp. 109–120.

[22] E. Koci, M. Thiele, W. Lehner, and O. Romero, "Table Recognition in Spreadsheets via a Graph Representation," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, 2018, pp. 139–144.

[23] I. A. Doush and E. Pontelli, "Detecting and Recognizing Tables in Spreadsheets," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, 2010, pp. 471–478.

[24] E. Koci, M. Thiele, O. Romero, and W. Lehner, "Table Identification and Reconstruction in Spreadsheets," in *Advanced Information Systems Engineering*, 2017, pp. 527–541.

[25] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 282–289.

[26] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[27] M. D. Adelfio and H. Samet, "Schema Extraction for Tabular Data on the Web," *Proc. VLDB Endow.*, vol. 6, no. 6, pp. 421–432, 2013.