ENTERPRISE DATA INTEGRATION

ON EXTRACTING DATA FROM HTML TABLES

JUAN C. ROLDÁN

UNIVERSITY OF SEVILLA, SPAIN

DOCTORAL DISSERTATION
SUPERVISED BY DR. RAFAEL CORCHUELO AND DR. PATRICIA JIMÉNEZ

JUNE, 2020

First published in June 2020 by
The Distributed Group
ETSI Informática
Avda. de la Reina Mercedes, s/n
Sevilla, E-41012. SPAIN

Classification (ACM 2012): 10003279 [Information systems] Data extraction and integration; 10003278 [Information systems] Site wrapping; 10011732 [Applied computing] Information integration and interoperability.

Support: Supported by the PIF programme of the University of Seville (grant PPI2016-II.2), the Spanish R&D&I programme (grants TIN2013-40848-R and TIN2016-75394-R) and the Andalusian R&D&I programme (grant P18-RT-1060). Additional support for research visit granted by the Fulbright organisation.

University of Sevilla, Spain

The committee in charge of evaluating the dissertation presented by Juan C. Roldán in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering, hereby recommends ______________________ of this dissertation and awards the author the grade ____.

Dr. Miguel Toro
Full Professor
University of Sevilla

Dr. Juan Manuel Corchado Full Professor University of Salamanca	Dr. Vitor Basto Associate Professor Instituto Universitario de Lisboa
Dr. Pedro Szekely Associate Professor University of Southern California	Dr. Paula Montoto Associate Professor University of Coruña

To put record where necessary, we sign minutes in _____________, __________________.

Extracting data from HTML tables, by Nicolás, aged six.

To my parents.
To Blanca.

Contents

List of Figures

List of Tables

Acknowledgements

When I was a kid, every time someone asked me what I wanted to be, the answer was the same: an inventor. After many years, study nights, exams, papers, and discussions, I am finally about to become an inventor, a qualified one. And this dream would not be possible without the support of those who helped me since I was a kid.

First of all, I would like to thank to my family. My parents, for their constant encouragement in my walk through life; my sister, for her mood and support; my grandfather Cayetano, for awakening my desire for knowledge with his sketches, creations, and Jules Verne novels; and Blanca, for helping me through the hardest moments and being the reason of the best ones.

I also wish to thank my friends. Pablo, for our never-ending discussions about research and many other topics; Ana, Clara, Fede, Javier, Jose, Nai, and Paco, for their warm company; Dani, my fellow partner during this journey; and all of the good friends I found during my time in Los Angeles, chiefly Alex and Basel: salamat/shukran to you both.

Special thanks go to the TDG folks. To Dr. Ruiz for introducing me to the academic world, to Dr. Ortega and Dr. Cimmino for their help during our beginning at Geozoco, and to all the members of this group who are now *distributed* around the globe.

I am also grateful to the ISI staff for helping me and making my time there so easy and enlightening. Thanks to Pedro and Craig, for giving me such a good opportunity and for sharing your knowledge and experience with me. Thanks to Majid and Ehsan for our discussions about research and non-research. And thanks to Alma and Amy for making everything go smooth during the stay.

Last but not least, I would like to thank my advisers, who are much more than advisers: they have been my friends and mentors. Dr. Rafael Corchuelo imprinted on me his perpetual longing for something better; thanks to you for your dedication and generosity. Dr. Patricia Jiménez taught me to say what I think without a hassle; thanks to you for your honesty and tireless efforts.

Abstract

The Web is a universal communication channel that provides a vast amount of valuable data about a plethora of topics. In recent years, there has been a quick rise of data-hungry products and services that have motivated the need for ways to extract web to feed them with as little effort as possible. HTML tables are a source of up-to-date data that is not being extracted and loaded into major knowledge bases in an automated manner. Extracting them is challenging because there are several common layouts in which data are displayed and they present several encoding and formatting problems; furthermore, the available general-purpose data extractors ignore the particularities of HTML table encodings and do not suffice to deal with the intricacies of web tables.

In this dissertation, we have studied the problem of extracting data from HTML tables with no supervision. After completing an extensive review of the literature, we realised that none of the available table-specific proposals provided a holistic approach to solve this problem. This motivated us to work on TOMATE, a table extraction proposal that encompasses every table extraction task with an emphasis in the crucial task of identifying cell functions. Our experimental analysis proved that we have advanced the state of the art with several proposals that are intended to help both researchers and practitioners.

While working on this dissertation, we have developed a number of marginal contributions, namely: Aquila, a proposal to synthesise meta-data tags for HTML documents; Kizomba, a general extraction proposal that was called; and Romulo, a proposal to cluster data. Furthermore, we have collaborated on the inception of a start-up project called Stargazr where we hope to put much of the knowledge generated in this dissertation into practice.

Resumen

La Web es una vía universal de comunicación que contiene un volumen de datos extraordinario sobre una gran variedad de temas. En los últimos años se ha producido un rápido aumento de los productos y servicios que consumen gran cantidad de datos, lo que ha motivado la necesidad de encontrar formas de extraerlos automáticamente. Las tablas HTML son una fuente de datos actualizados que no se está integrando de forma automatizada a las principales bases de conocimiento. La extracción de tablas resulta compleja ya que existe una gran variedad de estructuras y formas de presentar y codificar los datos. Usar extractores de propósito general no es una solución al problema, dado que ignoran las particularidades del rico lenguaje que se usa para representar tablas.

En esta tesis hemos estudiado el problema de extraer datos de tablas HTML sin supervisión. Al realizar un análisis exhaustivo de la literatura de extracción de tablas, hemos observado que ninguna de las propuestas disponibles resuelve el problema al completo. Esto nos ha motivado a desarrollar TOMATE, una propuesta de extracción de tablas que abarca todas las tareas involucradas, aunque pone el énfasis en la tarea crucial de identificar la función de las celdas. Nuestro análisis experimental ha demostrado que hemos dado un paso adelante en el estado del arte con varias propouestas que tienen por objeto ayudar a investigadores y profesionales del sector.

Mientras que trabajabamos en esta tesis, hemos desarrollado algunas contribuciones marginales, a saber: Aquila, una propuesta para sintetizar etiquetas de metadatos para ficheros HTML; Kizomba, un extractor general de datos de la Web; y Romulo, una propuesta para clusterizar datos. Además, hemos colaborado internacionalmente en un proyecto start-up denominado Stargazr en el que tenemos como objetivo poner en práctica gran parte del conocimiento que hemos generado en esta tesis.

Chapter 1

Introduction

This chapter introduces our PhD work. It is organised as follows: Section 1.1 introduces the context of our research work; Section 1.2 presents the hypothesis that has motivated our work and states our thesis; Section 1.3 summarises our contributions; Section 1.4 sketches the collaborations that we have conducted; finally, Section 1.5 describes the structure of this document.

1.1 Research context

During the last three decades, the Web has reshaped our way of communicating with each other, and the amount of data available has grown exponentially. These data have proven to be extremely valuable, which is the motivation behind a new research field that focuses on producing techniques to extract them [12]. Very often, the data are encoded using HTML tables designed to be human-friendly [31], which makes it difficult to integrate them into automated business processes. Consequently, several general-purpose and table-specific extraction techniques have been devised.

Broadly speaking, web extractors apply a set of rules to one or more input HTML documents and extract their data into records that are expected to be relatively easy to integrate. According to how the rules are devised, they can be classified as follows: hand-crafted, supervised, unsupervised, and heuristic-based proposals. Hand-crafted proposals require a person to devise a set of extraction rules for a particular set of documents. Supervised proposals require a person to provide a training set with annotations from which a machine learner infers the extraction rules (the annotations are samples of the data to be extracted). Unsupervised proposals require a training set without annotations to learn the rules, which typically attempt to extract as much data as possible. Heuristic-based proposals use rules that are derived from the experience with previous problems and have proven to be general enough to extract data from as many documents as possible. Generally speaking, the unsupervised and heuristic-based proposals require less human effort than the hand-crafted or the supervised proposals, but still require some effort to remove irrelevant data and to endow the extractions with semantics.

In the literature, there are several general-purpose proposals that can extract data that are formatted using arbitrary layouts [22, 43, 111, 116, 123]. A few of them are hand-crafted [28, 110], many are supervised or unsupervised [29, 56, 61, 62, 66, 71, 79, 86, 117, 118], and a few are heuristic-based [10, 90, 91, 98, 104–106, 115, 124]. The supervised and unsupervised proposals use a variety of approaches to the problem, including learning text anchors [71], inferring transducers [56], learning from reference sets [86], inducing grammars [29], analysing visual features [79], aligning text [118], training neural networks [117], matching trees [66], learning first-order rules [61], or learning propositio-relational rules [62].

Some authors have worked in proposals that are specifically targeted towards extracting data from tables [27, 39, 58, 82, 107, 138, 139]. The earliest ones focus on tables that are encoded using pre-formatted text, images, or general mark-up languages. Recent trends seem to suggest that HTML is becoming pervasive to encode the tables in the Web [19], which has motivated most authors to restrict their attention to tables that are encoded using this mark-up language [107]. There is a consensus regarding formalising table understanding as a pipeline that consists of the following tasks [27]: a) location, which identifies the excerpts of the input documents where the tables are located; b) discrimination, which discards the tables that do not seem to provide any useful data; c) segmentation, which transforms the excerpts into grids of cells; d) functional analysis, which identifies the functions of the cells, that is, whether they provide data or meta-data; e) structural analysis, which identifies groups of related meta-data cells (the headers) and groups of related data cells (the tuples); and f) interpretation, which transforms the input tables into record sets in which the components of the tuples are mapped onto their corresponding headers. The key task seems to be the functional analysis task because the quality of the resulting records clearly depends on the ability of this task to correctly set meta-data cells apart from data cells. We have found twelve proposals that provide a solution to the functional analysis task in the context of tables that are encoded using HTML [17, 18, 23, 24, 38, 40, 64, 67, 87, 96, 134, 136]. They address the problem using a variety of approaches that range from using heuristics to using deep-neural networks.

1.2 Research rationale

In this section, we present the hypothesis that has motivated our research work and we also state the thesis that we prove in the rest of the dissertation.

1.2.1 Hypothesis

In recent years, we have experienced an inexorable rise of data-hungry products and services that are based on advanced machine-learning methods [73]. This explains the need for datasets on a variety of topics, which are used both to learn good models and to exploit them [137]. Web information extractors are particularly useful to generate streams of real-time data on a variety of topics using that that are available on the Web in human-friendly formats [12].

Hand-crafted proposals are useful insofar they may be very effective, but they require the user to devise extraction rules for every subset of relevant documents, which is not feasible at web scale. Supervised proposals are simpler because providing a training set is expected to be easier than devising an extraction rule, but generating the annotations is time-consuming and error-prone, which makes it difficult to scale supervised methods to the Web. Unsupervised proposals and heuristic-based proposals are, in principle, easier to scale, but they require the user to sieve the useful data and to endow them with semantics. This, however, seems to require less effort and to be less error-prone than the hand-crafted or the supervised approaches, which is the reason why we focus on them.

According to Forrester [85], the industrial reliance on real-time streams of data about a variety of topics has grown significantly. We have realised that HTML tables are a source of up-to-date data that is not being extracted into major knowledge bases in an automated manner. This finding was also pointed out by Oulabi and Bizer [98], who analysed the Wikipedia tables in a few domains that are well supported by DBpedia and found 206 690 records with no matches in the knowledge base. Thus, finding an automated way to integrate them into automated business processes is a key to simplify the process of self-service data preparation [125].

According to the previous argumentation, we formulate this hypothesis:

There is a growing interest in extracting data from HTML tables automatically. The goal is to use them to feed business processes, with an emphasis on business analytic processes. General-purpose web extraction proposals do not suffice to deal with the particularities of HTML table extraction. Automatically extracting web data in human-friendly formats is a step towards leveraging the Humanity's most powerful knowledge repository.

1.2.2 Thesis

We have carried out a comprehensive study of the proposals to extract data from HTML tables [107]. It makes it clear that there are many methods to locate, segment, and discriminate them, which means that it is difficult to make novel contributions regarding those tasks. It also reveals that many authors pay little attention to the structural analysis or the interpretation tasks, which means that they are relatively easy to implement once the functionality of the cells is properly identified. Clearly, the functional analysis task is a cornerstone to success regarding extracting data from HTML tables. According to our survey [107], there are twelve proposals to implement this task [17, 18, 23, 24, 38, 40, 64, 67, 87, 96, 134, 136]. Some of them can deal with horizontal listings only [17, 18, 24, 38]; a few of them can also deal with vertical listings [134, 136]; a few others can also deal with matrices [87]; only a few can deal with any table layouts [23, 40, 64, 67, 96]. They generally have trouble to deal with the variety of formats used to display data in tables, including multi-line headers, no headers at all, repeated headers, context cells, or factorised cells; they also have trouble to deal with typical encoding problems, including tables that do not have the same number of columns for each row or the many tables that do not encode the cells using the appropriate tags [107]. According to our experience, the previous problems are very common. We assembled two experimental datasets with tables that were randomly selected from the Wikipedia and the Dresden Web Table Corpus and we found out that 64.41% of them have one or more of the previous problems.

According to the previous argumentation, we formulate this thesis:

It is possible to develop an HTML table extractor that solves the most important challenges automatically with high effectiveness and efficiency on current web documents. We conjecture that putting an emphasis in the functional analysis task is the key.

1.3 Summary of contributions

Next, we summarise the contributions we have made to prove our thesis.

State of the art: it is a comprehensive literature review of the HTML table extraction proposals that have been published between 2000 and 2018. It proposes a unified vocabulary to describe table extraction proposals, as well as a set of general and task-specific characteristics to compare them. We have a journal article [107] regarding this contribution.

TOMATE: it is proposal that extracts data from HTML tables with arbitrary layouts into schema-less records that can be easily integrated into automated business processes. It solves the most important challenges identified in the state of the art that have a non-negligible impact on the effectiveness of the process. We have a conference paper [109] and a journal article [108] regarding this contribution.

We also developed several marginal contributions while we were working on the main ones, namely: Aquila [109], which is a proposal to synthesise meta-data tags in HTML documents markup that relies on a novel embedding approach to find a path that identifies a subset of DOM nodes whose attributes help learn good taggers; Kizomba [105, 106], which is a general-purpose web extraction framework in which several extraction methods can be integrated, with an emphasis on being scalable and open; and Romulo [26], which is a proposal to support the identification of cell functions in TOMATE by applying a meta-heuristic approach to multi-way single-subspace clustering.

1.4 Collaborations

During the development of this dissertation, a nine-month Fulbright research visit was organised at the Information Sciences Institute, University of Southern California (USA). This visit was paid to the Centre on Knowledge Graphs, which is headed by Prof. Dr. Craig Knoblock and supervised by Prof. Dr. Pedro Szekely. The goal was to design the table extraction tool which is now the core of this dissertation after our review of the literature on this topic. This proposal was integrated into two IARPA projects. The first one is DataMart, which is a platform that automatically indexes datasets and APIs with semantic information so that they can be retrieved using a search engine. TOMATE was used to extract data from HTML tables from all over the Web in order to widen the number of supported sources. The second project is SAGE, a geo-political forecasting tool that used our proposal to feed the forecasts with data from Wikipedia tables. It also gave us a good perspective on which of the problems solved by our table extraction tools were more relevant.

Later, an on-going remote collaboration with the Technical University of Chemnitz was organised. Part of this work is developed by another doctoral student, Rafi Wadan, who is supervised by Prof. Dr. Uwe Göetze. Its goal is to extract automatically real-time data about job postings from different sites in order to compare the changing role of management accountants in Germany and the United States. Since the automatic extraction from several sites with different structures is a time consuming task, our web extraction framework Kizomba, along with the table extraction proposal TOMATE, was used to automate this process and test both proposals in a real-world task.

The latter collaboration has motivated the creation of a start-up called Stargazr, whose goal is to provide a set of business tools to analyse and forecast financial data to support the decision making process. These tools integrate the data extraction proposals described in this dissertation to extract and prepare data from the Web, amongst other sources.

1.5 Structure of this dissertation

This dissertation is organised as follows:

- The introduction comprises this chapter, in which we motivate our research work and conclude that there exists a need to devise a proposal to extract data from HTML tables that solve the most important challenges with high effectiveness and efficiency on current web documents in an automatic manner.

- Chapter 2 presents our study of the state of the art regarding HTML table extraction. It studies the existing surveys, defines a conceptual framework to describe proposals using the same terminology, introduces a taxonomy of tables, and summarises every proposal comparing them with the ones that perform the same task.

- Chapter 3 describes TOMATE, our table-specific clustering approach to extract data from tables. We present the motivation of our work, introduce some preliminaries required to understand our proposal, describe its details regarding each step, and illustrate how it works with a real world case study.

- Chapter 4 describes the experimental analysis of TOMATE. It introduces the proposals that were compared in the analysis; then, describes how we gathered our datasets, reports on the configuration of our proposal, defines our experimental methodology, analyses our experimental results, and performs some statistical analysis to confirm that our thesis is validated.

- Chapter 5 concludes this dissertation. It summarises our key findings and sketches some future work towards unsupervised web data extraction.

The dissertation includes three appendices that describe our marginal contributions.

Chapter 2

Background Information

Our purpose in this chapter is to introduce the background information regarding this dissertation. Section 2.1 introduces the chapter; Section 2.2 introduces the preliminaries required to understand this dissertation; Section 2.3 describes and compares the proposals that are related to the location task; Section 2.4 describes and compares the proposals that are related to the segmentation task; Section 2.5 describes and compares the proposals that are related to the discrimination task; Section 2.6 describes and compares the proposals that are related to the functional analysis task; Section 2.7 describes and compares the proposals that are related to the structural analysis task; Section 2.8 describes and compares the proposals that related to the interpretation task; and Section 2.9 summarises our conclusions.

2.1 Introduction

Web documents encode data in a way that facilitates rendering them in human-friendly formats. Data extractors help extract those data as datasets that are suitable to feed business applications [43].

Many web documents provide data in tabular formats. Understanding them has proven particularly useful for answering queries [17], creating summaries [80], saving storage space [15], adapting data to mobile devices [130], helping impaired people have access to many web sites [83], finding related data [14], or creating linked data [69], just to mention a few applications. Unfortunately, general-purpose data extractors do not deal well with the intricacies of HTML tables, which has motivated many authors to work on specific-purpose proposals [27, 39, 57, 81, 82, 138] that can be decomposed into the following tasks: locating tables in input documents, discriminating data tables, segmenting them into cells, analysing their functionality, analysing their structure, and interpreting them to produce datasets.

There are dozens of proposals to implement the previous tasks, which motivated Lopresti and Nagy [81, 82], Hurst [57], Zanibbi et al. [138], Costa-Silva et al. [27], and Embley et al. [39] to survey them. Unfortunately, the most recent survey was published in 2 003, which means that they have missed the steady shift from encoding tables using pre-formatted text or images to encoding tables using HTML tabular, listing, or block tags. Unfortunately, updating them is not enough because we have identified the following shortcomings: almost none of them provide a conceptual framework, none of them provide a taxonomy of tables, none of them analyse all of the tasks involved in extracting data from tables (but some of them report on complementary tasks), and almost none of them provide a comparison framework.

Our survey reports on 26 proposals that were published from 2 000 until 2 018; it then provides an up-to-date picture of the current state of the art that complements the proposals that were surveyed by other authors. It also addresses the shortcomings that we have found in the literature. We think that our review of the literature shall help both researchers and practitioners who need to have as a complete picture as possible of the current state of the art; researchers shall also have a conceptual framework that shall help them describe their proposals as homogeneously as possible and a taxonomy that shall help them delimit their scope of applicability; both researchers and practitioners shall be able to compare the many existing proposals side-by-side and practitioners shall be able to make informed decisions thanks to our comparison framework.

2.2 Conceptual framework

In this section, we first introduce the mathematical notation that we use to formalise our conceptual framework, then report on the data structures that we require, and then on the tasks involved in the extraction process.

2.2.1 Notation

We first introduce some basic mathematical notation regarding fundamental concepts and given sets that puts a foundation to the rest of our conceptual framework.

Notation 1 (Fundamentals) $A = A_1 \mid A_2 \mid \ldots \mid A_n$ $(n \geq 1)$ introduces a set definition, where A is the name of a new set and A_i $(i = 1 \mathinner{..} n)$ are set denotations. A definition is inductive if some, but not every A_i refers to set A. Set denotations are introduced as follows: $\emptyset$ is the empty set, $\{a_1, a_2, \ldots, a_n\}$ is a set extension $(n \geq 1)$, $A \cup B$ is the union of sets A and B, $A \to B$ is the set of total functions from set A onto set B, $A \nrightarrow B$ is the set of partial functions from set A onto set B, $A \times B$ is the Cartesian product of sets A and B, $\mathbb{F} A$ is the finite power set of set A, and $n \mathinner{..} m$ is the set of naturals from n up to m if $n \leq m$ and $\emptyset$ if $n > m$. The elements of sets $A \to B$, $A \nrightarrow B$, or $A \times B$ are denoted as tuples of the form (a, b), where $a \in A$ and $b \in B$.

Notation 2 (Given sets) We use the following given sets: $\mathsf{Natural}$, which denotes the set of natural numbers; $\mathsf{Document}$, which denotes the set of HTML documents, $\mathsf{Excerpt}$, which denotes the set of HTML excerpts, Type, which denotes a set of table types, Name, which denotes a set of names, and Value, which denotes a set of values. We also define the set of positions in a table as $\mathsf{Position} = \mathsf{Natural} \times \mathsf{Natural}$.

2.2.2 Data structures

Now, we introduce the data structures in our conceptual framework. We illustrate them with the table and the dataset in Figure 2.1, which were gathered from a share market domain. The table provides data about the volume and the risk of the shares of a company that is identified by means of a ticker; the data are sampled at different times and they are shown regarding today and yesterday. Some grid lines are greyed and dashed, which means that there are some cells in the encoding of the table whose corresponding grid lines are invisible when it is rendered.

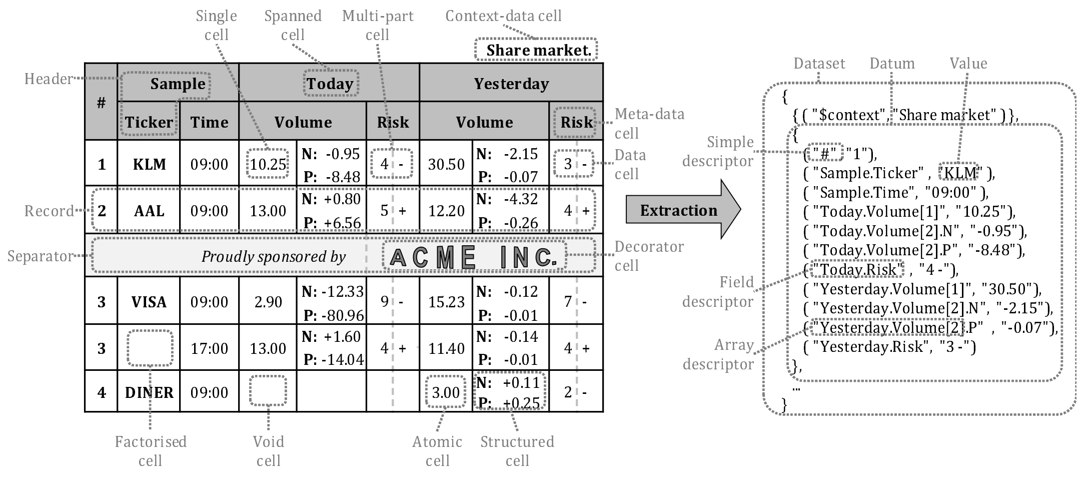


Figure 2.1: *Table extraction concepts.*

Definition 2.1 (Properties) *A property is a value that one of the tasks involved in the extraction process computes from a cell or a table so that other tasks can use it. Formally, we define* $\mathsf{Property} = \mathsf{Name} \times \mathsf{Value}$*; that is, a property* p *is represented as a tuple of the form* (n, v)*, where* n *denotes its name and* v *denotes its value. We assume that properties are used to store the function of each cell in a table and to group cells that must be interpreted together, but specific proposals might use them to store additional data.*

Definition 2.2 (Cells) *A cell is the smallest unit of which a table is composed. Formally, we define* $\mathsf{Cell} = \mathsf{Value} \times \mathbb{F}\,\mathsf{Property}$*; that is, a cell* c *is represented as a tuple of the form* (v, F)*, where* v *denotes its value and* F *denotes its set of properties. According to how they must be segmented, cells are classified as single cells, whose values occupy exactly one position, spanned cells, whose values spread across several positions, multi-part cells, whose values are encoded using several positions, and context-data cells, which are cells that do not occupy a position within the body of the table, but provide titles, notes, or related text that must be interpreted with the table. According to their function, cells are classified as meta-data cells, which provide values that help endow other cells with semantics, data cells, which provide the data to be extracted, and decorator cells, which provide irrelevant data. According to how their values must be interpreted, cells are classified as factorised cells, which are empty cells whose values must be borrowed from adjacent cells, void cells, whose values are missing, atomic cells,* *whose values cannot be decomposed further, and structured cells, whose values can be decomposed and may have a mixture of data and meta-data. Meta-data cells are grouped into headers, data cells are grouped into records, and decorator cells are grouped into separators.*

Definition 2.3 (Tables) *A table is a collection of cells that are laid out two-dimensionally in its body plus some context-data cells in its surroundings, if any. Formally, we define* $\mathsf{Table} = (\mathsf{Position} \nrightarrow \mathsf{Cell}) \times \mathbb{F}\,\mathsf{Cell} \times \mathbb{F}\,\mathsf{Property}$*; that is, a table* t *is represented as a tuple of the form* (D, C, P)*, where* D *maps its positions onto the cells in its body,* C *denotes its context-data cells, and* P *denotes its set of properties.*

Definition 2.4 (Datasets) *A dataset is a structured representation of the data that is provided by a single table. Formally, we define them using the following sets:* $\mathsf{Dataset} = \mathbb{F}\,\mathsf{Datum}$*, where* $\mathsf{Datum} = \mathsf{Descriptor} \nrightarrow \mathsf{Value}$*,* $\mathsf{Descriptor} = \mathsf{SimpleDescriptor} \mid \mathsf{FieldDescriptor} \mid \mathsf{ArrayDescriptor}$*,* $\mathsf{SimpleDescriptor} = \mathsf{Value}$*,* $\mathsf{FieldDescriptor} = \mathsf{Descriptor} \times \mathsf{Value}$*, and* $\mathsf{ArrayDescriptor} = \mathsf{Descriptor} \times \mathsf{Natural}$*. Simply put, a dataset is a set of data of the form* $\{(d_1, v_1), (d_2, v_2), \dots, (d_n, v_n)\}$*, where each* d_i *is a descriptor that helps endow value* v_i *with semantics (*$i = 1 \mathinner{..} n$*,* $n \geq 1$*).*

2.2.3 Extraction tasks

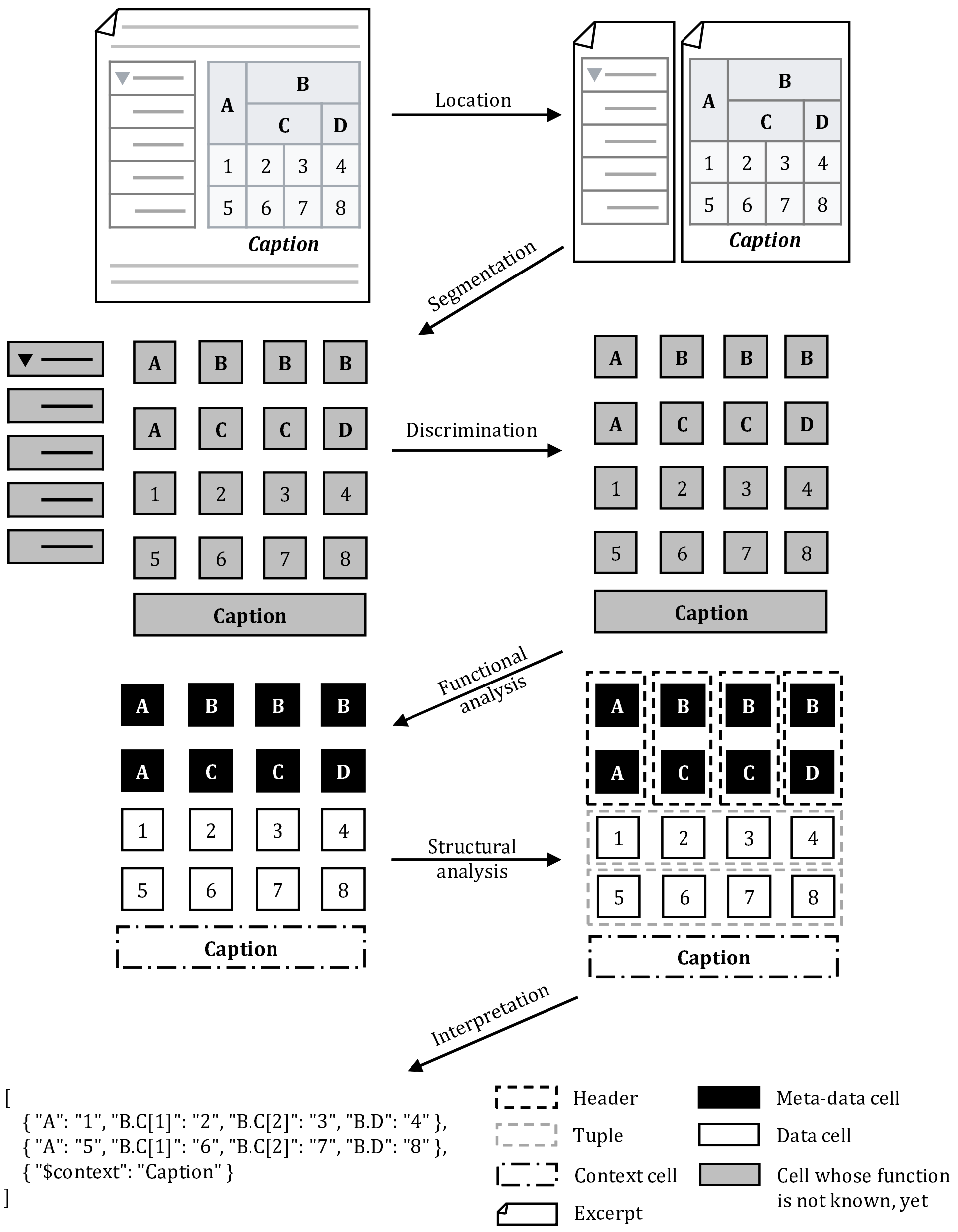
Figure 2.2 illustrates the tasks involved in the extraction process, which we define below. One proposal may address one or more tasks, cf. Table 2.1.

Definition 2.5 (Location) *This task searches an input document for excerpts that contain tables. Formally, we define* $\mathsf{Location} = \mathsf{Document} \rightarrow \mathbb{F}\,\mathsf{Excerpt}$*. Specific proposals differ regarding the encodings that they can identify regarding the bodies of the tables and their context-data cells.*

Definition 2.6 (Segmentation) *This task searches for cells in an excerpt that encodes a data table. Formally, we define* $\mathsf{Segmentation} = \mathsf{Excerpt} \rightarrow \mathsf{Table}$*. Specific proposals differ regarding how they deal with spanned, multi-part cells, and context-data cells.*

Definition 2.7 (Discrimination) *This task classifies a table according to a taxonomy. Formally,* $\mathsf{Discrimination} = \mathsf{Table} \rightarrow \mathsf{Type}$*. Every proposal must make non-data tables apart from data tables; specific proposals differ in the specific types of data tables that they can discriminate.*

Definition 2.8 (Functional analysis) *This task identifies the function of a cell. Formally, we define* $\mathsf{FunctionalAnalysis} = \mathsf{Table} \rightarrow \mathsf{Table}$*; note that we assume that the results of this task are stored using some properties of the*



```
[
   { "A": "1", "B.C[1]": "2", "B.C[2]": "3", "B.D": "4" },
   { "A": "5", "B.C[1]": "6", "B.C[2]": "7", "B.D": "8" },
   { "$context": "Caption" }
]
```

Figure 2.2: *Sample table extraction process.*

Reference	Location	Segmentation	Discrimination	Functional analysis	Structural analysis	Interpretation
Chen et al. 2000	Naive	Naive	Yes	Yes	Naive	Yes
Lerman et al. 2001	Yes	Yes	No	No	Yes	No
Penn et al. 2001	Naive	No	Yes	No	No	No
Yoshida and Torisawa 2001	No	No	No	Yes	Naive	No
Cohen et al. 2002	Naive	Yes	Yes	No	Yes	No
Hurst 2002	Naive	No	Yes	No	No	No
Wang and Hu 2002	Naive	No	Yes	No	No	No
Yang and Luk 2002	Naive	Naive	Yes	Yes	Yes	Yes
Lerman et al. 2004	Yes	Yes	No	No	Yes	No
Kim and Lee 2005	Naive	No	Yes	Yes	No	No
Jung and Kwon 2006	Naive	Naive	Yes	Yes	No	No
Gatterbauer et al. 2007	Yes	Yes	Yes	Naive	No	No
Okada and Miura 2007	Naive	No	Yes	No	No	No
Cafarella et al. 2008	Naive	No	Yes	Yes	No	Yes
Crestan and Pantel 2011	Naive	No	Yes	No	No	No
Elmeleegy et al. 2011	No	Yes	No	No	Naive	No
Fumarola et al. 2011	Yes	No	Yes	No	Yes	No
Lautert et al. 2013	No	No	Yes	No	No	No
Ling et al. 2013	Yes	Yes	No	Naive	Naive	No
Son and Park 2013	Naive	No	Yes	No	No	No
Braunschweig et al. 2015	No	No	No	Naive	Naive	No
Chu et al. 2015	Naive	Yes	No	No	Naive	No
Eberius et al. 2015	Naive	No	Yes	No	No	No
Wu et al. 2016	Naive	No	Yes	Naive	Naive	Yes
Nishida et al. 2017	No	No	Yes	No	No	No
Liao et al. 2018	Naive	No	Yes	No	No	No

Table 2.1: *Tasks addressed by each table extraction proposal.*

cells of the output table. Every proposal must make meta-data cells apart from data cells; specific proposals differ regarding their ability to identify decorator cells.

Definition 2.9 (Structural analysis) *This task groups cells. Formally, we define* $\mathtt{StructuralAnalysis} = \mathtt{Table} \to \mathtt{Table}$*; we also assume that the results are stored using some properties of the cells of the output table. Every proposal must be able to identify headers and records; specific proposals differ in their ability to identify different header structures and layouts, the dimensionality, orientation, and multiplicity of the records, and how they deal with separators.*

Definition 2.10 (Interpretation) *This task extracts the data in a table into a dataset. Formally, we define* $\mathtt{Interpretation} = \mathtt{Table} \to \mathtt{Dataset}$. *Every proposal must be able to create pairs of simple descriptors and values; specific proposals differ in their ability to create more complex descriptors and their* *ability to make factorised cells apart from void cells and to distinguish between atomic cells and structured cells.*

2.2.4 Comparison characteristics

We compare the proposals that we have surveyed by means of a comparison framework that includes both general and task-specific characteristics.

The general characteristics are the following: a) *Foundation:* it is a hint on the technique behind each proposal. b) *Tables required:* it is the minimum number of tables required for a proposal to work; the less tables required, the better. c) *Effectiveness:* it is the extent to which a proposal succeeds in implementing a task correctly according to an effectiveness measure; the higher the effectiveness, the better. d) *Efficiency:* it is the amount of computing power that a proposal requires to implement a task; the more efficient (i.e., the less computing power is required), the better. e) *Resources:* it refers to the resources that a user must provide so that a proposal can work properly; the less resources, the better. f) *Features:* it refers to the features onto which the input data must be projected in order to machine learn a predictor or to make a decision according to a heuristic. Features can be either structural, which are related to the HTML or the DOM representation of the input documents, visual, which are related to their rendering, or value features, which are related to the values of the cells. g) *Parameters:* it refers to the settings that must be tuned so that a proposal works well, which can be either pre-defined, learnable, or user-defined parameters. Pre-defined parameters have a value that the authors of a proposal have found generally appropriate; they are preferable to learnable parameters, whose values must be experimentally learnt by the user; in turn, they are preferable to user-defined parameters, which must be set by the user using his or her intuition; the less parameters, the better.

Note that it is easy to make decisions building on the general features that we presented above since we have characterised their preferred values; the same applies to the task-specific features that we describe in the following sub-sections. The only exceptions are the foundation characteristic and the features characteristic. The reason is that it is not generally clear whether a heuristic-based approach is preferable to a machine-learning approach or vice versa, or whether structural, visual, or value features are preferable to each other. Note, too, that effectiveness and efficiency are decision-making characteristics, but the figures provided by an author are not generally comparable to the figures provided by a different author because they evaluated their proposals using different machines, learning sets, and evaluation sets.

Property	Categories			
	Listing	Form	Matrix	Enumeration
Record dimensionality	1	1	2	0
Record multiplicity	*	1	1	*
Record orientation	Horizontal, Vertical	Horizontal, Vertical	None	None
Header layout	None, Single, Horizontally repeated, Vertically repeated, Split	None, Single, Split	Single	None

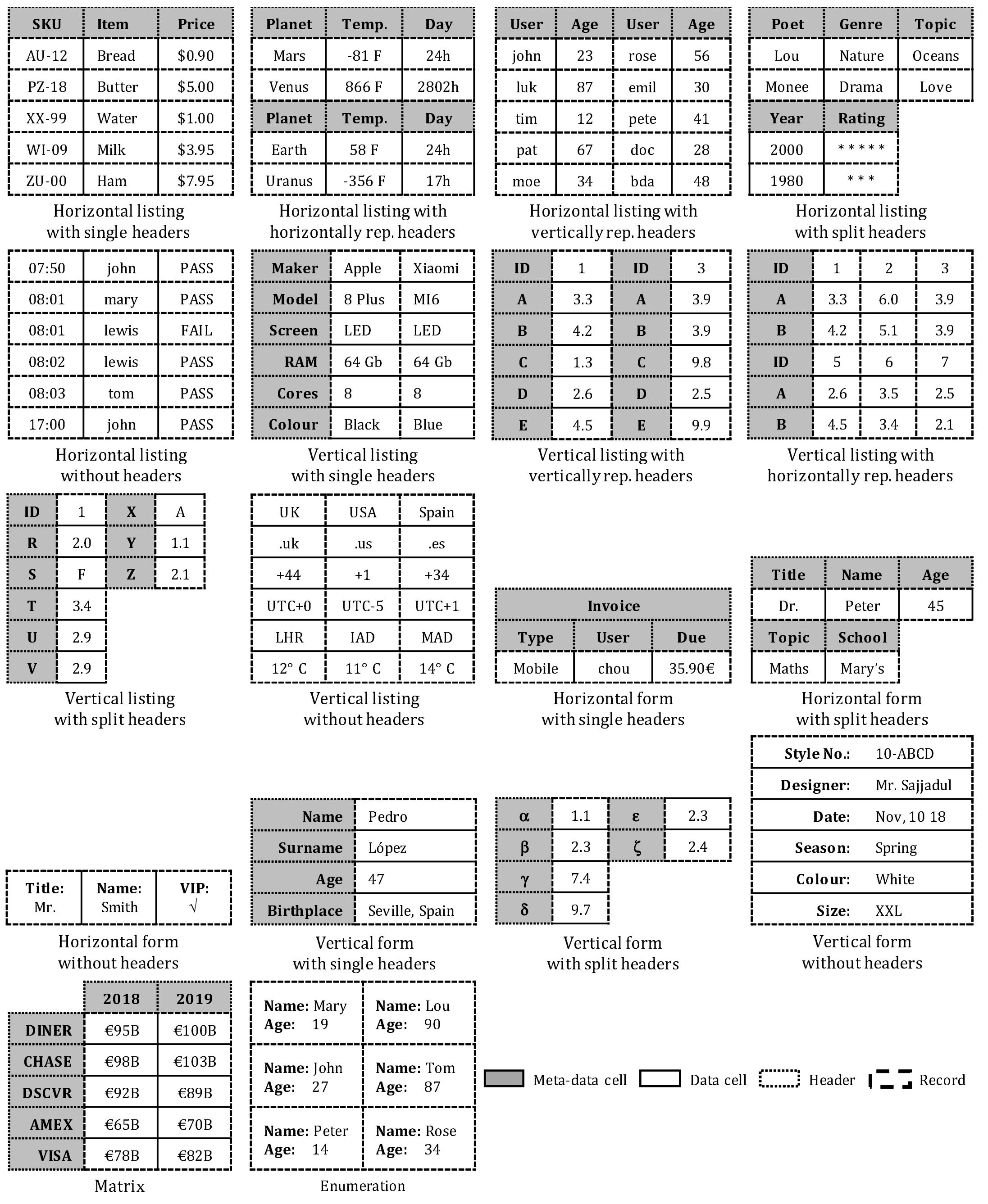
Table 2.2: *Taxonomy of data tables.*

2.2.5 Taxonomy of tables

HTML tables can be classified as non-data tables, which are used for layout purposes or to display utilities, and data tables, which are used to display relevant data.

We classify data tables as listings, forms, matrices, or enumerations building on the following properties, cf. Table 2.2 and Figure 2.3: a) *Record dimensionality:* it refers to the dimensions of the records shown in the table; its values are 0, which means zero-dimensional records that consist of single data cells, 1, which means one-dimensional records that consist of every data cell in a row or a column, and 2, which means two-dimensional records that consists of every data cell in a table. b) *Record multiplicity:* it refers to the number of records that a table is intended to show; its values are 1, which means exactly one record, or *, which means zero, one, or more records. Note that listings are data tables with record dimensionality 1 and record multiplicity *, forms are data tables with record dimensionality 1 and record multiplicity 1, matrices are data tables with dimensionality 2 and record multiplicity 1, and enumerations are data tables with record dimensionality 0 and record multiplicity *.

The variants depend on some additional properties: a) *Record orientation:* it refers to how the records are laid out; its values are none, which means that records are not explicitly oriented, horizontal, which means that the records are laid out in a row-wise manner, or vertical, which means that they are laid out in a column-wise manner. b) *Header layout:* it refers to how headers are laid out in a table; its values are none, which means that a table does not have any headers, single, which means that its headers lay at the first rows and/or columns, horizontally repeated, which means that they are



SKU	Item	Price
AU-12	Bread	$0.90
PZ-18	Butter	$5.00
XX-99	Water	$1.00
WI-09	Milk	$3.95
ZU-00	Ham	$7.95

Horizontal listing with single headers

Planet	Temp.	Day
Mars	-81 F	24h
Venus	866 F	2802h
Planet	Temp.	Day
Earth	58 F	24h
Uranus	-356 F	17h

Horizontal listing with horizontally rep. headers

User	Age	User	Age
john	23	rose	56
luk	87	emil	30
tim	12	pete	41
pat	67	doc	28
moe	34	bda	48

Horizontal listing with vertically rep. headers

Poet	Genre	Topic
Lou	Nature	Oceans
Monee	Drama	Love
Year	Rating	
2000	* * * * *	
1980	* * *	

Horizontal listing with split headers

07:50	john	PASS
08:01	mary	PASS
08:01	lewis	FAIL
08:02	lewis	PASS
08:03	tom	PASS
17:00	john	PASS

Horizontal listing without headers

Maker	Apple	Xiaomi
Model	8 Plus	MI6
Screen	LED	LED
RAM	64 Gb	64 Gb
Cores	8	8
Colour	Black	Blue

Vertical listing with single headers

ID	1	ID	3
A	3.3	A	3.9
B	4.2	B	3.9
C	1.3	C	9.8
D	2.6	D	2.5
E	4.5	E	9.9

Vertical listing with vertically rep. headers

ID	1	2	3
A	3.3	6.0	3.9
B	4.2	5.1	3.9
ID	5	6	7
A	2.6	3.5	2.5
B	4.5	3.4	2.1

Vertical listing with horizontally rep. headers

ID	1	X	A
R	2.0	Y	1.1
S	F	Z	2.1
T	3.4		
U	2.9		
V	2.9		

Vertical listing with split headers

UK	USA	Spain
.uk	.us	.es
+44	+1	+34
UTC+0	UTC-5	UTC+1
LHR	IAD	MAD
12° C	11° C	14° C

Vertical listing without headers

Invoice		
Type	User	Due
Mobile	chou	35.90€

Horizontal form with single headers

Title	Name	Age
Dr.	Peter	45
Topic	School	
Maths	Mary's	

Horizontal form with split headers

Title: Mr.	**Name:** Smith	**VIP:** √

Horizontal form without headers

Name	Pedro
Surname	López
Age	47
Birthplace	Seville, Spain

Vertical form with single headers

α	1.1	ε	2.3
β	2.3	ζ	2.4
γ	7.4		
δ	9.7		

Vertical form with split headers

Style No.:	10-ABCD
Designer:	Mr. Sajjadul
Date:	Nov, 10 18
Season:	Spring
Colour:	White
Size:	XXL

Vertical form without headers

	2018	2019
DINER	€95B	€100B
CHASE	€98B	€103B
DSCVR	€92B	€89B
AMEX	€65B	€70B
VISA	€78B	€82B

Matrix

Name: Mary **Age:** 19	**Name:** Lou **Age:** 90
Name: John **Age:** 27	**Name:** Tom **Age:** 87
Name: Peter **Age:** 14	**Name:** Rose **Age:** 34

Enumeration

▨ Meta-data cell ▭ Data cell ⬚ Header ⌐ ¬ Record

Figure 2.3: *Sample data tables.*

repeated every some rows, vertically repeated, which means that they are repeated every some columns, and split, which means that they spread across several non-adjacent rows or columns.

Note that once meta-data cells are made apart from data cells and headers are identified, it is relatively easy to identify the records and to classify a table. Headerless tables are a bit more difficult to classify because they require to analyse the homogeneity of their cells both from a lexical and a semantic point of view. For instance, independently from whether a listing has headers or not, its cells are expected to be column-wise homogeneous in a horizontal listing and row-wise homogeneous in a vertical listing; the key property of a form is that it is intended to display a single record, so the cells are not expected to be homogeneous; contrarily, an enumeration is expected to display multiple records that consists of only one cell, so they all are assumed to be homogeneous.

2.3 Location

Yoshida et al. [136], Elmeleegy et al. [38], Lautert et al. [74], Braunschweig et al. [14], and Nishida et al. [96] did not pay attention to the location task. Chen et al. [23], Cohen et al. [25], Hurst [58], Yang and Luk [134], Kim and Lee [67], Jung and Kwon [64], Okada and Miura [97], Cafarella et al. [17], Son and Park [119], Chu et al. [24], Eberius et al. [34], Wu et al. [131], and Liao et al. [77] reported on naive approaches that consisted in extracting every HTML excerpt with a table tag; Penn et al. [99], Wang and Hu [127], and Crestan and Pantel [31] followed the same approach but discarded tables with nested tables. The other proposals provide more sophisticated approaches.

2.3.1 Summary of proposals

Lerman et al. [75, 76] focused on tables that are encoded using listing tags. Their proposal works as follows: a) first, the input documents are first tokenised and each token is assigned one or more lexical types; b) then, the smallest input document is picked as a base template; c) the remaining documents are then iteratively compared to the template in order to make the sequences of tokens that appear exactly once apart from the others; d) finally, the excerpts of the document that have the largest repetitive sequence of tokens are returned. The authors did not evaluate their procedure in isolation, but their complete system.

Gatterbauer et al. [48] presented a visual approach that analyses the bounding boxes used to render the elements of a document in an attempt to identify tables, lists, and so-called aligned graphics. Their proposal works as follows: a) first, every excerpt that meets some user-defined constraints that typically hold for tables, lists, or aligned graphics is retained as a data table while the others are discarded as non-data tables (i.e., they do not have to wait for the discrimination task); b) some heuristics are then applied in order to find so-called frames, which consist of data and meta-data cells that are close to each other; c) finally, the frames are expanded to four orthogonal directions by finding elements whose bounding boxes are near to each other; d) finally, the corresponding excerpts are returned. The authors evaluated their proposal on 493 tables from their own repository plus 19 additional tables from Wang and Hu's [127] repository.

Fumarola et al. [45] also presented a visual approach to the problem. Their proposal works as follows: a) it first creates a bounding box that encloses the whole input document; b) it then iterates recursively and creates a bounding box for every element in that document; c) next, it analyses the positions of the inner bounding boxes and finds those that are laid out in a row- or a column-wise manner; d) then, the corresponding excerpts are returned. The authors did not evaluate their procedure in isolation, but their complete system.

Ling et al. [78] presented a proposal whose focus is on locating context-data cells. It works as follows: a) first, it locates the elements in the input document that have a table tag; b) then, it extracts some context data from the title tag; c) next, it segments the text around the tables and aligns the resulting segments using a multiple string alignment algorithm; d) finally, the segments that are repetitive enough are considered context-data cells. The authors did not evaluate their procedure in isolation, but their whole system.

2.3.2 Comparison

Table 2.3 summarises our comparison regarding location proposals. The task-specific characteristics are the following: a) *Body encodings:* it refers to how the tables that a proposal can locate must be encoded; the more kinds of encodings are identified, the better. b) *Context-data encodings:* it refers to how context-data cells are encoded; the more kinds of encodings are identified, the better.

Regarding the general characteristics, it is surprising that all of the location proposals are based on heuristics; there is no record in the literature of a single proposal that has tried a machine-learning approach. Most proposals can work on a single table, but the ones by Lerman et al. [75, 76] and

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features: Structural	Features: Visual	Features: Value	Parameters: Pre-defined	Parameters: Learnable	Parameters: User-defined	Body encodings	Context data encodings
Chen et al. 2000	Heuristics	1	N/A	N/A								Tabular tags	Caption tag
Lerman et al. 2001	Heuristics	2+	N/A	N/A								Listing tags	
Penn et al. 2001	Heuristics	1	N/A	N/A								Leaf tabular tags	
Cohen et al. 2002	Heuristics	1	N/A	N/A								Tabular tags	
Hurst 2002	Heuristics	1	N/A	N/A								Tabular tags	
Wang and Hu 2002	Heuristics	1	N/A	N/A								Leaf tabular tags	
Yang and Luk 2002	Heuristics	1	N/A	N/A								Tabular tags	
Lerman et al. 2004	Heuristics	2+	N/A	N/A								Listing tags	
Kim and Lee 2005	Heuristics	1	N/A	N/A								Tabular tags	
Jung and Kwon 2006	Heuristics	1	N/A	N/A								Tabular tags	
Gatterbauer et al. 2007	Heuristics	1	N/A	N/A		Presence of tag table; cell adjacency.	CSS attributes color, bgcolor, font-size, font-style, font-weight, font-family, and text-align; attribute href.		Maximum distance to adjacent cells			Any	
Okada and Miura 2007	Heuristics	1	N/A	N/A								Tabular tags	
Cafarella et al. 2008	Heuristics	1	N/A	N/A								Tabular tags	
Crestan and Pantel 2011	Heuristics	1	N/A	N/A						Minimum number of samples per node		Leaf tabular tags	
Fumarola et al. 2011	Heuristics	1	N/A	N/A								Any	
Ling et al. 2013	Heuristics	2+	N/A	N/A								Tabular tags	Tag title; surrounding text.
Son and Park 2013	Heuristics	1	N/A	N/A								Tabular tags	
Chu et al. 2015	Heuristics	1	N/A	N/A								Tabular tags	
Eberius et al. 2015	Heuristics	1	N/A	N/A								Tabular tags	
Wu et al. 2016	Heuristics	1	N/A	N/A								Tabular tags	
Liao et al. 2018	Heuristics	1	N/A	N/A								Tabular tags	

Table 2.3: *Comparison of location proposals.*

Ling et al. [78] require at least a pair of tables to perform table alignment. None of the proposals was presented in isolation, but as a component of a larger system, which is the reason why no author reported on effectiveness or efficiency. Realise that only the proposal by Gatterbauer et al. [48] projects the input documents onto structural and visual features in order to apply their heuristics; note, too, that it is the only one that requires a predefined parameter. The proposal by Crestan and Pantel [31] is the only that requires the user to set a learnable parameter.

Regarding the task-specific characteristics, most of the proposals locate tables that are encoded using tabular tags, a few focus on tables that are encoded using listing tags, and only Gatterbauer et al. [48] and Fumarola et al. [45] are independent from the tags used, since they analyse the rendering of the input documents, which makes them the only proposals that might deal with tables that are encoded using block tags. Note, too, that the vast majority of proposals focus on locating the tables themselves, not their context-data cells. Chen et al. [23] and Ling et al. [78] are the exceptions: the former presents a simple approach that searches for caption tags and the latter presents a more sophisticated approach that analyses the title tags and the text that surrounds the tables.

2.4 Segmentation

Penn et al. [99], Yoshida et al. [136], Hurst [58], Wang and Hu [127], Kim and Lee [67], Okada and Miura [97], Cafarella et al. [17], Crestan and Pantel [31], Fumarola et al. [45], Lautert et al. [74], Son and Park [119], Braunschweig et al. [14], Eberius et al. [34], Wu et al. [131], Nishida et al. [96], and Liao et al. [77] did not report on any proposals to implement this task. Chen et al. [23], Yang and Luk [134], Jung and Kwon [64] relied on a naive approach that searches for the cells as they are encoded in the input documents using specific-purpose tags. The other proposals provide more sophisticated approaches.

2.4.1 Summary of proposals

Lerman et al. [75] segmented tables that are encoded using listing tags building on the tags themselves and some punctuation symbols; implicitly, they assumed that records are shown in a row-wise manner. Prior to segmentation, the authors applied a document alignment method to detect the template of the documents and their repetitive segments, which are very likely to contain the lists. Once the lists are located, their proposal works as follows: a) the segments are clustered according to their separators to the left and to the right; b) then, the DataPro pattern-learning system is invoked on the previous clusters in an attempt to learn patterns that characterise their data; c) then, for each segment in each cluster, it computes binary features that indicate whether it matches the previous patterns or not; d) next, the AutoClass clustering algorithm is invoked to learn the optimal number of clusters and to learn a set of rules that assign new segments to the most similar cluster; e) finally, the data in each cluster is assumed to be a column of the corresponding table, which facilitates identifying the cells in a row-wise manner. The evaluation was performed on the tables from a repository with 50 documents that were taken from 14 different sources. The authors did not evaluate their procedure in isolation, but their complete system.

Cohen et al. [25] relied on some transformations that help normalise tables before they are segmented. Their proposal works as follows: a) the HTML structure is cleaned up using HTML Tidy and the extra cells generated by this tool are removed; b) structured cells are divided into multiple atomic cells by splitting inner tables, paragraphs, or pre-formatted text; c) spanned cells are split into several cells unless this results in more cells than the height or the width of the table. The authors did not evaluate their procedure in isolation, but their complete system.

Lerman et al. [76] segmented tables by learning a probabilistic model from the repetitive segments in which they decompose tables that are encoded using listing tags; they assumed that records are shown in a row-wise manner. Their proposal works as follows: a) lists are split into columns according to candidate separators, which can be tags or punctuation symbols; b) some value features are then computed on each column and their siblings; c) then, an inference algorithm learns a probabilistic model from the previous features; d) the parameters are then used to find the best column assignment for a segment, which is the one that maximises the probability of the features observed given the model. Their evaluation was performed on the tables from a repository with approximately 283 tables from 12 web sites on book sellers, property taxes, white reports, and corrections. They also experimented with a constrain satisfaction approach that was less accurate.

Gatterbauer et al. [48] presented a proposal that requires to identify the spatial relationships between the individual cells of a table. It works as follows: a) it computes the boxes that represent the elements of an input document, taking into account their contents area, padding, border, and margin areas according to the CSS2 visual formatting model; b) it then overlaps a grid that helps identify each box by means of the co-ordinates of its upper-left corner and its lower-right corner; c) then, it aligns the boxes according to their horizontal and vertical projections; d) next, an adjacency relation is computed according to how distant the cells are; e) finally, some cells are selected and a recurrent expansion algorithm is invoked in an attempt to explore the adjacency relation to find their neighbours. The authors evaluated their proposal on the tables provided by a repository with 1 537 documents that were retrieved from search engines, from Wang and Hu's [127] repository, or written by the authors. The authors did not evaluate their procedure in isolation, but their complete system.

Elmeleegy et al. [38] tried to find columns by checking how homogeneous the cells in a table are. The homogeneity is analysed through the data types, the syntax, and/or delimiters. To perform this step, the authors used two sources, a large-scale language model, which helps know sentences that should not be split because they have previously occurred within a cell, and a corpus of tables, which helps identify data that appear in the same column in other tables. Their proposal works as follows: a) each row is split into a (possibly different) number of columns using two scoring functions, namely: a field quality score, which measures the quality of an individual column candidate, and a field-to-field consistency score, which measures the likelihood that two column candidates are actually the same column; b) then, it sets the number of columns to the most frequent one; c) padding columns are added to rows that have less columns than expected and some columns are merged otherwise; d) finally, the segmentation of cells is refined by checking the consistency amongst the cells in a per-column basis; e) if the consistency check fails, the procedure is re-launched. Their evaluation was performed on 20 tables from 20 different domains plus 100 additional tables that were randomly sampled from the Web.

Ling et al. [78] assumed that HTML tables can be segmented very easily searching for td tags; their key contribution was regarding how to segment context-data cells. Their proposal works as follows: a) it first uses a number of heuristics to generate candidate context-data cells, namely: tokens in between some punctuation marks, the longest common sub-sequences, pieces of text that can be wikified [102], and pieces of text that vary from document to document but are located at the same position; b) then, the previous context-data cells are added to the original table as additional columns; c) finally, a pairwise adaptation of the Multiple Sequence Alignment algorithm is used to segment the context-data cells. The evaluation was performed on 20 000 tables that were picked from a repository with 130 million tables from 10 different web sites.

Chu et al. [24] also focused on finding the columns of a data table. Their proposal works as follows: a) each row is tokenised using a set of user-defined delimiters; b) then candidate columns are generated using two approaches: a seed record is provided and the system discards segmentations that are very different; a custom pruning procedure that borrows some ideas from the well-known A* procedure is also used; c) it then measures the homogeneity of each column using lexical and semantic similarity functions that are averaged (the former computes the difference regarding the number of tokens, characters, and tokens in a number of user-defined categories; the latter computes the point-wise mutual data function); d) the process is repeated until a segmentation that maximises similarity is found. Their evaluation was performed on 100 million data tables that were transformed into lists; they used 20 additional tables encoded as lists from five different domains.

2.4.2 Comparison

Table 2.4 summarises our comparison regarding segmentation proposals. The task-specific characteristics are the following: a) *Spanned cells:* it describes if a proposal is able to identify cells that have been merged to create a

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters			Spanned cells	Multi-part cells	Context-data cells
						Structural	Visual	Value	Pre-defined	Learnable	User-defined			
Chen et al. 2000	Heuristics	1	N/A	N/A								Yes	No	Yes
Lerman et al. 2001	Clustering; probabilistic model.	2+	N/A	N/A				DataPro patterns				No	No	No
Cohen et al. 2002	Heuristics	1	N/A	N/A								Yes	No	No
Yang and Luk 2002	Heuristics	1	N/A	N/A								Yes	No	No
Lerman et al. 2004	Clustering	2+	N/A	N/A				Token types				No	No	No
Jung and Kwon 2006	Heuristics	1	N/A	N/A								Yes	No	No
Gatterbauer et al. 2007	Heuristics	1	N/A	N/A			Bounding boxes; padding; borders; margins.					Yes	No	No
Elmeleegy et al. 2011	Heuristics	1	F_1=0.92	Average 8 seconds per list	Language model; corpus with 154M tables.				Min TC support	Best-first search nodes	Maximum number of repetitions	No	No	No
Ling et al. 2013	Heuristics	2+	P=0.62 R=0.70 F_1=0.66	N/A	Wikipedia corpus						Same heuristic score; empty segment score.	No	No	Yes
Chu et al. 2015	Heuristics	1	**Wiki** P=0.89 R=0.93 F_1=0.90 **Other** P=0.89 R=0.93 F_1=0.90	Average 2 seconds per row					Refinement required; syntactic distance weight; semantic distance weight.		Tokenisation delimiters	Yes	No	No

Table 2.4: *Comparison of segmentation proposals.*

larger cell; a proposal that can identify spanned cells is better than a proposal that cannot. b) *Multi-part cells:* it describes if a proposal is able to identify multiple cells must be interpreted as a single value; a proposal that can identify multi-part cells is better than a proposal that cannot. c) *Context-data cells:* it describes if a proposal can identify context-data cells or not; a proposal that can identify context-data cells is better than a proposal that cannot.

Regarding the general characteristics, it is easy to realise that only the proposals by Lerman et al. [75, 76] have tried a machine-learning approach; the others build on heuristics that their authors have proven to work well in practice. Furthermore, most of them can work on as few as one input table. Unfortunately, roughly 70% of the authors did not report on the effectiveness of their proposals; roughly 30% reported on precision, recall, and the F_1 score. Only Elmeleegy et al. [38] and Chu et al. [24] reported on the efficiency of their approaches; their figures reveal that the algorithms behind the scenes might not be scalable enough. Regarding the resources required, only the proposals by Elmeleegy et al. [38] and Ling et al. [78] require the user to provide a few, but they do not seem to be difficult to find. Only the proposals by Lerman et al. [75, 76] and Gatterbauer et al. [48] require to project the input tables onto some simple features. Regarding the parameters, only the proposals by Elmeleegy et al. [38], Ling et al. [78], and Chu et al. [24] need the users to set a few.

Regarding the task-specific characteristics, it is surprising that many proposals do not make an attempt to analyse spanned cells and that none of them can identify multi-part cells, both of which are very common in practice. It is also surprising that only the proposals by Chen et al. [23] and Ling et al. [78] can identify context-data cells, which are also very common in practice.

2.5 Discrimination

Lerman et al. [75], Yoshida et al. [136], Lerman et al. [76], Elmeleegy et al. [38], Ling et al. [78], Braunschweig et al. [14], and Chu et al. [24] did not pay attention the discrimination task. The other proposals provide more sophisticated approaches.

2.5.1 Summary of proposals

Chen et al. [23] devised a proposal to discriminate tables by means of heuristics. It works as follows: a) a cell similarity measure is computed by combining string similarity, named-entity similarity, and number similarity functions; b) then, the tables whose cells do not exceed a threshold regarding the number of similar neighbour cells are discarded; c) finally, tables with less than two cells or tables with many links, forms, or figures, are also discarded. The evaluation was performed on 3 218 tables from their own repository with documents on airlines from the Chinese Yahoo! site.

Penn et al. [99] also devised a heuristic-based approach. Their proposal works as follows: a) tables that do not have multiple rows and columns are discarded; b) tables whose cells have more than one non-text-formatting tag are also discarded; c) finally, tables whose cells have more than a user-defined number of words are also discarded. The authors also mentioned that a desirable feature is to have syntactic and semantic homogeneity into account, but they did not go further regarding this idea. They experimented with an unspecified number of tables from their own repository with documents from 75 sites on news, television, radio, and companies.

Cohen et al. [25] devised a proposal that builds on machine learning a classifier. It works as follows: a) some structural and value features are computed from a learning set with tables that are pre-classified as either data tables or non-data tables; b) then, several classifiers are then machine-learnt and evaluated; c) the classifier that achieves the best effectiveness is selected to implement the discrimination task. The authors experimented with Multinomial Naive Bayes, Maximum Entropy, Winnow, and a decision tree learner that was based on C4.5; their conclusion was that the best results were achieved using Winnow. They evaluated their proposal using a 5-trial approach on 339 tables from their own repository; in each trial, 75% of the tables were used for learning and the remaining 25% for evaluation purposes.

Hurst [58] presented a similar machine-learning approach. The difference is that his proposal also takes visual features into account. He performed his evaluation on 89 data tables and 250 non-data tables from his own repository, which were randomly grouped into five sets from which 25% of the tables were selected for learning purposes and 75% for evaluation purposes. The results confirmed that Naive Bayes achieved the best results when the whole set of features was used, whereas Winnow worked better when only a subset of geometric features was used.

Wang and Hu [127] devised another machine-learning proposal. They computed structural and value features that they transformed into new features using Naive Bayes or weighted k-NN; the resulting features were fed into a custom decision tree learner. Note that value features rely on the words found in the input documents, which requires a large learning set so as to minimise the chances that a classifier is applied to a document with a word that was not in the learning set. The evaluation was performed using 9-fold cross evaluation on 11 477 tables from their own repository with documents from Google's directories.

Yang and Luk [134] reported on another heuristic-based method. Their proposal works as follows: a) tables that have th tags are considered data tables; b) tables that do not only contain links, forms, or images are also considered data tables; c) meta-data and data cells are then located using some user-defined patterns; d) tables that do not have both meta-data and data cells are discarded. They evaluated their method on 1 927 tables from their own repository, which was assembled with random documents from the Web.

Kim and Lee [67] used heuristics and an algorithm to check how homogeneous the cells are. Their proposal works as follows: a) tables are considered data tables if they contain caption or th tags and there are td tags at the right or the bottom sides; b) they are discarded if they have a single cell, if they have nested tables, or if they seem to have meta-data cells only; c) if they have too many links, images, or empty cells, then they are also discarded; d) then, it checks that the cells in the tables selected previously are consistent using a number of user-defined patterns; e) if the degree of homogeneity per row or column does not exceed a predefined threshold, then the corresponding table is discarded. The evaluation was performed on the 11 477 tables from Wang and Hu's [127] repository.

Jung and Kwon [64] presented a machine-learning proposal. It works as follows: a) it first removes empty rows and columns, splits spanned cells by duplicating their values, and discards tables with only one cell; b) then, it computes many structural, visual, and value features of the table to find out if it has meta-data cells, in which case the table is assumed to have data; c) finally, a C4.5 learner is fed with the input features and the classified tables. The evaluation was performed using 10-fold cross evaluation on 10 000 tables from their own repository plus some tables from Wang and Hu's [127] repository.

Gatterbauer et al. [48] reported on an approach that identifies tables using some rendering heuristics. Their proposal works as follows: a) elements with td, th, and div tags are considered candidate tables; b) it tries to identify frames that rely on those elements, which are assumed to be tables; c) overlapping tables are discarded; d) tables are also discarded if, after removing separator columns and rows, they have less than three rows, a single cell is more than 40% the total size of the table, or they contain cells with more than 20 words. The evaluation was performed on 493 tables from a repository that was assembled by students.

Okada and Miura [97] devised another machine-learning approach that requires to binarise discrete features before feeding them into an ID3 learner. The evaluation was performed using 10-fold cross evaluation on 100 data tables and 100 non-data tables from their own repository.

Cafarella et al. [17] proposed another machine-learning approach. Their proposal works as follows: a) it considers tables that have at least four cells, are not embedded in forms, and do not seem to be calendars; b) the tables that meet the previous criteria are classified as either data or non-data tables by a person; c) then, a statistical classifier is machine-learnt from a dataset that vectorises the previous tables using both structural and value features that are intended to measure how consistent the cells are. They evaluated their proposal using 5-fold cross evaluation over several thousand tables from their own repository.

Crestan and Pantel [31] also presented a machine-learning proposal. It works as follows: a) tables that have less than four cells or have cells with more than 100 characters are discarded; b) next, some structural and value features are computed; c) then, a Gradient Boosted Decision Tree classification model is machine-learnt. The evaluation was conducted on 5 000 tables from their own repository by performing 20-fold cross evaluation without overlapping.

Fumarola et al. [45] proposed a heuristic-based approach. Their proposal works as follows: a) it selects the elements whose bounding boxes are arranged in a grid; b) it then computes their similarity by comparing their DOM trees; c) next, the number of nodes in each list is computed; d) if the similarity is above a user-defined threshold and the difference in the number of nodes is below another user-defined threshold, then the tables are considered data tables. The evaluation was performed on 224 tables from Gatterbauer et al.'s [48] repository.

Lautert et al. [74] devised a machine-learning proposal that builds on neural networks. It work as follows: a) it computes some structural, visual, and value features; b) then, it uses them to machine-learn a perceptron with one hidden layer and resilient propagation; c) it has one output neuron per type of data table, which is encoded using a score in range $[0.00 \,..\, 1.00]$; the classification is performed in two steps, namely: the first one uses the 25 features to classify the tables into the corresponding types and the second step uses the previous 25 features plus the type of table output by the previous classifier. The evaluation was performed on a repository with 342 795 tables that were sampled from many different sites.

Son and Park [119] also tried a machine-learning approach. Their proposal works as follows: a) it selects every DOM node with tag table and their corresponding parents; b) the features described by Wang and Hu [127] are then computed to create a learning set; c) finally, an SVM classifier is machine-learnt using a structure kernel that works with structural features plus a linear kernel that works with value features; the structure kernel is based on two other kernels, one of which works on the table nodes and the other on the corresponding parent nodes. The authors performed 10-fold cross evaluation on a subset of 11 477 tables from Wang and Hu's [127] repository; roughly 89% of the tables were used for learning purposes and roughly 11% were used for evaluation purposes.

Eberius et al. [34] devised a proposal that builds on machine learning a classifier. It works as follows: a) some heuristics are applied to filter most non-data tables out, namely: tables with less than two rows or columns, tables with an invalid HTML structure, and tables that cannot be displayed correctly; b) some structural and value features are then computed regarding the tables and some of their sub-regions in order to compute local features; c) two alternatives are now tried: learning one classifier for every table type or using one classifier to discriminate between data and non-data tables and an additional classifier to classify some kinds of data tables; d) several classifiers are machine-learnt and evaluated, namely: CART, C4.5, SVM, and Random Forest; e) the classifier that achieves the best effectiveness is selected to implement the discrimination task. They evaluated their proposal on a repository with 24 654 tables from the October 2014 Common Crawl. According to their experience, the best results were achieved with Random Forest.

Wu et al. [131] provided a method to cluster tables that are similar according to their structure. Their proposal works as follows: a) for every two tables, it computes the set of paths that corresponds to caption, td, and th tags; b) then, the similarity between the paths of every two tables is computed according to a normalised similarity function; c) in a first stage, tables are clustered according to their local density plus the previous similarities; d) for each cluster, a second clustering is performed building on the paths that lead to elements with tags li, span, or div; e) finally, a so-called artificial judgment method is used to decide on the class of each cluster. The authors used a repository with 5 000 tables from the Wikipedia to evaluate their system, but no results were provided regarding this task.

Nishida et al. [96] devised a proposal that analyses a subset of cells at the top-left corner of a table using a deep neural network. It works as follows: a) for each td or th tag, an embedding is generated by tokenising words, tags, and row and column indexes; b) each token is encoded as a one-hot vector; c) an LSTM with an attention mechanism is then used to obtain a semantic representation of each cell; d) a convolutional neural network is then connected to three residual units and applied to vectorise the input table; e) finally, a classification layer is used. The authors learnt the network using 3 567 tables from 200 web sites, and evaluated the results on 60 678 tables from 300 web sites; the documents were selected from the April 2016 Common Crawl. They also experimented with an ensemble of five neural networks, which attained the best results.

Liao et al. [77] presented a heuristic-based approach that takes into account the existence of nested data tables. It works as follows: a) tables with a th or caption tag are considered data tables; b) tables with a large number of pictures, frames, forms, or script tags are discarded; c) tables with a small number of elements or many empty cells are discarded, too; d) tables with too many consistent values in their rows are considered incomplete data tables, which must be stitched to other sibling tables to create a complete data table. They evaluated their method on 226 tables from 50 different sites.

2.5.2 Comparison

Tables 2.5 and 2.6 summarise our comparison regarding discrimination proposals. The only task-specific characteristic is *Types of data tables*, which refers to the kinds of data tables that a proposal can discriminate; the more types can be discriminated, the better.

Regarding the general characteristics, it is easy to realise that 64% of the proposals use a machine-learning approach and 36% use heuristic-based approaches. The former require at least two tables to learn a predictor that implements the discrimination task, whereas the latter can generally work on a single table. Except for Wu et al.'s [130], the other authors report on effectiveness measures that are specific to this task; most of the authors selected precision, recall, and the F_1 score as effectiveness measures; the exceptions are Cohen et al. [25], Lautert et al. [74], and Nishida et al. [96], who report on the F_1 score only, Okada and Miura [97], who reported on accuracy, and Fumarola et al. [45], who reported on recall only. Apparently, the effectiveness of the machine-learning proposals seems very high compared to heuristic-based ones; however, due to the differences in the evaluation processes, this conclusion is not sound. Unfortunately, only Son and Park [119] and Eberius et al. [34] reported on the efficiency of their proposals, which does not seem to be very good according to their figures; Wu et al. [130] did not report on the efficiency of their proposal but they mentioned that it relies on a linear clustering

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters			Types of data tables
						Structural	Visual	Value	Pre-defined	Learnable	User-defined	
Chen et al. 2000	Heuristics	1	P=0.92 R=0.80 F_1=0.86	N/A					Thresholds to the number of links, forms, figures, and neighbour similar cells; string similarity threshold; named entity similarity threshold; number category similarity threshold.			
Penn et al. 2001	Heuristics	1	P=0.86 R=0.89 F_1=0.88	N/A							Average word length threshold	
Cohen et al. 2002	Winnow	2+	F_1=0.79-1.00	N/A		Number of rows; number of columns.		Ratio of columns with alphabetic content; ratio of simple cells.			Winnow threshold	
Hurst 2002	Naive Bayes; Winnow.	2+	**Naive Bayes** P=0.95 R=0.94 F_1=0.94 **Winnow** P=1.00 R=0.92 F_1=0.96	N/A		Number of tr tags; maximum number of th/td tags; bag-of-tags beneath the table tag.	Border; bag-of-attributes found for each tag below the table tag; number of rows or columns according to a geometric model	String content ratio; singular cell ratio.				
Wang and Hu 2002	Custom decision tree learner	2+	P=0.97 R=0.94 F_1=0.95	N/A		Average and standard deviation of the number of cells, cells per row and cells per column; average cumulative length consistency.		Histogram and consistency of each value type; word group frequency features.	Naive Bayes parameters; maximum impurity reduction; maximum depth of the tree; minimum number of samples used			
Yang and Luk 2002	Heuristics	1	P=0.94 R=1.00 F_1=0.97	N/A							Attribute-indicating entity patterns Value-indicating entity patterns	
Kim and Lee 2005	Heuristics	1	P=0.97 R=0.99 F_1=0.98	N/A					Table homogeneity threshold; syntactic homogeneity threshold.		Threshold to the number of link/image/empty cells; homogeneity patterns.	

Table 2.5: *Comparison of discrimination proposals (Part 1).*

Reference	Foundation	Tables required	Effectiveness	Efficiency	Res.	Features			Parameters			Types of data tables
						Structural	Visual	Value	Pre-defined	Learnable	User-defined	
Jung and Kwon 2006	C4.5	2+	P=0.94 R=0.96 F_1=0.95	N/A		Presence of a caption, th, or table tag; tag type; ratio of spanned cells; likelihood of headers; index based on the number of columns and rows; tag consistency.	Border	Presence of rows and columns with numerical data; ratio of empty, string-only, symbol-only, and digit-only cells; ratio of cells with images or links; number of sentences longer than 40 characters; average standard deviation of the cell length; consistency of content type.				
Gatterbauer et al. 2007	Heuristics	1	P=0.68 R=0.81 F_1=0.74	N/A					Maximum cell coverage; maximum word length.			
Okada and Miura 2007	ID3	2+	Acc=0.91	N/A		Presence of nested tables, images, spanned cells, or th tag; number of HTML tags and rows	Presence of visible border lines or caption tag; specification of width or height					
Cafarella et al. 2008	Statistic model	2+	P=0.41 R=0.81 F_1=0.54	N/A		Dimensions		Ratio of rows with mostly void cells and non-string data; average cell length and standard deviation; number of columns with basic types.				
Crestan and Pantel 2011	Gradient Boosted Decision Tree	2+	P=0.80 R=0.76 F_1=0.78	N/A		Maximum number of cells per row and column; ratio of cells with th, a, img, input, select, br, or formatting tags; ratio of colspans and rowspans.		Maximum cell length; average and variance of cell lengths; ratio of distinct tags, strings, cells with trailing colons, cells with digits, numerical cells, and non-empty cells.	Number of trees; minimum samples per node, best-first search nodes at classification.			Listings; forms; matrices; enumerations.
Fumarola et al. 2011	Heuristics	2+	R=0.79	N/A							Value similarity threshold; structure similarity threshold.	
Lautert et al. 2013	Neural networks	2+	F_1=0.22-0.95	N/A		Maximum number of cells per row and column; ratio of cells with th, a, img, input, select, br, or formatting tags; colspan vs. rowspan ratio; type of table.	Position of inner tables	Maximum cell length; average and variance length; ratio of distinct tags, trailing colon cells, cells with digits, numerical cells, non-empty cells; ratio of cells with commas and brackets; ratios of cells with unorederd and ordered lists.	Number of hidden layers.			Listings; forms; matrices.

Table 2.6: *Comparison of discrimination proposals (Part 2).*

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters			Types of data tables
						Structural	Visual	Value	Pre-defined	Learnable	User-defined	
Son and Park 2013	SVM	2+	P=0.98 R=0.98 F_1=0.98	**Learning** 20h **Evaluation** 2h (1100 tables)		Average and standard deviation of the number of cells, cells per row and cells per column; average cumulative length consistency.		Histogram and consistency of each value type; word group frequency features.	Parse tree kernel decay factor; structure kernel mixing ratio; composite kernel mixing ratio.			
Eberius et al. 2015	Random Forest	2+	**One classifier** P=0.84 R=0.82 F_1=0.82 **Two classifiers** P=0.83 R=0.83 F_1=0.83	10,238h (millions of tables)	Wikipedia	Maximum, average, and standard deviation of row, column, and cell lengths; ratio of images, forms, hyperlinks, and other cells; presence of th tags; local cell length average and variance; local ratio of span, th, a, img, input, select, font, br, ul, or ol tags.		Ratio of alphabetic, digit, and empty cells; local ratio of colons, commas, brackets, numeric characters, and non-empty cells.	Local regions			Listings; forms; matrices.
Wu et al. 2016	Clustering	2+	N/A	N/A		Similarity to other tables; local cluster density.					Cut-off distance	
Nishida et al. 2017	Deep neural network	2+	**Single** F_1=0.88 **Ensemble** F_1=0.91	N/A	Wikipedia				Token embedding size; LSTM output size; number of residual units; number of fully connected layers; number of weighted layers; CNN filters; SGD momentum; SGD minibatch size.			Listings; forms; matrices.
Liao et al. 2018	Heuristics	1	P=0.94 R=0.93 F_1=0.94	N/A					Empty cell threshold		Layout tag threshold; consistency threshold.	Complete; incomplete.

Table 2.7: *Comparison of discrimination proposals (Part 3).*

algorithm. The only proposals that require resources are the ones by Eberius et al. [34] and Nishida et al. [96]; fortunately, they do not seem to be a major obstacle since they consists of a corpus that was gathered from the Wikipedia. The ones that rely on machine learning methods need to project the input data onto a space of structural, visual, and/or value features that seems simple to compute. Regarding their parameters, most of them have predefined parameters for which the authors recommend some values that they have found out to work well; none of the proposals require any learnable parameters, but a few require to provide some user-defined parameters.

Regarding the task-specific characteristics, the only proposals that can sub-classify data tables are the following ones: Crestan and Pantel [31] distinguish amongst listings, forms, matrices, and enumerations; Lautert et al. [74], Eberius et al. [34], and Nishida et al. [96] distinguish amongst listings, forms, and matrices; and Liao et al. [77] distinguishes between complete and incomplete tables (which are encoded as independent tables, but must be stitched together so that they can be properly interpreted).

2.6 Functional analysis

Lerman et al. [75], Penn et al. [99], Cohen et al. [25], Hurst [58], Wang and Hu [127], Lerman et al. [76], Okada and Miura [97], Crestan and Pantel [31], Elmeleegy et al. [38], Fumarola et al. [45], Lautert et al. [74], Son and Park [119], Chu et al. [24], Eberius et al. [34], Nishida et al. [96], and Liao et al. [77] did not report on any proposals to implement the functional analysis task. Gatterbauer et al. [48] presented a naive approach that matches the structure of a table to a number of pre-defined structures in which it is relatively easy to set the meta-data cells apart from the data cells. Ling et al. [78] and Wu et al. [131] assumed that meta-data cells can be easily located by searching for th tags. Braunschweig et al. [14] also presented a naive solution since they assumed that meta-data cells are located on the first row. The other proposals provide more sophisticated approaches.

2.6.1 Summary of proposals

Chen et al. [23] devised a proposal that is based on computing row/column similarities. It works as follows: a) if there is only one row or column, then the first cell in that row or column is considered to be a meta-data cell; b) otherwise, the last row is compared to the others; if at least half the rows are similar and the similarity of the first and the last rows is smaller than the average, then the top-most row is assumed to be composed of meta-data cells; c) otherwise, the last column is compared to the others; if at least half the columns are similar and the similarity of the first and the last columns is smaller than the average, then the left-most column is assumed to be composed of meta-data cells; otherwise, the first row is considered to be composed of meta-data cells. The evaluation was not performed on this task, but on their whole system.

Yoshida et al. [136] suggested using ontologies. Their proposal works as follows: a) for each cell in a table, it computes the ratio of times that its value is recorded in the ontology; b) these ratios are then used to feed the Expectation-Maximisation algorithm in order to learn a classifier that makes a few sub-types of listings apart; c) once the exact type of listing is clear, identifying meta-data cells is relatively easy and the rest of cells are assumed to be data cells. (Note that the authors assume that the input tables are data tables, which is the reason why this cannot be considered a discrimination proposal.) They evaluated their proposal on 175 tables that were randomly sampled from a repository with 35 232 tables.

Yang and Luk [134] applied some heuristics to differentiate rows with meta-data cells from rows with data cells. Their proposal works as follows: a) a row is considered to have meta-data cells if it has at most 50% the average number of cells per row, if it contains no structured cells, or if the visual features are different from the visual features of the others rows; b) then, it tries to detect if the input table is a listing or a matrix; c) once the table structure is identified, it is easy to identify the meta-data. (Note that the authors assume that the input tables are data tables, which is the reason why this cannot be considered a discrimination proposal.) The authors did not report on their experimental results regarding this task, but their whole system.

Kim and Lee [67] devised a proposal that first attempts to classify the input table. It works as follows: a) in the case of tables with one single row/column, the first cell is considered to be a meta-data cell and the rest are considered to be data cells; b) in the case of tables with two rows and two columns that do not have any spanned cells, the first row/column is assumed to have meta-data cells and the second row/column is assumed to have data cells; c) tables with two rows/columns and three or more columns/rows whose upper-left cell spans a whole row/column are discarded altogether; otherwise if the first row/colum has some spanned cells (but not all), then the first column/row is assumed to have meta-data cells and the others are assumed to have data cells; d) otherwise, the homogeneity of the cells is checked per rows and columns using the following functions: a lexical similarity function that focuses on the data types and the length of the values, and a semantic similarity function that builds on some user-provided key words and patterns. The authors did not provide any experimental results regarding this task.

Jung and Kwon [64] proposed a heuristic-based technique to locate the meta-data within the tables. Their proposal works as follows: a) cells with a `th` tag are assumed to have meta-data; b) if the table can be partitioned into two blocks with the same background colour or font, then the top and/or the left blocks are assumed to contain meta-data; c) if the cells in a row or column have some user-defined values or match some user-defined patterns, then they are also considered to contain meta-data; d) spanned cells that are embedded in `td` tags are also assumed to have meta-data as long as they are located at the top-left areas of the table; its adjacent cells are also considered to have meta-data; e) if the top-right cell is empty, then it is likely that the cells in the first row or column have meta-data; f) a probability is finally computed for every cell building on the previous heuristics and the cells whose probability exceeds a threshold are then considered to be meta-data cells whereas the others are assumed to be data cells. The evaluation was performed using 10-fold cross evaluation on 10 000 tables from their own repository plus some tables from Wang and Hu's [127] repository.

Cafarella et al. [17] devised a machine-learning proposal. It works as follows: a) a learning set is assembled with data tables in which the cells are classified as either meta-data or data cells; b) in cases in which a table does not have any meta-data cells, synthetic cells are created and the meta-data is fed from a separate database with similar tables; c) some structural and value features are computed for each cell; d) a classifier is machine-learnt from the previous features; e) the results of the classifier are used to enrich the other database. The authors evaluated their proposal by means of 5-fold cross evaluation on a repository with 1 000 tables that were gathered from the Web.

2.6.2 Comparison

Table 2.8 summarises our comparison regarding functional analysis proposals. The only task-specific characteristic is *Decorators*, which refers to the ability of a proposal to identify decorator cells; a proposal that can find decorators is better than a proposal that cannot.

Regarding the general characteristics, 80% of the proposals rely on heuristics and 20% rely on machine-learning approaches. Most of them can work on as few as a single table. Many of the authors report on the effectiveness of

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters			Decorators
						Structural	Visual	Value	Pre-defined	Learnable	User-defined	
Chen et al. 2000	Heuristics	1	N/A	N/A							Minimum required similarity	No
Yoshida et al. 2001	Expectation maximisation	1	P=0.79 R=0.85 Acc=0.78	N/A	Domain-specific ontology				EM θ parameter (auto adjusted)			No
Yang and Luk 2002	Heuristics	1	N/A	N/A					Cell count threshold			No
Kim and Lee 2005	Heuristics	1	N/A	N/A					Cell homogeneity threshold		Key words; lexical patterns.	No
Jung and Kwon 2006	Heuristics	1	P=0.86 R=0.88 F_1=0.87	N/A							Meta-data text and patterns; similarity threshold.	No
Gatterbauer et al. 2007	Heuristics	1	P=0.79 R=0.85 Acc=0.78	N/A	Domain-specific ontology							No
Cafarella et al. 2008	Detect classifier; reference matching.	2+	**Tables with headers** P=0.89 R=0.85 **Tables without headers** P=0.75 R=0.80	N/A	ACSDb database	Dimensions		Ratio of columns with lowercase, puntuation, or non-string data in the first row; ratio of columns with non-string data in body; ratio of columns with length smaller than the first row.				No
Ling et al. 2013	Heuristics	2+	N/A	N/A							Semantic equivalence threshold	No
Braunschweig et al. 2015	Heuristics	2+	N/A	N/A								No
Wu et al. 2016	Heuristics	1	N/A	N/A								No

Table 2.8: *Comparison of functional analysis proposals.*

their proposals, which is not as good as was the case with the previous tasks. Unfortunately, no result on their efficiency is available. Regarding the resources required, Yoshida et al.'s [136] and Gatterbauer et al.'s [48] proposals require domain-specific ontologies, whereas Cafarella et al.'s [18] requires a publicly-available database. The proposal by Cafarella et al. [18] is the only that projects the input data onto a space of simple structural and value features. The proposal by Yoshida et al. [136] requires a pre-defined parameter that is auto-adjusted, and the proposals by Yang and Luk [134] and Kim and Lee [67] require another pre-defined parameter for which the authors provide a default value; the only proposals that require user-defined parameters are the ones by Chen et al. [23], Kim and Lee [67], Jung and Kwon [64], and Ling et al. [78].

Regarding the task-specific characteristics, note that none of the proposals that we have surveyed seem to be able to identify decorator cells, which are very common in many tables, chiefly due to the abundance of separators that contain advertisements in many popular sites.

2.7 Structural analysis

Chen et al. [23], Penn et al. [99], Hurst [58], Wang and Hu [127], Kim and Lee [67], Jung and Kwon [64], Gatterbauer et al. [48], Okada and Miura [97], Cafarella et al. [17], Crestan and Pantel [31], Lautert et al. [74], Son and Park [119], Eberius et al. [34], Nishida et al. [96], and Liao et al. [77] did not report on any proposals to implement the structural analysis task. Yoshida et al. [136] presented a naive proposal that classifies tables in a number of categories, which makes identifying the records quite a trivial task. Elmeleegy et al. [38] also assumed that the records within tables that are encoded as lists are always laid out row-wise. Ling et al. [78] and Braunschweig et al. [14] assumed that records are displayed row-wise or column-wise depending on the number of meta-data or data cells found in the first few rows or columns. Chu et al. [24] also presented a naive approach that assumes that the records within tables that are encoded as lists are always laid out row-wise. Wu et al. [131] presented an additional naive approach since they just identify records in horizontal listings. The other proposals provide more sophisticated approaches.

2.7.1 Summary of proposals

Lerman et al. [75] used a couple of algorithms to detect row-wise records. Their proposal works as follows: a) first, it uses DataPro to find the patterns that describe the data in each column; b) such patterns can be interpreted as tags that allow to transform a table into a sequence of symbols; c) then, a version of ALERGIA is used to infer a finite automaton from those sequences; d) the automaton is then transformed into a regular expression; e) finally, it identifies repeating sub-patterns that correspond to the records in the original table. No experimentation was performed regarding this task.

Cohen et al. [25] presented a proposal that relies on four so-called builders, namely: a builder focuses on meta-data cells that cut in on the table, one that focuses on columns of headers, another that focuses on rows of headers, and an additional one that takes the tag paths into account. The builders are fed into a FOIL-based inductive logic programming system in order to learn a classification rule that allows to identify both horizontal and vertical records. No experimental results were reported regarding this specific task, but their whole system.

Yang and Luk [134] presented a proposal that specialises in numerical tables. It works as follows: a) first, it removes the headers of the input table; b) then, it checks whether the records seem to be one-dimensional or two-dimensional using some heuristics; c) the type of cells is analysed using pre-defined patterns in order to label numeric data cells; d) given the types of cells and the dimensionality of the records, their proposal tries to match a number of pre-defined patterns that help identify the records. The evaluation was performed on 169 one-dimensional and 50 two-dimensional tables.

Lerman et al. [76] devised two proposals to identify records, namely: a constraint solving technique and a probabilistic technique. The first proposal works as follows: a) it models the cells in the tables using Boolean variables; b) it then adds constraints on those variables that ensure that each cell belongs to a single record, only contiguous cells can be assigned to the same record, and two cells cannot be at the same position in the same table; c) then a constraint solver is used to find a solution to the constraints. The second proposal works as follows: a) it uses a set of observable variables that model the types of tokens in the data cells, as well as a set of hidden variables that provide the record number or the column number of every cell; b) a probabilistic model is then learnt by assuming a number of dependencies between token types, cells, columns, neighbour columns, format, or record numbers; c) finally, the values of the hidden variables are inferred building on the probabilistic model. Their evaluation was performed on the tables from their own repository, which were gathered from 12 web sites on book sellers, property taxes, white reports, and corrections.

Fumarola et al. [45] presented a proposal that was described very shallowly. It seems to work on so-called candidate lists, which are sets of cells that correspond to different columns and form a single record; each candidate list is a sub-tree of the DOM tree and they all are required to satisfy some structural similarity constraints, including a minimum size in terms of nodes. The evaluation was performed on 224 tables from Gatterbauer et al.'s [48] repository.

2.7.2 Comparison

Table 2.9 summarises our comparison regarding structural analysis proposals. The task-specific characteristics are the following: a) *Header structure:* it describes the kinds of headers that a proposal can identify according to their structure, namely: none, which means that it can analyse tables without headers, simple, which means that it can analyse simple headers that consists of one meta-data cell only, and complex, which means that it can identify complex headers that consists of multiple meta-data cells; the more header structures a proposal can identify, the better. b) *Header layout:* it describes the kinds of headers that a proposal can identify according to how they are laid out, namely: none, which means that it can identify that a table does not have any headers, single, which means that it can identify headers in the first rows and/or columns of a table, horizontally repeated, which means that it can identify that the headers are repeated every some rows, vertically repeated, which means that it can identify that the headers are repeated every some columns, and split, which means that it can identify series of headers that are split across several non-adjacent rows or columns; the more header layouts a proposal can identify, the better. c) *Record dimensionality:* it describes the dimensionality of the records that a proposal can identify, namely: 0 if it can identify the records in an enumeration, 1 it can identify the records in a listing or a form, and 2 if it can identify the records in a matrix; the more record dimensionalities a proposal can identify, the better. d) *Record multiplicity:* it describes the number of records that a table is intended to show, namely: 1 in the case of forms and matrices, and * in the case of listings and enumerations; the more record multiplicities a proposal can identify, the better. e) *Record orientation:* it describes the orientations that it can identify, namely: none in the case of matrices and enumerations, horizontal or vertical in the case of listings and forms; the more record orientations a proposal can identify, the better. f) *Separators:* it describes whether a proposal can identify separator rows and/or columns; a proposal that can identify separators is better than a proposal that cannot.

Regarding the general characteristics, many proposals rely on heuristic-based approaches; the exceptions are the proposals by Lerman et al. [75, 76],

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters			Header structure	Header layout	Record dimensionality	Record multiplicity	Record orientation	Separators
						Structural	Visual	Value	Pre-defined	Learnable	User-defined						
Lerman et al. 2001	DataPro Grammar induction; optimisation.	2+	N/A	N/A								None	None	1	*	Horizontal	No
Yoshida et al. 2001	Heuristics	2+	N/A	N/A								Simple	None; single; split.	1	*	Horizontal; vertical.	No
Cohen et al. 2001	Inductive logic programming	1	N/A	N/A								Simple	Single	1	*	Horizontal; vertical.	No
Yang and Luk 2002	Heuristics	1	**1 dim.** P=0.92 R=0.96 F_1=0.94 **2 dim.** P=0.90 R=0.98 F_1=0.94	N/A					Average cell count threshold		Cell type patterns	None	None	1, 2	*	Horizontal; vertical.	No
Lerman et al. 2004	Constraint Satisfaction; Hidden Markov Models.	2+	N/A	N/A								None	None	1	*	Horizontal	No
Elmeleegy et al. 2011	Heuristics	1	P=0.64 R=0.63 F_1=0.63	N/A								Simple	Single	1	*	Horizontal	No
Fumarola et al. 2011	Heuristics	2	N/A	N/A							Minimum nodes threshold	None	None	0	*	None	No
Ling et al. 2013	Heuristics	1	N/A	N/A								Simple	Single	1	*	Horizontal	No
Braunschweig et al. 2015	Heuristics	1	N/A	N/A								Simple	Single	1	*	Horizontal	No
Chu et al. 2015	Heuristics	1	N/A	N/A								None	None	1	*	Horizontal	No
Wu et al. 2016	Heuristics	1	N/A	N/A								Simple	Single	1	*	Horizontal	No

Table 2.9: *Comparison of structural analysis proposals.*

which leverage some grammar induction techniques, and Cohen et al.'s [25] proposal, which leverages inductive logic programming. Most of the proposals require as few as one input table. Unfortunately, only Yang and Luk [134] and Elmeleegy et al. [38] reported on the effectiveness of their proposals, and none of the authors reported on their efficiency. Note that none of the proposals require to project the input data onto a space of features and that only Yang and Luk's [134] and Fumarola et al.'s [45] proposals require some parameters to be set.

Regarding the task-specific characteristics, it is surprising that all of the proposals assume that the tables do not have any headers or they are simple; it is also surprising that the only proposal that can identify single and split headers is the one by Yoshida et al. [136]. Regarding the record dimensionality, only the proposal by Yang and Luk [134] can make uni-dimensional records apart from two-dimensional records; the proposal by Fumarola et al. [45] implicitly assumes that the records in a table are zero-dimensional and does not make an attempt to analyse the structure of the corresponding cells; the other proposals implicitly assume that the records are uni-dimensional. Regarding the record multiplicity, it is interesting to see that all of the proposals assume that tables may display more than one record; simply put, they cannot make listings apart from forms. Regarding the record orientation, most proposals implicitly assume that the records are oriented horizontally; the only exceptions are the proposals by Yoshida et al. [136], Cohen et al. [25], and Yang and Luk [134], which can make horizontal records apart from vertical records. It is surprising that none of the proposals that we have surveyed can identify separators, even though they are very common in practice.

2.8 Interpretation

Lerman et al. [75], Penn et al. [99], Yoshida et al. [136], Cohen et al. [25], Hurst [58], Wang and Hu [127], Lerman et al. [76], Kim and Lee [67], Jung and Kwon [64], Gatterbauer et al. [48], Okada and Miura [97], Crestan and Pantel [31], Elmeleegy et al. [38], Fumarola et al. [45], Lautert et al. [74], Ling et al. [78], Son and Park [119], Braunschweig et al. [14], Chu et al. [24], Eberius et al. [34], Nishida et al. [96], and Liao et al. [77] did not report on this task. The other proposals provide more sophisticated approaches.

2.8.1 Summary of proposals

Chen et al. [23] presented an atypical proposal whose goal is to interpret tables as bags of key-value pairs, without making an attempt to identify records. It works as follows: a) if there is only one header row/column, then the data cells in the remaining rows/columns are assigned simple descriptors on a per row/column basis; b) if there are both header rows and columns, then the table is assumed to be a matrix and the data are assigned field descriptors that merge the meta-data in the corresponding column and row header. The authors did not evaluate their proposal.

Yang and Luk [134] proposed a procedure that is similar to Chen et al.'s [23] procedure, but takes multiple header rows or columns into account, in which case the cells are simply merged to create field descriptors, as well as cells that contain both meta-data and data, in which case the meta-data are transformed into simple descriptors. The authors did not report on the evaluation of this task.

Cafarella et al. [17] followed Chen et al.'s [23] procedure, too, but they went a step forward in cases in which a table does not provide any meta-data cells. In such cases, they collect the data on a per-column basis and attempt to find the most similar data in the ACSDb database, which is a resource that has many data with correct descriptors. The authors did not report on the evaluation of this task.

Wu et al. [131] suggested to use an ad-hoc interpretation method depending on the structure of the table identified in the discrimination task. They only reported on a method to extract data from horizontal listings with headers using some heuristics that are related to how the th and the td tags encode a subject-predicate-object relation. They conducted their experimentation on a repository with 100 horizontal listings from Wikipedia. The authors did not report on the evaluation of this task.

2.8.2 Comparison

Table 2.10 summarises our comparison regarding interpretation proposals. The task-specific characteristics are the following: a) *Descriptors:* it reports on the kind of descriptors that a proposal can assign to the data in a table; the more kinds of descriptors a proposal can generate, the better. b) *Empty values:* it refers to the ability of a proposal to make a difference between empty cells whose values are factorised and cells that are actually empty; a proposal that can make a difference between factorised and void cells is better than another proposal that cannot. c) *Value structure:* it refers to the ability of a proposal to make a difference between cells whose value is an atomic piece of data and cells whose values are structured and can be decomposed further; a proposal that can make a difference between cells with an atomic value and cells with a structured value is better than a proposal that cannot.

Reference	Foundation	Tables required	Effectiveness	Efficiency	Resources	Features			Parameters				Empty values	Value structure
						Structural	Visual	Value	Pre-defined	Learnable	User-defined	Descriptors		
Chen et al. 2000	Heuristics	1	N/A	N/A								Simple, Field	No	No
Yang and Luk 2002	Heuristics	1	N/A	N/A								Simple, Field	No	Yes
Cafarella et al. 2008	Reference matching	1	N/A	N/A	ACSDb database							Simple	No	No
Wu et al. 2016	Heuristics	1	P=0.66	N/A								Simple	No	No

Table 2.10: *Comparison of interpretation proposals.*

Regarding the general characteristics, most proposals rely on heuristics that have proven to work well in practice; the only exception is the proposal by Cafarella et al. [18], which uses a reference matching approach. Wu et al. [130] were the only authors who reported on effectiveness, but they measured precision only; unfortunately, none of the proposals report on efficiency. Cafarella et al.'s [18] proposal is the only one that requires a publicly-available resource. None of the proposals project the input data onto a feature space and none of them require any parameters to be set.

Regarding the task-specific characteristics, note that Chen et al.'s [23] and Yang and Luk's [134] proposals are the only that can generate field descriptors by analysing headers that are composed of several cells; the remaining can only generate simple descriptors that consists of the value of a single cell. Unfortunately, none of the proposals can make a difference between factorised cells and void cells. Regarding making a difference amongst atomic and structured cells, it seems that only the proposal by Yang and Luk [134] can deal with this interpretation problem.

2.9 Summary

The Web provides many data in user-friendly tabular formats that are encoded using HTML. Data extractors are intended to extract those data as datasets that can feed business applications. There exist many proposals to implement them, which has motivated several previous surveys. Unfortunately, they are outdated and we do not think that it suffices to update them because they do not provide a good conceptual framework, they do not provide a taxonomy of tables, they do not analyse the exact tasks involved, and they do not provide a good comparison framework. In this chapter, we have presented a review of the literature that does not have any of the previous problems, which we hope will be useful to both researchers and practitioners.

Chapter 3

TOMATE: the HTML table extractor

This chapter describes TOMATE, which is a proposal to extract data from HTML tables with no supervision. It is organised as follows: Section 3.1 introduces our proposal; Section 3.2 presents some preliminaries; Section 3.3 describes the location task; Section 3.4 describes the segmentation task; Section 3.5 describes the discrimination task; Section 3.6 describes the functional analysis task; Section 3.7 describes the structural analysis task; Section 3.8 describes the interpretation task; Section 3.9 presents a case study; finally, Section 3.10 summarises our conclusions.

3.1 Introduction

We are interested in datasets that are encoded using HTML tables, with an emphasis on tables that are generated by a person instead of a machine. Very often, the data in those tables cannot be found in any knowledge bases, which makes it difficult to use them in automated business processes.

Data extractors are software components that analyse web documents and output their data as records that facilitate integrating them into common knowledge bases. There are many proposals to implement data extractors: some of them are general-purpose, since they do not expect the data to be rendered using any specific layouts [22, 43, 111, 123]; some others focus on data that are laid out in tables [27, 39, 58, 82, 107, 138, 139]. The table-specific proposals implement a pipeline that is commonly known as table understanding. It consists of the following tasks [107]: locating the tables in the input documents, segmenting them into cells, discriminating the ones that provide data, analysing the functions of their cells, analysing their structure, and, finally, interpreting the data and outputting the corresponding records.

We have analysed many of the table-understanding proposals in the literature [27, 39, 58, 82, 107, 138, 139]. Our focus is on the proposals that provide a solution to analyse the functions of the cells [17, 18, 23, 24, 38, 40, 64, 67, 87, 96, 134, 136], since the other tasks have been extensively studied in the literature and there are many alternatives to implement them. Unfortunately, only a few can deal with arbitrary table layouts (horizontal listings, vertical listings, or matrices), they have problems to deal with the many different formats used to display the data (multi-line headers, no headers at all, context cells, repeated headers, or factorised cells), and they also have problems with typical encoding problems (inconsistent row lengths or incorrect uses of tags td and th) [107]. Simply put: the previous proposals cannot deal well with 64.41% of the tables in our experimental datasets.

In this chapter, we present TOMATE[†1], which is a new proposal to extract data from tables that are encoded using HTML. Our key contribution is regarding functional analysis; we have devised a novel approach that relies on clustering the cells using a variety of features and then applying some

[†1]TOMATE is a recursive acronym that stands for "TOMATE: Online Multi-document Automated Table Extraction". Its implementation and instructions on how to install and run it are available at https://pypi.org/project/tablextract. The experimental datasets are available at http://datasets.tdg-seville.info and the tool to assemble them is available at http://tomatera.tdg-seville.info.

heuristics to correct the results. Our proposal can deal with any table layouts and the problems that we mentioned before regarding the formats used to display the data and how they are encoded. Our experimental analysis reveals that it can attain an F_1 score of 89.50% and takes an average of 0.09 CPU seconds to process each table. That F_1 score is 24.11% better than the best proposal that does not require supervision and 5.68% better than the F_1 score attained by the best of the supervised proposals; the average time to process each table is slightly worse, but good enough for practical purposes and clearly compensates for the improvement regarding the F_1 score. The difference in effectiveness was proven to be statistically significant at the standard confidence level. To illustrate its practical value, we present a case study in which we describe how TOMATE is being used to feed a real-world IARPA geopolitical forecasting system with data that are extracted from Wikipedia tables. Most such tables are not generated using Wikipedia's pre-defined templates, which means that their data cannot be extracted and integrated by the data extractors that are currently deployed at major knowledge bases.

3.2 Preliminaries

In this section, we introduce the mathematical notation used in this article and define the main concepts behind our proposal.

Definition 3.1 *Mathematical notation We represent a vector* m *as a map of the form* $\{k_i : v_i\}_{i=1}^{p}$, *where each* k_i *is a key and each* v_i *is a value* ($p \geq 1$). *We denote its domain as* $\operatorname{dom} m = \{k_i\}_{i=1}^{p}$ *and its range as* $\operatorname{ran} m = \{v_i\}_{i=1}^{p}$. *Given key* k, *we denote its value in vector* m *as* $m[k]$. *Given a real number* z *and vectors* m, m_1, *and* m_2, *we define the following operations:*

$$\begin{aligned}
m + z &= \{k_i : m[k_i] + z \mid k_i \in \operatorname{dom} m\} \\
m - z &= \{k_i : m[k_i] - z \mid k_i \in \operatorname{dom} m\} \\
z\, m &= \{k_i : z\, m[k_i] \mid k_i \in \operatorname{dom} m\} \\
{}^{m}\!/_{z} &= \left\{k_i : {}^{m[k_i]}\!/_{z} \mid k_i \in \operatorname{dom} m\right\} \\
m_1 + m_2 &= \{k_i : m_1[k_i] + m_2[k_i] \mid k_i \in \operatorname{dom} m_1 \cap \operatorname{dom} m_2\} \\
m_1 - m_2 &= \{k_i : m_1[k_i] - m_2[k_i] \mid k_i \in \operatorname{dom} m_1 \cap \operatorname{dom} m_2\} \\
\operatorname{dist}(m_1, m_2) &= \sqrt{\sum_{k \in \operatorname{dom} m_1 \cap \operatorname{dom} m_2} (m_1[k] - m_2[k])^2}
\end{aligned}$$

A matrix M *is a map of the form* $\{(i,j) : m_{i,j}\}_{i=1,j=1}^{\alpha,\beta}$ ($\alpha \geq 1, \beta \geq 1$). *Given two indices* $1 \leq i \leq \alpha$ *and* $1 \leq j \leq \beta$, *we denote the component of matrix* M *at position* (i,j) *as* $M[i,j]$.

Category	Attributes	Components	Range
Style	Font-color, background-color, border-color, outline-color	R, G, B	[0, 255]
	Padding, margin, border-weight	Top, bottom, left, right	[0, ∞]
	Font-size, font-weight		[0, ∞]
	Display, text-align, vertical-align, font-family, text-decoration, text-transform		Categorical
Structural	Tag		Categorical
	Number of children		[0, ∞]
	Rowspan, row index		[1, ∞]
	Colspan, column index		[1, ∞]
	Is node repeated?		Boolean
Lexical	Number of alphanumeric, digit, lowercase, uppercase, symbol, and whitespace characters		[0, ∞]
	Number of tokens		[0, ∞]
	Number of stopwords		[0, ∞]
	Is first/last character type alphanumeric, digit, lowercase, uppercase, symbol, or whitespace?		Boolean
	Is the content capitalised, all capitals, an amount, a range, a date, money, or empty?		Boolean
Miscellaneous	Number of nouns, verbs, adjectives, adverbs, and other parts-of-speech		[0, ∞]
	Likelihood of being meta-data		[0, 1]

Table 3.1: *Taxonomy of DOM node attributes.*

Definition 3.2 *Documents A document is a text file whose contents are encoded using the HTML mark-up language, which allows to represent it using a DOM tree. Given a node* a *in a DOM tree, we denote its sequence of children as* $\mathtt{childen}(\mathsf{a})$*, the area of its bounding box as* $\mathtt{area}(\mathsf{a})$*, and its attribute vector as* $\mathtt{attr}(\mathsf{a})$*. Table 3.1 presents the attributes that we take into account, namely: style attributes, which are related to their display properties, structural attributes, which are related to their DOM-tree properties, lexical attributes, which are related to properties of their textual contents, and miscellaneous attributes, which are related to some syntax and statistical properties.*

Definition 3.3 *Tables and cells A table is a grid that can be used to display data or to position other elements on the screen. They are represented as*

DOM sub-trees with a table root tag and their cells are represented as DOM sub-trees with either td or th root tags. We are interested in tables that are used to display data, which are commonly referred to as data tables. There are three common layouts for data tables, namely: horizontal listings, which are row-wise tables, vertical listings, which are column-wise tables, and matrices, which are table-wise tables. According to the estimates by Crestan and Pantel [31], the previous table layouts amount to 86.30% *of the data tables in the Web; the remaining* 13.70% *of the tables form a long tail in which there are a variety of layouts whose individual frequencies are negligible; it is also arguable whether some such layouts are actually data tables, positioning grids, or other types of diagrams [39]. The cells can be either meta-data cells, which provide semantic hints to understand the meaning of the data, or data cells, which provide the data themselves*[†2]*. A group of related meta-data cells is a header and a group of related data cells is a tuple.*

Definition 3.4 *Features A feature is a numeric characteristic of a cell that is computed from the attributes of its DOM node, cf. Table 3.1. They can be either base features, which are computed directly from the attributes, or deviation features, which measure how different the base features of a cell are from the features of the other cells in the same row, column, or table. Hereinafter, we use κ to denote the number of base features; that is, there are a total of 4κ features per cell if we take the deviation features into account.*

Definition 3.5 *Record sets A record set is a collection of records, which are vectors of the form $\{d_i : v_i\}_{i=1}^r$, where each d_i is a descriptor and each v_i is a value ($r \geq 1$). The descriptors are computed from the meta-data cells, some of which may be generated artificially in cases in which the original table does not provide them; the values are computed from the data cells. Simply put: a record highlights the relational nature of the data by making it explicit the relation between the components of a tuple and their corresponding headers.*

Definition 3.6 *Table understanding Table understanding is the set of tasks involved in the process of obtaining a record set given a document that contains tables. It encompasses the following tasks with no specific order: a) location, which identifies the excerpts of the input documents where the tables are located; b) discrimination, which discards the tables that do not seem*

[†2]In theory, tag th must be used to encode meta-data cells and tag td must be used to encode data cells. Unfortunately, 34.96% of the tables in our experimental datasets encode meta-data cells with td tags and 11.74% of them encode data cells with th tags. This clearly motivates the need for a method to analyse the function of the cells independently from how they are encoded.

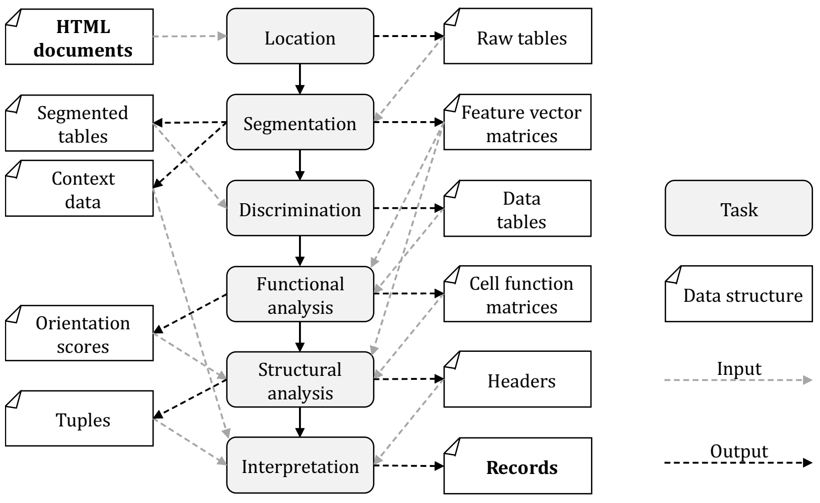


Figure 3.1: *Pipeline and data flow of TOMATE.*

to provide any useful data; c) segmentation, which transforms the excerpts into grids of cells; d) functional analysis, which identifies the functions of the cells, that is, whether they provide data or meta-data; e) structural analysis, which identifies groups of related meta-data cells (the headers) and groups of related data cells (the tuples); and f) interpretation, which transforms the input tables into record sets in which the components of the tuples are mapped onto their corresponding headers. TOMATE implements all of these tasks in the order described in Figure 3.1.

Definition 3.7 *Variation point A variation point is a stage of a proposal with multiple interchangeable alternatives in which none of them stands out from a conceptual point of view. Variation points require some experimentation to decide the alternative to be used in the implementation. A configuration maps every variation point of a proposal onto a specific alternative. After the implementation of a proposal, a search algorithm is performed to find the best configuration. Two notable examples of those algorithms are grid search, which tests every possible configuration; and sequential search, which starts using a default configuration, and then adjusts every variation point individually while updating the default configuration.*

3.3 Location

This task works on the input HTML documents and returns their raw HTML tables, if any. It performs the following steps:

i. The input documents are downloaded using a headless browser, which injects a script that decorates each HTML element with its attributes before saving them, cf. Table 3.1.

ii. The documents are then transformed into their corresponding DOM trees using a standard HTML parser.

iii. Every sub-tree whose root node has tag `table` is extracted using an appropriate CSS selector.

These steps can be implemented very straightforwardly using commodity technology, so we do not provide any additional details.

3.4 Segmentation

This task works on the tables returned by the previous task; for each of them, it returns a new table in which every row has the same number of columns and empty and factorised cells have been processed; it also returns a matrix with the feature vectors of each cell and the context data identified surrounding the table. This task performs the following steps:

i. The table is applied several pre-processing operations.

ii. The base features are computed.

iii. The deviation features are computed.

iv. All of the features are normalised.

v. Empty and factorised cells are processed.

Next, we provide additional details on each step.

Step 1: pre-processing operations This step applies several pre-processing operations to the input table and outputs a table with α rows and β columns ($\alpha \geq 1, \beta \geq 1$) using the following procedure:

i. If the dir attribute is present in either the table tag or any of its ancestors and its value is rtl, then the table is flipped horizontally. Thus, the first column is always the left-most column.

ii. The maximum cell span is set to 200 cells. We think that it is very unlikely that a correct table has such large cell spans; setting this limit helps avoid overhead when processing tables that are incorrectly encoded.

iii. The cells whose span is greater than one are replicated accordingly and their span is set to one.

iv. The rows that are shorter than the largest row are padded to the right using empty cells, which prevents outputting ragged tables.

v. Duplicated rows or columns are removed, except for the top-most and the left-most ones.

Step 2: computing base features This step analyses the cells in the input tables and outputs a matrix with their feature vectors.

First, it computes the attributes of the DOM nodes, cf. Table 3.1. The style attributes are computed using a headless browser and most of the other attributes are computed using straightforward procedures. The only attributes that deserve a little more explanations are the "Is node repeated?" structural attribute and the "Likelihood of being meta-data" miscellaneous attribute. The former is a Boolean attribute that indicates whether more nodes with the same path and content can be found in other tables in the set of input documents; the latter was pre-computed as follows using the Dresden Web Table Corpus [35]:

i. The tables in the corpus were cleaned by lower-casing their contents, stripping white spaces, and replacing the digits with a generic "D" token.

ii. Every cell in the first row or column was flagged as a meta-data cell and the remaining ones were flagged as data cells.

iii. Then, the meta-data likelihood of every cell was computed by dividing the number of occurrences of its content in a meta-data cell by the total number of occurrences of that content in the corpus.

iv. The previous likelihood was adjusted using another heuristic, namely: every cell in a row or column with an average meta-data likelihood greater than 0.50 was considered meta-data; otherwise, it was considered data.

v. The previous two steps were repeated five times, which was enough for the likelihoods to stabilise.

Please, note that we do not claim that the likelihoods computed by the previous procedure are strongly correlated to the probability of a particular content to be meta-data or data. It simply provides an estimate that has proven to work well in practice when it is combined with the other attributes that our proposal takes into account.

Next, we project the nodes onto feature vectors that are computed from their attributes. Before computing them, we need to replace the composite attributes by new attributes that represent their individual components and the categorical attributes by binary attributes that represent them using one-hot encoding. This simplifies working with the attributes since they all can be assumed to be numeric from now on.

Then, for every node a, we compute its base features as follows:

$$\begin{aligned}
\mathtt{bfeat}(a) &= \omega_1(a)\,\mathtt{attr}(a) + \sum_{b \in \mathtt{children}(a)} \omega_2(a,b)\,\mathtt{bfeat}(b) \\
\omega_1(a) &= 1 - \sum_{b \in \mathtt{children}(a)} \omega_2(a,b) \\
\omega_2(a,b) &= \frac{\mathtt{area}(b)}{\mathtt{area}(a)}
\end{aligned}$$

Note that the base features are computed as the weighted average of the values of the attributes of node a and the base features of its children. The weight of the attributes and the base features is computed taking into account the relative area of the bounding box of the corresponding node. $\omega_1(a)$ denotes the weight of the attributes of node a, which is computed as the percentage of the area of the bounding box of node a that depends exclusively on that node, not its children. Similarly, $\omega_2(a, b)$, where $b \in \mathtt{children}(a)$, denotes the percentage of area of the bounding box that depends on node b relative to the total area of node a.

Step 3: computing deviation features This step takes the matrix of base features as input and returns a new matrix in which each component has the base features plus the deviation features.

For every base feature f, we compute three additional deviation features, namely: f^r, which measures the deviation of feature f in a cell with regard to the other cells in the same row, f^c, which measures the deviation of feature f in a cell with regard to the other cells in the same column, and f^t, which measures the deviation of feature f in a cell with regard to the other cells in the same table. To compute them, we need the following ancillary variables:

$$R_i = \sum_{j=1}^{\beta} {}^1\!/_{\beta}\, M[i,j] \;\; (1 \le i \le \alpha)$$
$$C_j = \sum_{i=1}^{\alpha} {}^1\!/_{\alpha}\, M[i,j] \;\; (1 \le j \le \beta)$$
$$T = \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} {}^1\!/_{(\alpha\,\beta)}\, M[i,j],$$

where M denotes the matrix returned by the previous step and R_i, C_i, and T are vectors with the per-row, per-column, and per-table averages of the base features, respectively.

Assume now that $\{f_p : v_p\}_{p=1}^{\kappa}$ is the base feature vector computed for the cell at position (i, j). The deviation features are then defined as follows:

$$f_p^r = (v_p - R_i[f_p])^2$$
$$f_p^c = (v_p - C_j[f_p])^2$$
$$f_p^t = (v_p - T[f_p])^2,$$

where $1 \le p \le \kappa$.

Step 4: normalising features This step takes the $\alpha \times \beta$ matrix with the 4κ-dimensional vectors computed by the previous step and normalises the values of the features so that they all range in interval $[0.00, 1.00]$.

In the literature, we have found the following common approaches, where Z denotes the set of values of a feature in a table, Z' denotes its set of normalised values, Z^* denotes the set of values of that feature across all of the tables in our experimental datasets, and clamp denotes a function that clamps the value to which it is applied into interval $[0.00, 1.00]$:

Global min-max: it normalises the values of a feature taking into account its minimum and its maximum across all of the tables in our datasets, namely:

$$Z' = \left\{ \frac{z - \min Z^*}{\max Z^* - \min Z^*} \mid z \in Z \right\}$$

Local min-max: it is similar to the previous one, but the minimum and the maximum are computed on a per-table basis, namely:

$$Z' = \left\{ \frac{z - \min Z}{\max Z - \min Z} \mid z \in Z \right\}$$

Standard normal: it transforms the values of a feature into a set of values that are distributed according to a standard normal distribution that is shifted to and clamped into our target interval, namely:

$$Z' = \left\{ \text{clamp} \left(\frac{z - \text{mean}\, Z + \text{stdev}\, Z}{2\ \text{stdev}\, Z} \right) \mid z \in Z \right\}$$

Softmax: it transforms the values of a feature into a set of probabilities that are proportional to the exponential of each feature value, namely:

$$Z' = \left\{ \frac{e^{z_1}}{\sum_{z_2 \in Z} e^{z_2}} \mid z_1 \in Z \right\}$$

Since of them stands out from a conceptual point of view, this is a variation point.

Step 5: processing empty and factorised cells This step works on the input table and returns a new table in which empty cells are identified and removed if necessary and factorised cells are made explicit.

The first operation performed by this step consists in identifying the cells that can be considered empty. Our proposal is to flag as such every cell whose content consists exclusively of white spaces, dashes, asterisks, or question marks. If the lang attribute is present in the table tag or any of its ancestors, then other language-dependent symbols are also considered, e.g., "N/A" and "void" in English, "N/A" and "N/D" in Spanish, or "ND" and "S.O." in French.

The second operation processes the factorised cells as follows:

i. First, full-span rows are identified. They are rows that originally consisted of just one spanned cell. According to their positions, they can be classified as top full-span rows, middle full-span rows, and bottom full-span rows.

ii. If there is at least one middle full-span row and it is repeated every two rows, then we analyse how similar the middle full-span rows are to the last top full-span row and the first bottom full-span row by measuring the Euclidean distance of their feature vectors. If they are more similar to the former, then the last top full-span row and every middle full-span row are appended to the end of the next row as an additional column. If they are more similar to the latter, then the first bottom full-span row and every middle full-span row are appended to the end of the previous row as an additional column.

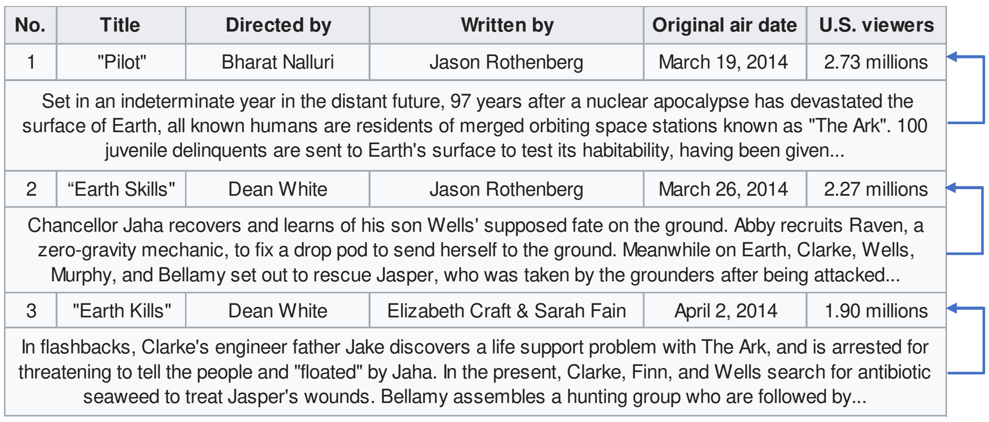
iii. If the full-span rows are not repeated periodically, then the last top full-span row and every middle full-span row are appended to the following rows as additional columns until another full-span row is found. The new column can be placed either at the beginning or the end of the row. To determine its position, we need to compute $\bar{c}$ as the average of the deviation features per column, and $\bar{r}$ as the average of the deviation features per row, excluding full-span rows in both cases. If $\bar{r} > \bar{c}$, then the column is placed at the last position since the table is more likely to be a horizontal listing and the factorised cells are more likely to be spanned data. Otherwise, the column is placed at the first position, since the table is more likely to be a vertical listing and the factorised cells are more likely to be spanned meta-data.

The procedure is then repeated on a per-column basis to deal with tables that have full-span columns. Next, the rows and columns in which every cell is empty or the result of repeating a spanned cell are removed. The remaining full-span rows or columns are removed and saved as context cells that will be used in the interpretation task since they usually provide captions or footnotes that are worth preserving.

Example 3.1 *Figures 3.2, 3.3, and 3.4 illustrate how to process factorised cells.*

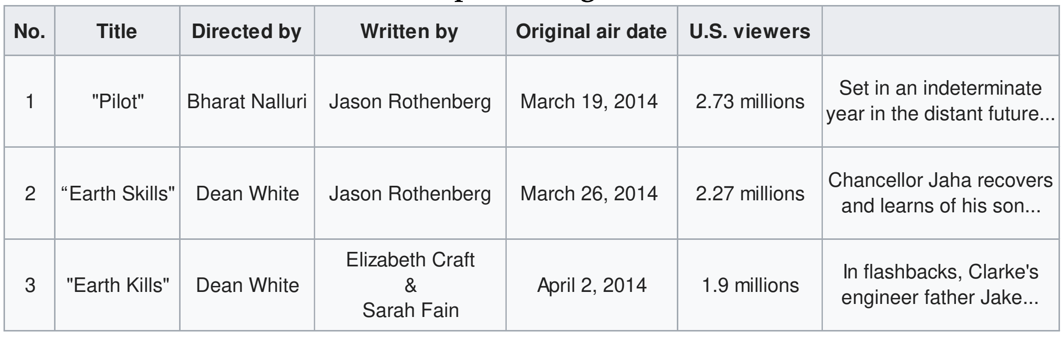
The table in Figure 3.2.a shows data about a TV show. There is a middle full-span row that is repeated every two rows and it is clearly more similar to the bottom full-span row than to the top full-span row. Our heuristic indicates that they must be added to the right of the previous rows as an additional column, cf. Figure 3.2.b.

The table in Figure 3.3.a shows data about the medals won by a basketball player, grouped by competition. In this case, the middle full-span rows are not repeated periodically and it is easy to realise that the per-row deviation is significantly higher, since the columns are very homogeneous. According to our heuristic, this implies that a new column must be added to the right of the table and the contents of the full-span rows must be replicated for each row until a new full-span row is found, cf. Figure 3.3.b.



No.	Title	Directed by	Written by	Original air date	U.S. viewers
1	"Pilot"	Bharat Nalluri	Jason Rothenberg	March 19, 2014	2.73 millions
Set in an indeterminate year in the distant future, 97 years after a nuclear apocalypse has devastated the surface of Earth, all known humans are residents of merged orbiting space stations known as "The Ark". 100 juvenile delinquents are sent to Earth's surface to test its habitability, having been given...					
2	"Earth Skills"	Dean White	Jason Rothenberg	March 26, 2014	2.27 millions
Chancellor Jaha recovers and learns of his son Wells' supposed fate on the ground. Abby recruits Raven, a zero-gravity mechanic, to fix a drop pod to send herself to the ground. Meanwhile on Earth, Clarke, Wells, Murphy, and Bellamy set out to rescue Jasper, who was taken by the grounders after being attacked...					
3	"Earth Kills"	Dean White	Elizabeth Craft & Sarah Fain	April 2, 2014	1.90 millions
In flashbacks, Clarke's engineer father Jake discovers a life support problem with The Ark, and is arrested for threatening to tell the people and "floated" by Jaha. In the present, Clarke, Finn, and Wells search for antibiotic seaweed to treat Jasper's wounds. Bellamy assembles a hunting group who are followed by...					

a) Before processing the table.



No.	Title	Directed by	Written by	Original air date	U.S. viewers	
1	"Pilot"	Bharat Nalluri	Jason Rothenberg	March 19, 2014	2.73 millions	Set in an indeterminate year in the distant future...
2	"Earth Skills"	Dean White	Jason Rothenberg	March 26, 2014	2.27 millions	Chancellor Jaha recovers and learns of his son...
3	"Earth Kills"	Dean White	Elizabeth Craft & Sarah Fain	April 2, 2014	1.9 millions	In flashbacks, Clarke's engineer father Jake...

b) After processing the table.

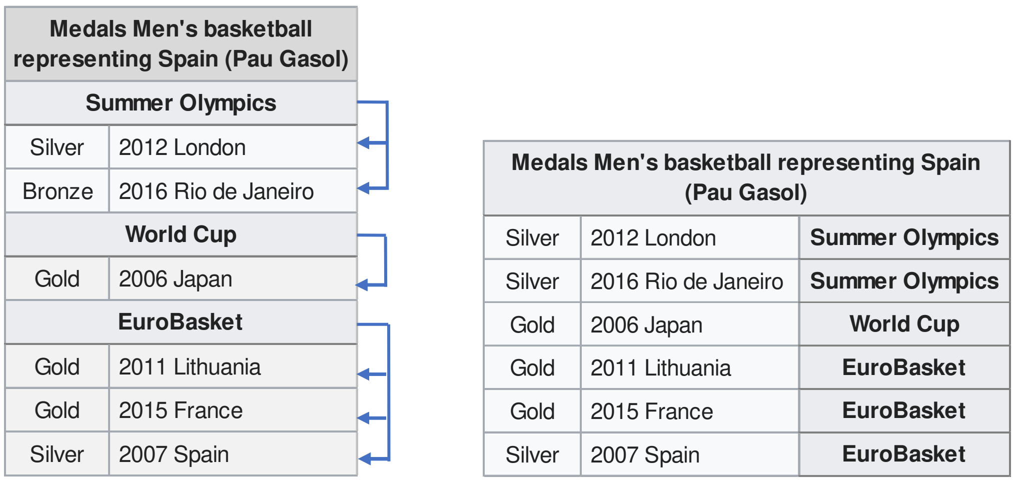
Figure 3.2: *Factorisation of periodic full-span cells.*

The table in Figure 3.4.a shows data about a comparative analysis of vacuum cleaners. In this case, the full-span rows are not repeated periodically and the deviation per row seems to be smaller than the deviation per column. Thus, our heuristic suggests that a new column must be added to the left and the contents of the full-span rows must be replicated to the new cells until a new full-span row is found, cf. Figure 3.4.b.

3.5 Discrimination

This task takes a table returned by the segmentation task and keeps it or removes it from the pipeline flow depending on whether it can be considered a data table or not. It performs the following steps:

i. Tables whose width or height attributes are 0px or whose display attribute



Medals Men's basketball representing Spain (Pau Gasol)	
Summer Olympics	
Silver	2012 London
Bronze	2016 Rio de Janeiro
World Cup	
Gold	2006 Japan
EuroBasket	
Gold	2011 Lithuania
Gold	2015 France
Silver	2007 Spain

a) Before processing the table.

Medals Men's basketball representing Spain (Pau Gasol)		
Silver	2012 London	**Summer Olympics**
Silver	2016 Rio de Janeiro	**Summer Olympics**
Gold	2006 Japan	**World Cup**
Gold	2011 Lithuania	**EuroBasket**
Gold	2015 France	**EuroBasket**
Silver	2007 Spain	**EuroBasket**

b) After processing the table.

Figure 3.3: *Factorisation of non-periodic full-span cells to the right.*

is none are discarded[†3].

ii. Tables that have a single row or a single column are discarded.

iii. Tables with an ancestor table tag, a descendant textarea tag, or a descendant input tag with attribute type="text" are also discarded[†3]. This proved very useful to remove utilities like pagination controls that are formatted as inner tables and input forms.

3.6 Functional analysis

This task takes a data table and its corresponding feature vector matrix and returns an $\alpha \times \beta$ matrix in which each component identifies the function of the corresponding cell plus some scores that help guess the orientation of the input table. This task performs the following steps:

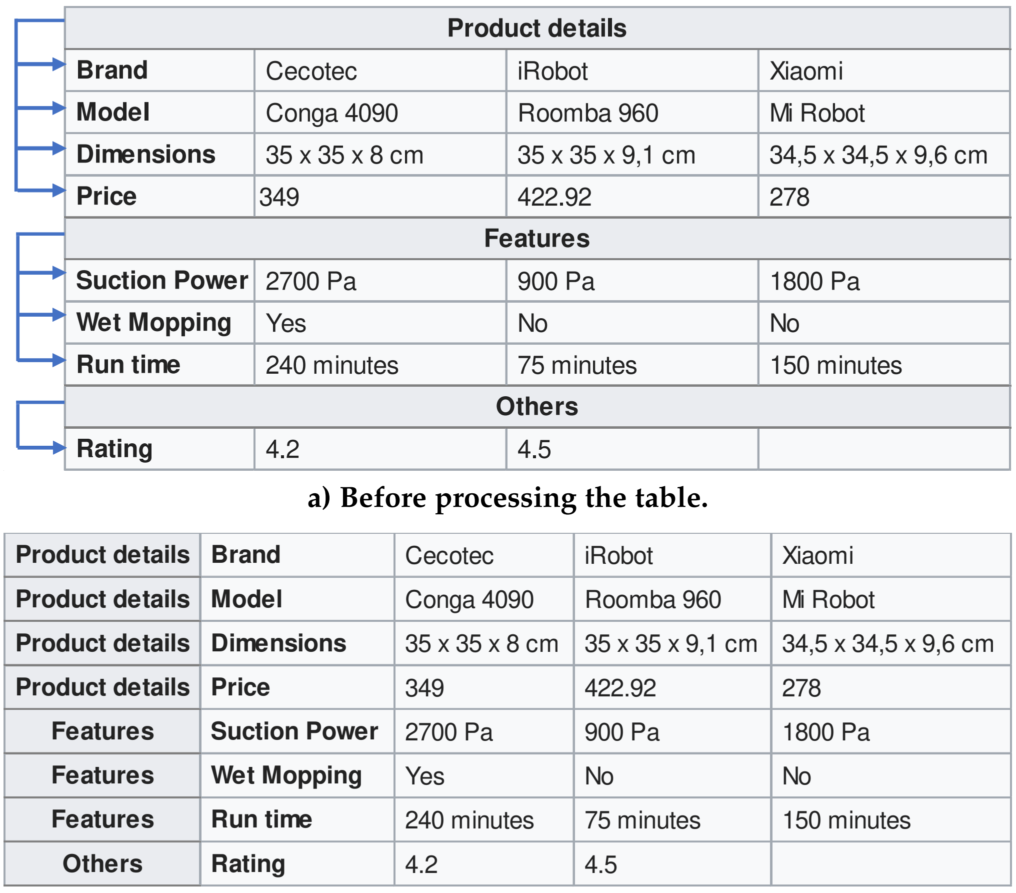
i. The dimensionality of the feature vectors is reduced.

ii. Candidate cell functions are computed.

iii. Cell functions are identified.

Next, we provide additional details on each step.

[†3]For the sake of efficiency, TOMATE implements this operation during the location task since this helps discriminate some tables without segmenting them.



Product details			
Brand	Cecotec	iRobot	Xiaomi
Model	Conga 4090	Roomba 960	Mi Robot
Dimensions	35 x 35 x 8 cm	35 x 35 x 9,1 cm	34,5 x 34,5 x 9,6 cm
Price	349	422.92	278
Features			
Suction Power	2700 Pa	900 Pa	1800 Pa
Wet Mopping	Yes	No	No
Run time	240 minutes	75 minutes	150 minutes
Others			
Rating	4.2	4.5	

a) Before processing the table.

Product details	**Brand**	Cecotec	iRobot	Xiaomi
Product details	**Model**	Conga 4090	Roomba 960	Mi Robot
Product details	**Dimensions**	35 x 35 x 8 cm	35 x 35 x 9,1 cm	34,5 x 34,5 x 9,6 cm
Product details	**Price**	349	422.92	278
Features	**Suction Power**	2700 Pa	900 Pa	1800 Pa
Features	**Wet Mopping**	Yes	No	No
Features	**Run time**	240 minutes	75 minutes	150 minutes
Others	**Rating**	4.2	4.5	

b) After processing the table.

Figure 3.4: *Factorisation of non-periodic full-span cells to the left.*

Step 1: reducing feature dimensionality This step takes the $\alpha \times \beta$ matrix with the 4κ-dimensional vectors computed by the segmentation task and returns an $\alpha \times \beta$ matrix with new feature vectors that result from selecting some features and then reducing their dimensionality.

Regarding feature selection, recall that we take four categories of features into account, namely: style, structural, lexical, and miscellaneous. Conceptually, there is not a clear argument in favour of using any of them, so we need to introduce a variation point in which we consider the following alternatives: a) using all of the features, b) using a single category of features, c) using combinations of two categories, and d) using combinations of three categories. This results in a total of 15 alternatives.

Regarding dimensionality reduction, we need to introduce a new variation point with the following alternatives: a) not performing dimensionality reduction, b) using Principal Component Analysis, which creates new uncorrelated features that consecutively maximise variance, and c) using feature agglomeration, which merges the features that are similar using an agglomerative clustering approach. We do not consider the more recent t-SNE method because our preliminary experiments confirmed that its results are very poor in our context; this method is very good at reducing a dataset to two or three dimensions, which is appropriate for visualisation purposes but clearly not enough to compute the functionality of the cells.

Step 2: computing candidate cell functions This step takes the $\alpha \times \beta$ matrix with vectors computed by the previous step and returns an $\alpha \times \beta$ matrix in which each component is the candidate function of the corresponding cell.

First, we cluster the cells into two clusters K_1 and K_2. The exact method used is another variation point since it is not clear which one performs the best. We consider the following alternatives: a) k-means using k-means^{++} as the initialisation method and mini-batches when dealing with more than one thousand cells, which is a well-known approach that typically performs very well; and b) agglomerative clustering, which recursively merges pairs of clusters that minimally increase the linkage distance until two clusters are found. We could not use alternatives like DBSCAN, OPTICS, Mean Shift, or Affinity Propagation because they are intended to find the optimal number of clusters, which typically resulted in more than two clusters.

Finally, we need to determine which cluster corresponds to meta-data cells and which one corresponds to data cells. We use a heuristic that builds on the following variables: a) $r_i = (^{a_i}/_{\beta} + {}^{b_i}/_{\alpha})/2$, where a_i and b_i denote the number of cells in the first row and column, respectively, that are in cluster K_i; intuitively, r_i measures the ratio of cells in cluster K_i that are in the first row and column; thus, the higher r_i, the higher the chances that cluster K_i consists of meta-data cells since such cells are typically placed in the first row or column ($i \in \{1, 2\}$). b) $s_i = 1 - {}^{|K_i|}/_{\alpha\,\beta}$; intuitively, s_i measures the one-complement of the relative size of cluster K_i; thus, the higher s_i, the higher the chances that cluster K_i consists of meta-data cells since these cells are typically a minority ($i \in \{1, 2\}$). c) $c_i = 1 - \sum_{(p,q) \in K_i} {}^{\sqrt{p^2+q^2}}/_{|K_i|}$; intuitively, c_i measures the closeness of a cluster to the top-left corner of a table; thus, the higher c_i, the higher the chances that cluster K_i consists of meta-data cells since such cells are typically near the top-left corner of the tables ($i \in \{1, 2\}$). Our heuristic assumes that the meta-data cells are in cluster K_1

and the data cells in cluster K_2 if ${}^{(r_1+s_1+c_1)}\!/_3 \geq {}^{(r_2+s_2+c_2)}\!/_3$; otherwise, we assume that the data cells are in cluster K_1 and the meta-data cells are in cluster K_2.

Step 3: identifying cell functions We identify the functions of the cells building on the previous candidate functions and the orientation of the table.

The orientation is guessed using three scores, namely: w^r, which stands for row-wise orientation, w^c, which stands for column-wise orientation, and w^t, which stands for table-wise orientation. To compute them, we split the table into the following regions: A_1, which consists of the cell at position $(1, 1)$, A_2, which consists of the set of cells at positions $(1, j)$, $2 \leq j \leq \beta$, A_3, which consists of the set of cells at positions $(i, 1)$, $2 \leq i \leq \alpha$, and A_4, which consists of the set of cells at positions (i, j), $2 \leq i \leq \alpha, 2 \leq j \leq \beta$. Then, we use two methods to compute the scores depending on the size of the table.

The first method is used with tables that have more than two rows and more than two columns. It computes the scores as follows, where silh denotes the Silhouette coefficient:

$$w^r = \text{silh}\{A_1 \cup A_2, A_3 \cup A_4\}$$
$$w^c = \text{silh}\{A_1 \cup A_3, A_2 \cup A_4\}$$
$$w^t = \text{silh}\{A_1, A_2, A_3, A_4\}$$

The intuition behind the previous formulation is that the Silhouette score is expected to be the highest when each of the clusters has only regions with the same function; otherwise, it should drop. If a table has row-wise orientation, then regions A_1 and A_2 should consist almost exclusively of meta-data cells, whereas regions A_3 and A_4 should have a majority of data cells; that is: clustering $\{A_1 \cup A_2, A_3 \cup A_4\}$ should have a better Silhouette coefficient than the others. If a table has column-wise orientation, then regions A_1 and A_3 should consist almost exclusively of meta-data cells, whereas regions A_2 and A_4 should have a majority of data cells; that is: clustering $\{A_1 \cup A_3, A_2 \cup A_4\}$ should have a better Silhouette coefficient than the others. Finally, if a table has table-wise orientation, then we have experimentally found that computing the Silhouette coefficient of clustering $\{A_1, A_2, A_3, A_4\}$ provides a better estimate than the other clusterings. We also explored computing w^t as silh$\{A_1 \cup A_2 \cup A_3, A_4\}$ since our intuition suggested that this would result in the highest Silhouette score in the case of table-wise tables, but our experience confirmed that the Silhouette coefficient of clustering $\{A_1, A_2, A_3, A_4\}$ works much better.

However, the Silhouette coefficient tends to increase with the number of clusters, which makes the w^t score artificially higher than it should be in the case of tables with only two rows or only two columns. We use a different method with such tables. We first compute the average feature vector of the cells that belong to each area A_i (which is denoted as u_i), and then compute the orientation scores by measuring the differences between the four regions as follows ($1 \leq i \leq 4$):

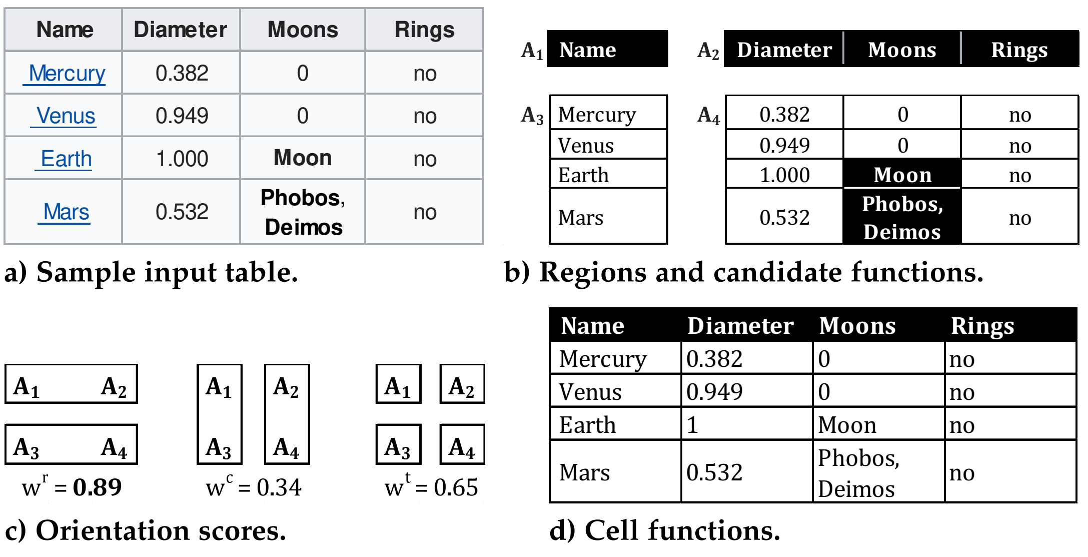
$$\begin{aligned} w^r &= \text{dist}(u_1, u_3) + \text{dist}(u_2, u_4) - \text{dist}(u_1, u_2) - \text{dist}(u_3, u_4) \\ w^c &= \text{dist}(u_1, u_2) + \text{dist}(u_3, u_4) - \text{dist}(u_1, u_3) - \text{dist}(u_2, u_4) \\ w^t &= \text{dist}(u_3, u_4) + \text{dist}(u_2, u_4) - \text{dist}(u_1, u_2) - \text{dist}(u_1, u_3) \end{aligned}$$

Assuming that each region has the function that corresponds to the majority vote provided by its cells, the intuition is that we expect the score for a given orientation to be the highest when we maximise the distance between two regions that have different functions and when we minimise the distance between two regions that have the same functions. Simply put, if we are dealing with a row-wise table, then we expect the cells in regions A_1 and A_2 to be meta-data cells and the cells in regions A_3 and A_4 to be data cells. Therefore, their distances should be minimal. Any other combination of two regions should have different functions with regard to each other and, therefore, their distance should be larger. Thus, we expect to maximise w^r if the distances between u_1 and u_3 and between u_2 and u_4 are high and the distances between u_1 and u_2 and between u_3 and u_4 are low, as long as the correct orientation is row-wise. The reasoning is similar in the case of column-wise or table-wise tables.

The procedure to identify the functions of the cells works as follows: a) if the orientation is guessed to be row-wise or table-wise, then we set R to the set of row indices in which the majority of cells are candidate meta-data cells; otherwise, we set $R = \emptyset$; if the orientation is column-wise or table-wise, then we set C to the set of column indices in which the majority of cells are candidate meta-data cells; otherwise, we set $C = \emptyset$. b) next, we declare as data the cells at positions (i, j) such that $i \notin R$ and $j \notin C$; c) now, we declare as meta-data the cells at positions (i, j) such that $i \in R$ and $1 \leq j \leq \beta$ or $1 \leq i \leq \alpha$ and $j \in C$; d) finally, if both R and C are non-empty sets, then we declare as data the cells at positions (i, j) such that $i \in R$ and $1 + \max C \leq j \leq \beta$ or $1 + \max R \leq i \leq \alpha$ and $j \in C$.

Example 3.2 *Figures 3.5 and 3.6 illustrate our method to identify the functionality of the cells.*

Figure 3.5.a shows a table with data about the terrestrial planets of the solar system. Figure 3.5.b shows the four regions in which we divide the table



Name	Diameter	Moons	Rings
Mercury	0.382	0	no
Venus	0.949	0	no
Earth	1.000	**Moon**	no
Mars	0.532	**Phobos, Deimos**	no

a) Sample input table.

A_1	Name	A_2	Diameter	Moons	Rings
A_3	Mercury	A_4	0.382	0	no
	Venus		0.949	0	no
	Earth		1.000	Moon	no
	Mars		0.532	Phobos, Deimos	no

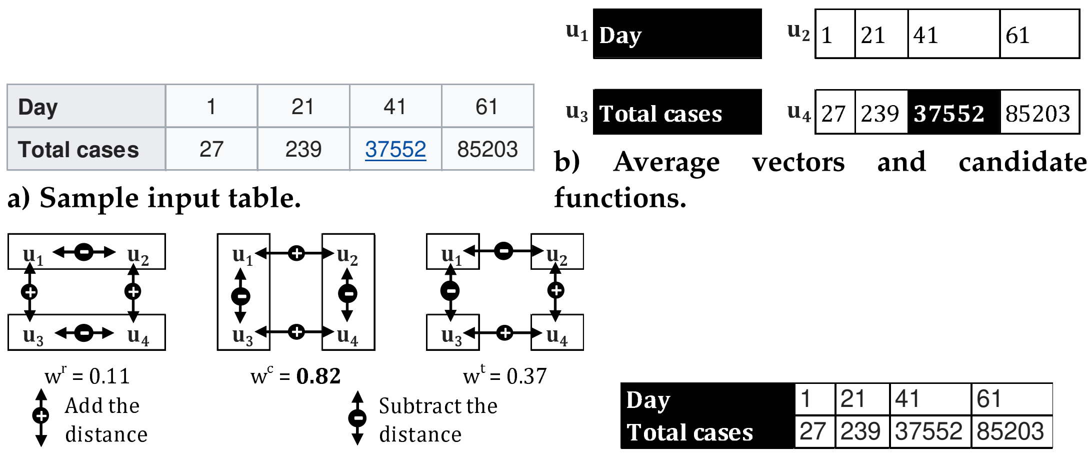
b) Regions and candidate functions.

c) Orientation scores.

Name	Diameter	Moons	Rings
Mercury	0.382	0	no
Venus	0.949	0	no
Earth	1	Moon	no
Mars	0.532	Phobos, Deimos	no

d) Cell functions.

Figure 3.5: *First method to correct cell functions.*



Day	1	21	41	61
Total cases	27	239	37552	85203

a) Sample input table.

u_1	Day	u_2	1	21	41	61
u_3	Total cases	u_4	27	239	37552	85203

b) Average vectors and candidate functions.

c) Results of the area distances.

Day	1	21	41	61
Total cases	27	239	37552	85203

d) Corrected table functions.

Figure 3.6: *Second method to correct cell functions.*

and the candidate functions of the cells (black cells correspond to meta-data cells and white cells correspond to data cells). Note that most candidate functions are correct, except for the highlighted cells on the third column, which seem more similar to the meta-data cells at the top than to the remaining data

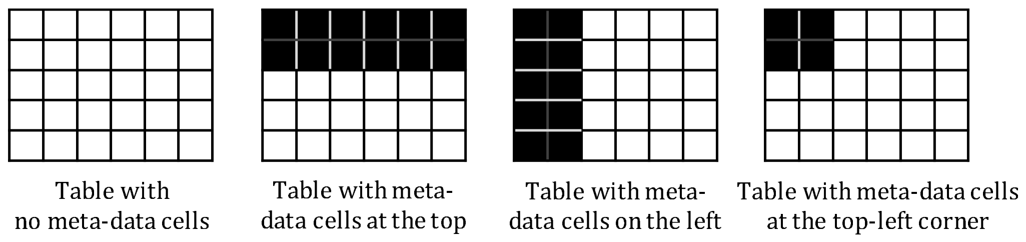


Table with no meta-data cells
Table with meta-data cells at the top
Table with meta-data cells on the left
Table with meta-data cells at the top-left corner

Figure 3.7: *Canonical tables after functional analysis.*

cells. To identify the correct functionality, we have to guess the orientation of the table. This table has more than two rows and more than two columns, so the first method must be used. Recall that it computes the three scores using the Silhouette coefficient of three different clusters and guesses the orientation according to the highest one. Figure 3.5.c illustrates the three clusterings that are analysed and their corresponding scores. In this example, the highest score is w^r, which means that the table is assumed to be row-wise. Therefore, the candidate functionality is corrected according to the majority vote on a per-row basis; note that this assigns the correct functionality to the two highlighted cells in the third row, cf. Figure 3.5.d.

Figure 3.6.a shows a table with data about the COVID-19 outbreak. It has two rows only, so the second method to guess the orientation is used. If the first method was used, the orientation would be estimated as table-wise, since having so few rows makes the difference between the silhouette coefficient of clusterings with different number of clusters artificially high and results in the clustering with four clusters being selected. Figure 3.6.b illustrates the four average feature vectors that are computed for the cells in each of the regions. Note that vectors u_1 and u_3 refer to the regions that have the meta-data cells, whereas vectors u_2 and u_4 refer to the regions that have the data cells. Figure 3.6.c illustrates the operations performed to compute the distance of the average feature vectors of the table. Since vectors u_1 and u_3 are very similar, the highest score is w^c, which means that our heuristic considers that the table is oriented column-wise. Figure 3.6.d shows the functionality returned by the functional analysis task after the candidate functions are corrected.

Example 3.3 *Figure 3.7 illustrates the four types of tables that result from the functional analysis task, namely: a table with no meta-data cells, a table in which they are at the top rows, a table in which they are on the left columns, or a table in which they are at the top-left corner. The first case cor-*

responds to tables in which the clustering process cannot identify any rows or columns with a majority of meta-data cells; such tables are common in cases in which the reader can be assumed to understand the meaning of the cells using their background knowledge as well as the context around the table. The second and the third cases correspond to horizontal and vertical listings, respectively; note that there are some listings in which the meta-data cells are repeated several times for the sake of readability, but the repetitions were removed by the segmentation task because they do not provide any data of interest for the remaining tasks. The fourth case corresponds to a matrix, which is typically a table with a few meta-data cells and many data-cells; furthermore, the data cells in the first few rows and columns act as indexers for the remaining data cells.

3.7 Structural analysis

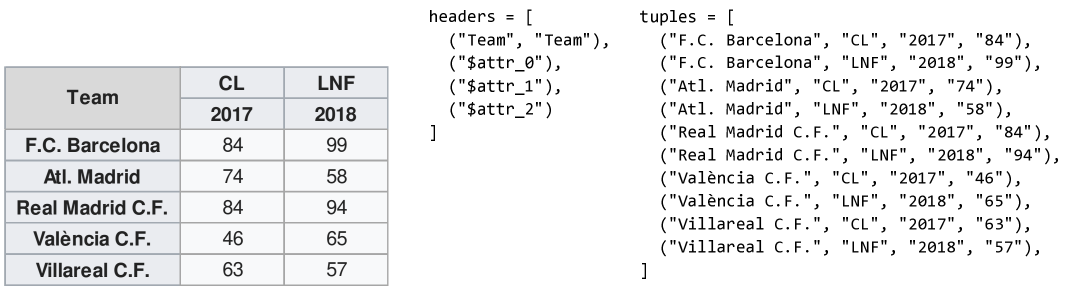
This task works on a cell function matrix, a feature vector matrix, and the orientation scores of a table. It returns the collection of headers and the collection of tuples in the input table.

The procedure that this task executes is as follows. Note that it requires to create some artificial meta-data cells. Such cells have contents that are generated according to a user-defined pattern. (Any pattern may be used as long as the artificial meta-data can be clearly identified later.)

i. If there are not any meta-data cells, then we proceed as follows: if the w^r score that was computed by the functional analysis task is greater than or equal to the w^c score, then a new row of artificial meta-data cells is added at the top of the table; otherwise, a new column of artificial meta-data cells is added to the left of the table. In the first case, the headers are the column-wise blocks of artificial meta-data cells and the tuples are the row-wise blocks of data cells; in the second case, the headers are the row-wise blocks of artificial meta-data cells and the tuples are the column-wise blocks of data cells.

ii. If there are one or more rows of meta-data cells at the top, then each column-wise block of such meta-data cells is a header and each row-wise block of data cells is a tuple.

iii. If there are one or more columns of meta-data cells on the left, then each row-wise block of such meta-data cells is a header and each column-wise block of data cells is a tuple.



Team	CL	LNF
	2017	2018
F.C. Barcelona	84	99
Atl. Madrid	74	58
Real Madrid C.F.	84	94
València C.F.	46	65
Villareal C.F.	63	57

a) Sample input table.

```
headers = [
  ("Team", "Team"),
  ("$attr_0"),
  ("$attr_1"),
  ("$attr_2")
]
```

```
tuples = [
  ("F.C. Barcelona", "CL", "2017", "84"),
  ("F.C. Barcelona", "LNF", "2018", "99"),
  ("Atl. Madrid", "CL", "2017", "74"),
  ("Atl. Madrid", "LNF", "2018", "58"),
  ("Real Madrid C.F.", "CL", "2017", "84"),
  ("Real Madrid C.F.", "LNF", "2018", "94"),
  ("València C.F.", "CL", "2017", "46"),
  ("València C.F.", "LNF", "2018", "65"),
  ("Villareal C.F.", "CL", "2017", "63"),
  ("Villareal C.F.", "LNF", "2018", "57"),
]
```

b) Collections of headers and tuples.

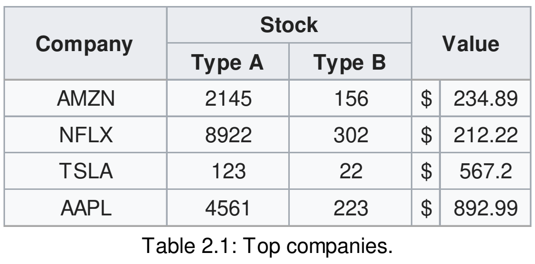
Figure 3.8: *Sample results of structural analysis.*

iv. Otherwise, there is a block of meta-data cells at the top-left corner. In this case, we set R and C to the sets of row indices and column indices with meta-data cells, respectively. Given a cell at position (i, j) such that $i \notin R$ and $j \notin C$, we define its row indexers as the set of cells at positions $\{(i, k) \mid k \in C\}$ and its column indexers as the set of cells at positions $\{(k, j) \mid k \in R\}$. In this case, the headers are the column-wise blocks of meta-data cells plus $|R| + 1$ artificial headers and there is a tuple for each data cell at position (i, j) $(i \notin R, j \notin C)$ that is composed of its row indexers, plus its column indexers, plus the cell itself.

Example 3.4 *Figure 3.8.a shows a sample table with some statistics about a few Spanish soccer teams. Our proposal identifies it as a table-wise table that has one meta-data cell at the upper-left corner and many data cells; the meta-data cell is split into two individual cells by the segmentation task to take its two-row span into account. It then computes four headers and ten tuples. The first header is composed of the two meta-data cells; the others were generated artificially using pattern $attr_$i$, where i denotes a sequential index ($i \geq 0$). The tuples correspond to the cells in the body of the matrix plus their corresponding row and column indexers.*

3.8 Interpretation

This task takes the context cells output by the segmentation task and the collections of headers and tuples returned by the structural analysis task as input. It returns a collection of records that are expected to facilitate processing the data in the input table automatically.



Company	Stock		Value	
	Type A	Type B		
AMZN	2145	156	$	234.89
NFLX	8922	302	$	212.22
TSLA	123	22	$	567.2
AAPL	4561	223	$	892.99

Table 2.1: Top companies.

a) Sample input table.

```
[
  {"Company": "AMZN", "Stock / Type A": "2,145", "Stock / Type B": "156", "Value / 1": "$", "Value / 2": "234.89"},
  {"Company": "NFLX", "Stock / Type A": "8,922", "Stock / Type B": "302", "Value / 1": "$", "Value / 2": "212.22"},
  {"Company": "TSLA",  "Stock / Type A": "123", "Stock / Type B": "22", "Value / 1": "$", "Value / 2": "567.20"},
  {"Company": "AAPL",  "Stock / Type A": "4,561", "Stock / Type B": "223", "Value / 1": "$", "Value / 2": "892.99"},
  {"$context": "Table 2.1: Top companies."}
]
```

b) Result of the interpretation task.

Figure 3.9: *Sample result of the interpretation task.*

Recall that records are modelled as maps of the form $\{d_i : v_i\}_{i=1}^{r}$, where each d_i is a descriptor and each v_i is a value ($r \geq 1$). The descriptors are computed from the headers as follows: the meta-data cells in a header are collapsed by catenating their contents using an appropriate user-defined separator (any separator can be used as long as it can be clearly identified in the descriptors); in cases in which two adjacent meta-data cells have the same content, the second one is ignored since, very likely, it resulted from splitting a spanned cell; in cases in which two resulting descriptors are the same, a sequential index is added to disambiguate them. There must also be a user-defined descriptor to represent the data in the context cells; as usual, any descriptor is valid as long as it can be clearly identified.

The procedure to compute the output record set is straightforward: it creates a new record per tuple in which each component takes its value from the corresponding data cell in the tuple and its descriptor from the corresponding header, as explained before.

Example 3.5 *Figure 3.9.a shows a sample table with some stock market data. Our proposal identifies it as a row-wise table in which the first two rows are composed of meta-data cells, then come some tuples, and there is a final context cell. Note that the meta-data cells in the first and the last column have vertical spans, which means that they are segmented as two independent cells each. Note, too, that the meta-data cell in the last column actually spans* *two columns because the author of the table decided to format the values using a column to display the currency symbol and another column to display the nominal value. The interpretation task results in the record set in Figure 3.9.b. Realise that the headers are "Company", which corresponds to the first component of the tuples, then come "Stock / Type A" and "Stock / Type B", which correspond to the second and the third component of the tuples, and finally come "Value / 1" and "Value / 2", which correspond to the fourth and the fifth component of the tuples. In this example, we use symbol "/" to catenate the contents of the original meta-data cells and create the descriptors. Note that the caption to the table is identified as a context cell, which is added to the resulting record set using the user-defined "$context" descriptor.*

3.9 Case study

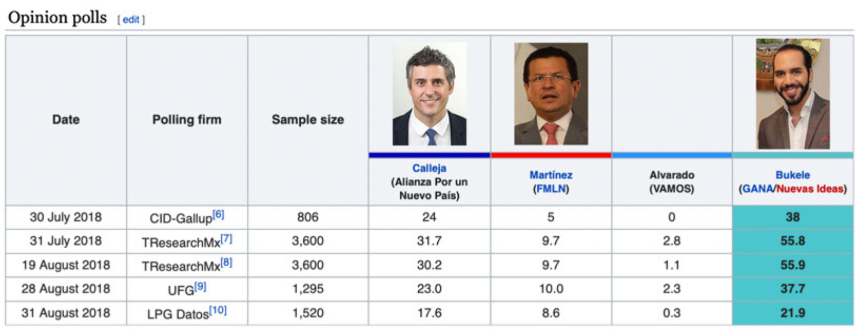
Our proposal was integrated into a hybrid geopolitical forecasting system [89]. It gets from 5 to 10 multiple-choice questions from IARPA every week and forwards them to a crowdsourcing platform where the users are asked to answer them. To help the users provide informed answers, it presents charts with quantitative data and forecasts using machine-learning models. HTML tables are plenty of relevant data for many of those questions.

Automation is very important because the questions posed to the system are about current events and the data are not expected to be generally available in any major knowledge bases. Furthermore, the users are only expected to perform clerical tasks, e.g., specifying the URL of a document with a table of interest or mapping its headers onto the choices for a given question.

Our proposal helps extract data from the HTML tables that feed the system, so that it can provide charts and forecasts. Figure 3.10 illustrates the system in the context of El Salvador 2019 presidential election. Given a question, the system searches the Wikipedia for documents that match that question using a standard search engine. The user may select some documents with tables, like the one in Figure 3.10.a, and then activate TOMATE, which analyses it as follows:

Location: it fetches the document and makes its attributes explicit, then parses it to create its DOM tree representation, and finally selects the sub-trees whose roots have tag table.

Segmentation: the first step pre-processes the input document as follows: in this case there is no dir attribute, so the table does not need to be flipped



Opinion polls [edit]

Date	Polling firm	Sample size	Calleja (Alianza Por un Nuevo País)	Martínez (FMLN)	Alvarado (VAMOS)	Bukele (GANA/Nuevas Ideas)
30 July 2018	CID-Gallup[6]	806	24	5	0	38
31 July 2018	TResearchMx[7]	3,600	31.7	9.7	2.8	55.8
19 August 2018	TResearchMx[8]	3,600	30.2	9.7	1.1	55.9
28 August 2018	UFG[9]	1,295	23.0	10.0	2.3	37.7
31 August 2018	LPG Datos[10]	1,520	17.6	8.6	0.3	21.9

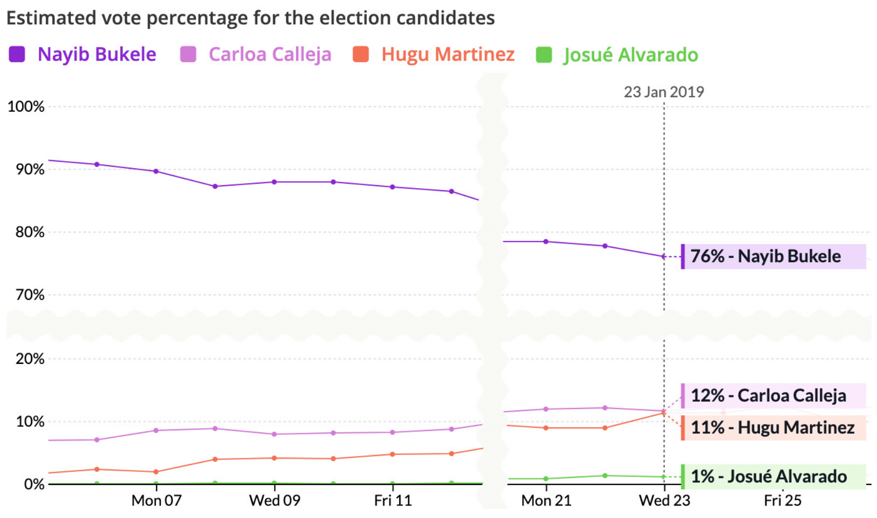
a) Sample Wikipedia table with opinion polls.

Date	**Polling firm**	**Sample size**	Carlos Calleja	Hugo Martínez		Nayib Bukele
Date	**Polling firm**	**Sample size**	Calleja (Ali...)	Martínez (FMLN)	Alvarado (VAMOS)	Bukele (GANA)
30 July 2018	CID-Gallup	806	24	5	0	38
31 July 2018	TResearchMx	3600	31.7	9.7	2.8	55.8
19 August 2018	TResearchMx	3600	30.2	9.7	1.1	55.9
28 August 2018	UFG	1295	23	10	2.3	37.7
31 August 2018	LPG Datos	1520	17.6	8.6	0.3	21.9

b) Results of functional analysis.

```
headers = [               tuples = [
  ("Date"),                 ("30 July 2018", "CID-Gallup", "806",
  ("Polling firm"),          "Carlos Calleja", "Calleja (Alianza ...)", "24"),
  ("Sample size"),          ("30 July 2018", "CID-Gallup", "806",
  ("$attr_0"),               "Hugo Martínez", "Martínez (FMLN)", "5"),
  ("$attr_1"),              ...
  ("$attr_2")             ]
]
```

c) Table after performing structural analysis.



d) Results of structural analysis.

Figure 3.10: *Sample opinion poll from Wikipedia.*

horizontally; neither are there any cells that span more than 200 rows or columns, so no correction must be performed; the three cells at the top-left corner span three rows each, so they are replicated accordingly (note that there are three rows at the top of the table: the first one has the pictures of the politicians, the second one has the corresponding party colour, and the third one has the names of the politicians and their parties); in this case all of the rows have the same number of columns, so no correction is performed; neither are there any duplicated rows or columns, so none of them needs to be removed. The second step computes the base features from the attributes of the DOM tree. The third step computes the deviations of the previous features on a per-row, per-column, and per-table basis. The fourth step identifies the empty cells and processes the factorised cells; note that the second row of the table consists exclusively of empty cells (with the party colours) and cell that have resulted from spanning the "Data", "Polling firm", and "Sample size" cells; that is, our heuristic removes it from the table; there are not any full-span rows or columns, so there are not any factorised cells to process in this example.

Discrimination: the table in our sample document must be passed on to the next task because it is visible to the user, it has more than one row and one column, does not have any ancestor table, and it does not contain any input elements.

Functional analysis: first, this task reduces the dimensionality of the features that represent the cells, next computes the candidate cell functionality, and then computes the final functionality. Figure 3.10.b illustrates its results. Note that the table is identified as a table-wise table, which is correct but not evident on a first sight. Realise that the six cells at the top-left corner are clearly meta-data cells that provide a semantic hint for the data cells that are below them; but the remaining cells in the corresponding rows do not actually provide any meta-data, but data that consists of the pictures of the politicians plus their names and the names of their parties.

Structural analysis: since the functional analysis identifies a unique block of meta-data cells at the top-left corner, the table is a matrix. The structural analysis task creates three headers from the meta-data cells, i.e., ("Date"), ("Polling firm"), and ("Sample size"), plus two artificial headers for the data cells in the first two rows, i.e., ("$attr_0") and ("$attr_1"), plus an additional artificial header for the cells in the body of the matrix, i.e., ("$attr_2"). The tuples combine each of the cells in the body of the matrix with their corresponding row and column indexers. Figure 3.10.c illustrates the results of this task.

Interpretation: this task leverages the results of the previous tasks and creates a record set in which the data in the tuples are explicitly associated with the descriptors that are computed from their corresponding headers. For instance, this is the first record returned by this task:

```
{
   "Date": "30 July 2018",
   "Polling firm": "CID-Gallup",
   "Sample size": "806",
   "$attr_0": "Carlos Calleja",
   "$attr_1": "Calleja (Alianza por un nuevo país)",
   "$attr_2": 24
}
```

The system shows the user the results of the interpretation task and helps him or her decide on which records and which of their components must be used to feed the machine-learning engine that is responsible for generating the forecasts, which are shown in charts like the one in Figure 3.10.d. More sophisticated automatic approaches are currently being evaluated [94, 121].

3.10 Summary

In this chapter we have presented TOMATE, which is an automated method to extract data from HTML tables. Given one or more input HTML documents, it locates the tables, segments them, discriminates the ones that provide data, analyses the function of their cells, finds their structure, and provides an interpretation as a record. Our most important contribution is regarding the functional analysis task: we project the cells of the input tables onto a feature space in which we first perform reduction and then clustering to find the candidate functions; the final functions are computed taking into account the orientation of the input tables.

TOMATE can deal with all of the common problems that we have found with other proposals in the literature: regarding the table layouts, it can deal with horizontal listings, vertical listings, and matrices; regarding the formatting problems, it can deal with tables that have multi-line headers, no headers at all, context cells, repeated headers, or factorised cells; regarding encoding problems, it can deal with tables that have inconsistent row lengths or tables that use tags td and th incorrectly.

Chapter 4

Experimental analysis

A good experimental study is mandatory to prove that our proposal performs correctly and outperforms the others in the literature, which is our goal in this chapter. It is organised as follows: Section 4.1 introduces this chapter; Section 4.4 presents the proposals with which we have confronted TOMATE; Section 4.2 reports on the datasets on which the experiments were run; Section 4.3 describes the variation points in TOMATE and how we configured them; Section 4.5 presents our results; Section 4.6 confirms that our results are statistically sound; and Section 4.7 summarises our conclusions.

4.1 Introduction

Our experiments consisted in running TOMATE and the proposals that we presented previously on our two datasets. The proposals were implemented using the Python 3.6 language, Selenium, BeautifulSoup, and the Scientific Python ecosystem. The experiments were run on a Windows 10 Pro computer that was equipped with an AMD Ryzen 5 3600X processor with six 3.79 GHz cores and 16 GiB of DDR4 RAM memory.

We addressed the evaluation using the 3-fold cross method. In the case of TOMATE and the proposals by Yoshida et al. [136], Jung and Kwon [64], and Embley et al. [40] the training splits were ignored because these proposals do not require to learn any models; in the case of Nishida et al.'s [96] proposal, the training splits were used to fit their neural network.

We measured the performance of the proposals in terms of effectiveness and efficiency. Regarding effectiveness, we computed precision (P), recall (R), and the F_1 score (F_1); regarding efficiency, we computed the average number of CPU seconds to process each table (Time). The effectiveness measures were computed by averaging the results on the three testing splits that we got from each dataset. The efficiency measures were also averaged across the three testing splits; in the case of Nishida et al.'s [96] proposal, the training time was apportioned amongst all of the tables in the corresponding testing splits.

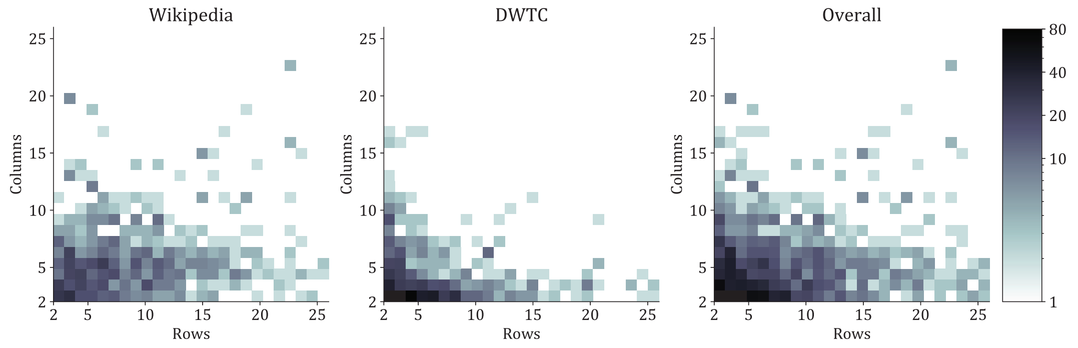
To confirm that our conclusions are sound, we performed a statistical analysis at the standard confidence level. The ultimate goal of such an analysis is to compute a ranking of proposals in which the differences in rank are proven to be statistically significant. Simply put, the goal is to discern if the differences regarding a particular performance variable are inherent to a particular proposal or an unfortunate consequence of outliers that may distort the average. We used a common approach in the literature to perform the analysis [47, 112], namely: first, the empirical ranking is computed using each of the performance measures; second, an omnibus test is used to determine if there are any significant differences in the empirical ranks; if there are, then it proceeds to perform a post-hoc test to compare the proposals to each other and find out which exact differences in rank are statistically significant. Which tests must be used depends on whether the experimental results are normally and homocedastically distributed or not, which can be checked using Shapiro-Wilk's and Levene's tests, respectively. If they are, one can use ANOVA-F as the omnibus test and Dunnett's as the post-hoc test; otherwise, one can use Iman-Davenport's as the omnibus test and Bergman-Hommel's as the post-hoc test.

4.2 Datasets

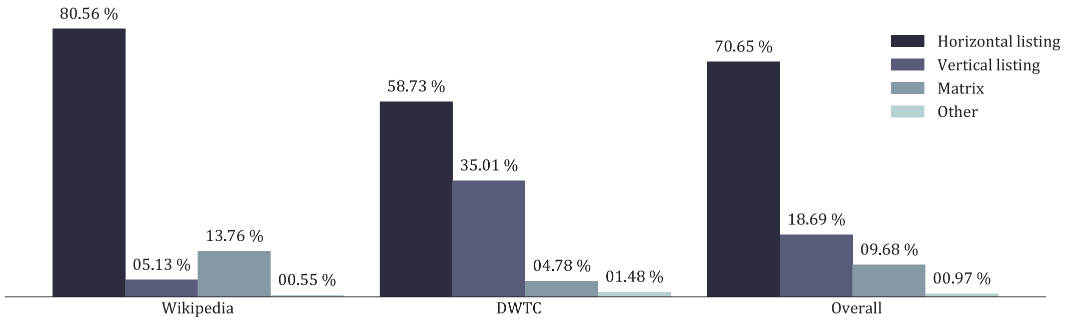
We assembled two large experimental datasets with real-world tables that were fetched from the English Wikipedia [128] and the subset of English documents in the Dresden Web Table Corpus [35]. We sampled enough documents from these repositories to assemble two datasets with 1 496 and 1 513 tables, respectively. In the case of the Wikipedia, we only selected user-generated tables, since the data in tables that are generated using templates like infobox, navbox, or sistersitebox can be readily extracted by major knowledge bases.

We created a ground truth from the previous datasets with the help of four independent annotators. They were presented the tables found and the segmentation computed by TOMATE; they had to classify the tables according to their layout as horizontal listings, vertical listings, matrices, or unknown, and their cells as either meta-data or data cells. Their agreement to annotate the ground truth was very good: they annotated 84.24% of the tables with the same layout and 97.58% of the cells with the same functionality; the Krippendorff Alpha coefficients were 80.36% and 96.11%, respectively. This degree of agreement was considered very good and enough to trust our ground truth for evaluation purposes. The relative disagreement regarding table layouts was mainly due to matrices, since the authors of the original tables used a variety of formats that made it difficult to agree on whether they were actual matrices or listings in many cases; however, the agreement regarding the cell functionality is almost perfect, which makes our datasets excellent to assess how the compared proposals perform on real-world tables.

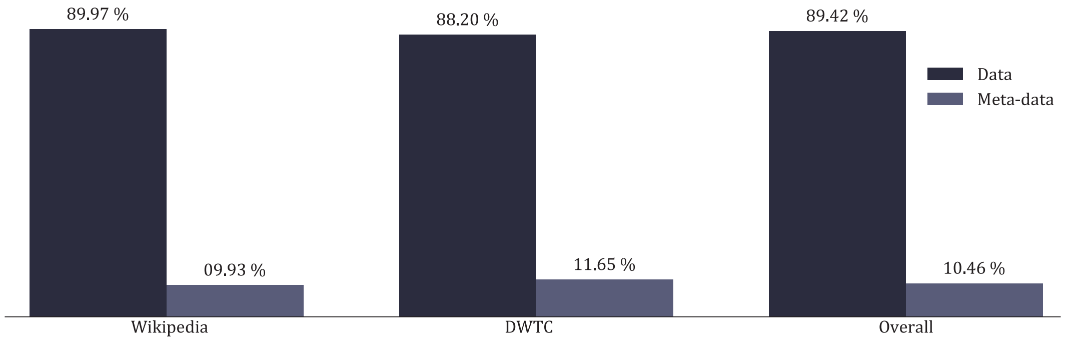
Figure 4.1.a reports on the dimensions of the tables in terms of rows and columns: the darker a dot, the more tables with the corresponding dimensionality, but realise that the scale of colours is exponential due to the large number of tables in our dataset. Note that most of the tables have less than 10 columns and less than 10 rows and that the DWTC tables are usually smaller and more unbalanced regarding their dimensions. Figure 4.1.b reports on the table layouts in our datasets, namely: 70.65% of the tables are horizontal listings, 18.69% are vertical listings, 9.68% are matrices, and 0.97% of the tables have an unknown layout. Clearly, horizontal listing is the most common layout, chiefly in the case of the Wikipedia dataset. Figure 4.1.c reports on the cells in our datasets: overall, there are 190 600, 89.42% of which are data cells and 10.46% of which are meta-data cells. This distribution is not surprising since, typically, a single meta-data cell can be used to endow many more data cells with semantics.



a) Number of tables according to their dimensions.



b) Frequency of table layouts.



c) Frequency of cell functions.

Figure 4.1: *Comparative statistics about our datasets.*

Table 4.1 provides an additional insight into the formatting and encoding problems considered in this article. Note that 64.41% of the tables have one or more extraction problems, which is a very large percentage.

Regarding the formatting problems, it is interesting to note that 14.39% of the tables have multi-line headers, but the figure raises to 22.54% of the tables in the Wikipedia dataset; the reason might be that the authors of

Source	Problem	Percentage of tables		
		Wikipedia	DWTC	Overall
Formatting	Multi-line headers	22.54%	6.15%	14.39%
	No headers	4.69%	13.70%	9.17%
	Context cells	13.62%	6.62%	10.14%
	Factorised cells	0.79%	2.87%	1.82%
	Repeated headers	0.79%	0.87%	0.83%
Encoding	Inconsistent row lengths	25.38%	3.94%	14.72%
	Meta-data cells with td tag	16.97%	53.15%	34.96%
	Data cells with th tag	20.70%	2.68%	11.74%
	At least one problem	58.89%	69.99%	64.41%

Table 4.1: *Frequency of table extraction problems by dataset.*

Wikipedia tables tend to provide a homogeneous format and they commonly use structured headers that provide as much context as possible to interpret the corresponding data cells; in the case of the DWTC dataset, only a 6.15% of the tables rely on multi-line headers; the reason might be that the tables were gathered from many different sites in the Web, which means that they are not homogeneous at all. The previous hypothesis might be partially supported by the percentage of tables with no headers: note that only 4.69% of the tables in the Wikipedia dataset do not provide any headers, whereas this indicator raises to 13.70% in the case of the DWTC dataset. Finally, note that there are about 1.82% and 0.83% of tables that have factorised cells or repeated headers; the figures are not as large as was the case with the previous problems, but they definitely contribute to the number of different formatting problems with which a good proposal to extract data from tables must deal.

Regarding the encoding problems, it is interesting to realise that 25.38% of the tables in the Wikipedia dataset and 3.94% of the tables in the DWTC dataset have inconsistent row lengths; this is clearly due to the fact that the tables in the Wikipedia dataset were encoded by a person, whereas many of the tables in the DWTC dataset were encoded by a machine. It is also interesting to realise that 34.96% of the meta-data cells are encoded using td tags and 11.74% of the data cells are encoded using th tags. These figures are not surprising in the case of the Wikipedia tables because they are encoded manually; in the case of the DWTC the conclusion is that developers seem to forget that they must use the th tag to encode meta-data cells (note that the figure raises to 53.15% of the tables in this dataset). Our conclu-

Variation point	Description	Alternative	P	R	F_1	Time
Normalisation	Process to normalise features to the [0, 1] interval.	Min-max global	**73.60%**	**88.31%**	**77.77%**	**1164.60**
		Min-max local	70.77%	85.06%	74.41%	1181.79
		Standard	66.17%	81.02%	68.29%	1216.21
		SoftMax	70.91%	82.23%	74.02%	1235.13
Feature selection	Feature groups used as input for the clustering algorithm.	Style	67.02%	80.53%	69.41%	1155.20
		Lexical	67.42%	81.73%	69.92%	1135.26
		Structural	70.34%	84.21%	73.76%	**1040.06**
		Misc.	68.26%	82.88%	71.16%	1056.46
		Style, lexical	68.61%	82.72%	71.50%	1286.45
		Style, structural	71.87%	84.79%	75.36%	1221.37
		Style, misc.	69.95%	83.92%	73.20%	1201.84
		Lexical, structural	70.88%	84.80%	74.44%	1177.58
		Lexical, misc.	68.56%	83.27%	71.53%	1161.94
		Structural, misc.	72.02%	85.28%	75.57%	1098.79
		Style, lexical, structural	71.84%	85.03%	75.35%	1323.50
		Style, lexical, misc.	69.77%	84.01%	72.96%	1342.89
		Style, structural, misc.	**72.57%**	**85.59%**	**76.15%**	1232.07
		Lexical, structural, misc.	71.65%	85.19%	75.19%	1196.99
		All	72.28%	85.53%	75.81%	1353.56
Dimensionality reduction	Reduction applied to the input before clustering.	None	69.64%	83.36%	72.70%	1223.35
		PCA	70.05%	83.86%	73.22%	1235.08
		Feature agglomeration	**70.75%**	**84.45%**	**74.13%**	**1145.16**
Clustering method	Method applied to cluster the cells.	K-means	70.10%	83.65%	73.29%	1288.54
		Agglomerative	**70.18%**	**84.09%**	**73.39%**	**1121.15**

Table 4.2: *Average performance of each alternative to configure TOMATE.*

sion is that these figures clearly justify the need for specific-purpose methods to identify the functionality of the cells.

4.3 Variation points

We performed a grid search to find the best alternatives to implement the variation points in TOMATE. There are several variation points in TOMATE, which allow to fine tune it. We experimented on both datasets, performing a grid search to implement the variation points in TOMATE. Recall that there are four variation points, namely: how to normalise the features (four alternatives), which categories of features must be selected to produce the candidate functionalities (fifteen alternatives), which technique must be used to reduce their dimensionality (three alternatives), and which technique must be used to perform the clustering (two alternatives). This results in a total of 360 configurations that were explored in sequence.

Table 4.2 shows a summary with the average performance attained by each alternative on our experimental datasets, which was computed as the average performance in all of the experiments in which that alternative was used.

Regarding normalisation, we found that the min-max global alternative was the best. We realised that min-max local normalisation or the SoftMax normalisation usually give more importance to features that are not significant for a person, e.g., padding, which usually ranges from 1 to 5 pixels. We also realised that the standard normalisation works significantly worse because there are many feature values that are normalised as $+1.00$, which results in clusters that group cells that are actually very different.

Regarding feature selection, we found that there are two good alternatives, namely: using the four categories of features together or leaving the lexical features out. Furthermore, we observed that using a single feature group does not suffice in most experiments because each group of features provides some discriminatory power that cannot be replaced by any of the other groups.

Regarding feature reduction, we found that using a dimensionality reduction technique improves the results. The best result was obtained using feature agglomeration, both regarding effectiveness and efficiency.

Regarding the clustering method, we found that both alternatives provide fairly similar results. However, the performance of the agglomerative clustering method is slightly better than k-means. We profiled our implementation and we found out that the difference was due to the initialisation procedure in k-means.

The previous analysis helps have an overall understanding of how each alternative performs. To make a decision, we need to analyse the top configurations. Table 4.3 reports on the top 10 configurations according to their F_1 score. The best two alternatives provide very similar precision, recall, and F_1 score, but the second one is faster. We profiled our implementation and we found out that the reason is that the second alternative does not require to perform any regular expression match because it does not use any lexical features. The decision regarding which of the alternatives should be selected was difficult. We finally leaned towards the top one because we estimated that difference regarding efficiency would not be significant for practical purposes, but the difference regarding effectiveness would be.

Normalisation	Feature selection	Dimensionality reduction	Clustering method	P	R	F_1	Time
Min-max global	All	PCA	K-means	**76.93%**	**90.89%**	**81.49%**	**1489.09**
Min-max global	Style, structural, misc.	Feature agglomeration	Agglomerative	76.91%	90.74%	81.45%	1006.14
Min-max global	All	None	K-means	76.66%	90.74%	81.21%	1426.72
Min-max global	Style, structural, misc.	None	K-means	76.31%	89.96%	80.75%	1275.88
Softmax	Style, structural, misc.	Feature agglomeration	K-means	77.38%	85.92%	80.73%	1394.67
Min-max global	Style, lexical, structural	Feature agglomeration	Agglomerative	75.99%	90.13%	80.47%	1092.55
Min-max global	Style, structural, misc.	Feature agglomeration	K-means	75.92%	90.18%	80.41%	1295.06
Min-max global	Style, lexical, misc.	None	K-means	75.77%	90.13%	80.25%	1362.69
Softmax	Style, structural	Feature agglomeration	Agglomerative	76.61%	85.93%	80.14%	1072.34
Min-max global	Style, structural	Feature agglomeration	K-means	75.83%	88.78%	80.10%	1220.67

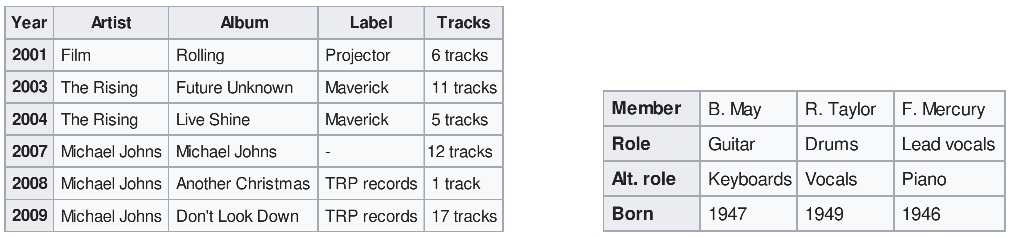
Table 4.3: *Top* 10 *TOMATE configurations according to the* F_1 *score.*

4.4 Compared proposals

Assuming that several proposals are confronted with exactly the same tables, their overall effectiveness depends completely on their ability to perform the functional analysis task correctly, since the structural analysis and the interpretation tasks can be implemented using a common procedure. Our experimental approach borrowed the location, segmentation, and discrimination tasks from TOMATE, since this allowed us to confront different proposals on exactly the same tables; we then implemented the functional analysis task using the proposals by Yoshida et al. [136], Jung and Kwon [64], Embley et al. [40], and Nishida et al. [96], which resulted in different subsets of meta-data and data cells; finally, we applied the structural and interpretation procedures in TOMATE to extract the data. Below, we provide an overall picture of the competitors; the reader should consult the references to learn the details.

Yoshida et al. [136] devised an Expectation Maximization method in which cell functions and table layouts are iteratively adjusted using a probability model that relies on the contents of the cells. The authors did not describe the initialisation strategy or the stopping criterion used, so we resorted to a common approach in the literature: we initialised the probabilities with random values, we adjusted them in 10 iterations, we repeated the process 100 times, and we kept the best result only.

Jung and Kwon [64] proposed seven heuristics to identify meta-data cells. They rely on style and syntax features, and each of them results in a binary table in which the cells are labelled as either data or meta-data. Then, these



Year	Artist	Album	Label	Tracks
2001	Film	Rolling	Projector	6 tracks
2003	The Rising	Future Unknown	Maverick	11 tracks
2004	The Rising	Live Shine	Maverick	5 tracks
2007	Michael Johns	Michael Johns	-	12 tracks
2008	Michael Johns	Another Christmas	TRP records	1 track
2009	Michael Johns	Don't Look Down	TRP records	17 tracks

Member	B. May	R. Taylor	F. Mercury
Role	Guitar	Drums	Lead vocals
Alt. role	Keyboards	Vocals	Piano
Born	1947	1949	1946

a) Table with a highlighted numeric column. b) Table with non-repeated values.

Figure 4.2: *Sample tables from Wikipedia.*

tables are combined using a linear interpolation method whose weights were estimated by the authors using a hill-climbing procedure.

Embley et al. [40] devised an algorithm whose emphasis is on identifying a number of critical cells that help delimit the header and the data areas of a table. The clever part of this proposal is the procedure that shifts the critical cells so as to identify which regions of the table can be used as indexers for the other regions, which basically helps make meta-data cells apart from data cells.

Nishida et al. [96] proposed a deep neural network that encodes the cells as fixed-size vectors using a long short-term memory approach to compute their semantic representations. Then, a third-order tensor that represents the table was connected to a convolutional layer and a sequence of filters to compute a likelihood for every table layout. The cell functions can then be easily inferred according to the table layout.

4.5 Experimental results

Table 4.4 presents our experimental results. For each proposal, we report on its performance measures in the Wikipedia and the DWTC datasets independently, plus its overall average performance on both datasets. We first group the results according to the table layout and then present the aggregated results on all layouts. The best results in each group are highlighted in boldface.

Our proposal attains the best overall precision, recall, and F_1 score in both datasets. Regarding horizontal listings, Nishida et al.'s [96] proposal attains a better precision and F_1 score in the Wikipedia dataset, since simple horizontal

Layout	Proposal	Wikipedia				DWTC				Overall			
		P	R	F_1	Time	P	R	F_1	Time	P	R	F_1	Time
Horizontal listing	P0	90.52%	**95.25%**	91.43%	0.12	**85.07%**	**87.45%**	**84.01%**	0.09	**88.06%**	**91.72%**	**88.08%**	0.11
	P1	66.45%	73.35%	66.41%	0.53	63.73%	68.68%	63.66%	0.29	65.22%	71.24%	65.17%	0.42
	P2	70.08%	74.95%	70.33%	0.01	47.69%	56.76%	50.77%	**0.00**	59.97%	66.74%	61.50%	0.01
	P3	62.57%	79.12%	58.80%	**0.00**	65.19%	70.17%	57.72%	**0.00**	63.75%	75.08%	58.31%	**< 0.01**
	P4	**91.97%**	94.86%	**92.07%**	9.26	74.76%	75.53%	73.77%	9.41	84.20%	86.13%	83.81%	9.33
Vertical listing	P0	**88.05%**	**91.29%**	**87.99%**	0.05	**93.09%**	**93.78%**	**92.41%**	0.02	**90.33%**	**92.41%**	**89.99%**	0.04
	P1	44.52%	50.08%	43.60%	0.53	46.86%	49.47%	45.68%	0.29	45.58%	49.80%	44.54%	0.42
	P2	46.47%	58.53%	48.76%	0.00	28.59%	49.90%	35.18%	**0.00**	38.40%	54.63%	42.63%	**< 0.01**
	P3	77.46%	83.27%	73.72%	**0.00**	83.32%	81.76%	79.28%	**0.00**	80.11%	82.59%	76.23%	**< 0.01**
	P4	61.65%	64.16%	57.09%	9.26	88.73%	89.42%	87.90%	9.41	73.88%	75.56%	71.00%	9.33
Matrix	P0	**90.19%**	**96.76%**	**91.13%**	0.12	93.98%	94.36%	92.75%	0.09	**91.90%**	95.68%	91.86%	0.11
	P1	54.38%	59.68%	51.23%	0.53	51.20%	59.09%	51.87%	0.30	52.94%	59.41%	51.52%	0.43
	P2	58.48%	65.69%	56.40%	0.00	47.55%	58.26%	48.94%	0.00	53.55%	62.34%	53.03%	**< 0.01**
	P3	87.77%	96.11%	89.28%	**0.00**	**94.52%**	**98.01%**	**95.84%**	**0.00**	90.82%	**96.97%**	**92.24%**	**< 0.01**
	P4	83.43%	81.01%	75.69%	9.27	83.39%	77.58%	76.15%	9.40	83.41%	79.46%	75.90%	9.33
All	P0	**90.34%**	**95.24%**	**91.21%**	0.11	**88.36%**	**90.04%**	**87.43%**	0.07	**89.45%**	**92.89%**	**89.50%**	**0.09**
	P1	63.56%	70.16%	63.04%	0.53	57.12%	61.39%	56.69%	0.29	60.65%	66.20%	60.17%	0.42
	P2	67.17%	72.76%	67.20%	0.01	40.89%	54.40%	45.14%	0.00	55.30%	64.47%	57.24%	0.01
	P3	66.89%	81.71%	63.86%	**0.00**	73.06%	75.65%	67.24%	**0.00**	69.68%	78.97%	65.39%	**< 0.01**
	P4	89.11%	91.24%	87.86%	9.26	80.15%	80.57%	78.91%	9.41	85.06%	86.42%	83.82%	9.33

Legend: P0 = TOMATE; P1 = Yoshida et al.'s [136] proposal; P2 = Jung and Kwon's [64] proposal; P3 = Embley et al.'s [40]; P4 = Nishida et al.'s [96] proposal.

Table 4.4: *Experimental results of table extraction proposals.*

listings are much more frequent. We have found that the problem with this layout is that our orientation detection method tends to classify some horizontal listings as matrices when the first rows and columns are very different from the others; for instance, see the "Year" column in the table in Figure 4.2.a.

The proposal by Yoshida et al. [136] did not attain the best result regarding any of the measures since it builds on the assumption that the meta-data cells usually have the same common contents across different tables, but the variety of topics covered by Wikipedia tables does not meet this assumption. Their frequency-based approach also has problems when processing cells with numeric contents, since many of them occur only once in the datasets.

The proposal by Jung and Kwon [64] works reasonably well on horizontal listings, but it is less effective when processing other layouts. We found that the main source of errors are some data cells that the user highlights in a way that resembles meta-data cells; for instance, see the "Year" column in the table in Figure 4.2.a. Style and structural features provide good hints to identify the function of a cell, but they are very dependent on an author's preferences. Contrarily, lexical features are less dependent on such preferences.

The proposal by Embley et al. [40] is significantly faster than the others, but it has some problems when dealing with listings. We found that tables with no repeated contents are interpreted as matrices with one row and one column of headers, e.g., see the table in Figure 4.2.b. This technique was devised to extract data from spreadsheets by analysing the column and row content repetitions, and trying to find a set of rows and columns that index the rest of the data. However, the tables in our dataset do not typically have many repeated values and matrices are not the most frequent layout.

The proposal by Nishida et al. [96] is the one that obtains the closest results to TOMATE. However, it seems to be the most inefficient because it relies on using neural networks that are very complex from a computational point of view. This proposal works well when extracting tables with a layout that is similar to other tables in the training set. Note that it attains good results with the tables in the Wikipedia dataset in which there are many similar tables regarding the climate, sports results, books, or TV shows, just to mention a few examples.

4.6 Statistical analysis

In this section, we analyse our experimental results from a statistical point of view [47, 112]. Our goal is to compute a ranking of the proposals that we have compared and to make sure that it is statistically sound.

We first checked our experimental results for normality and homoscedasticity using Shapiro-Wilk's and Levene's tests, respectively. We do not provide a specific chart regarding the results of these tests because the p-value that we got was zero or nearly zero in every case, which is a strong indication that the experimental results are neither normal nor homoscedastic. It then proceeds to use Iman-Davenport's as the omnibus test and Bergman-Hommel's as the post-hoc test.

Tables 4.5.a–d show the results of the post-hoc tests regarding our performance measures. In each case, we present the empirical ranking that is computed from our experimental results in the first two columns; the next two columns present the F statistic that is computed by Iman-Davenport's test and its corresponding p-value; the following three columns report on the proposals compared, the Z statistic that is computed by Bergman-Hommel's test and its corresponding p-value. The cells that correspond to

Ranking		Iman-Davenport		Bergmann-Hommel		
Proposal	Rank	F	P-Value	Comparison	Z	P-value
P0	1.94	970.98	0.0E+00	P0 - P1	0.37E+02	0.22E-307
P4	2.29			P0 - P2	0.42E+02	0.22E-307
P3	3.38			P0 - P3	0.33E+02	0.22E-307
P1	3.60			P0 - P4	0.78E+01	0.14E-13
P2	3.79			P1 - P2	0.44E+01	0.97E-05
				P1 - P3	0.48E+01	0.13E-05
				P1 - P4	0.30E+02	0.72E-191
				P2 - P3	0.93E+01	0.76E-19
				P2 - P4	0.34E+02	0.10E-251
				P3 - P4	0.25E+02	0.80E-134

a) Analysis of precision.

Ranking		Iman-Davenport		Bergmann-Hommel		
Proposal	Rank	F	P-Value	Comparison	Z	P-value
P0	2.01	805.34	0.0E+00	P0 - P1	0.37E+02	0.22E-307
P4	2.34			P0 - P2	0.37E+02	0.22E-307
P3	3.35			P0 - P3	0.30E+02	0.39E-200
P2	3.65			P0 - P4	0.75E+01	0.24E-12
P1	3.66			P1 - P2	0.66E-01	0.95E+00
				P1 - P3	0.69E+01	0.12E-10
				P1 - P4	0.30E+02	0.19E-192
				P2 - P3	0.69E+01	0.12E-10
				P2 - P4	0.30E+02	0.68E-192
				P3 - P4	0.23E+02	0.10E-113

b) Analysis of recall.

Ranking		Iman-Davenport		Bergmann-Hommel		
Proposal	Rank	F	P-Value	Comparison	Z	P-value
P0	1.96	863.15	0.0E+00	P0 - P1	0.36E+02	0.46E-280
P4	2.32			P0 - P2	0.37E+02	0.22E-307
P1	3.55			P0 - P3	0.36E+02	0.75E-286
P3	3.56			P0 - P4	0.82E+01	0.13E-14
P2	3.61			P1 - P2	0.13E+01	0.56E+00
				P1 - P3	0.38E+00	0.70E+00
				P1 - P4	0.28E+02	0.13E-167
				P2 - P3	0.94E+00	0.56E+00
				P2 - P4	0.29E+02	0.19E-183
				P3 - P4	0.28E+02	0.48E-172

c) Analysis of the F_1 score.

Ranking		Iman-Davenport		Bergmann-Hommel		
Proposal	Rank	F	P-Value	Comparison	Z	P-value
P3	1.43	917.75	0.0E+00	P0 - P1	0.23E+02	0.10E-113
P2	1.64			P0 - P2	0.30E+02	0.99E-192
P0	2.96			P0 - P3	0.34E+02	0.11E-258
P1	3.97			P0 - P4	0.46E+02	0.22E-307
P4	5.00			P1 - P2	0.52E+02	0.22E-307
				P1 - P3	0.57E+02	0.22E-307
				P1 - P4	0.23E+02	0.38E-119
				P2 - P3	0.48E+01	0.14E-05
				P2 - P4	0.76E+02	0.22E-307
				P3 - P4	0.81E+02	0.22E-307

d) Analysis of time.

Legend: P0 = TOMATE; P1 = Yoshida et al.'s [136] proposal; P2 = Jung and Kwon's [64] proposal; P3 = Embley et al.'s [40]; P4 = Nishida et al.'s [96] proposal.

Table 4.5: *Statistical analysis of table extraction proposals.*

p-values that are smaller than the standard significance level ($\alpha = 0.05$) are highlighted in grey to facilitate the interpretation.

Note that TOMATE ranks the first regarding precision, recall, and the F_1 score, but the third regarding time. Iman-Davenport's test returns a p-value of 0.00 in every case, which is a very strong indication that there are experimental differences in the empirical rankings. It then proceeds to apply Bergman-Hommel's test to compare the empirical rank of every proposal to every other. Note that the differences are statistically significant in almost every case, which confirms that TOMATE actually ranks the first regarding precision, recall, and F_1 score and also that the differences in rank regarding to the techniques immediately after is statistically significant.

Unfortunately, the improvement regarding effectiveness comes at the cost of some inefficiency when compared to the other proposals. The empirical ranking establishes that Embley et al.'s [40] proposal (P3) is the quickest one, but realise that it ranks at the fourth position regarding the effectiveness measures; it also establishes that Jung and Kwon's [64] proposal (P2) ranks at the second position regarding efficiency, but it ranks the third regarding the effectiveness measures. Bergman-Hommel's test confirms that the previous differences in rank are statistically significant.

Summing up: the statistical analysis confirms that TOMATE is 5.68% more effective than the second best proposal in the ranking, which is a supervised one. Note, too, that it is 24.11% better than the third proposal in the ranking, which does not require any supervision. This improvement makes it a bit more inefficient, but it takes an average of 0.09 CPU seconds to process each of the tables in our datasets, which we think is good for practical purposes.

4.7 Summary

In this chapter, we have compared TOMATE with 5 state-of-the-art competitors. First, we have performed a grid search in order to find the best configuration of TOMATE. Then, we have evaluated every alternative with two large table datasets, to measure both their efficiency and their performance overall and in a per-layout basis. Our proposal attains the best overall precision, recall, and F_1 score in both datasets, and our statistical analysis confirmed that TOMATE ranks the best among the compared proposals. Note that it is 24.11% better than the next proposal in the ranking which does not require any supervision and 5.68% more effective than the best supervised one. This improvement makes it a bit slower, but it takes an average of 0.09 CPU seconds to process each of the tables in our datasets, which we think is good for practical purposes.

Chapter 5 Conclusions

The Web has become one of the most common communication means. It provides a vast repository of data about a plethora of topics. Many such data are encoded using human-friendly HTML tables, which makes it difficult to integrate them into automated business processes. There has been a rapid rise of data-hungry products and services in the recent years, which have motivated the need for automatic web data extractors.

In this dissertation, we have studied the problem of extracting data from HTML tables with no supervision. Our extensive review of the literature revealed that none of the available proposals provided a holistic approach to solve this problem since most of them have difficulties to deal with the long tail of challenges to be taken into account. This motivated us to work on a proposal to overcome them.

We have devised and implemented TOMATE, which is a table extraction proposal that encompasses every task, from locating the HTML excerpts containing tables to transforming them into schema-less records that can be easily adapted for the application of interest. It focuses on the crucial cell function analysis step by clustering the cells according to their features. In future, we would like to explore unsupervised learning approaches as a way to deal with the long tail of minor problems without exploring each of them individually.

Furthermore, we produced some marginal contributions while working on the main ones. Initially, we designed AQUILA, a supervised general-purpose extraction tool intended to synthesise meta-data tags for HTML documents. The amount of effort required to build and curate annotations motivated us to work on Kizomba, an unsupervised take on the previous problem that is based on identifying recurrent representation patterns in the Web. Web tables were one of the most frequent patterns, and while studying how to extract them we realised that their difficult encodings required a specific-purpose proposal. During the development of TOMATE, we studied better approaches to feature selection and clustering, which motivated us to work on Romulo, which is a multi-way single-subspace approach to clustering.

Last, but not least, we would like to highlight that the work that was carried out during the development of the previous results led to setting up a start-up company in an international context.

Summing up, assuming that our research hypothesis is accepted, we think that we have sufficiently proven our thesis. We hope that our results can effectively help companies reduce the effort involved in obtaining up-to-date data about a plethora of topics. We also think that we have opened up an interesting research path that may soon lead to new research results.

Appendix A

Aquila: synthesis of meta-data tags for HTML files

Aquila is a supervised proposal to generate meta-data tags from HTML documents. It is organised as follows: Section A.1 introduces our proposal; Section A.2 reports on the related work; Section A.3 provides an overall picture of its design; Section A.4 reports on our experiments; finally, Section A.5 presents our conclusions and some future work.

A.1 Introduction

The data in an HTML document can be easily endowed with semantics by using meta-data tags that can be encoded using RDFa, JSON-LD, Microdata, or Microformats. Such tags either provide links to the corresponding entities in an external knowledge graph or to the corresponding types in a vocabulary. Simply put, they pave the way for software agents that can easily sift through HTML documents and understand their data, which is particularly interesting in the case of search engines. Unfortunately, a recent analysis of the November 2018 Common Crawl revealed that only 29.20% of the domains provide meta-data tags [11].

In the literature, there are a few approaches to overcome this problem, namely: proposals to synthetise meta-data tags that target Content Management Systems (CMS) [5, 93], text editors that authors can use to add meta-data tags to their documents [37, 52], proposals that attempt to link data records in an HTML document to entities in an external knowledge graph [16, 36], and a recent proposal [98] that can identify new entities in tables that are not in the external knowledge graph and map their properties onto the corresponding vocabulary. However, none of these proposals provides a general solution to the problem of synthetising meta-data tags for HTML documents with arbitrary structures. This motivated researchers to exploring graph embedding techniques [20, 49, 126] because the problem of synthesising meta-data tags in a DOM tree can be straightforwardly mapped onto a node classification problem. Unfortunately, the results were not good at all since the best F_1 score attained on our evaluation dataset was 0.69.

The previous problems inspired us to work on a new proposal to synthesise meta-data tags that links the data in an HTML document to their types in a vocabulary. It targets documents that provide semi-structured data and it does not require any external resources. It builds on a new graph embedding technique that preserves the original attributes of the nodes and does not require to preset an embedding size. Basically, it finds a common path in the DOM trees of a set of HTML documents that identifies a subset of DOM nodes whose attributes help learn a good tagger. We have experimented with four state-of-the-art embedders on a repository with 40 datasets on eight topics that were collected from a variety of popular sites; our conclusion is that our proposal can attain an F_1 score as high as 0.78, whereas the best F_1 score attained by the other embedders is 0.69. Summing up, we think that ours is a novel and promising approach to the problem.

A.2 Related work

A few natural language tagging proposals provide a solution to this problem. Dodero et al. [33] described a methodology that can be used to re-engineer a software system so that it produces HTML documents with meta-data tags. However, we restrict our attention to proposals that can be plugged into existing systems. Adrian et al. [5] presented a method that matches noun phrases in an HTML document onto a knowledge graph, Ngomo et al. [93] presented the architecture of a module that can be integrated in a CMS so that it can generate HTML documents with meta-data tags, Eldesouky et al. [37] and Guerrero-Contreras et al. [52] decribed editors that helps content producers include meta-data tags in their text without requiring any technical knowledge. Unfortunately, these proposals cannot be applied in our context because they all rely on natural-language components, but do not work well with semi-structured data, which is a non-grammatical encoding.

More recent tagging approaches focus on semi-structured data. Burget [16] devised a proposal that computes a collection of tags for non-overlapping rectangular areas of HTML documents. Efthymiou et al. [36] focused on HTML tables by using three basic approaches that were also hybridised, namely: a look-up method, a word-embedding method, and an ontology-matching method. Oulabi and Bizer [98] presented a proposal focused on HTML tables that performs schema matching and clustering in an attempt to augment an external knowledge graph. Unfortunately, these proposals require a knowledge graph per domain and all of the entities in the input HTML document must be recorded in that knowledge graph. Furthermore, the proposals that focuses on tables [36, 98] assume that data are organised in rows, each column corresponds to an attribute, and there are headers that provide hints on the attributes.

We also studied graph embedding proposals [20, 49, 126], since the problem of synthesising meta-data tags in a DOM tree can be straightforwardly mapped onto a node classification problem. We have implemented a tagger using some state-of-the-art graph embedders but the results were not good at all since the best F_1 score attained on our evaluation dataset was 0.69. We believe that the problem with such embedders is that they just attempt to preserve the distance between the nodes connected by an edge, but the classification power of the original attributes is lost in the embedding.

A.3 Our proposal

Aquila is intended to learn a meta-data tagger (or tagger for short), which is a classifier that works on an embedding of a node; the embedding, in turn, is a map with its attributes and the attributes of its neighbours along a given path. The key of our proposal is to find the path that results in an embedding from which the best possible classifier can be learnt.

To do so, it first learns a classifier given a ground truth and a path. The documents in the ground truth have annotations, which are mappings that make it explicit the meta-data tag that corresponds to each DOM node together with its attributes. The documents in the ground truth are split into a learning set and a validation set. To learn a classifier it uses a base learner [129] on the learning set and assesses how good it is on the documents in the validation set; the assessment is computed as a normalised score in interval $[0.00 \,..\, 1.00]$ (the greater, the better). Note that the classifier returned by the first step is learnt using an empty path, which means that it is learnt from the attributes of the nodes themselves.

Then, it iteratively attempts to extend the path by finding neighbours whose attributes help learn a better classifier. On each iteration, it explores the links that extend the current path to the parent node, the left node, and the right node. For each such extended path, it learns a new classifier building on the attributes of the nodes and their neighbours along the extended path. If the score of the new classifier is greater than the previous one, it is then kept; otherwise, it is discarded.

The method returns the best classifier found and its corresponding path. Given a new document, it must be transformed into a embedding set with the attributes of the nodes and their neighbours along the path and then the classifier must applied in order to predict a meta-data tag for each node. Note that it finishes when no extension of the current path allows to learn a better classifier. This is guaranteed to happen because the method works on DOM trees, which are finite by definition. That means that the method eventually reaches the root node, which does not have any parent, left, or right siblings.

A.4 Experimental analysis

Figure A.1.a summarises the datasets that we used to evaluate our proposal. Each dataset consists of 30 documents that were downloaded from a

Schema (from Schema.org)		# DOM	# Attr.	# Node	# Attr.
Name	**Attributes**	**Nodes**	**Values**	**edges**	**edges**
Book	Name, author, ISBN.	175749	83272	198704	1354053
Car	Color, number of doors, vehicle engine, model, mileage from odometer, number of forward gears.	255822	79281	106587	515670
Doctor	Name, address, telephone, additional type.	143794	43154	187690	914165
Event	Name, start date, location, URL.	59257	40047	231132	1081792
Movie	Name, director, actor, copyright year, duration.	171187	93968	155676	800797
JobPosting	Company, job location, ocupational category.	158810	75828	244218	1059515
Person	Name, birth date, height, weigth, nationality.	300321	68661	259529	1182927
Property	Address, number of rooms, floor size.	331154	90464	178223	1076395

a) Description of our web data extraction datasets.

Base tagger		P1	P2	P3	P4	P5	Base tagger		P1	P2	P3	P4	P5
Bayes network	P	0.14	0.15	0.16	0.15	**0.80**	Naïve Bayes	P	0.14	0.71	0.18	0.24	**0.81**
	R	0.92	0.92	**0.99**	0.93	0.78		R	0.91	0.69	0.70	**0.92**	0.80
	F_1	0.24	0.25	0.27	0.25	**0.78**		F_1	0.24	0.69	0.28	0.35	**0.79**
K-nearest neighbours	P	0.20	0.19	0.16	0.44	**0.80**	PART decision lists	P	0.17	0.20	0.17	0.26	**0.79**
	R	0.91	0.89	**0.99**	0.91	0.81		R	0.88	0.88	**0.93**	0.87	0.76
	F_1	0.25	0.28	0.28	0.57	**0.77**		F_1	0.24	0.32	0.29	0.40	**0.76**
C4.5	P	0.17	0.20	0.17	0.26	**0.77**	Random forests	P	0.14	0.22	0.16	0.19	**0.79**
	R	0.88	0.87	**0.96**	0.88	0.79		R	0.92	0.91	**0.99**	0.91	0.80
	F_1	0.24	0.31	0.29	0.38	**0.78**		F_1	0.24	0.28	0.28	0.28	**0.77**
Repeated incremental pruning to produce error	P	0.10	0.26	0.18	0.24	**0.79**	Sequential minimal optimisation	P	0.14	0.19	0.17	**0.38**	0.71
	R	0.36	0.76	0.73	**0.86**	0.80		R	**0.91**	0.89	0.97	0.90	0.67
	F_1	0.14	0.35	0.28	0.36	**0.78**		F_1	0.24	0.26	0.29	0.51	**0.68**

P1 = ComplEx, P2 = HolE, P3 = Node2Vec, P4 = TransD, P5 = Aquila

b) Performance results of tagger proposals.

Table A.1: *Experimental results of Aquila.*

popular web site in the corresponding category. The categories were randomly selected from The Open Directory and the web sites were randomly selected from the 100 best ranked sites in each category according to Google's search engine. The datasets were split into learning sets with four documents, validation sets with two documents, and testing sets with 24 documents. Since the majority of nodes have a null tag, we balanced the datasets by re-weighting the classes using Weka's class balancer [129].

We selected four state-of-the-art graph embedders as baselines: ComplEx [122], HolE [95], Node2Vec [51], and TransD [60]. To generate weighted graphs we transformed each DOM node into a graph node; connected them according to their links in the DOM tree (parent, left, right); and computed their weights by measuring the similarity of the corresponding nodes in terms of their attributes. To generate labelled graphs from the input documents, we transformed the DOM nodes and the values of their attributes into graph nodes, connected the nodes according to their links in the DOM tree or the values of their attributes; and labelled the edges with the corresponding link or attribute.

Figure A.1.b shows the performance results that we computed from running the baselines and our proposal on our datasets. Regarding the baselines, they attain an average recall as high as 0.87 but an average precision as low as 0.21; in other words, the taggers learnt from their embeddings tend to return many nodes for each tag and have little ability to make them apart. The results using our proposal are clearly better in every case: our average precision is 0.78; our average recall is 0.78; and our average F_1 score is 0.76. Our best F_1 score was attained using Naive Bayes base learner.

Our conclusion from the previous results is that the embeddings computed by the baselines do not capture well the features of the original documents so that a good tagger can be inferred from them, whereas the embeddings computed by our proposal do. We conjecture that it is the ability of our proposal to find a path that maximises the performance of the classifier returned and its ability to preserve the original attributes that helps it learn better taggers.

A.5 Summary

In this appendix, we have presented a new proposal to synthesise metadata tags that link the data in an HTML document to their types in a vocabulary. It relies on a novel embedding technique that preserves the original attributes and does not require to preset an embedding size. Our experimental results prove that our approach can attain an F_1 score on our evaluation dataset that is better than using four state-of-the-art embedders, independently from the base learner used. In future, we are planning on doing additional research regarding the following ideas: exploring additional links, exploring backtracking, and exploring different strategies to compare the scores.

Appendix B

Kizomba: general-purpose web data extraction

Kizomba is a heuristic-based proposal to extract data from HTML documents that relies on recurrent representation patterns. It is described as follows: Section B.1 introduces our proposal; Section B.2 describes the related work; Section B.3 describes its details; Section B.4 reports on the results of our experiments; finally, Section B.5 concludes our work.

B.1 Introduction

In the Web, there are many semi-structured documents that provide data that might help humans make decisions. Data extraction techniques are required to analyse those documents and transform them into structured records that can be fed into typical data extraction systems [22, 116, 123]. Nowadays, the web data extraction field is evolving towards developing proposals at web scale. A proposal is considered to work at web scale, when it requires as little user intervention as possible and can extract data from as many sites as possible.

In the literature, there are some unsupervised proposals that do not require the user to annotate the learning set [8, 30, 66, 118], but they have an important drawback: they focus on extracting the information that varies from a document to the rest, but they focus on data and forget about their structure and the labels that endow them with semantics. The Open Information Extraction (OIE) proposals attempt to solve the previous problems [42] but most of them extract information encoded using natural language, but do not work well with semi-structured data, with cannot be processed using POS tagging methods. The only proposal that can work with HTML document structures requires human effort in order to generate an extractor for a new domain or language.

We bet on heuristic-based proposals because they are the closest to a truly web scalable proposal since they do not require any learning phases, data models or schemas, and labelling the information that they extract is getting easier thanks to automatic semantic typers [120]. In this paper, we present an unsupervised heuristic-based technique that extracts information building on common web representation patterns, which is a term that we use to refer to the typical regular templates used to render information in web documents, e.g., an attribute-value table, a variable list, HTML meta-information, and so on. Since the heuristics are used to extract the information instead of learning a rule, a learning step is not required. Our technique does not require any human intervention since it extracts the information as it processes the web documents, which makes it appropriate to perform OIE from semi-structured documents at web scale. Furthermore, it can make a difference between data themselves and the labels used to endow them with semantics.

B.2 Related work

Typical information extractors are machine-learnt from a set of web documents, known as learning set. Many proposals require the learning set to be gathered from a single web site and can work on documents from that site only; many also require the user to annotate the learning set with labels that endow the information to be extracted with semantics, known as supervised approaches. These supervised approaches are not scalable to the Web because they require too much user intervention to annotate the learning set. There are some unsupervised proposals that do not require the user to annotate the learning set [8, 30, 66, 118]; furthermore, they can be applied to a variety of sites and are not bound with a predefined data model. This makes them quite appealing to extract information at web scale, but they have an important drawback: they focus on extracting the information that varies from a document to the rest, but do not take their structure into account. In other words, they can extract attributes and group them into records without any semantics, but do not attempt to analyse the actual structure of the information so that the resulting records make sense. Simply put: they focus on data and forget about their structure and the labels that endow them with semantics.

The proposals in the OIE field attempt to solve the previous problems [42]. Unfortunately, an immense majority of proposals in this field focus on web documents in which the information to be extracted is rendered in natural language passages [6, 41, 135]; they typically require the text to be POS tagged in order to identify patterns from which the information of interest can be extracted. To the best of our knowledge, there has been only one attempt to devise an OIE proposal for semi-structured documents [21]. However, this proposal requires human effort in order to generate an extractor for a new domain or language. It is not clear whether it is general enough to be applied to a whole domain, instead of a subset of sites from that domain. That is, the technique seems to be domain- or site-dependent, which prevents it from working at web-scale. Furthermore, it cannot be applied to documents with multiple records because it is only intended to extract attributes without a structure. It has trouble dealing with optional attributes, cannot deal with multi-valued ones and it requires attributes to be fully-contained within the context of a DOM node. Nodes that do not appear in at least 50% of the documents are discarded, which is a strong assumption, since there are cases in which relevant attributes appear in less than a half of the documents. Last, but not least, it is unclear whether the proposal is resilient to changes.

B.3 Our proposal

In this section we describe Kizomba, which works on sets of two or more structurally-similar documents within a web site. It applies a number of heuristics that we identified during the development of previous web data extractors [61–63, 103, 114, 115, 117, 118].

Our heuristics have proven to be general enough so as to be applicable to a variety of web sites since they focus on identifying representation patterns, which are recurrent web design patterns that people are familiar with. They provide structural and behavioural common features across different sites. Identifying these patterns is essential in order to identity records and attributes. A record instance is a piece of text that encapsulates the data regarding an entire item. Record instances might rely on subpieces of text with an atomic structure, namely, attribute instances. Next, we provide some details on the representation patterns covered by Kizomba.

Data nodes: We align nodes that have the same tag paths to identify the ones that are likely to carry some data of interest, that is, those that contain data that varies from one document to another. These nodes are marked as data nodes, and their ancestors are marked as child-data nodes.

Key-value tables: This representation pattern attempted to extract the most frequent table structure in e-commerce sites, which are tables that display attributes about one or more records. The first column encodes the names of the attributes, while the next columns present the values of such attributes in a record. Since the input documents are structurally similar, we can collect every child-data table with the same tag path across different documents. Then, we can compute the variability ratio of the nodes of each column of the tables as n/k, where n denotes the total number of unique values, and k denotes the number of values. In a key-value table, the variability ratio of the first column must be smaller than the variability ratio of the other columns. This pattern was an early version of TOMATE, c.f. Chapter 3.

Description lists: This heuristic is similar to the previous one, since a description list can be seen as a two-column table in which the terms constitute the first column and the descriptions the second column. So, it also relies on computing and checking variability ratios. The name of a property is encoded using dt tags and their values are encoded using dd tags. Alternatively name-value pairs can be encoded usign tags ol or ul, wrapping the names with b nodes.

Proposal	Average			Standard deviation		
	P	R	F_1	P	R	F_1
RoadRunner	0.53	0.76	0.59	0.21	0.28	0.24
FivaTech	0.68	0.86	0.72	0.26	0.14	0.25
Trinity	0.81	**0.91**	0.85	0.10	**0.08**	**0.07**
Our proposal	**0.95**	0.87	**0.90**	**0.07**	0.14	0.08

a) Effectiveness results.

Documents	50	100	150	200	300	400	500
Total time	119.08	169.20	282.79	370.23	498.22	622.17	702.91
Per document	2.38	1.69	1.89	1.85	1.66	1.56	1.41

b) Scalability analysis.

Table B.1: *Experimental results of Kizomba.*

Global data: There are some nodes that are used to provide global data of the document, such as title, h1, and meta nodes. They typically contain data about more than one property, so they are aligned across the documents and split according to the set of tokens that does not vary from one document to another.

Breadcrumbs: A breadcrumb is a hint that helps users to keep track of their location within a site. The path consists of a sequence of hyperlinked words or short phrases that are separated by a glyph, plus a final word or short sentence that is not hyperlinked but is separated by the same glyph. We identify breadcrumbs by searching for such sequences and requiring the degree of variability of the components of the path to increase monotonically from left to right.

Remaining data nodes: After the previous heuristics have been applied, some data nodes may remain. Although this remaining data is not structured according to a representation pattern, it might still be relevant for further analysis. They are extracted using their CSS selector as their field name and their own text as their values.

B.4 Experimental analysis

To evaluate our results, we compared our proposal with some unsupervised data extraction proposals that have been widely used in the literature for empirical comparison purposes [30, 66, 118]. We ran the data extractors on a dataset with 20 sites with 30 documents each from multiple domains.

Table B.1.a reports on the effectiveness of the proposals that we have analysed. The precision and recall of each site were computed as the average of the precision and recall of each value of an attribute extracted in each document. The per-document standard deviation was also computed. To measure the efficiency of our proposal, we applied it to a collection of 500 documents from the book domain. Starting with 50 documents, we increased the number of documents by 100 up to 500. The performance results are shown in Figure B.1.b.

Our conclusion is that the precision of our proposal is better the other proposals. Despite the recall falls in some sites, the F_1 score stands out over the results of the other proposals. Since our proposal is based on a catalogue of heuristics, at the time of performing the experimentation, some representation patterns were not identified. In practice, our results suggest that our proposal can be used to extract data from semi-structured documents at web scale with good precision and recall, since it scales in a linear manner.

B.5 Summary

Feeding decision support systems with web data typically requires sifting through an unwieldy amount of data that is available in human-friendly formats only. In this appendix, we have presented a proposal to extract data from semi-structured documents in a structured format, with an emphasis on it being scalable and open. In the literature, there is only one open but not scalable proposal, since it requires human supervision on a per-domain basis. In this paper, we present a new proposal that relies on a number of heuristics to identify patterns that are typically used to represent the data in a web document. Our experimental results confirm that our proposal is very competitive in terms of effectiveness and efficiency.

Appendix C

Romulo: clustering in the context of data lakes

Romulo is a novel approach to multi-way single-subspace clustering using a meta-heuristic. It is described in this chapter, which is organised as follows: Section C.1 introduces our proposal; Section C.2 analyses the related work; Section C.3 presents the proposal; Section C.4 shows the experimental analysis; finally, Section C.5 concludes the chapter.

C.1 Introduction

The Web is currently the most important data source since it provides a plethora of datasets on virtually any topic of interest. Data lake is an umbrella term that refers to an array of technologies and techniques that help data engineers store and understand the datasets that they collect from different repositories [68, 84, 100]. Typically, they are used to store as many datasets as possible, with as many data and attributes as possible since the ultimate goal is not to miss any opportunities to leverage them; unfortunately, this makes it very difficult to explore them. Usually, engineers start exploring the datasets in a data lake by identifying a subspace of informative attributes, projecting the dataset onto that subspace, and clustering the resulting projected dataset [54].

In the literature, there are many clustering proposals to address the previous problem [2–4, 7, 13, 32, 46, 50, 55, 59, 70, 92, 101, 113, 132, 133]. These methods are classified depending on whether they execute the steps independently or co-ordinately (single-way or multi-way), whether they compute a single subspace or multiple subspaces of attributes (single-subspace or multi-subspace), and whether they require an engineer to provide the number of clusters or not (manual or automatic). Many of them use algorithmic approaches; but most recent proposals use meta-heuristic approaches that map the problem onto nature-inspired processes. It is particularly shinning that no meta-heuristic approach has been explored in the context of multi-way single-subspace automatic clustering. Abualigah [1] highlighted that meta-heuristic approaches have been related to unsatisfactory outcomes and inaccurate clusters in the past and the reason might be that they have not been sufficiently studied in this context.

This chapter presents Romulo, which is the first attempt in the literature to use a meta-heuristic to address the problem of multi-way single-subspace automatic clustering. The approach was confronted with four strong competitors to find subspaces of informative attributes and perform the clustering, as well as P3C, which is a proposal for multi-way multi-subspace automatic clustering. The evaluation was performed on a challenging data lake with 139 datasets from 10 different repositories. The Silhouette coefficient attained by Romulo was 0.57, which is 18.75% better than the score attained by the best competitor. These results make Romulo a good contribution to the array of techniques that data engineers can use to explore their data lakes.

C.2 Related work

It is also clear that most authors have focused on devising single-way clustering proposals [59, 132]. Originally, most approaches to clustering were algorithmic, but meta-heuristic approaches [7, 13, 44, 46, 55, 92, 101] have found their way into this field because their nature-inspired approaches are very good to explore complex search spaces and they can be implemented very efficiently using current multi-threaded CPUs. Unfortunately, they are not the most appropriate in the context of data lakes since their datasets commonly consist of many data with many attributes that are dumped from their original repositories with as little processing as possible. In such a context, multi-way clustering is a must since it helps find the subspaces of attributes that result in the best possible clusters co-ordinately.

Single-way clustering is a hard problem, but multi-way clustering [32, 46, 59, 113] is even harder. Most of them use algorithmic approaches. It is then surprising that meta-heuristics have been explored so little in this context. For instance, there is not a single record in the literature of a meta-heuristic approach to multi-way single-subspace clustering. Abualigah [1] mentioned that such approaches have commonly been related to poor results, very likely because they have not been sufficiently studied in this context. Furthermore, the existing algorithmic approaches have a number of problems that hinder their applicability to real-world datasets, namely: they neglect between-cluster isolation and they do not deal well with imbalanced datasets, not to mention their sensitivity to the configuration parameters [32, 70, 113].

The analysis of the literature that was presented in the previous sections makes it clear that there are a variety of approaches to clustering. Unfortunately, there is not a single proposal that can be considered universal. Typically, the motivation to work on a new proposal originates from the inability of another proposal to deal with a particular kind of datasets. This has resulted in many approaches that commonly require fine tuning their configuration parameters so that they can find the clusters in the input datasets [113]. The previous findings make it clear that there is a research niche regarding multi-way single-subspace automatic clustering. This was the motivation to explore this problem using a genetic approach that led to the development of Romulo. The experimental results prove that it is a good tool for data engineers who have to explore complex data lakes. This, in turn, motivates some future work regarding how to improve it by incorporating a data engineer's background knowledge in the search for subspaces and clusters, as well as exploring other nature-inspired approaches.

C.3 Our proposal

This section introduces the core algorithm of Romulo. It works on an input dataset and outputs a subset of attributes and its corresponding clustering. It relies on a genetic strategy [9] that creates an initial population, evolves it genetically, and selects the best individual to compute the output.

The first step consists in generating the initial population of random individuals. They are tuples of the form (k, A), where k denotes the number of clusters and A is a vector of the form $(a_1, a_2, \ldots, a_d)$ in which each a_i is a Boolean that denotes whether the corresponding attribute must be used or not to compute the clusters; d represents the number of attributes in the input dataset ($d \geq 1$).

Next, the population is evolved multiple times. In each generation the offspring is computed by applying one of the following genetic operations: crossing two existing individuals, mutating an individual, or randomly picking an individual. The crossover is performed as follows: first, two individuals are sampled from the population along with their components (k_1, A_1) and (k_2, A_2); then, it computes the number of attributes in the population and generates a random natural p in interval $[1, d-1]$; finally, it generates two offsprings $(k_2, A_2[1:p] \bullet A_1[p+1:d])$ and $(k_1, A_1[1:p] \bullet A_2[p+1:d])$, where $\bullet$ is the vector catenation operator. The mutation is performed as follows: first, an individual is picked from the population along with its components (k, A); next, it computes a random natural in interval $[k-1, k+1]$ which is set as the number of clusters in the offspring; finally, it flips a random attribute from A in the offspring.

Then, the best individuals from both the current population and the offspring are selected to generate the next population. It is implemented using the version of the Lexicase method that was described by la Cava et al. [72]. Helmuth et al. [53] published an in-depth analysis of Lexicase; their conclusion was that this method is very appropriate in cases in which it is not easy to aggregate multiple quality indicators into a single value. This is the case of Romulo, whose fitness function relies on the number of clusters to compute and the resulting Silhouette coefficient. Aggregating both indicators into a meaningful single value is not easy because they range in different intervals and assess the quality of the clusters from very different perspectives. In such cases, the Lexicase procedure provides a simple yet effective solution to implement the fitness function, which was confirmed by the experimental analysis.

Finally, the result is computed very straightforwardly: first, it selects the best individual (k, A) from the last generation using the Lexicase method; second, it projects the input dataset onto the attributes selected by A; third, it computes k clusters on the projected dataset using any single-way manual clustering proposal; and, finally, it assembles the result and returns it.

C.4 Experimental analysis

The experiments were executed on a data lake with 139 datasets that were randomly collected from 10 different repositories. The datasets range in size from 15 to 20 640 data, with an average of 2 248.36 data per dataset. The number of attributes ranges from 4 to 1 196, with an average of 106.26 attributes per dataset. According to the GSPPCA method, 39.09% of the attributes were estimated not to be informative. Summing up, the data lake provides an assorted collection of heterogeneous datasets that ensures that no bias was introduced in the experimentation.

The analysis of the related work reveals that Romulo is the only proposal in the literature that addresses the multi-way single-subspace automatic clustering problem using a meta-heuristic approach. As a conclusion, there are not any direct competitors with which it can be compared empirically. The comparison was then carried out using five closely-related proposals that were adapted to deal with multi-way single-subspace automatic clustering problems. Four of the competitors resulted from combining the recent GSPPCA method to find subspaces of informative attributes and AffinityPropagation, DBScan, MeanShift, and OPTICS-XI to perform the clustering. The fifth competitor is an algorithmic proposal for multi-way multi-subspace auto clustering that is known as P3C [88]. Every proposal was configured by performing a grid search on a subset of ten datasets that were randomly picked from the data lake.

Table C.1 presents the performance results achieved on the repositories. The columns report on the repositories and the performance results achieved by the proposals that were compared, namely: the Silhouette coefficient ($\mathtt{Silh.}$), which captures well the idea that clusters must be compact and isolated [65]; it ranges in interval $[-1.00, +1.00]$, where the lower bound indicates that the clusters evaluated are neither compact nor isolated and the upper bound indicates that they are very compact and very isolated; and the execution time ($\mathtt{Time}$), which was measured in CPU seconds. A cell with "N/A" means that the corresponding proposal threw an exception and could not process the corresponding dataset. The averages are provided in the last row. The best results per repository or dataset are highlighted in grey.

Repository	**Romulo**		**GSPCCA + Aff.Prop.**		**GSPCCA + DBScan**		**GSPCCA + MeanShift**		**GSPCCA + OPTICS-XI**		**P3C**	
	Silh.	**Time**	**Silh.**	**Time**	**Silh.**	**Time**	**Silh.**	**Time**	**Silh.**	**Time**	**Silh.**	**Time**
UCI	0.44	17.02	0.30	18.08	0.28	**7.23**	**0.45**	13.53	0.01	7.40	-0.14	104.23
EU-Open-Data	0.57	95.53	0.61	16.38	**0.79**	54.88	0.59	16.49	0.07	**15.78**	0.10	118.33
SciKit	0.23	164.63	0.36	368.19	**0.39**	**9.15**	0.32	65.64	-0.21	20.66	-0.16	110.68
TOMATE	**0.93**	**4.87**	0.59	6.54	0.50	6.45	0.61	7.16	0.26	6.85	-0.06	109.76
Conferences	**0.49**	92.90	0.29	130.81	0.44	**9.96**	0.39	29.31	0.09	16.93	-0.06	113.75
DFT-UK	**0.71**	18.97	0.25	21.99	0.45	**11.70**	0.56	17.45	0.08	13.39	-0.10	92.92
Open-ML	**0.48**	80.60	0.32	93.01	0.43	**9.13**	0.38	66.75	0.12	29.67	0.00	94.62
Microsoft	**0.56**	64.77	0.30	35.19	0.05	**8.05**	0.53	63.58	-0.11	29.32	-0.04	121.02
UBER-Trips	0.27	89.09	0.26	112.39	**0.35**	**16.80**	0.24	210.69	0.19	114.34	-0.01	109.94
WHO-COVID	**0.71**	152.29	0.47	384.78	0.69	**6.95**	0.65	83.84	0.26	29.31	0.03	125.88
Average	**0.54**	78.07	0.37	118.74	0.44	**14.03**	0.47	57.44	0.08	28.36	-0.04	110.11

Table C.1: *Experimental results of Romulo.*

Regarding effectiveness, Romulo ranks the first one because it attains an average Silhouette coefficient of 0.54 on the repositories. Regarding efficiency, Romulo ranks fourth because it takes 78.07 and 71.50 CPU seconds in average. The intuitive conclusion is that Romulo outperforms the other proposals in terms of effectiveness while it remains efficient enough for practical purposes. Note that it is 14.89% more effective than the next proposal in the ranking. Obviously, this improvement comes at the cost of some inefficiency since it takes an average of 20.63 more CPU seconds. This drop regarding efficiency clearly compensates for the increase regarding effectiveness.

C.5 Summary

This appendix has introduced Romulo, which is a proposal that helps data engineers analyse the datasets in their data lakes by finding a subspace of attributes from which it extracts the best possible clusters of data. It is the first attempt in the literature to explore this problem using a meta-heuristic approach. The proposal was confronted with five strong competitors on a challenging real-world data lake. They all were configured using grid search on a small subset of datasets. The conclusion is that Romulo is the most effective proposal, which clearly compensates for the extra time that it requires. Summing up: Romulo is a good contribution to a data engineer's toolbox in the context of data lakes.

Bibliography

[1] L. M. Abualigah. *Feature selection and enhanced krill herd algorithm for text document clustering*. Springer, 2018. 10.1007/978-3-030-10674-4.

[2] L. M. Abualigah and A. T. Khader. Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for text clustering. *The Journal of Supercomputing*, 73:4773–4795, 2017. 10.1007/s11227-017-2046-2.

[3] L. M. Abualigah, A. T. A. Khader, and E. S. Hanandeh. Hybrid clustering analysis using improved krill herd algorithm. *Applied Intelligence*, 48:4047–4071, 2018. 10.1007/s10489-018-1190-6.

[4] L. M. Abualigah, A. T. A. Khader, and E. S. Hanandeh. A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. *Eng. Appl. Artif. Intell.*, 73:111–125, 2018. 10.1016/j.engappai.2018.05.003.

[5] B. Adrian, J. Hees, I. Herman, M. Sintek, and A. Dengel. Epiphany: adaptable RDFa generation linking the Web of Documents to the Web of Data. In *EKAW*, pages 178–192, 2010. 10.1007/978-3-642-16438-5_1.

[6] E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, and A. Voskoboynik. Snowball: a prototype system for extracting relations from large text collections. In *SIGMOD Conference*, page 612, 2001. 10.1145/375663.375774.

[7] S. Alam, G. Dobbie, Y. S. Koh, P. Riddle, and S. U. Rehman. Research on particle swarm optimization based clustering: a systematic review of literature and techniques. *Swarm and Evolutionary Computation*, 17: 1–13, 2014. 10.1016/j.swevo.2014.02.001.

[8] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, pages 337–348, 2003. URL http://doi.acm.org/10.1145/872757.872799.

[9] H. Beyer and H. Schwefel. Evolution strategies: a comprehensive introduction. *Nat. Comput.*, 1(1):3–52, 2002. 10.1023/A:1015059928466.

[10] C. S. Bhagavatula, T. Noraset, and D. Downey. TabEL: entity linking in web tables. In *ISWC*, pages 425–441, 2015. 10.1007/978-3-319-25007-6_25.

[11] C. Bizer, R. Meusel, and A. Primpel. Web Data Commons: RDFa, Microdata, and Microformat data sets. Technical report, University of Mannheim, 2019. Available at http://webdatacommons.org/structureddata/2018-12/stats/stats.html.

[12] C. Bizer, A. Primpeli, and R. Peeters. Using the Semantic Web as a source of training data. *Datenbank-Spektrum*, 19(2):127–135, 2019. 10.1007/s13222-019-00313-y.

[13] C. Bong and M. Rajeswari. Multi-objective nature-inspired clustering and classification techniques for image segmentation. *Appl. Soft Comput.*, 11(4):3271–3282, 2011. 10.1016/j.asoc.2011.01.014.

[14] K. Braunschweig, M. Thiele, and W. Lehner. From web tables to concepts: a semantic normalization approach. In *ER*, pages 247–260, 2015. 10.1007/978-3-319-25264-3_18.

[15] A. L. Buchsbaum, D. F. Caldwell, K. W. Church, G. S. Fowler, and S. Muthukrishnan. Engineering the compression of massive tables: an experimental approach. In *SODA*, pages 175–184, 2000. URL http://dl.acm.org/citation.cfm?id=338219.338249.

[16] R. Burget. Information extraction from the Web by matching visual presentation patterns. In *ISWC*, pages 10–26, 2016. 10.1007/978-3-319-68723-0_2.

[17] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: exploring the power of tables on the Web. *PVLDB*, 1(1):538–549, 2008. 10.14778/1453856.1453916.

[18] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational Web. In *WebDB*, pages 1–6, 2008. URL http://webdb2008.como.polimi.it/images/stories/WebDB2008/paper30.pdf.

[19] M. J. Cafarella, A. Y. Halevy, H. Lee, J. Madhavan, C. Yu, D. Z. Wang, and E. Wu. Ten years of web tables. *PVLDB*, 11(12):2140–2149, 2018. 10.14778/3229863.3240492.

[20] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018. 10.1109/TKDE.2018.2807452.

[21] A. Carlson and C. Schafer. Bootstrapping information extraction from semi-structured web pages. In *ECML/PKDD (1)*, pages 195–210, 2008. 10.1007/978-3-540-87479-9_31.

[22] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.*, 18(10):1411–1428, 2006. 10.1109/TKDE.2006.152.

[23] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai. Mining tables from large scale HTML texts. In *COLING*, pages 166–172, 2000. 10.3115/990820.990845.

[24] X. Chu, Y. He, K. Chakrabarti, and K. Ganjam. TEGRA: table extraction by global record alignment. In *SIGMOD*, pages 1713–1728, 2015. 10.1145/2723372.2723725.

[25] W. W. Cohen, M. Hurst, and L. S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *WWW*, pages 232–241, 2002. 10.1145/511446.511477.

[26] R. Corchuelo, P. Jiménez, and J. C. Roldán. Romulo: a clustering proposal in the context of data lakes. *Eng. Appl. of AI*, pages 1–39, 2020. Under review.

[27] A. Costa-Silva, A. M. Jorge, and L. Torgo. Design of an end-to-end method to extract information from tables. *IJDAR*, 8(2-3):144–171, 2006. 10.1007/s10032-005-0001-x.

[28] V. Crescenzi and G. Mecca. Grammars have exceptions. *Inf. Syst.*, 23(8):539–565, 1998. 10.1016/S0306-4379(98)00028-3.

[29] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1&2):21–52, 2008. 10.1080/08839510701853093.

[30] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: automatic data extraction from data-intensive web sites. In *SIGMOD Conference*, page 624, 2002. 10.1145/564691.564778.

[31] E. Crestan and P. Pantel. Web-scale table census and classification. In *WSDM*, pages 545–554, 2011. 10.1145/1935826.1935904.

[32] Z. Deng, K. Choi, Y. Jiang, J. Wang, and S. Wang. A survey on soft subspace clustering. *Inf. Sci.*, 348:84–106, 2016. 10.1016/j.ins.2016.01.101.

[33] J. M. Dodero, I. Ruiz-Rube, M. Palomo-Duarte, and J. Vázquez-Murga. Open linked data model revelation and access for analytical web science. In *MTSR*, pages 105–116, 2011. 10.1007/978-3-642-24731-6_10.

[34] J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner. Building the Dresden web table corpus: a classification approach. In *BDC*, pages 41–50, 2015. 10.1109/BDC.2015.30.

[35] J. Eberius, M. Thiele, K. Braunschweig, and W. Lehner. Top-k entity augmentation using consistent set covering. In *SSDBM*, pages 8:1–8:12, 2015. 10.1145/2791347.2791353.

[36] V. Efthymiou, O. Hassanzadeh, M. Rodríguez-Muro, and V. Christophides. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *ISWC*, pages 260–277, 2017.

[37] B. Eldesouky, M. Bakry, H. Maus, and A. Dengel. Seed: an end-user text composition tool for the Semantic Web. In *ISWC*, pages 218–233, 2016.

[38] H. Elmeleegy, J. Madhavan, and A. Y. Halevy. Harvesting relational tables from lists on the Web. *VLDB*, 20(2):209–226, 2011. 10.1007/s00778-011-0223-0.

[39] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy. Table-processing paradigms: a research survey. *IJDAR*, 8(2-3):66–86, 2006. 10.1007/s10032-006-0017-x.

[40] D. W. Embley, S. C. Seth, and G. Nagy. Transforming web tables to a relational database. In *ICPR*, pages 2781–2786, 2014. 10.1109/ICPR.2014.479.

[41] O. Etzioni, M. J. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *WWW*, pages 100–110, 2004. 10.1145/988672.988687.

[42] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: the second generation. In *IJCAI*, pages 3–10, 2011.

[43] E. Ferrara, P. de Meo, G. Fiumara, and R. Baumgartner. Web data extraction, applications, and techniques: a survey. *Knowl.-Based Syst.*, 70: 301–323, 2014. 10.1016/j.knosys.2014.07.007.

[44] E. Figueiredo, M. Macedo, H. V. Siqueira, C. J. Santana, A. Gokhaled, and C. J. Bastos-Filhoa. Swarm intelligence for clustering: a systematic review with new perspectives on data mining. *Eng. Appl. of AI*, 82: 313–329, 2019. 10.1016/j.engappai.2019.04.007.

[45] F. Fumarola, T. Weninger, R. Barber, D. Malerba, and J. Han. Extracting general lists from web documents: a hybrid approach. In *IEAAIE*, pages 285–294, 2011. 10.1007/978-3-642-21822-4_29.

[46] A. J. García and W. Gómez-Flores. Automatic clustering using nature-inspired metaheuristics: a survey. *Appl. Soft Comput.*, 41:192–213, 2016. 10.1016/j.asoc.2015.12.001.

[47] S. García and F. Herrera. An extension on "Statistical comparisons of classifiers over multiple datasets" for all pair-wise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[48] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *WWW*, pages 71–80, 2007. 10.1145/1242572.1242583.

[49] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: a survey. *Knowl.-Based Syst.*, 151:78–94, 2018.

[50] R. Greenlaw and S. Kantabutra. Survey of clustering: algorithms and applications. *Journal of Information Retrieval Research*, 3(2):1–29, 2013. 10.4018/ijirr.2013040101.

[51] A. Grover and J. Leskovec. Node2Vec: scalable feature learning for networks. In *SIGKDD*, pages 855–864, 2016.

[52] G. Guerrero-Contreras, J. L. Navarro-Galindo, J. Samos, and J. L. Garrido. A collaborative semantic annotation system in health: towards a SOA design for knowledge sharing in ambient intelligence. *Mobile Information Systems*, 2017:1–10, 2017.

[53] T. Helmuth, L. Spector, and J. Matheson. Solving uncompromising problems with Lexicase selection. *IEEE Trans. Evolutionary Computation*, 19(5):630–643, 2015. 10.1109/TEVC.2014.2362729.

[54] C. Hennig. What are the true clusters? *Pattern Recognition Letters*, 64: 53–62, 2015. 10.1016/j.patrec.2015.04.009.

[55] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 39(2):133–155, 2009. 10.1109/TSMCC.2008.2007252.

[56] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Inf. Syst.*, 23(8):521–538, 1998. 10.1080/08839510701853093.

[57] M. Hurst. Layout and language: challenges for table understanding on the Web. In *WDA*, pages 27–30, 2001. URL http://wda2001.csc.liv.ac.uk/Papers/12_hurst_wda2001.pdf.

[58] M. Hurst. Classifying table elements in HTML. In *WWW*, 2002. URL http://wwwconference.org/proceedings/www2002/poster/115/index.html.

[59] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. 10.1016/j.patrec.2009.09.011.

[60] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, pages 687–696, 2015.

[61] P. Jiménez and R. Corchuelo. On learning web information extraction rules with TANGO. *Inf. Syst.*, 62:74–103, 2016. 10.1016/j.is.2016.05.003.

[62] P. Jiménez and R. Corchuelo. Roller: a novel approach to web information extraction. *Knowl. Inf. Syst.*, 49(1):197–241, 2016. 10.1007/s10115-016-0921-4.

[63] P. Jiménez, R. Corchuelo, and H. A. Sleiman. ARIEX: automated ranking of information extractors. *Knowl.-Based Syst.*, 93:84–108, 2016. 10.1016/j.knosys.2015.11.004.

[64] S.-W. Jung and H.-C. Kwon. A scalable hybrid approach for extracting head components from web tables. *IEEE Trans. Knowl. Data Eng.*, 18(2):174–187, 2006. 10.1109/TKDE.2006.19.

[65] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Wiley, 2nd edition, 2005.

[66] M. Kayed and C.-H. Chang. FiVaTech: page-level web data extraction from template pages. *IEEE Trans. Knowl. Data Eng.*, 22(2):249–263, 2010. 10.1109/TKDE.2009.82.

[67] Y.-S. Kim and K.-H. Lee. Detecting tables in web documents. *Eng. Appl. of AI*, 18(6):745–757, 2005. 10.1016/j.engappai.2005.01.009.

[68] M. Klettke, H. Awolin, U. Störl, D. Müller, and S. Scherzinger. Uncovering the evolution history of data lakes. In *BigData*, pages 2462–2471, 2017. 10.1109/BigData.2017.8258204.

[69] C. A. Knoblock, P. A. Szekely, E. E. Fink, D. Degler, D. Newbury, R. Sanderson, K. Blanch, S. Snyder, N. Chheda, N. Jain, R. R. Krishna, N. B. Sreekanth, and Y. Yao. Lessons learned in building linked data for the American Art Collaborative. In *International Semantic Web Conference (2)*, pages 263–279, 2017. 10.1007/978-3-319-68204-4_26.

[70] H. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 3(1):1:1–1:58, 2009. 10.1145/1497577.1497578.

[71] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *IJCAI (1)*, pages 729–737, 1997. URL ftp://ftp.cs.washington.edu/tr/1997/11/UW-CSE-97-11-04.pdf.

[72] W. la Cava, T. Helmuth, L. Spector, and J. H. Moore. A probabilistic and multi-objective analysis of Lexicase selection and ϵ-Lexicase selection. *Evolutionary Computation Journal*, pages 1–26, 2019. 10.1162/evco_a_00224.

[73] D. Laney and A. Jain. One hundred data and analytics predictions through 2021. Technical report, Gartner, 2020. URL https://www.gartner.com/en/doc/3746424-100-data-and-analytics-predictions-through-2021.

[74] L. R. Lautert, M. M. Scheidt, and C. F. Dorneles. Web table taxonomy and formalization. *SIGMOD Record*, 42(3):28–33, 2013. 10.1145/2536669.2536674.

[75] K. Lerman, C. Knoblock, and S. Minton. Automatic data extraction from lists and tables in web sources. In *IJCAI*, 2001. URL http://www.isi.edu/integration/papers/lerman01-atem.pdf.

[76] K. Lerman, L. Getoor, S. Minton, and C. A. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *SIGMOD*, pages 119–130, 2004. 10.1145/1007568.1007584.

[77] T. Liao, T. Liu, S. Zhang, and Z. Liu. Research on web table positioning technology based on table structure and heuristic rules. In *AISC*, pages 351–360, 2018. 10.1007/978-3-319-67071-3_41.

[78] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the Web. In *IJCAI*, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6758.

[79] W. Liu, X. Meng, and W. Meng. ViDE: a vision-based approach for deep web data extraction. *IEEE Trans. Knowl. Data Eng.*, 22(3):447–460, 2010. 10.1109/TKDE.2009.109.

[80] M.-L. Lo, K.-L. Wu, and P. S. Yu. TabSum: a flexible and dynamic table summarization approach. In *ICDCS*, pages 628–635, 2000. 10.1109/ICDCS.2000.840979.

[81] D. P. Lopresti and G. Nagy. Automated table processing: an (opinionated) survey. In *GREC*, pages 109–134, 1999. URL http://www.cse.lehigh.edu/~lopresti/Publications/1999/grec99.pdf.

[82] D. P. Lopresti and G. Nagy. A tabular survey of automated table processing. In *GREC*, pages 93–120, 2000. 10.1007/3-540-40953-X_9.

[83] J. Mankoff, H. Fait, and T. Tran. Is your web page accessible? A comparative study of methods for assessing web page accessibility for the blind. In *CHI*, pages 41–50, 2005. 10.1145/1054972.1054979.

[84] C. Mathis. Data lakes. *Datenbank-Spektrum*, 17(3):289–293, 2017. 10.1007/s13222-017-0272-7.

[85] C. McKinnon, D. Hong, H. Colin, and M. Bakalar. The five key trends that will shape your 2020 content services strategy. Technical report, Forrester, 2020. URL https://www.forrester.com/report/The+Five+Key+Trends+That+Will+Shape+Your+2020+Content+Services+Strategy/-/E-RES83441.

[86] M. Michelson and C. A. Knoblock. Unsupervised information extraction from unstructured, ungrammatical data sources on the World Wide Web. *IJDAR*, 10(3-4):211–226, 2007. 10.1007/s10032-007-0052-2.

[87] N. Milošević, C. Gregson, R. Hernández, and G. Nenadic. Disentangling the structure of tables in scientific literature. In *NLDB*, pages 162–174, 2016. 10.1007/978-3-319-41754-7_14.

[88] G. Moise, J. Sander, and M. Ester. P3C: a robust projected clustering algorithm. In *ICDM'06*, pages 414–425, 2006. 10.1109/ICDM.2006.123.

[89] F. Morstatter, A. Galstyan, G. Satyukov, D. M. Benjamin, A. Abeliuk, M. Mirtaheri, P. Szekely, E. Ferrara, A. Matsui, M. Steyvers, S. Bennet, D. Budescu, M. Himmelstein, M. D. Ward, A. Beger, M. Catasta, R. Sosic, J. Leskovec, P. Atanasov, R. Joseph, R. Sethi, and A. Abbas. SAGE: a hybrid geopolitical event forecasting system. In *IJCAI (1)*, 2019. URL https://www.ijcai.org/Proceedings/2019/0955.pdf.

[90] E. Muñoz, A. Hogan, and A. Mileo. Triplifying Wikipedia tables. In *LDIE*, pages 26–37, 2013. URL http://dl.acm.org/citation.cfm?id=2874472.2874476.

[91] E. Muñoz, A. Hogan, and A. Mileo. Using linked data to mine RDF from Wikipedia tables. In *WSDM*, pages 533–542, 2014. 10.1145/2556195.2556266.

[92] S. J. Nanda and G. Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evol. Comp.*, 16: 1–18, 2014. 10.1016/j.swevo.2013.11.003.

[93] A.-C. N. Ngomo, N. Heino, K. Lyko, R. Speck, and M. Kaltenböck. SCMS: semantifying content management systems. In *ISWC*, pages 189–204, 2011. 10.1007/978-3-642-25093-4_13.

[94] P. Nguyen, N. Kertkeidkachorn, R. Ichise, and H. Takeda. MTab: matching tabular data to knowledge graph with probability models. In *SemTab@ISWC*, pages 191–192, 2019. URL http://ceur-ws.org/Vol-2553/paper2.pdf.

[95] M. Nickel, L. Rosasco, and T. A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961, 2016. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewFile/12484/11828.

[96] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *AAAI*, pages 168–174, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14396.

[97] H. Okada and T. Miura. Detection of layout-purpose table tags based on machine learning. In *UAHCI*, pages 116–123, 2007. 10.1007/978-3-540-73283-9_14.

[98] Y. Oulabi and C. Bizer. Extending cross-domain knowledge bases with long tail entities using web table data. In *EDBT*, pages 385–396, 2019. 10.5441/002/edbt.2019.34.

[99] G. Penn, J. Hu, H. Luo, and R. T. McDonald. Flexible web document analysis for delivery to narrow-bandwidth devices. In *ICDAR*, pages 1074–1078, 2001. 10.1109/ICDAR.2001.953951.

[100] C. Quix. Data lakes: a solution or a new challenge for big data integration? In *DATA*, page 7, 2016.

[101] S. Rana, S. Jasola, and R. Kumar. A review on particle swarm optimization algorithms and their applications to data clustering. *Artif. Intell. Rev.*, 35(3):211–222, 2011. 10.1007/s10462-010-9191-9.

[102] L.-A. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation of Wikipedia. In *ACL*, pages 1375–1384, 2011. URL http://www.aclweb.org/anthology/P11-1138.

[103] A. M. Reina, P. Jiménez, and R. Corchuelo. A novel approach to web information extraction. In *BIS*, pages 152–161, 2015. 10.1007/978-3-319-19027-3_13.

[104] D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML tables to DBpedia. In *WIMS*, pages 10:1–10:6, 2015. 10.1145/2797115.2797118.

[105] J. C. Roldán. Kizomba: an unsupervised heuristic-based web information extractor. In *PAAMS*, pages 383–385, 2015. ISBN 978-3-319-40159-1.

[106] J. C. Roldán, P. Jiménez, and R. Corchuelo. Extracting web information using representation patterns. In *HotWeb*, pages 1–5, 2017. 10.1145/3132465.3133840.

[107] J. C. Roldán, P. Jiménez, and R. Corchuelo. On extracting data from tables that are encoded using HTML. *Knowl.-Based Syst.*, pages 1–19, 2019. https://doi.org/10.1016/j.knosys.2019.105157.

[108] J. C. Roldán, P. Jiménez, P. Szekely, and R. Corchuelo. Tomate: Online Multi-document Automated Table Extraction. *Information Sciences*, pages 1–39, 2020. Under review.

[109] J. C. Roldán, P. Jiménez, P. Szekely, and R. Corchuelo. Unsupervised data extraction from web tables. In *VLDB*, pages 1–10, 2021. Under review.

[110] A. Sahugueta and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data & Knowl. Eng.*, 36:283–316, 2001. 10.1016/S0169-023X(00)00051-3.

[111] S. Sarawagi. Information extraction. *Foundations and trends in databases*, 1(3):261–377, 2008. 10.1561/1900000003.

[112] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC, 4th edition, 2007.

[113] K. Sim, V. Gopalkrishnan, A. Zimek, and G. Cong. A survey on enhanced subspace clustering. *Data Min. Knowl. Discov.*, 26(2):332–397, 2013. 10.1007/s10618-012-0258-x.

[114] H. A. Sleiman and R. Corchuelo. An information extraction framework. In *PAAMS (Workshops)*, pages 149–156, 2012. 10.1007/978-3-642-28795-4_18.

[115] H. A. Sleiman and R. Corchuelo. TEX: an efficient and effective unsupervised web information extractor. *Knowl.-Based Syst.*, 39:109–123, 2013. 10.1016/j.knosys.2012.10.009.

[116] H. A. Sleiman and R. Corchuelo. A survey on region extractors from web documents. *IEEE Trans. Knowl. Data Eng.*, 25(9):1960–1981, 2013. 10.1109/TKDE.2012.135.

[117] H. A. Sleiman and R. Corchuelo. A class of neural-network-based transducers for web information extraction. *Neurocomputing*, 135: 61–68, 2014. 10.1016/j.neucom.2013.05.057.

[118] H. A. Sleiman and R. Corchuelo. Trinity: on using trinary trees for unsupervised web data extraction. *IEEE Trans. Knowl. Data Eng.*, 26(6): 1544–1556, 2014. 10.1109/TKDE.2013.161.

[119] J. W. Son and S.-B. Park. Web table discrimination with composition of rich structural and content information. *Appl. Soft Comput.*, 13 (1):47–57, 2013. 10.1016/j.asoc.2012.07.025.

[120] M. Taheriyan, C. A. Knoblock, P. A. Szekely, and J. L. Ambite. Learning the semantics of structured data sources. *J. Web Sem.*, 37:152–169, 2016. 10.1016/j.websem.2015.12.003.

[121] A. Thawani, M. Hu, E. Hu, H. Zafar, N. T. Divvala, A. Singh, E. Qasemi, P. A. Szekely, and J. Pujara. Entity linking to knowledge graphs to infer column types and properties. In *SemTab@ISWC*, pages 25–32, 2019. URL http://ceur-ws.org/Vol-2553/paper4.pdf.

[122] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080, 2016. URL http://proceedings.mlr.press/v48/trouillon16.pdf.

[123] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Comput. Surv.*, 38(2):#4, 2006. 10.1145/1132956.1132957.

[124] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the Web. *VLDB*, 4: 528–538, 2011. 10.14778/2002938.2002939.

[125] M. J. Walker. 2017 hype cycles highlight enterprise and ecosystem digital disruptions. Technical report, Gartner, 2017. URL https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/.

[126] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: a survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743, 2017. 10.1109/TKDE.2017.2754499.

[127] Y. Wang and J. Hu. Detecting tables in HTML documents. In *DAS*, pages 249–260, 2002. 10.1007/3-540-45869-7_29.

[128] Wikipedia. On downloading the Wikipedia database, 2020. URL https://en.wikipedia.org/wiki/Wikipedia:Database_download.

[129] I. Witten, E. Frank, M. Hall, and C. Pal. *Data mining*. Morgan Kaufman, 2016.

[130] K.-L. Wu, S.-K. Chen, and P. S. Yu. Dynamic refinement of table summarization for m-commerce. In *WECWIS*, pages 179–186, 2002. 10.1109/WECWIS.2002.1021257.

[131] X. Wu, C. Cao, Y. Wang, J. Fu, and S. Wang. Extracting knowledge from web tables based on DOM tree similarity. In *KSEM*, pages 302–313, 2016. 10.1007/978-3-319-47650-6_24.

[132] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015. 10.1007/s40745-015-0040-1.

[133] R. Xu and D. C. Wunsch. Survey of clustering algorithms. *IEEE Trans. Neural Networks*, 16(3):645–678, 2005. 10.1109/TNN.2005.845141.

[134] Y. Yang and W.-S. Luk. A framework for web table mining. In *WIDM*, pages 36–42, 2002. 10.1145/584931.584940.

[135] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, and S. Soderland. TextRunner: open information extraction on the Web. In *HLT-NAACL (Demonstrations)*, pages 25–26, 2007. URL https://www.aclweb.org/anthology/N07-4013.pdf.

[136] M. Yoshida, K. Torisawa, and J. Tsujii. A method to integrate tables of the World Wide Web. In *WDA*, pages 31–34, 2001. URL http://wda2001.csc.liv.ac.uk/Papers/13_yoshida_wda2001.pdf.

[137] N. Yuhanna, G. Leganza, R. Perdoni, and M. Bakalar. The Forrester Wave: data management for analytics, Q1 2020. Technical report, Forrester, 2018. URL https://www.forrester.com/report/The+Forrester+Wave+Data+Management+For+Analytics+Q1+2020/-/E-RES157286.

[138] R. Zanibbi, D. Blostein, and J. R. Cordy. A survey of table recognition. *IJDAR*, 7(1):1–16, 2004. 10.1007/s10032-004-0120-9.

[139] S. Zhang and K. Balog. Web table extraction, retrieval, and augmentation: a survey. *ACM Trans. Intell. Syst. Technol.*, 11(2):13:1–13:35, 2020. 10.1145/3372117.

This document was typeset on July 9, 2020 at 18:09 using class RC–BOOK α2.15 for LaTeX$_{2\epsilon}$. As of the time of writing this document, this class is not publicly available since it is in alpha version. Only members of The Distributed Group are using it to typeset their documents. Should you be interested in giving forthcoming public versions a try, please, do contact us at contact@tdg-seville.info. Thanks!