Information Sciences 577 (2021) 49–68

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

TOMATE: A heuristic-based approach to extract data from HTML tables

Juan C. Roldán [a,*], Patricia Jiménez [a], Pedro Szekely [b], Rafael Corchuelo [a]

[a] University of Seville, ETSI Informática, Avda. Reina Mercedes s/n, Sevilla 41012, Spain
[b] University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA

ARTICLE INFO

Article history:
Received 28 May 2020
Received in revised form 24 February 2021
Accepted 28 April 2021
Available online 04 May 2021

Keywords:
HTML tables
Data extraction

ABSTRACT

Extracting data from user-friendly HTML tables is difficult because of their different layouts, formats, and encoding problems. In this article, we present a new proposal that first applies several pre-processing heuristics to clean the tables, then performs functional analysis, and finally applies some post-processing heuristics to produce the output. Our most important contribution is regarding functional analysis, which we address by projecting the cells onto a high-dimensional feature space in which a standard clustering technique is used to make the meta-data cells apart from the data cells. We experimented with two large repositories of real-world HTML tables and our results confirm that our proposal can extract data from them with an F_1 score of 89.50% in just 0.09 CPU seconds per table. We confronted our proposal with several competitors and the statistical analysis confirmed its superiority in terms of effectiveness, while it keeps very competitive in terms of efficiency.

1. Introduction

Machine learning is boosting many services on the Web. This explains the increasing need for datasets that are used both to train and to exploit them. We are interested in datasets that are encoded using HTML tables, with an emphasis on user-generated tables. Many such tables provide data that cannot be found in common knowledge bases, which makes it difficult to use them in automated processes [15]. Data extractors are software components that analyse web documents and return their data as records that facilitate automatic processing. Many of them are general-purpose since they do not focus on a particular data layout [20,8], but there are also many table-specific proposals [18,25].

We have analysed many table-specific proposals [18] and our conclusion is that they typically rely on the following tasks: location, segmentation, discrimination, functional analysis, structural analysis, and interpretation. The literature provides many solutions to implement them; our focus is on the functional analysis task since it has been paid less attention, which provides enough room for improvement [3,24,23,11,10,1,2,6,7,4,12,14]. Unfortunately, only a few of the existing proposals can deal with arbitrary table layouts (e.g., horizontal listings, vertical listings, or matrices), they have problems to deal with the many different formats used to display the data (e.g., multi-line headers, no headers at all, context cells, repeated headers, or factorised cells), and they also have trouble with typical encoding problems (e.g., inconsistent row lengths or incorrect

* Corresponding author.
E-mail address: jcroldan@us.es (J.C. Roldán).

https://doi.org/10.1016/j.ins.2021.04.087
0020-0255/

uses of tags `td` and `th`). Simply put: they may easily have problems with 64.41% of the tables in our experimental repositories.

TOMATE[1] is a new proposal that is specifically devised to deal with HTML tables and improves on our previous results [17–20,18]. Its key contribution is regarding the functional analysis task: it projects the cells onto a rich feature space in which a standard clustering technique plus some custom heuristics are used to make meta-data cells apart from data cells. Our experimental analysis reveals that it can attain an F_1 score of 89.50%, which is 5.68% better than the best supervised competitor and 24.11% better than the best unsupervised competitor; furthermore, it takes an average of 0.09 CPU seconds per table. (The differences were proven statistically significant at the standard confidence level.) Currently, it is being used to feed an IARPA system with data that are extracted from Wikipedia tables; most such tables are user-generated, which means that the data extractors that are currently deployed at major knowledge bases cannot deal with them.

The rest of the article is organised as follows: Section 2 reviews the related work; Section 3 describes how TOMATE implements the data-extraction tasks; Section 4 shows the results of our experimental analysis; Section 5 presents our case study; finally, Section 6 concludes the article.

2. Related work

In this section, we present the related work. We first summarise the literature and then discuss and compare the existing proposals to ours.

2.1. Summary of the literature

A data extractor is a tool that uses rules to seek for data in HTML documents and returns them as records [20,8,18,25]. They can be classified as hand-crafted, if the user must devise the rules, supervised, if the user provides a learning set with annotations, unsupervised, if the user provides a learning set without annotations, and heuristic-based, if the rules are actually built-in heuristics that have proven to work well with many different documents.

The literature provides many general-purpose proposals [20,8]. Unfortunately, they do not take the intricacies of tables into account. This motivated many authors to work on specific-purpose proposals [18,25]. Simply put: their goal is to accurately group the meta-data cells into headers, the data cells into tuples, and then return records that reflect the relationships between the headers and the components of the tuples.

Initially, the challenge was to extract data from tables that were encoded using pre-formatted text or images. More recently, the focus has shifted towards tables that are encoded using mark-up languages like HTML, XML + XLST, or the OpenDoc spreadsheet standards. Our focus is on tables that are encoded using HTML. They have some unique characteristics that make it difficult to extract data from them, including: captions, context data outside the tables, cells that are encoded using inappropriate tags, or richer cell contents (including images or forms), not to mention that many HTML tables are not used to display data but to position other elements on the screen. We do not focus on the important challenge of understanding spreadsheet tables, since it would require to address some extra problems: inferring table delimiters, identifying different and more varied table layouts, a higher frequency of context cells, and the need for different features related to the predefined formats, formulae, conditional formatting, and sorting criteria.

Many researchers address this problem using a pipeline with the following tasks: 1) location, which returns the excerpts of the input documents that contain tables; 2) segmentation, which transforms the excerpts into grids of cells; 3) discrimination, which discards non-data tables; 4) functional analysis, which identifies whether the cells provide data or meta-data; 5) structural analysis, which identifies groups of related meta-data cells (the headers) and groups of related data cells (the tuples); and 6) interpretation, which transforms the input tables into record sets. We identified this pipeline after analysing and comparing the trends in the field over the last 20 years of research [18].

Our focus is on the functional analysis task because our experience suggests that the quality of the results 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 this task [3,24,23,11,10,1,2,6,7,4,12,14]. They address the problem using a variety of approaches that range from using heuristics that are tailored for a particular table repository to using deep-neural networks.

2.2. Discussion and comparison

The literature provides many general-purpose proposals to extract data from arbitrary documents. Unfortunately, they do not seem to work well with tables [1], which we could confirm in our previous work [17–20]. Some unsupervised proposals are based on analysing the structure of the HTML, which is problematic insofar HTML is intended to describe the tables on a per-row basis, but their layout (as interpreted by a person) can be a horizontal listing, a vertical listing, or a matrix. Other unsupervised proposals seek to identify differences amongst several HTML documents that are generated by the same template, which is also problematic insofar the meta-data cells are typically the same. The supervised proposals might, in

[1] TOMATE is publicly available at http://tomatera.tdg-seville.info.

principle, be trained to correctly interpret HTML tables, but they involve too much human effort and cannot be easily adapted to the existing variety of layouts, formats, and encoding problems.

The previous problems have motivated many authors to work on proposals that are specifically targeted to extract data from HTML tables [18,25]. Our focus is on the unsupervised and heuristic-based ones because the human effort involved in hand-crafted or supervised proposals is not scalable to the Web. The literature provides many such approaches to locate, segment, and discriminate tables, which means that it is difficult to make novel contributions regarding those tasks; furthermore, many authors pay little attention to the structural analysis or the interpretation tasks, which means that they are relatively easy to implement once the function of the cells is properly identified. There is, however, some room for improvement regarding the functional analysis task: some of the existing proposals can deal with horizontal listings only [1,2,4,6]; a few can also deal with vertical listings [23,24]; a few others can also deal with matrices [12]; only a few can deal with any table layouts [3,11,10,7,14]. Generally speaking, they 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 [18]. Unfortunately, roughly 64.41% of the tables in our experimental repositories have one or more of the previous problems.

This motivated us to work on TOMATE, which can deal with all of the problems holistically. It differs from the existing approaches in that it first projects the cells onto a rich feature space in which the meta-data and data cells can be made apart using a standard clustering technique plus some custom heuristics. This approach is novel and has proven to work very well in practice.

3. Our proposal

In this section, we describe our proposal. We first present some preliminary concepts and then delve into our solution to implement each of the tasks involved in the table understanding pipeline, cf. Fig. 1.

3.1. Preliminaries

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

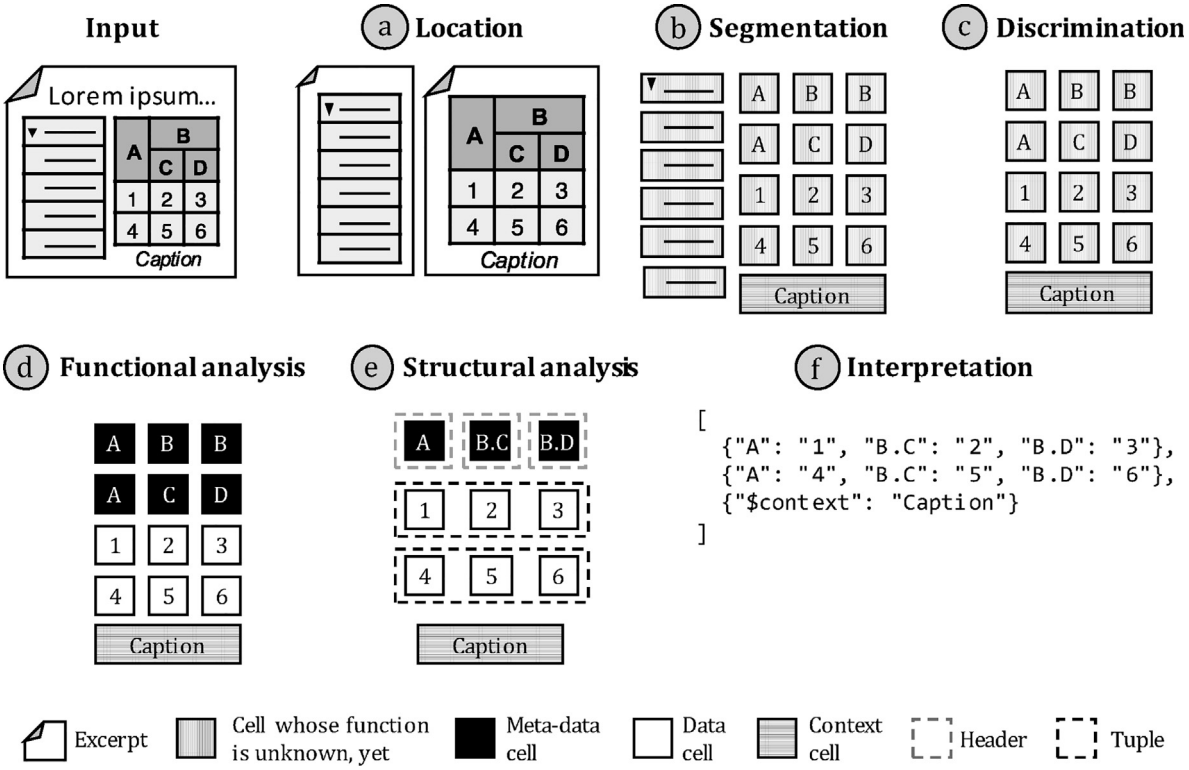


Fig. 1. Sketch of our proposal.

Definition 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 ($p \geqslant 1$). We denote its domain as $\text{dom } m = \{k_i\}_{i=1}^{p}$ and its range as $\text{ran } m = \{v_i\}_{i=1}^{p}$. 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 (where θ denotes the addition, the subtraction, or the multiplication operators):

$$\begin{aligned} z\,\theta\, m = &\quad \{k \,:\, z\,\theta\, m[k] \mid k \in \text{dom } m\}, \\ m_1\,\theta\, m_2 = &\quad \{k \,:\, m_1[k]\,\theta\, m_2[k] \mid k \in \text{dom } m_1 \cap \text{dom } m_2\}, \text{ and} \\ dist(m_1,\, m_2) = &\quad \sqrt{\sum_{k \in \text{dom } m_1 \cap \text{dom } m_2} (m_1[k] - m_2[k])^2}. \end{aligned}$$

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

Definition 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 $children(a)$, the area of its bounding box as $area(a)$, and its attribute vector as $attr(a)$. Table 1 presents the attributes that we take into account, namely: style attributes, which are related to display properties, structural attributes, which are related to DOM-tree properties, lexical attributes, which are related to textual content features, and miscellaneous attributes, which are related to some syntax and statistical properties.

Definition 3. Tables and cells. A table is a structure that helps position other elements on a grid. We are interested in tables whose elements are relevant data to be extracted. There are three common layouts for data tables, namely: horizontal listings, vertical listings, and matrices. 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. A group of related meta-data cells is a header and a group of related data cells is a tuple.

Definition 4. Features. A feature is a numeric characteristic of a cell that is computed from the attributes of its DOM node, cf. Table 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 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 ($r \geqslant 1$). 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 headers and the components of the tuples.

Table 1
Taxonomy of attributes.

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, ∞]

3.2. Task a: location

This task works on the input HTML documents and returns their tables, if any. It performs the following steps: 1) the input documents are downloaded using a headless browser, which runs a script to make the attributes of the HTML elements explicit; 2) the documents are then transformed into their corresponding DOM trees using a standard HTML parser; and 3) the tables are extracted using a CSS selector that fetches the sub-trees whose root node has tag `table`. These steps can be implemented very straightforwardly using commodity technology, so we do not provide any additional details.

3.3. Task b: 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. This task performs the following steps: 1) the table is pre-processed; 2) the base features are computed; 3) the deviation features are computed; 4) the features are normalised; and 5) the 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 ($\alpha \geqslant 1, \beta \geqslant 1$) using the following procedure:

1. If the `table` tag or any of its ancestors has attribute `dir` set to `rtl`, then it is flipped horizontally so that the first column is always the left-most one.
2. The maximum cell span is set to 200 cells, which helps avoid overhead when processing tables that are incorrectly encoded.
3. The cells whose span is greater than one are replicated accordingly and their span is set to one.
4. The rows that are shorter than the largest row are padded to the right using empty cells, which prevents outputting ragged tables.
5. 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 base feature vectors.

First, it computes the attributes of the DOM nodes, cf. Table 1. The style attributes are computed using a headless browser and the other attributes are computed using simple procedures, except for attribute "Likelihood of being meta-data", which was pre-computed as follows:

1. The tables in the repositories were cleaned by lower-casing their contents, stripping white spaces, and replacing the digits with a generic "DIGIT" token.
2. Every cell in the first row or column was flagged as a meta-data cell and the remaining ones were flagged as data cells.
3. 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 repositories.
4. The previous likelihood was adjusted as follows: 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.
5. The previous two operations were repeated five times, which was enough for the likelihoods to stabilise.

Please, note that we do not claim that the previous likelihoods are good meta-data probability estimates. They simply help compute features that have proven to work well when they are combined with the other features.

Next, we 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:

$$bfeat(a) = \omega_1(a)\, attr(a) + \sum_{b \in children(a)} \omega_2(a,b)\, bfeat(b),$$

where $\omega_1(a) = 1 - \sum_{b \in children(a)} \omega_2(a,b)$ and $\omega_2(a,b) = area(b)/area(a)$. Note that $\omega_1(a)$ 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 $b \in children(a)$, denotes the percentage of area of the bounding box that depends on node b relative to the total area of node a. Simply put: 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, where the weights are the relative area of the bounding boxes of the corresponding nodes.

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 cell c and base feature f, we compute the following deviation features: f^r, f^c, and f^t; respectively, they measure the deviation of f in c with respect to the cells in the same row, column, and table. To compute them, we need the following ancillary variables:

$$\begin{aligned} R_i &= \sum_{j=1}^{\beta} 1/\beta M[i,j] \ (1 \leqslant i \leqslant \alpha), \\ C_j &= \sum_{i=1}^{\alpha} 1/\alpha M[i,j] \ (1 \leqslant j \leqslant \beta), \text{and} \\ T &= \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} 1/(\alpha\beta) M[i,j], \end{aligned}$$

where M denotes the matrix returned by the previous step and R_i, C_j, 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) $(1 \leqslant i \leqslant \alpha, 1 \leqslant j \leqslant \beta)$. The deviation features are then defined as follows:

$$\begin{aligned} f_p^r &= \left(v_p - R_i[f_p]\right)^2, \\ f_p^c &= \left(v_p - C_j[f_p]\right)^2, \text{ and} \\ f_p^t &= \left(v_p - T[f_p]\right)^2, \end{aligned}$$

where $1 \leqslant p \leqslant \kappa$.

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, there are several approaches to perform the normalisation, e.g., global min–max, local min–max, standard normal, or softmax. Unfortunately, none of them stands out from a conceptual point of view, which means that this is a variation point. Thus, the decision regarding which of the alternatives must be used requires some experimentation and must be delayed to the following section.

Step 5: processing empty cells. This step works on the input table and returns a new table in which irrelevant empty cells are made explicit.

Every cell whose content consists exclusively of white spaces, dashes, asterisks, or question marks is flagged as empty. 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 cells that are considered empty preserve their features, but they are not taken into account for interpretation purposes.

Step 6: processing factorised cells. This operation processes the factorised cells as follows:

1. First, full-span rows are identified. They are rows that originally consisted of one single cell that spanned a row. According to their positions, they can be classified as top full-span rows, which are full-span rows in the first row or in a row in which every previous row up to the first row is also a full-span row, bottom full-span rows, which are full-span rows at the last row or in a row from which the remaining rows until the last one is also a full-span row, or middle full-span rows, which are full-span rows with no full-span rows before or after them.
2. 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 to the first bottom full-span row using the Euclidean distance amongst their corresponding feature vectors. If they are more similar to the former, then every middle full-span row and the last top 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 every middle full-span row and the first bottom full-span row are appended to the end of the previous row as an additional column.
3. If the full-span rows are not repeated periodically, then every middle full-span row and the last top full-span row are appended to the following rows as an additional column until another full-span row is found. To determine the position of the new column, we compute $\overline{c}$ as the average of the deviation features per column and $\overline{r}$ as the average of the deviation features per row, excluding full-span rows. If $\overline{r} > \overline{c}$, then the new column is placed on the right 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 on the left since the table is more likely to be a vertical listing and the factorised cells are more likely to be spanned metadata.
4. The procedure is then repeated on a per-column basis to deal with tables that have full-span columns.
5. Finally, the rows and columns in which every cell is empty or the result of repeating a spanned or factorised cell are removed. The remaining full-span rows or columns are removed and saved as context cells since they usually provide captions or footnotes that are worth preserving. Note that the cells identified as factorised cells preserve their original features.

Example 1. Figs. 2–4 illustrate how factorised cells are processed.

The table in Fig. 2 shows information 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. The second operation indicates that they must be added to the right of the previous rows as an additional column.

The table in Fig. 3 shows information about the medals won by a basketball player. 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. 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. Only a full-span row is kept after factorisation. It is not difficult to realise that the content of this row is the caption of the table, which is preserved as a context cell.

The table in Fig. 4 shows information 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. According to our heuristics, 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.

3.4. Task c: discrimination

This task takes a table returned by the segmentation task and discriminates it as follows: 1) the tables whose `width` or `height` attributes are `0px` or whose `display` attribute is `none` are discarded; 2) the tables with an ancestor `table` tag or a descendant user-input tag are also discarded; 3) the tables that have a single row or a single column are discarded. For the sake of efficiency, our proposal implements the first two operations during the location task since they do not require to segment the table.

3.5. Task d: functional analysis

This task takes a table produced by the segmentation task 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: 1) the dimensionality of the feature vectors is reduced; 2) candidate cell functions are computed; and 3) cell functions are identified. Next, we provide additional details on each step, cf. Fig. 5.

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

Regarding feature selection, we did not find any clear arguments in favour of using a particular group of features. Thus, we need to introduce a variation point in which we consider the following alternatives: 1) using all of the groups, 2) using a single group, 3) using combinations of two groups, and 4) using combinations of three groups. This results in a total of 15 alternatives.

Regarding dimensionality reduction, neither is there a clear decision, so we introduce a new variation point with the following alternatives: 1) not performing dimensionality reduction, 2) using Principal Component Analysis, and 3) using feature agglomeration.[2]

Step 2: computing candidate cell functions. This step takes the $\alpha \times \beta$ matrix with vectors computed by the previous task 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 with the following alternatives: 1) k-means using k-means^{++} as the initialisation method and mini-batches when dealing with more than one thousand cells; and 2) agglomerative clustering, which recursively merges pairs of clusters that minimally increase the linkage distance until two clusters are found.[3]

Next, 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: 1) $r_i = (a_i/\beta + b_i/\alpha)/2$, 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 ($i \in \{1,2\}$). 2) $s_i = 1 - |K_i|/\alpha\beta$; 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 ($i \in \{1,2\}$). 3) $c_i = 1 - \sum(p,q) \in K_i \sqrt{p^2+q^2}/|K_i|$; 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 $(r_1 + s_1 + c_1)/3 \geqslant (r_2 + s_2 + c_2)/3$; otherwise, we assume that the data cells are in cluster K_1 and the meta-data cells are in cluster K_2.

[2] We do not consider t-SNE because our experiments on a random sample of 200 tables resulted in an F_1 score of 23.12%. This method is good to reduce a dataset to two or three dimensions for visualisation, which is insufficient to detect the function of the cells.

[3] 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.

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...					

Fig. 2. Factorised cells that must be added to the right of the previous rows.

Medals won by 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

Fig. 3. Factorised cells that must be added to the right of the following rows.

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	

Fig. 4. Factorised cells that must be added to the left of the following rows.

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 $(1,j)$, $2 \leqslant j \leqslant \beta$, A_3, which consists of the set of cells at positions $(i,1)$, $2 \leqslant i \leqslant \alpha$, and A_4, which consists of the set of cells at positions (i,j), $2 \leqslant i \leqslant \alpha, 2 \leqslant j \leqslant \beta$. 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:

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

The intuition behind the previous formulation is that the Silhouette coefficient 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 $\text{silh}\{A_1 \cup A_2 \cup A_3, A_4\}$ since our intuition suggested that this would result in the highest Silhouette coefficient 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. In such cases, 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 ($1 \leqslant i \leqslant 4$):

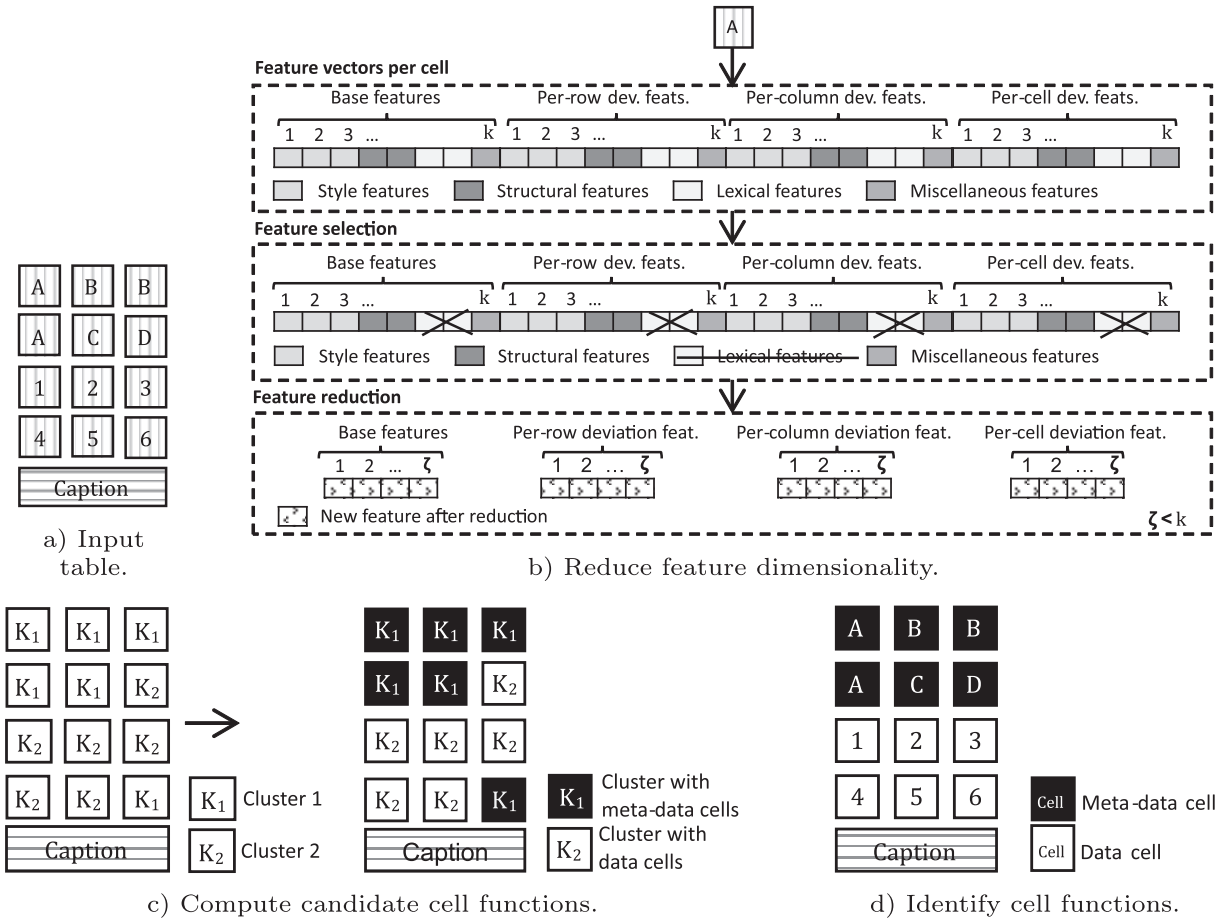
$$w^r = \text{dist}(u_1, u_3) + \text{dist}(u_2, u_4) - \text{dist}(u_1, u_2) - \text{dist}(u_3, u_4),$$
$$w^c = \text{dist}(u_1, u_2) + \text{dist}(u_3, u_4) - \text{dist}(u_1, u_3) - \text{dist}(u_2, u_4), \text{ and}$$
$$w^t = \text{dist}(u_3, u_4) + \text{dist}(u_2, u_4) - \text{dist}(u_1, u_2) - \text{dist}(u_1, u_3).$$

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 is as follows: 1) if the orientation is 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 $R = \varnothing$; 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 $C = \varnothing$; 2) next, we declare as data the cells at positions (i,j) such that $i \notin R$ and $j \notin C$; 3) now, we declare as meta-data the cells at positions (i,j) such that $i \in R$ and $1 \leqslant j \leqslant \beta$ or $1 \leqslant i \leqslant \alpha$ and $j \in C$; 4) finally, if both R and C are not empty, then we declare as data the cells at positions (i,j) such that $i \in R$ and $1 + \max C \leqslant j \leqslant \beta$ or $1 + \max R \leqslant i \leqslant \alpha$ and $j \in C$.

Example 2. Figs. 5–7 illustrate our approach to functional analysis.

Fig. 5 provides an overview of the task. Fig. 5.a shows a sample table with data. Fig. 5.b shows how each cell is represented as a vector with 4κ features that are grouped according to their types. After the feature vectors are computed, we perform feature selection to determine which groups of features can be dropped, if any. Then, we perform feature reduction to transform the original space of selected features into a smaller-dimensional space that makes computations faster. Fig. 5.c shows a sample clustering that is computed from the reduced feature vectors; note that the clusters simply make two groups of cells apart, whose functions are identified by means of the heuristics that were explained before. Fig. 5.d, shows the result of identifying the cell functions, which requires correcting the function of the cells that are isolated within a region in which most neighbours have the complementary function. The last step is a bit more involved. We provide additional details in Figs. 6 and 7.



a) Input table.

b) Reduce feature dimensionality.

c) Compute candidate cell functions.

d) Identify cell functions.

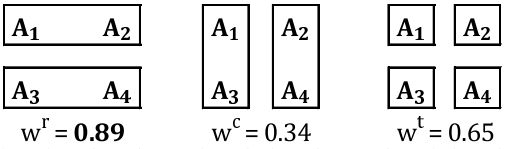
Fig. 5. Diagram of the functional analysis.

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.

Fig. 6. Correcting cell functions using the first method.

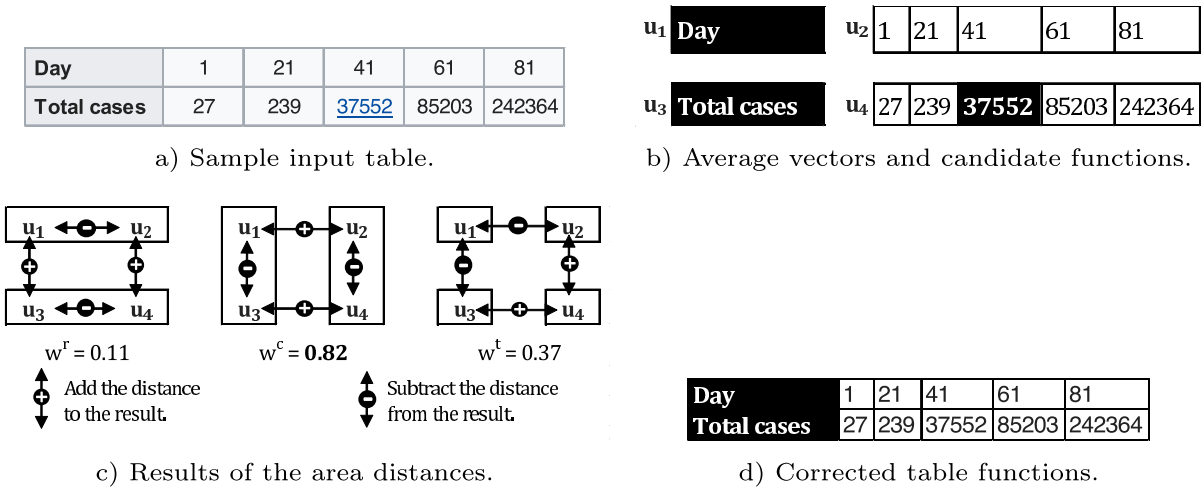


Fig. 7. Correcting cell functions using the second method.

Fig. 6.a shows a table with data about the terrestrial planets of the solar system. Fig. 6.b shows the four regions in which we divide the table and the candidate functions of the 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 cells. To identify the correct function, 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 clusterings and guesses the orientation according to the highest one. Fig. 6.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 can be assumed to be row-wise. Therefore, the candidate function is corrected according to the majority vote on a per-row basis, which assigns the correct function to the highlighted cells in the third row, cf. Fig. 6.d. Fig. 7.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 were used, the orientation would be estimated as table-wise because 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. Fig. 7.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, but vectors u_2 and u_4 refer to the regions that have the data cells. Fig. 7.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. Fig. 7.d shows the function returned by the functional analysis task after the candidate functions are corrected.

3.6. Task e: structural analysis

This task works on the table returned by the segmentation task plus the function matrix and the scores computed by the functional analysis task. It returns the headers and tuples in the input table. It performs the steps below:

1. 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. (The artificial meta-data can be generated using a pattern like $\$\texttt{attr\_}i$, where i denotes a sequential index.) 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.
2. 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.
3. 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.
4. 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 index-

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.

Fig. 8. Sample results of structural analysis.

ers, plus the cell itself. The proposal by Rastan et al. [16] provides further details on how to generate more descriptive headers in this case.

Example 3. Fig. 8.a shows a table with statistics about a few soccer teams. We identify 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 cells by the segmentation task since it is spanned. It then computes four headers and ten tuples. The first header is composed of the two meta-data cells, whereas the others were generated artificially. The tuples correspond to the cells in the body of the matrix plus their corresponding row and column indexers.

3.7. Task f: 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 with the extracted data.

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 ($r \geqslant 1$). The descriptors are computed from the headers as follows: the meta-data cells in a header are collapsed by catenating their contents using a user-defined separator, e.g., symbol "/"; if two meta-data cells that should be catenated have the same content, the second one is ignored since it typically results from a spanned cell; if 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, e.g., `$context`. Computing the output record set is now simple: just create 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.

Example 4. Fig. 9.a shows a table with stock market data. We identify 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 columns have vertical spans, which means that they are split into several cells. 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 Fig. 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. Note that the caption to the table is identified as a context cell, which is also added to the resulting record set.

4. Experimental analysis

In this section, we present our experimental analysis. First, we describe our experimental setting; next, we report on the variation points; then, we present our experimental results; finally, we present our statistical analysis.

4.1. Experimental setting

We implemented the proposals by Yoshida et al. [24], Jung and Kwon [10], and Embley et al. [7], which are well-known unsupervised techniques, as well as Nishida et al.'s [14] proposal, which is the most advanced supervised proposal of which we are aware. Unfortunately, there are not any public implementations of the proposals in the related work section; thus, we restricted our attention to the ones that are described with enough details to implement them accurately. The unsupervised

Company	Stock		Value	
	Type A	Type B		
AMZN	2145	156	$	234.89
NFLX	8922	302	$	212.22
TSLA	123	22	$	567.20
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.

Fig. 9. Sample result of the interpretation task.

proposals rely on algorithmic approaches that build on the Expectation Maximisation method, custom heuristics, or a custom search algorithm; the supervised one relies on a deep neural network.

We assembled two large experimental repositories with real-world tables that were fetched from the English Wikipedia [22] and the subset of English documents in the Dresden Web Table Corpus [5]. We randomly sampled enough documents to assemble two repositories with 1 496 and 1 513 tables, respectively. In the case of the Wikipedia, we limited our extraction to user-generated tables, since the data in tables that are generated using templates like `infobox`, `navbox`, or `sistersitebox` can be easily extracted.

To create our ground truth, we asked four people to annotate the tables in our repositories according to their layouts and the cells according to their functions.[4] Their agreement was very good since the Krippendorff Alpha coefficients were 80.36% and 96.11%, respectively. The disagreement was mainly due to matrices: many of them can be easily interpreted as listings and, some times, it was not clear if a cell had meta-data or data. The tables in our ground truth are distributed as follows: 70.65% are horizontal listings, 18.69% are vertical listings, 9.68% are matrices, and 0.97% have unknown layouts; the cells are distributed as follows: 10.46% are meta-data cells and 89.42% are data cells.

Table 2 provides an insight into the formatting and encoding problems found in our repositories. 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 repository; the reason might be that the authors of Wikipedia tables tend to use homogeneous formats 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 repository, only 6.15% of the tables have 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 repository are headless, whereas this indicator raises to 13.70% in the case of the DWTC repository. Finally, note that about 1.82% and 0.83% tables 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 repository and 3.94% of the tables in the DWTC repository have inconsistent row lengths; this is clearly due to the fact that the tables in the Wikipedia repository were encoded by a person, whereas many of the tables in the DWTC repository 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 repository). Summing up: 64.41% of the tables have one or more intricacies.

Our experiments consisted in running our proposal (TOMATE) and the competitors on the two repositories. The proposals were implemented using the Python 3.7 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 validation method. In

[4] Our datasets are publicly available at http://datasets.tdg-seville.info.

Table 2
Frequency of problems by repository.

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%

the case of unsupervised proposals, the training splits were ignored because they do not require to learn any models; in the case of the supervised proposal, the training splits were used to fit its neural network. We measured their effectiveness in terms of 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 measures were averaged from the results computed for each split; in the case of the supervised proposal, the training time was apportioned amongst the tables in the testing splits.

4.2. Variation points

We performed a grid search to find the best alternatives to implement the variation points in our proposal, namely: how to normalise the features, which groups of features must be selected to produce the candidate cell functions, which technique must be used to reduce their dimensionality, and which technique must be used to perform the clustering. The variation points resulted in 360 configurations that were explored in sequence, cf. Table 3.

Regarding normalisation, we found that the global min–max alternative was the best one. We realised that the local min–max normalisation or the softmax normalisation usually give more importance to features that are not significant for a person, e.g., a padding that 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.

Table 3
Average performance with each alternative.

Variation point	Description	Alternative	P	R	F1	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**

Regarding feature selection, we found that there are two good alternatives, namely: using the four groups of features or leaving the lexical features out. Furthermore, we observed that using a single feature group does not suffice in most experiments because each of them 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, cf. Table 4. 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 matches 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 the difference regarding efficiency would be negligible in practice, but the difference regarding effectiveness would not.

4.3. Experimental results

Table 5 presents our experimental results. For each proposal, we report on its performance measures in the Wikipedia and the DWTC repositories independently, plus its overall average performance on both repositories. 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.

Table 4
Top 10 configurations according to the F_1 score.

Normalisation	Feature selection	Dimensionality reduction	Clustering method	P	R	F1	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 5
Experimental results.

Layout	Proposal	Wikipedia				DWTC				Overall			
		P	R	F1	Time	P	R	F1	Time	P	R	F1	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 [24] proposal; P2 = Jung and Kwon's [10] proposal; P3 = Embley et al.'s [7]; P4 = Nishida et al.'s [14] proposal.

Our proposal attains the best overall precision, recall, and F_1 score in both repositories. Regarding horizontal listings, TOMATE attains the best overall effectiveness results although the proposal by Nishida et al. [14] is more competitive in the Wikipedia repository, since simple horizontal 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 Fig. 10.a. Regarding vertical listings, TOMATE has proven to be the most effective proposal in every repository. Regarding matrices, the proposal by Embley et al. [7] provides the best overall results but TOMATE provides very similar results.

The proposal by Yoshida et al. [24] 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 repositories.

The proposal by Jung and Kwon [10] 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 author highlights in a way that resembles meta-data cells; for instance, see the "Year" column in the table in Fig. 10.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 them.

The proposal by Embley et al. [7] 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 Fig. 10.b. This technique was devised to extract data from tables that resemble 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.

Nishida et al.'s [14] proposal attains the closest overall results to TOMATE. However, it seems to be the most inefficient one 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 repository, which provides many similar tables regarding the climate, sports results, books, or TV shows, just to mention a few examples.

4.4. Statistical analysis

We first checked our experimental results for normality and homoscedasticity using Shapiro–Wilk's and Levene's tests, respectively. 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 perform non-parametrical tests, e.g., Iman-Davenport's and Bergmann-Hommel's tests.

Tables 6.a–d show the results of the analysis. The first two columns present the empirical ranking; 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 p-values that are smaller than the standard significance level ($\alpha = 0.05$) are highlighted in boldface.

Note that TOMATE ranks the first regarding precision, recall, and 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; they also confirm that the differences in rank regarding the techniques immediately after is statistically significant.

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

a) Table with a highlighted numeric column.

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

b) Table with non-repeated values.

Fig. 10. Sample tables from Wikipedia.

Table 6
Statistical analysis.

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**
P1	3.60			P0 - P2	0.42E+02	**0.22E-307**
P2	3.79			P0 - P3	0.33E+02	**0.22E-307**
P3	3.38			P0 - P4	0.78E+01	**0.14E-13**
P4	2.29			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**
P1	3.66			P0 - P2	0.37E+02	**0.22E-307**
P2	3.65			P0 - P3	0.30E+02	**0.39E-200**
P3	3.35			P0 - P4	0.75E+01	**0.24E-12**
P4	2.34			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**
P1	3.55			P0 - P2	0.37E+02	**0.22E-307**
P2	3.61			P0 - P3	0.36E+02	**0.75E-286**
P3	3.56			P0 - P4	0.82E+01	**0.13E-14**
P4	2.32			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
P0	2.96			P0 - P1	0.23E+02	**0.10E-113**
P1	3.97			P0 - P2	0.30E+02	**0.99E-192**
P2	1.64			P0 - P3	0.34E+02	**0.11E-258**
P3	1.43	917.75	**0.0E+00**	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 [24] proposal; P2 = Jung and Kwon's [10] proposal; P3 = Embley et al.'s [7]; P4 = Nishida et al.'s [14] proposal.

Unfortunately, the improvement regarding effectiveness induces some inefficiency. The empirical ranking establishes that Embley et al.'s [7] proposal is the quickest one, but realise that it ranks at the fourth position regarding the effectiveness measures; it also establishes that Yung and Kwon's [10] proposal 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 best proposal with which we have compared it, which is a supervised one. It is 24.11% better than the next 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 table, which we think is good for practical purposes.

5. Case study

TOMATE was integrated into an IARPA system [13] whose goal is to help experts answer key political questions. The experts are expected to perform clerical tasks only, e.g., entering keywords to search the Wikipedia and selecting the tables that they think might provide relevant data. TOMATE extracts the data in the selected tables automatically and feeds the system so that it can perform forecasts that are shown using tables and charts.

Fig. 11 illustrates the system in the context of El Salvador 2019 presidential election. Let us assume that an expert has been posed a question and he or she has selected the table in Fig. 11.a. TOMATE analyses it as follows:

– *Location:* it fetches the document, makes its attributes explicit, parses it into a DOM tree, and selects the sub-trees with tag `table`.
– *Segmentation:* the first step pre-processes the input document as follows: there is no `dir` attribute, so the table does not need to be flipped 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); since all of the rows have the same number of columns, no correction is required; 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 their deviations 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 cells that 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, it is not within another table, and it does not contain any input elements.

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.

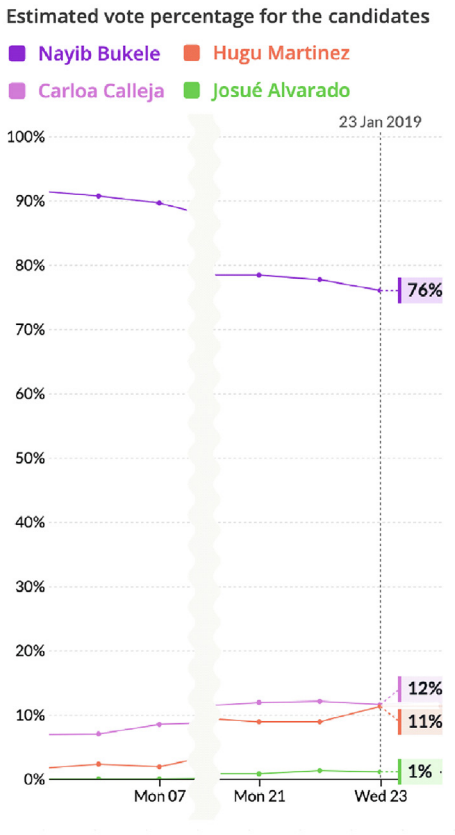
Date	Polling firm	Sample size	Carlos Calleja	Hugo Martínez		Nayib Bukele
Date	Polling firm	Sample size	Calleja (Ali...)	Martínez (FMLN)	Alvarado (VAMOS)	Bukele (GANA)
30 July 2018	CID-Gallup	806	24	5	0	38
31 July 2018	TResearchMx	3600	31.7	9.7	2.8	55.8
19 August 2018	TResearchMx	3600	30.2	9.7	1.1	55.9
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 = [
  ("Date"),
  ("Polling firm"),
  ("Sample size"),
  ("$attr_0"),
  ("$attr_1"),
  ("$attr_2")
]
```

```
tuples = [
  ("30 July 2018", "CID-Gallup", "806",
   "Carlos Calleja", "Calleja (Alianza ...)", "24"),
  ("30 July 2018", "CID-Gallup", "806",
   "Hugo Martínez", "Martínez (FMLN)", "5"),
  ("30 July 2018", "CID-Gallup", "806", "",
   "Alvarado (VAMOS)", "0"),
  ("30 July 2018", "CID-Gallup", "806",
   "Nayib Bukele", "Bukele (GANA)", "38"),
  ("31 July 2018", "TResearchMx", "3600",
   "Carlos Calleja", "Calleja (Alianza ...)", "31.7"),
  ("31 July 2018", "TResearchMx", "3600",
   "Hugo Martínez", "Martínez (FMLN)", "9.7"),
  ("31 July 2018", "TResearchMx", "3600",
   "", "Alvarado (VAMOS)", "2.8"),
  ("31 July 2018", "TResearchMx", "3600",
   "Nayib Bukele", "Bukele (GANA)", "55.8"),
  ...
]
```

c) Table after performing structural analysis.



d) Resulting chart.

Fig. 11. Sample opinion poll from Wikipedia.

– Functional analysis: first, this task reduces the dimensionality of the features that represent the cells, next computes the candidate cell functions, and then computes the final functions. Fig. 11.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 semantic hints 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., ("`$attr_2`*"). The tuples combine each of the cells in the body of the matrix with their corresponding row and column indexers. Fig. 11.c illustrates the results of this task.*
– Interpretation: this task leverages the results of the previous tasks and creates a record set in which the data in the tuples are explicitly associated with the descriptors that are computed from their corresponding headers. For instance, this is the first record returned: {`"Date": "30 July 2018", "Polling firm": "CID-Gallup", "Sample size": "806", "$attr_0": "Carlos Calleja", "$attr_1": "Calleja (Alianza por un nuevo país)", "$attr_2": 24`*}.*

The system shows the results of the interpretation task and helps the user decide on which records and which of their components must be used to feed the machine-learning engine that generates the forecasts, which are shown in charts like the one in Fig. 11.d. New approaches are currently being studied regarding entity matching [21] and system evaluation [9].

6. Conclusions

In this article, we have introduced TOMATE, which is an automated method to extract data from HTML tables. It can deal with all of the problems with which other proposals have difficulties: 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. Our experimental results prove that TOMATE ranks the first in terms of effectiveness since it can attain an F_1 score of 89.50%, which is 5.68% better than the best supervised competitor and 24.11% better than the best unsupervised competitor. It is also efficient enough for practical purposes, since it requires an average of 0.09 CPU seconds per table.

We have identified a number of problems that deserve to be studied in future, namely: identifying marginal table layouts, finding as much context data as possible, finding whether an empty cell is actually empty or factorised, dealing with tables that have multiple blocks of non-repeated headers, and removing decorator cells that are commonly used for advertisement purposes. None of them seems to be frequent enough to be a serious problem, but they all constitute a long tail of problems that deserve attention. Furthermore, we would like to study how to adapt this proposal to analyse data from spreadsheets, since many web tables are encoded using this approach.

CRediT authorship contribution statement

Juan C. Roldán: Conceptualization, Software, Validation, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Patricia Jiménez:** Conceptualization, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Pedro Szekely:** Conceptualization, Methodology, Investigation, Resources, Writing - original draft, Writing - review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Rafael Corchuelo:** Conceptualization, Methodology, Validation, Data curation, Investigation, Writing - original draft, Writing - review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work by Juan C. Roldán was partially supported by the Fulbright organisation and the University of Seville. The work done by Juan C. Roldán, Patricia Jiménez, and Rafael Corchuelo was supported by grants TIN2013-40848-R, TIN2016-75394-R, and P18-RT-1060. The work by Patricia Jiménez and Rafael Corchuelo was also supported by grant PID2020-112540RB-C44. The work by Pedro Szekely was supported by the FA8650-17-C-7715 USAF contract. Dinamic Area, S.L. provided some of the machinery that was used to perform the experimentation.

References

[1] M.J. Cafarella, A.Y. Halevy, D.Z. Wang, E. Wu, Y. Zhang, WebTables: exploring the power of tables on the Web, PVLDB 1 (1) (2008) 538–549, https://doi.org/10.14778/1453856.1453916.
[2] M.J. Cafarella, A.Y. Halevy, Y. Zhang, D.Z. Wang, and E. Wu, Uncovering the relational Web, in: WebDB, 2008, pp. 1–6. http://webdb2008.como.polimi.it/images/stories/WebDB2008/paper30.pdf..
[3] H.-H. Chen, S.-C. Tsai, J.-H. Tsai, Mining tables from large scale HTML texts, in: COLING, 2000, pp. 166–172. doi: 10.3115/990820.990845..
[4] X. Chu, Y. He, K. Chakrabarti, K. Ganjam, TEGRA: table extraction by global record alignment, in: SIGMOD, 2015, pp. 1713–1728. doi: 10.1145/2723372.2723725..
[5] J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, Top-*k* entity augmentation using consistent set covering, in: SSDBM, 2015, pp. 8:1–8:12. doi: 10.1145/2791347.2791353..
[6] H. Elmeleegy, J. Madhavan, A.Y. Halevy, Harvesting relational tables from lists on the Web, VLDB 20 (2) (2011) 209–226, https://doi.org/10.1007/s00778-011-0223-0.

[7] D.W. Embley, S.C. Seth, G. Nagy, Transforming web tables to a relational database, in: ICPR, 2014, pp. 2781–2786. doi: 10.1109/ICPR.2014.479..
[8] E. Ferrara, P. de Meo, G. Fiumara, R. Baumgartner, Web data extraction, applications, and techniques: a survey, Knowl.-Based Syst. 70 (2014) 301–323, https://doi.org/10.1016/j.knosys.2014.07.007.
[9] P. Jiménez, R. Corchuelo, H.A. Sleiman, ARIEX: automated ranking of information extractors, Knowl.-Based Syst. 93 (2016) 84–108, https://doi.org/10.1016/j.knosys.2015.11.004.
[10] S.-W. Jung, H.-C. Kwon, A scalable hybrid approach for extracting head components from web tables, IEEE Trans. Knowl. Data Eng. 18 (2) (2006) 174–187, https://doi.org/10.1109/TKDE.2006.19.
[11] Y.-S. Kim, K.-H. Lee, Detecting tables in web documents, Eng. Appl. AI 18 (6) (2005) 745–757, https://doi.org/10.1016/j.engappai.2005.01.009.
[12] N. Milošević, C. Gregson, R. Hernandez, G. Nenadic, Disentangling the structure of tables in scientific literature, in: NLDB, 2016, pp. 162–174. doi: 10.1007/978-3-319-41754-7_14..
[13] 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, A. Abbas, SAGE: a hybrid geopolitical event forecasting system, IJCAI 1 (2019) 6557–6559, https://doi.org/10.24963/ijcai.2019/955.
[14] K. Nishida, K. Sadamitsu, R. Higashinaka, Y. Matsuo, Understanding the semantic structures of tables with a hybrid deep neural network architecture, in: AAAI, 2017, pp. 168–174..
[15] Y. Oulabi, C. Bizer, Extending cross-domain knowledge bases with long tail entities using web table data, in: EDBT, 2019, pp. 385–396..
[16] R. Rastan, H.-Y. Paik, J. Shepherd, A. Haller, Automated table understanding using stub patterns, in: DSAA, 2016, pp. 533–548. doi: 10.1007/978-3-319-32025-0_33..
[17] J.C. Roldán. Kizomba: An unsupervised heuristic-based web information extractor, in: PAAMS, 2016, pp. 383–385..
[18] J.C. Roldán, P. Jiménez, R. Corchuelo, On extracting data from tables that are encoded using HTML, Knowl.-Based Syst. (2019) 1–19, https://doi.org/10.1016/j.knosys.2019.105157.
[19] H.A. Sleiman, R. Corchuelo, TEX: an efficient and effective unsupervised web information extractor, Knowl.-Based Syst. 39 (2013) 109–123, https://doi.org/10.1016/j.knosys.2012.10.009.
[20] H.A. Sleiman, R. Corchuelo, A survey on region extractors from web documents, IEEE Trans. Knowl. Data Eng. 25 (9) (2013) 1960–1981, https://doi.org/10.1109/TKDE.2012.135.
[21] A. Thawani, M. Hu, E. Hu, H. Zafar, N.T. Divvala, A. Singh, E. Qasemi, P.A. Szekely, J. Pujara, Entity linking to knowledge graphs to infer column types and properties, in: SemTab@ISWC, 2019, pp. 25–32..
[22] Wikipedia. Database download, 2020. URL https://en.wikipedia.org/wiki/Wikipedia:Database_download..
[23] Y. Yang, W.-S. Luk, A framework for web table mining, in: WIDM, 2002, pp. 36–42. doi: 10.1145/584931.584940..
[24] M. Yoshida, K. Torisawa, J. Tsujii, A method to integrate tables of the World Wide Web, in: WDA, 2001, pp. 31–34..
[25] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Trans. Intell. Syst. Technol. 11 (2) (2020) 13:1–13:35, https://doi.org/10.1145/3372117.