A clustering approach to extract data from HTML tables

Patricia Jiménez [*], Juan C. Roldán, Rafael Corchuelo

University of Seville, ETSI Informática, Avda. Reina Mercedes, s/n. Sevilla E-41012, Spain

Keywords:

HTML tables
Data extraction
Clustering
Genetic algorithms

A B S T R A C T

HTML tables have become pervasive on the Web. Extracting their data automatically is difficult because finding the relationships between their cells is not trivial due to the many different layouts, encodings, and formats available. In this article, we introduce Melva, which is an unsupervised domain-agnostic proposal to extract data from HTML tables without requiring any external knowledge bases. It relies on a clustering approach that helps make label cells apart from value cells and establish their relationships. We compared Melva to four competitors on more than 3 000 HTML tables from the Wikipedia and the Dresden Web Table Corpus. The conclusion is that our proposal is 21.70% better than the best unsupervised competitor and equals the best supervised competitor regarding effectiveness, but it is 99.14% better regarding efficiency.

1. Introduction

1.1. Context and motivation

We are interested in relational HTML tables, that is, tables that are encoded using HTML and are used to display relational data instead of arranging other elements on the screen (Milošević, Gregson, Hernández, & Nenadic, 2016; Wu, Cao, Wang, Fu, & Wang, 2016; Zhang & Balog, 2020). Several authors have crawled the Web and have found millions of relational tables, namely: Cafarella, Halevy, Zhang, Wang, and Wu (2008) found 154 million relational tables, Crestan and Pantel (2011) found 1.3 billion relational tables, Pimplikar and Sarawagi (2012) found 25 million relational tables, and the Web Table Corpora initiative found 233 million relational tables. This has recently boosted their popularity because they have helped publish many relational data so that they can be easily consumed through web browsers (Cafarella et al., 2018; Zhang & Balog, 2020).

Unfortunately, it is difficult for software agents to leverage their data (Braunschweig, Thiele, & Lehner, 2015; Milošević et al., 2016), chiefly in the cases in which they are not available in any knowledge bases. The previous case is not uncommon at all; for instance, Oulabi and Bizer (2019) analysed the relational HTML tables in a few domains that are well supported by DBpedia and extracted 206 690 data records with no matches in the knowledge base. Extracting those data in a format that is amenable for further processing is appealing in many contexts, including (meta-)data search, query expansion, document summarisation, question answering, knowledge discovery, synonym finding, improving accessibility, textual advertising, or creating linked data (Nishida, Sadamitsu, Higashinaka, & Matsuo, 2017; Oulabi & Bizer, 2019; Roldán, Jiménez, & Corchuelo, 2020; Wu et al., 2016).

In the literature, there are many proposals to extract data from HTML documents in general, not specifically tables (Ferrara, de Meo, Fiumara, & Baumgartner, 2014; Sleiman & Corchuelo, 2013a). They rely on text alignment (Sleiman & Corchuelo, 2013b), neural networks (Sleiman & Corchuelo, 2014), learning first-order rules (Jiménez & Corchuelo, 2016a), inferring propositio-relational rules (Jiménez & Corchuelo, 2016b), learning decision trees (Uzun, Agun, & Yerlikaya, 2013), embedding graphs (Jiménez,

[*] Corresponding author.
E-mail address: patriciajimenez@us.es (P. Jiménez).

Roldán, Gallego, & Corchuelo, 2020), or using n-grams and rendering information (Figueiredo, Assis, & Ferreira, 2017), to mention a few. Unfortunately, they do not seem to be appropriate to extract the underlying relationships between the cells in HTML tables (Cafarella et al., 2018), which motivated much work on table-understanding (Roldán et al., 2020; Zhang & Balog, 2020).

1.2. *Research goals and contributions*

Most of the table-understanding proposals revolve around three tasks: finding relational HTML tables in the input documents, identifying their cells and their corresponding roles, and generating the output data records. The first and the last tasks are relatively easy to implement using the current state of the art; the second one is the most challenging because none of the table-understanding proposals are able to deal with all of the most common table layouts, neither work they accurately when they face some common encoding or format problems. The main challenges are threefold: (a) to identify the roles of the cells and to uncover their relationships (Zhang & Balog, 2020), which are not explicitly encoded in HTML; (b) to work as unsupervisedly as possible since supervision (Cafarella et al., 2008; Nishida et al., 2017) requires human effort to assemble the learning datasets; and (c) to work on as many domains as possible, since proposals that were devised to extract data from a particular site, type of table, or rely on a particular knowledge base are not generally applicable (Milošević et al., 2016; Yang & Luk, 2002).

Our contribution is Melva. It is a proposal that can identify the roles of the cells, make their relationships explicit in a totally unsupervised manner, and is not bound with a particular domain. It is novel in that there are not any previous attempts in the literature to identify the role of a cell building on a clustering approach, which we address using a genetic strategy as the meta-heuristic to select the most informative features of the cells and to cluster them simultaneously. Our proposal proved to work and scale very well on more than 3000 tables that were gathered from the Wikipedia web site and from 781 other sites in the Dresden Web Table Corpus. Its hyper-parameters were set using grid search, but our results prove that it is not very sensitive to their exact values; this is important since it helps our proposal to deal with a variety of tables with minimal configuration effort. It achieved an average F_1 score of 0.84 and took an average time of 0.11 CPU seconds per table. We compared it to four competitors using a statistically-sound method. The conclusions were that our proposal can beat the best unsupervised proposal by 21.70% regarding effectiveness, while it is statistically indistinguishable from the state-of-the-art supervised proposal, but performs 99.14% more efficiently.

1.3. *Structure of the article*

Section 2 describes and compares the related work; Section 3 presents the details of our proposal; Section 4 reports on our experimental results; Section 5 presents our conclusions and some future work.

2. Review of the literature

In this section, we first report on the state of the art regarding extracting data from HTML tables and then discuss on it; next, we report on the state of the art regarding clustering and then discuss on it.

2.1. *Related work on extracting data from HTML tables*

The key to extract data from HTML tables is to identify the roles of the cells. Apart from some naive approaches (Braunschweig et al., 2015; Wu et al., 2016) that basically assume that the roles can be easily identified using the encoding or the position of the cells, the literature reports on several more involved heuristic-based and machine-learning approaches.

The heuristic-based approaches rely on extraction rules that have been devised by a person. Chen, Tsai, and Tsai (2000) devised a proposal that measures the similarity of the cells per rows/columns. The first rows/columns that are more dissimilar to each other are assumed to have the label cells and the others the value cells. Yang and Luk (2002) devised a proposal for numeric tables. It assumes that value cells contain a number or a range of numbers with an optional measurement unit, or strings like “N/A” or “nil”; otherwise, they are label cells. Kim and Lee (2005) presented a proposal where the topmost rows and the leftmost columns are classified as label cells as long as they meet some criteria regarding spanning and pattern-based coherency. Otherwise, they are classified as value cells. Jung and Kwon (2006) developed a proposal that attempts to identify two regions within the tables by analysing their tags, styles, and contents. It seeks for regions with spanned cells, common background colours or fonts, empty cells, and regions with cells whose contents match some patterns. When most of the heuristics agree to divide a table into two regions, the topmost and leftmost region is assumed to have the label cells and the other the value cells. Gatterbauer, Bohunsky, Herzog, Krüpl, and Pollak (2007) defined a number of common tables types and used several discrimination heuristics based on style features to classify the input tables. The authors revealed that their proposal needs to improve in order to work more effectively in a domain-independent manner. Embley, Seth, and Nagy (2014) developed a heuristic-based proposal that looks for four so-called critical cells. The first and fourth critical cells are set by default to the upper left corner and to the bottom right corner, respectively. The other two critical cells update their positions until the first two index every single cell in the region delimited by the last two critical cells, which is assumed to provide the value cells. Milošević et al. (2016) restricted their attention to the tables provided by the PubMed Central repository. They measure the similarity of the syntactic types of the cells contents on a per-column basis. When a fixed-size window contains cells with the same syntactic types, it stops, and the cells above that window are considered to be the label cells. The cells in a full-spanned row and in subsequent rows that also have spanned cells, the cells that are encoded in thead tags, or the cells that are in a row that is enclosed in horizontal lines are also considered label cells.

The machine-learning approaches rely on learning the extraction rules from annotated learning datasets. Yoshida, Torisawa, and Tsujii (2001) used the Expectation Maximisation method to estimate the probability that a cell contains a label or a value, which helps classify the input tables into several pre-defined types and then understand them. Cafarella et al. (2008) devised a supervised proposal that learns a classifier from a training set that provides many cells with structural and content-based features plus additional user-provided annotations. The classifier can then be used on unseen tables to classify their cells as either label or value cells. Nishida et al. (2017) devised the current state-of-the-art proposal, which relies on a deep neural network. They use one-hot encoding to represent the tokens in the table. Then they reduce the dimensionality with an embedding layer. The embeddings are then fed into a recurrent neural network that uses long short-term memory cells with an attention mechanism. This layer is followed by a convolutional layer and some filters. Finally, they use a fully-connected layer and a softmax activation layer to compute the probability that the input table matches a pre-defined layout. Once the layout is guessed, determining the role of the cells is relatively straightforward.

2.2. *Discussion on extracting data from HTML tables and implications*

A recent survey (Roldán et al., 2020) highlights that identifying the roles of the cells is paramount to extracting data from HTML tables. The problem has been addressed using different approaches (Braunschweig et al., 2015; Cafarella et al., 2008; Chen et al., 2000; Embley et al., 2014; Gatterbauer et al., 2007; Jung & Kwon, 2006; Kim & Lee, 2005; Milošević et al., 2016; Nishida et al., 2017; Wu et al., 2016; Yang & Luk, 2002; Yoshida et al., 2001), but they all have some inherent drawbacks regarding layouts, encodings, and formats.

Regarding the layouts, the proposals by Braunschweig et al. (2015) and Cafarella et al. (2008) cannot deal with vertical listings or matrices, the proposal by Milošević et al. (2016) cannot deal with vertical listings, and the proposals by Gatterbauer et al. (2007) and Yoshida et al. (2001) cannot deal with matrices. Wu et al.'s (2016) proposal can work on any layouts, but the authors mentioned that it may have difficulties to distinguish vertical listings from matrices.

Regarding the encoding, it is not difficult to find tables that use tag td to encode both label and value cells and tag th to highlight some value cells. This is a problem for the proposals by Jung and Kwon (2006), Milošević et al. (2016), and Wu et al. (2016). It is also very common to find tables that have heterogeneous row/column lengths because some of their cells are incorrectly spanned. This is problematic for the proposals by Braunschweig et al. (2015), Cafarella et al. (2008), and Yoshida et al. (2001).

Regarding the formats, it is usual to find tables that have multiple rows/columns of label cells, which helps structure the labels so that they are easier to understand; this is a problem with the proposal by Braunschweig et al. (2015). It is also common to find tables that do not have any label cells because the semantics are implicit in the context of the table or the data themselves; this is a problem with the proposals by Braunschweig et al. (2015), Chen et al. (2000), Embley et al. (2014), Jung and Kwon (2006), Kim and Lee (2005), Milošević et al. (2016), Yang and Luk (2002), and Yoshida et al. (2001).

Last, but not least, some proposals are supervised (Cafarella et al., 2008; Nishida et al., 2017). Sometimes, their effectiveness is superior to the unsupervised proposals, but they require a person to provide a learning dataset, which is a time-consuming and error-prone task. Furthermore, there are proposals whose heuristics are intended to deal with a subset of tables only, which is the case of Yang and Luk's (2002) proposal to extract data from numeric tables or Milošević et al.'s (2016) proposal to extract data from the tables in the PubMed Central repository. There are also a variety of emerging approaches that rely on external knowledge bases (Bhagavatula, Noraset, & Downey, 2015; Oulabi & Bizer, 2019; Ritze & Bizer, 2017; Zhang, 2017). Generally speaking, they aim at linking tables/rows/columns to the classes and/or properties and the data rows/columns to the entities in the knowledge base (Martínez-Rodríguez, Hogan, & López-Arévalo, 2020). Unfortunately, such approaches are not appropriate in our context because there are many tables whose data are not available in any external knowledge bases (Oulabi & Bizer, 2019).

Unfortunately, we have not found a proposal that overcomes all of the previous problems, which motivated us to work on it.

2.3. *Related work on clustering data*

There are many proposals to cluster data, which has motivated many authors to work on surveys that provide a taxonomy and highlight their key points, namely: Jain (2010) and Xu and Tian (2015) put an emphasis on describing the algorithmic approaches, whereas Alam, Dobbie, Koh, Riddle, and Rehman's (2014), Figueiredo et al.'s (2019), and García and Gómez-Flores's (2016) focused on meta-heuristic approaches; Deng, Choi, Jiang, Wang, and Wang (2016), García and Gómez-Flores (2016), and Sim, Gopalkrishnan, Zimek, and Cong (2013) put their emphasis on the proposals that can select the most informative features and cluster the data at the same time; Bong and Rajeswari (2011) put the emphasis on proposals that use multi-objective fitness functions.

Our analysis of the existing surveys suggests that the clustering techniques can be categorised along several dimensions, namely: (a) automatic or manual, depending on whether the number of clusters is learnt or provided by the user; (b) single-solution or multi-solution, depending on whether they return one single solution or multiple ones, (c) hierarchical or partitional, depending on whether they organise the clusters into a tree-like structure or a flat partition. (Partitioning algorithms can be further classified as hard or overlapping depending on whether a datum can belong to a single cluster or multiple clusters; partitional overlapping proposals can be further subdivided into fuzzy or soft proposals depending on whether the membership of a datum to a cluster is expressed as a likelihood or a binary value.); (d) single-way or multi-way, depending on whether they perform feature selection as a pre-processing step or during clustering; (e) algorithmic or meta-heuristic, depending on whether they compute the clusters using a standard algorithm or map the problem onto a nature-inspired model; and (f) single-objective or multi-objective, according to whether their fitness function relies on a single or multiple indicators.

Fitness functions help guide clustering towards the best possible results. They range from simple scores (Luna-Romera, García-Gutiérrez, Martínez-Ballesteros, & Riquelme-Santos, 2018; Xu & Tian, 2015) to complex multi-objective functions (Cava, Helmuth, Spector, & Moore, 2019). The main problem with multi-objective functions is that they combine a number of indicators that range in different intervals in which the interpretation of the lower and upper bounds (if any) is not homogeneous; this makes it difficult to project them onto a single-value score that can be compared to others using the standard greater-than-or-equal-to operator (Maulik, Bandyopadhyay, & Mukhopadhyay, 2011). This has motivated some research that led to the Lexicase procedure (Cava et al., 2019), which has proven particularly effective in the context of meta-heuristic approaches.

2.4. Discussion on clustering and implications

In our context, the clustering approach used must be manual (since we only need two clusters of cells), single-solution (since we only need to explore the best possible clustering), and hard-partitional (since the cells are expected to be either label or value cells, not both). In principle, the clustering technique used can be single- or multi-way, algorithmic or meta-heuristic, single- or multi-objective; however, we lean towards a multi-way, meta-heuristic, multi-objective technique.

We lean towards using a multi-way technique because the input cells are projected onto a high-dimensional feature space in which there are typically many uninformative features that introduce noise and contribute to ineffectiveness. Using a single-way technique is problematic insofar most existing feature selection procedures are supervised, but we are interested in devising a completely unsupervised technique; Sim et al.'s (2013) conclusions suggest that implementing feature selection independently from clustering typically leads to poor results, which justifies requiring a multi-way technique to select the most informative features in co-ordination with the clustering. We lean towards using meta-heuristic techniques because clustering is an NP-hard problem (García & Gómez-Flores, 2016; Jain, 2010) and meta-heuristics can naturally explore large search spaces using procedures that can be easily implemented using multiple threads. We lean towards a multi-objective fitness function because there is not a single indicator that works well in every case (Luna-Romera et al., 2018). Thus, it makes sense to combine several ones using the Lexicase procedure (Cava et al., 2019).

Unfortunately, we have not found a proposal that meets the previous requirements, which motivated us to work on it.

3. Our proposal

In this section, we first present some preliminary concepts; next, we describe our extraction method; then, we introduce our clustering approach; finally, we present our complexity analysis.

3.1. Preliminaries

Definition 1 (*Core Mathematical Concepts*)**.** A vector v is a tuple of the form $(v_1, v_2, \dots, v_d)$ $(d \geq 0)$. Its dimensionality is denoted as $\dim v = d$. Given vector v and a natural number i, its component at position i is denoted as $v[i]$ $(1 \leq i \leq \dim v)$. A matrix m is a vector of vectors $((v_{1,1}, v_{1,2}, \dots, v_{1,p}), \dots, (v_{n,1}, v_{n,2}, \dots, v_{n,p}))$ $(n \geq 1, p \geq 1)$. Given a matrix m and two natural numbers i and j, $m[i,j]$ denotes the component of m at position (i,j). We use two random variables: $\mathscr{B}_\beta$, whose distribution is a Bernoulli with mean β $(0.00 \leq \beta \leq 1.00)$, and $\mathscr{U}_{\beta_1,\beta_2}$, which is uniformly distributed in integer interval $[\beta_1, \beta_2]$ $(\beta_1 \leq \beta_2)$. $\square$

Definition 2 (*Documents, Tables, Cells, and Data Records*)**.** 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. A table is an HTML element with tag table, which can be used to display data or to arrange other elements on the screen. The former are called relational tables and the latter are called non-relational tables. The role of a cell within a table can be either label, which means that it provides a semantic hint, or value, which means that it provides a datum. Depending on how the label and value cells are arranged in a table, it is common to make three layouts apart, namely: horizontal listing, vertical listing, and matrices. A data record is a map of the form $\{h_i : v_i\}_{i=1}^k$ $(k \geq 0)$ that represents the data that have been extracted from a table. Each h_i is a header and each v_i is a value $(1 \leq i \leq k)$; the headers result from catenating one or more labels to form a descriptor that provides a semantic hint for the corresponding value. $\square$

Definition 3 (*Features*)**.** A feature is a property of a cell. We use feature vectors and feature matrices to represent the features of a single cell or all of the cells in a table, respectively. The features can be intra-cell features or inter-cell features. An intra-cell feature is computed from the attributes of a single cell; it can be visual, if it is computed from the rendering attributes, structural, if it is computed from the DOM tree structural attributes, or lexical/syntax, if it is computed from the content attributes. Inter-cell features are computed as the deviation of the intra-cell features of a cell with respect to the intra-cell features of the cells in the same row, column, or table. $\square$

3.2. Extracting data from HTML tables

This section describes our method to extract data from HTML tables, cf. Fig. 1. It works on an input set of documents D and returns a set of data records R, which is a map in which each input document and table within that document is mapped onto the corresponding data records.

```
method $melva(D)$ returns $R$
    $R := \emptyset$
    for each $d$ in $D$ do
        – Task 1: locate relational tables
        $T$ := locate relational tables in $d$
        for each $t$ in $T$ do
            – Task 2: identify cells and roles
            $m$ := identify the cells in $t$
            $c := identifyRoles(m)$
            – Task 3: generate data records
            $X$ := generate data records using $t$, $m$, and $c$
        end
        $R := R \cup \{(d, t, X)\}$
    end
end
```

Fig. 1. The main method.

Task 1: locating relational tables. This task gets an input document *d* and returns a collection of relational tables *T* using the following procedure: It loads and transforms the input document into a DOM tree; next, it renders the DOM tree on a virtual canvas and makes it explicit the attributes of the DOM nodes; finally, it locates the HTML tables by finding the DOM nodes with a table tag.

We implemented the following heuristics to discard non-relational tables: we discard tables whose width or height attributes are 0px or whose display attribute is none because they are not visible; we also discard the tables that have one single row/column because they are typically used to display listings without any data; finally, tables that are nested within other tables are also discarded, because they are typically used to show utilities without any data.

Task 2: identifying cells and roles. This task works on an input table *t* and results in a clustering *c* in which each cell is assigned an explicit role.

The first step consists in identifying the cells in table *t*, which results in a matrix *m* whose components provide the feature vectors of the cells. It iterates over the rows of the input table and looks for nodes with th and td tags. (Our experience proves that we cannot safely assume that th nodes encode label cells and td nodes encode value cells; thus, we simply assume that these nodes encode cells without an explicit role, yet.) The spanned cells are replicated according to their rowspan or colspan attributes and the rows/columns that are shorter than expected are padded using empty cells. If rows/columns consist of empty cells only, they are removed. If rows/columns are duplicated, then their replica after the topmost row or the leftmost column are removed. Finally, the intra-cell and the inter-cell features are computed and stored in matrix *m*.

The second step consists in computing a clustering *c* for the components of matrix *m*, which is expected to make the label cells apart from the value cells. We describe the details in the next subsection.

Task 3: generating data records. This task works on an input table *t*, its feature matrix *m*, and its clustering *c*; it returns a set of records that provide the data in the input table in a structured format.

The first step classifies the input table, which is simple if it has both label and value cells, but a bit more involved if it does not have any label cells. Such cases are easily detected because one of the clusters is empty and we proceed as follows: we compute the average of the inter-cell features per column, per row, and per table; if the per-column average is the smallest one, then we add a row of artificial label cells at the top; if the per-row average is the smallest one, then we add a column of artificial label cells on the left; otherwise, we add both a row and a column of artificial label cells. Classifying the input table is now simple: if the first topmost rows consists of label cells, then it is a horizontal listing; if the leftmost columns consists of label cells, then it is a vertical listing; otherwise it is a matrix.

The second step consists in creating the data records. If the input table is a horizontal/vertical listing, the label cells in the first few rows/columns are catenated to form the headers and then mapped onto the contents of the corresponding value cells. (We disambiguate the headers using sequential indices, if necessary.) In other words, each record has a number of components that map the contents of the label cells in a row/column onto the contents of the corresponding value cells. If the input table is a matrix, then each individual value cell results in a record in which its unique component has two headers that correspond to catenating the label cells in the corresponding row and column.

3.3. Identifying the roles of the cells

Fig. 2 shows the method to identify the roles of the cells, which works on a feature matrix *m* and outputs a clustering *c*. It uses a $(\mu + \lambda)$ genetic strategy that generates an initial population, evolves it iteratively, and computes the roles from the final population (Maulik et al., 2011).

```
method $identifyRoles(m)$ returns $c$
  – Step 1: generate the initial population
  $P := generatePopulation(m)$
  – Step 2: evolve the population
  loop $NGEN$ times unless population $P$ does not improve for $ES$ iterations
    $S := P \cup generateOffspring(P)$
    $P = evaluate(m, S)$
  end
  – Step 3: compute the roles
  $c := computeRoles(P, m)$
end
```

Fig. 2. Method to identify the roles.

```
method $generatePopulation(m)$ returns $P$
  $d := \dim m[1, 1]$
  $P := \emptyset$
  repeat $PSIZE$ times
    $a := (\mathcal{B}_{0.50}, \overset{d}{\ldots}, \mathcal{B}_{0.50})$
    $P := P \cup \{a\}$
  end
end
```

a) Method to generate a population.

```
method $generateOffspring(P)$ returns $S$
  $S := \emptyset$
  repeat $\lceil LAMBDA\ PSIZE \rceil$ times
    if $\mathcal{B}_{CXPB}$ then
      $\{a_1, a_2\} := crossover(P)$
      $S := S \cup \{a_1, a_2\}$
    else if $\mathcal{B}_{MUTPB}$ then
      $a := mutate(P)$
      $S := S \cup \{a\}$
    else
      $\{a\}$ := pick 1 individual from $P$
      $S := S \cup \{a\}$
    end
  end
end
```

b) Method to generate an offspring.

Fig. 3. Genetic operators (part 1/3).

Step 1: generating the initial population. Fig. 3.a presents the method to generate the initial population. It works on a feature matrix m and returns a population P, which is a set of individuals. Each individual is represented as a vector of Booleans that indicate which features must be selected.

The method works as follows: it first gets the dimension of the components in the feature matrix and initialises the population to an empty set. It then iterates several times according to hyper-parameter *PSIZE*, which denotes the number of individuals in the initial population. In each iteration, it creates a single individual a according to a Bernoulli random variable with mean probability 0.50. Simply put: the individuals consider each feature to be informative or uninformative with the same probability, which results in an initial population in which the individuals are as assorted as possible. Finally, population P is returned.

Step 2: evolving the population. The second step iterates a maximum number of times that is controlled by hyper-parameter *NGEN*. In each iteration an offspring S is generated from the current population and evaluated. It may terminate in advance if the population does not improve for *ES* consecutive iterations, which is another hyper-parameter.

Fig. 3.b shows the method to generate an offspring. It first initialises the offspring S to the empty set and then iterates $\lceil LAMBDA\, PSIZE \rceil$ times, where *LAMBDA* is a hyper-parameter that denotes the percentage of offsprings to be generated relative to the size of the initial population (*PSIZE*). In each iteration, it makes a decision on whether the offspring must be generated using crossover, mutation, or cloning. Note that the first two decisions are made according to Bernoulli random variables whose means are controlled by means of hyper-parameters *CXPB* and *MUTPB*, respectively; the third case is the default decision, which simply picks an arbitrary individual from the current population and clones it into the offspring.

Fig. 4.c shows the crossover method. First, it picks any two individuals a_1 and a_2 from population P. Then, it computes the number of features d of individual a_1 (they all have the same dimensionality) and generates a random natural p in interval $[1, d-1]$ that denotes the crossover point. The offsprings are obtained by slicing vectors a_1 and a_2 and catenating the results. Given a vector v and two natural numbers i and j, $v[i : j]$ denotes the following slice of v: $(v[i], v[i+1], \ldots, v[j])$ if $1 \leq i \leq j \leq \dim v$ or $()$ in other cases. Given two vectors $u = (u_1, u_2, \ldots, u_{d_1})$ and $v = (v_1, v_2, \ldots, v_{d_2})$, their catenation is defined as follows: $u \cdot v = (u_1, u_2, \ldots, u_{d_1}, v_1, v_2, \ldots, v_{d_2})$ $(d_1 \geq 0,\ d_2 \geq 0)$.

Fig. 4.d shows the method to perform mutation. It picks an arbitrary individual a from population P. Then, it gets the dimension d of a to compute a random natural p in interval $[1, d]$. Finally, it generates the offspring by catenating the results of slicing vector a from 1 to $p-1$, flipping the pth component of the individual, and then slicing vector a from $p+1$ to d.

```
method $crossover(P)$ returns $\{a'_1, a'_2\}$
    $\{a_1, a_2\}$ := pick 2 individuals from $P$
    $d := \dim a_1$
    $p := \mathcal{U}_{1,d-1}$
    $a'_1 := a_2[1 : p] \cdot a_1[p+1 : d]$
    $a'_2 := a_1[1 : p] \cdot a_2[p+1 : d]$
end
```

c) Method to perform crossover.

```
method $mutate(P)$ returns $a'$
    $\{a\}$ := pick 1 individual from $P$
    $d := \dim a$
    $p := \mathcal{U}_{1,d}$
    $a' := a[1 : p-1] \cdot \neg a[p] \cdot a[p+1 : d]$
end
```

d) Method to perform mutation.

Fig. 4. Genetic operators (part 2/3).

```
method $evaluate(m, S)$ returns $R$
    – Step 1: initialisation
    $n$ := get number of CPU cores available
    $chunks$ := split the individuals in $S$ in $n$ chunks of similar size
    $cache := \emptyset$
    – Step 2: scatter the evaluation
    for each $chunk$ in $chunks$ do concurrently
        for each individual $a$ in $chunk$ do
            if $a$ is not in $cache$ then
                $m'$ := project $m$ according to $a$
                $c$ := cluster $m'$ using $BCLU$
                $s$ := compute the Silhouette score using $c$ and $m'$
                $db$ := compute the Davies-Bouldin score using $c$ and $m'$
                $ch$ := compute the Caliński-Harabasz score using $c$ and $m'$
                $cache(a) := (s, db, ch)$
            end
        end
    end
    – Step 3: gather the results
    $R$ := select $\lceil MU\ PSIZE \rceil$ individuals from $cache$ using Lexicase
end
```

e) Method to perform evaluation.

Fig. 5. Genetic operators (part 3/3).

```
method $computeRoles(P, m)$ returns $c$
    $\{a\}$ := select 1 individual from $P$ using Lexicase
    $m'$ := project $m$ according to $a$
    $c'$ := cluster $m'$ using $BCLU$
    $c$ := correct clustering $c'$ using $m'$
end
```

Fig. 6. Method to compute the roles.

Fig. 5.e shows the evaluation method. It gets a feature matrix m and an offspring S, and returns a set R with the best individuals in S. The first step initialises n to the number of CPU cores, *chunks* to an n-split of offspring S, and *cache* to an empty map. The second step scatters the evaluation of the chunks to the available CPU cores. For each individual a in a chunk, the feature matrix is projected according to its selected features; then, the clusterer indicated by hyper-parameter *BCLU* is invoked to clusterise the projected feature matrix; next, the Silhouette, the Davies–Bouldin, and the Caliński–Harabasz scores are computed; finally, the cache is updated with the evaluation of individual a. The third step uses the Lexicase procedure (Cava et al., 2019) to select $\lceil MU\ PSIZE \rceil$ individuals from the cache.

Step 3: computing the roles. This step clusters the cells to assign them a role. It uses the procedure that is presented in Fig. 6. It works on a population P and a feature matrix m and outputs a clustering c.

The method first selects the best individual a using the Lexicase procedure; then, the feature matrix m is projected onto the features selected by a, which results in a new feature matrix m'; next, the base clusterer indicated by the *BCLU* hyper-parameter is invoked on m', which produces a candidate clustering c'.

Unfortunately, candidate clustering c' is often noisy, which requires to homogenise the roles per row/column using the majority vote in each row/column. This simple approach proved to work very well in practice.

3.4. Complexity analysis

Next, we analyse the complexity of our proposal. We first present some lemmata regarding the ancillary methods and then a theorem that analyses the overall complexity; we also provide a corollary that refines the results taking our experience into account. In the analysis, ν denotes the number of DOM nodes in a document and ρ denotes the number of cells in a table.

Lemma 1. *Task 1 does not take more than $O(\nu)$ time per document.*

Proof. The task to locate relational tables involves parsing, rendering, selecting the table nodes, and discarding non-relational tables; the first three steps can be implemented with industrial components that are not expected to take more than $O(\nu)$ time. Discriminating tables checks a few conditions on the DOM nodes, which does not take more than $O(\nu)$ time. Consequently, Task 1 will not take more than $O(\nu + \nu) \subseteq O(\nu)$ time to locate the relational tables. $\square$

Lemma 2. *Task 2 does not take more than $O(\rho^2)$ time per table.*

Proof. The first operation identifies the cells by iterating over the rows and looking for th and td nodes, which does not take more than $O(\nu)$ time. For each cell, it must compute its κ intra-cell features plus its $3\,\kappa$ inter-cell features, which can be implemented in $O(4\,\kappa\,\rho) \subseteq O(\rho)$ time, and then normalise them, which does not take more than $O(4\,\kappa\,\rho) \subseteq O(\rho)$ time. (Note that κ is a constant in our proposal.) Therefore, the operation to identify the cells does not take more than $O(\nu + \rho + \rho) \subseteq O(\nu + \rho)$ time.

The second operation performs the following steps in sequence:

1. Generating the population entails creating *PSIZE* individuals with $4\,\kappa$ features each, which takes $O(\mathit{PSIZE}\,4\,\kappa)$ time. Note that *PSIZE* is a hyper-parameter that must be set prior to running our proposal, which means that it is independent of the documents; note, too, that $4\,\kappa$ is also independent from the documents since it denotes the number of features. Thus, generating the population takes $O(1)$ time.
2. Evolving the population entails generating the offspring and then evaluating the current population and its offspring. Generating the offspring involves iterating $O(\mathit{LAMBDA}\,\mathit{PSIZE})$ times; in each iteration, it crosses two individuals over, mutates an individual, or clones an individual. Note that *LAMBDA* and *PSIZE* are hyper-parameters that are independent from the documents; note, too, that crossing, mutating, or cloning the individuals does not take more than $O(4\,\kappa)$ time. That is, generating the offspring does not take more than $O(1)$ time. Evaluating the population or the offspring is a bit more involved, namely:

   (a) The initialisation chunks the set of input individuals, which can be implemented in $O(1)$ time because the size of the population depends on hyper-parameters *LAMBDA* and *PSIZE* only.

   (b) The evaluation itself is first scattered across the available cores. It involves projecting the feature matrix according to the individual being evaluated, clustering the cells, and then computing the scores. Projecting the feature matrix does not take more than $O(4\,\kappa\,\rho) \subseteq O(\rho)$ time, because it iterates over a feature matrix with ρ cells and the projection works on feature vectors with $4\,\kappa$ components. Clustering the cells can be performed in $O(4\,\kappa\,\rho) \subseteq O(\rho)$ time according to Maulik and Bandyopadhyay (2002).[1] Computing the Silhouette score does not take more than $O(4\,\kappa\,\rho^2) \subseteq O(\rho^2)$ time since it measures the distances of the cells within a cluster and the distances of the cells in the other clusters; computing the Davies–Bouldin score does not take more than $O(4\,\kappa\,\rho^2) \subseteq O(\rho^2)$ since it computes the average similarity of each cluster with its most similar cluster, where similarity is measured as the ratio of the within-cluster distances to the between-cluster distances; finally, computing the Caliński–Harabasz score does not take more than $O(4\,\kappa\,\rho + k\,4\,\kappa + k\,4\,\kappa\,\rho) \subseteq O(\rho)$ time, since it involves computing the centroids of the clusters and the global centroid, then the inter-cluster variance, and then the intra-cluster variance with $k = 2$. Therefore, computing the three scores does not take more than $O(\rho^2 + \rho^2 + \rho) \subseteq O(\rho^2)$ time. Thus, scattering the evaluation does not take more than $O(\rho + \rho + \rho^2 + \rho^2 + \rho) \subseteq O(\rho^2)$ time.

   (c) Finally, the results must be gathered, which does not take more than $O(1)$ time. Note that the Lexicase procedure takes $O(nf)$ time, where n refers to the size of the set to which it is applied and f refers to its dimensionality. In our case, the Lexicase procedure is applied to the results of evaluating the current population and its offspring; simply put: $n \in O(\mathit{PSIZE}) \subseteq O(1)$ and $f = 3$ because each evaluation has three scores, which implies that the procedure executes in $O(1)$ time.

   That is, evaluating the current population and its offsprings does not take more than $O(1+\rho^2+1) \subseteq O(\rho^2)$ time. Thus, evolving the population does not take more than $O(1 + \rho^2) \subseteq O(\rho^2)$ time.

[1] According to Maulik and Bandyopadhyay (2002), clustering takes $O(k\,d\,n\,i)$ time, where k denotes the number of clusters, d refers to the dimensionality of the data, n the number of data, and i denotes the maximum number of iterations performed. In our context, $k = 2$, $d = 4\,\kappa$, $n = \rho$, and i is a small constant. Thus, clustering does not take more than $O(8\,\kappa\,\rho\,i) \subseteq O(\rho)$ because both κ and i are independent from the documents.

3. Computing the roles of the cells entails projecting the feature matrix according to the best individual, which can be performed in $O(\rho)$ time, clustering the projected matrix, which does not take more than $O(\rho)$ time, and correcting the clustering, which does not take more than $O(\rho)$ time. Thus, computing the roles does not take more than $O(\rho)$ time.

Therefore, the operation to identify the roles of the cells does not take more than $O(1 + \rho^2 + \rho) \subseteq O(\rho^2)$ time. Consequently, Task 2 does not take more than $O(\nu + \rho + \rho^2) \subseteq O(\rho^2)$ time to identify the cells of a table and their roles. □

Lemma 3. *Task 3 does not take more than $O(\rho)$ time per table.*

Proof. This step involves classifying the input table and generating the output data records. The former does not take more than $O(\rho)$ time if none of the clusters of cells is empty, since it only has to find the smallest cluster that is closest to the top rows and/or left columns; if a cluster is empty, then it involves computing the average of the deviation features per row, per column, and per cell, which does not take more than $O(4\,\kappa\,\rho) \subseteq O(\rho)$ time. Therefore, classifying the table does not take more than $O(\rho)$ time. The latter involves iterating over the cells to create the headers, the tuples, and associating them both, which does not take more than $O(\rho)$ time. Consequently, Task 3 does not take more than $O(\rho + \rho) \subseteq O(\rho)$ time to generate the data records. □

Theorem 1. *Melva does not take more than $O(\nu + \rho^2)$ time per document.*

Proof. The proof follows from the previous lemmata: Melva does not take more than $O(\nu)$ time to locate the relational tables in the documents, plus $O(\rho^2)$ time to identify their cells and their corresponding roles, plus $O(\rho)$ time to generate the data records. That is: Melva does not take more than $O(\nu + \rho^2 + \rho) \subseteq O(\nu + \rho^2)$ time to process a document. □

Corollary 1. *In our experimental repositories, we have found that the average number of DOM nodes per document is* 973.97 ± 1430.00 *nodes, whereas the average number of cells per table is* 60.14 ± 133.86 *cells. As a conclusion, the complexity of Melva is expected to be dominated by $O(\rho^2)$.*

4. Experimental analysis

In this section, we describe our experimental setup, how we configured our proposal, and then compare it experimentally.[2]

4.1. Experimental setup

The experiments were run on a Windows 10 computer that was equipped with an AMD Ryzen 7 2700X processor with eight hyper-threaded cores and 16 GiB of DDR4 RAM memory. We implemented our proposal using Python 3.7. We used the DEAP 1.2.2 framework to implement our genetic strategy and SciKit Learn 0.20.3 to leverage some popular clustering algorithms. The statistical tests were performed using the SCMAMP library.

We assembled two repositories with 1496 tables from the Wikipedia and 1513 tables from 781 random sites in the Dresden Web Table Corpus. They provide data on domains like competitions, governments, sports players, politicians, singers, biographies, magazines, movies, radio shows, songs, tours, cars, places, or courses, to mention a few. The tables were annotated by four independent judges whose degree of inter-agreement on a random subset of 300 tables was 96.11%.

We collected the F_1 score and the prediction time. The former was computed as the harmonic mean of precision and recall. The latter was computed as the average number of CPU seconds required to predict the roles of the cells in a table. In the case of supervised proposals, the learning time was apportioned across all of the tables. The repositories were partitioned into three equal-size splits each. Two splits were used for learning purposes in the case of supervised proposals, but discarded in the case of unsupervised proposals; the third one was used to compute the performance measures.

To confirm that our conclusions were sound, we performed a statistical analysis at the standard significance level ($\alpha = 0.05$) (Sheskin, 2011), namely: we first computed the empirical ranking for each performance measure and then performed Hommel's test to compare the best-ranking proposal to the others. The differences in rank are statistically significant if the resulting *p*-value is smaller than the significance level and not significant otherwise.

4.2. Configuring our proposal

We used a stratified grid search procedure to find a good configuration. The results are shown in Table 1. The rows report on the configurations that were tested and their results. The simple columns are the following: Config. ID, which is a unique identifier for each configuration; Explored hyper-parameter, which describes the hyper-parameter whose values are explored in each row; Alternatives, which shows the different values that were tested for the corresponding hyper-parameter; Default hyper-parameters, which shows the values that were used for the other hyper-parameters. The last two columns report on the F_1 score and the prediction

[2] Melva, its competitors, and the experimental repositories are publicly available at GitHub (https://github.com/patriciajimenez/Melva/releases/tag/v1.0) and the Mendeley hub (https://data.mendeley.com/datasets/77dz5skjbj/1). The tool that was used to create the ground truth is also available (http://tomatera.tdg-seville.info).

Table 1
Configuring the hyper-parameters of our proposal.

Config. ID	Explored hyper-parameters	Alternatives	Default hyper-parameters	F_1 score				Prediction time			
				Avg.	Std. Dev.	Rank	P-Value	Avg.	Std. Dev.	Rank	P-Value
C00		0.25	MUTPB = 0.05	0.8366	0.2400	3.0026	**0.4974**	0.6003	0.1957	4.5936	0.0000
C01	CXPB Crossover probability	0.20	NGEN = 30	0.8358	0.2417	3.0075	**0.4974**	0.5369	0.1776	3.6535	0.0000
C02		0.15	PSIZE = 30 MU = 0.90	0.8376	0.2395	2.9951	NA	0.5066	0.1807	3.0342	0.0000
C03		**0.10**	LAMBDA = 0.90	0.8360	0.2415	2.9961	**0.4974**	0.4269	0.1636	1.4577	NA
C04		0.05	BCLU = K-means ES = −1	0.8364	0.2409	2.9988	**0.4974**	0.4688	0.1681	2.2610	0.0000
C05		0.05	CXPB = 0.10	0.8379	0.2392	2.9976	**0.4599**	0.4416	0.1688	3.1798	0.0000
C06	MUTPB Mutation probability	**0.04**	NGEN = 30 PSIZE = 30	0.8372	0.2406	3.0012	**0.4599**	0.4223	0.1606	2.8367	NA
C07		0.03	MU = 0.90	0.8384	0.2398	2.9996	**0.4599**	0.4228	0.1642	2.8911	**0.2064**
C08		0.02	LAMBDA = 0.90 BCLU = K-means	0.8356	0.2417	3.0071	**0.4599**	0.4329	0.1626	3.1085	0.0000
C09		0.01	ES = −1	0.8381	0.2396	2.9945	NA	0.4272	0.1681	2.9839	0.0015
C10		30	CXPB = 0.10	0.8334	0.2435	3.0100	**0.3576**	0.4231	0.1678	4.7522	0.0000
C11	NGEN Maximum number of generations	25	MUTPB = 0.04 PSIZE = 30	0.8370	0.2408	3.0047	**0.3576**	0.3646	0.1592	3.9574	0.0000
C12		20	MU = 0.90	0.8385	0.2393	2.9996	**0.3576**	0.3130	0.1597	3.0798	0.0000
C13		15	LAMBDA = 0.90 BCLU = K-means	0.8381	0.2396	2.9976	**0.3576**	0.2554	0.1545	2.0688	0.0000
C14		**10**	ES = −1	0.8390	0.2387	2.9880	NA	0.2022	0.1556	1.1419	NA
C15		30	CXPB = 0.10	0.8387	0.2396	2.9974	**0.4591**	0.1993	0.1533	4.6504	0.0000
C16	PSIZE Size of the initial population	25	MUTPB = 0.04 NGEN = 10	0.8369	0.2397	3.0055	**0.4591**	0.1738	0.1524	3.9748	0.0000
C17		20	MU = 0.90	0.8382	0.2401	3.0008	**0.4591**	0.1440	0.1516	2.9141	0.0000
C18		15	LAMBDA = 0.90 BCLU = K-means	0.8397	0.2384	2.9931	NA	0.1247	0.1511	2.1232	0.0000
C19		**10**	ES = −1	0.8384	0.2376	3.0031	**0.4591**	0.1247	0.1502	1.3375	NA
C20	MU	0.90	CXPB = 0.10	0.8381	0.2394	2.9990	**0.4581**	0.1034	0.1498	3.0969	0.0000
C21	Ratio of individuals to	0.80	MUTPB = 0.04 NGEN = 10	0.8370	0.2391	3.0083	**0.4581**	0.1022	0.1492	2.9955	0.0044
C22	select	0.70	PSIZE = 10	0.8389	0.2384	2.9937	**0.4591**	0.1036	0.1517	3.0822	0.0000
C23	(relative to PSIZE)	0.60	LAMBDA = 0.90 BCLU = K-means	0.8403	0.2376	2.9900	NA	0.1023	0.1541	2.9601	0.0079
C24		**0.50**	ES = −1	0.8360	0.2409	3.0090	**0.4591**	0.1011	0.1540	2.8654	NA
C25	LAMBDA	**0.90**	CXPB = 0.10	0.8346	0.2414	1.1120	NA	0.1073	0.1595	3.4061	NA
C26	Ratio of offsprings to	0.80	MUTPB = 0.04 NGEN = 10	0.8264	0.2413	2.5582	0.0000	0.1042	0.1529	3.2355	0.0000
C27	generate	0.70	PSIZE = 10	0.8211	0.2404	2.9972	0.0000	0.1034	0.1542	3.1152	0.0000
C28	(relative to PSIZE)	0.60	MU = 0.50 BCLU = K-means	0.8118	0.2402	3.4517	0.0000	0.1007	0.1506	3.3121	0.0000
C29		0.50	ES = −1	0.7721	0.2407	4.8809	0.0000	0.0957	0.1506	1.9312	0.0000
C30		**K-means**	CXPB = 0.10	0.8347	0.2413	3.3044	**0.0697**	0.1072	0.1595	3.1101	NA
C31	**BCLU** Base	Agg. complete	MUTPB = 0.04 NGEN = 10	0.8361	0.2397	3.2478	**0.0929**	0.1291	0.1557	4.0100	0.0000
C32	clusterer	Agg. single	PSIZE = 10	0.8454	0.2349	3.1826	NA	0.1283	0.1553	4.0210	0.0000
C33		Agg. ward	MU = 0.50	0.8329	0.2458	3.2551	**0.0597**	0.1292	0.1555	4.0792	0.0000
C34		Birch	LAMBDA = 0.90 ES = −1	0.8407	0.2393	3.1897	**0.3604**	0.1294	0.1580	4.0725	0.0000
C35		Spectral		0.7459	0.1525	4.8204	0.0000	0.0593	0.0617	1.7072	0.0000
C36	**ES**	−1	CXPB = 0.10	0.8367	0.2397	2.0531	**0.4737**	0.1236	0.1581	3.9912	0.0000
C37	Number of generations	4	MUTPB = 0.04 NGEN = 10	0.8371	0.2399	2.0568	**0.4737**	0.1134	0.1553	3.3630	0.0000
C38	without	**3**	PSIZE = 10	0.8367	0.2408	2.0517	NA	0.1074	0.1604	2.9768	NA
C39	improvements	2	MU = 0.50 LAMBDA = 0.90	0.7887	0.2410	4.4123	0.0000	0.1014	0.1550	2.6010	0.0000
C40		1	BCLU = K-means	0.7878	0.2414	4.4261	0.0000	0.0942	0.1606	2.0680	0.0000

time attained by each configuration. They are complex columns that are composed of the following subcolumns: Avg., which is the average of the measure, Std. Dev., which is the standard deviation with respect to the average, Rank, which is the empirical rank, and P-Value, which is the *p*-value computed by Hommel's test when comparing the best-ranking proposal to the others (the cells with NA correspond to the comparison of the best-ranking proposal to itself).

Table 2
Comparing our proposals to others.

Proposal	F_1 score				Prediction time			
	Avg.	Std. Dev.	Rank	P-Value	Avg.	Std. Dev.	Rank	P-Value
Melva	0.8367	0.2408	2.3070	**0.1049**	0.1074	0.1604	3.0387	NA
Yoshida et al.	0.6154	0.2586	3.4137	0.0000	0.3089	0.0885	3.9603	0.0000
Jung and Kwon	0.5692	0.2630	3.5572	0.0000	0.0018	0.0054	1.5499	0.0000
Embley at al.	0.6552	0.2490	3.4955	0.0000	0.0002	0.0018	1.4511	0.0000
Nishida et al.	0.8467	0.2180	2.2266	NA	12.4708	0.1104	5.0000	0.0000

In each stratum, we selected the best-ranking configuration according to the F_1 score and the succeeding configurations that are statistically indistinguishable; we then selected the best-ranking configuration according to the prediction time. To facilitate the interpretation of Table 1, we have highlighted the cells that correspond to the winner configuration in each stratum using boldface (columns Config. ID and Alternatives). We have also highlighted the cells that correspond to the p-values that are greater than or equal to the significance level, since this helps identify the configurations whose difference in ranking with respect to the best-ranking configuration is not statistically significant.

The best configuration was $C38$, which attained an F_1 score of 0.84 and a prediction time of 0.11 CPU seconds. Realise that the results with the best configuration are not very different from the results with the other configurations, which means that our proposal is not very sensitive to the exact values to which its hyper-parameters are set.

4.3. Comparing our proposal to others

We compared Melva to the unsupervised proposals by Embley et al. (2014), Jung and Kwon (2006), and Yoshida et al. (2001), as well as the state-of-the-art supervised proposal by Nishida et al. (2017). We configured them using the guidelines provided by the authors. Unfortunately, Yoshida et al.'s (2001) guideline was incomplete, so we made some decisions that are in accordance with the common practices in the literature, namely: we initialised the probabilities of their Expectation-Maximisation method with random values, we adjusted them in 10 iterations, we repeated the process 100 times, and we kept the best result only.

Table 2 shows the results. Each row corresponds to the results attained by a different proposal. The first column shows the name of the proposals and the next two columns report, respectively, on their F_1 score and their prediction time. These complex columns should be read/interpreted as in Table 1. The cells that are highlighted correspond to p-values that are greater than or equal to the significance level, i.e., proposals whose differences in rank are not statistically significant. We used the F_1 score as the main performance indicator; in the case of draws, we used the prediction time as the ancillary performance indicator.

Regarding the F_1 score, Nishida et al.'s (2017) proposal ranks the first at position 2.23 and it is closely followed by Melva at position 2.31; Yoshida et al.'s (2001) proposal ranks at position 3.41 and it is followed by Embley et al.'s (2014) proposal at position 3.50 and Jung and Kwon's (2006) proposal at position 3.56. Realise that Hommel's test returns a p-value of 0.10 when comparing Nishida et al.'s (2017) proposal to ours, which is clearly above the significance level. Simply put, the difference in rank between both proposals is not statistically significant, which proves that Melva can attain the state-of-the-art F_1 score in a completely unsupervised manner. Note that the p-values of the remaining comparisons are clearly smaller than the significance level, which means that the differences are statistically significant with regard to the other unsupervised proposals.

Thus, it proceeds to compare Nishida et al.'s (2017) proposal to Melva regarding the prediction time. Our proposal ranks at position 3.04 and Nishida et al.'s (2017) proposal ranks at position 5.00. Note that Hommel's test returns p-value 0.00 regarding the comparison, which clearly indicates that the differences in rank are statistically significant.

5. Conclusions

We have presented Melva, which is an unsupervised proposal to extract data from HTML tables. It provides an effective and efficient solution to find the relational tables in the input documents, identify their cells and roles, and generate data records. Its key innovation is that it addresses the problem of identifying the role of the cells using a clustering approach that has proven to work very well in practice without adapting its hyper-parameters to any particular site or domain. It relies on a $(\mu + \lambda)$ genetic strategy that can select the most informative features and cluster the cells simultaneously.

Our research plans include exploring how to perform semi-supervised role classification if a person can provide some sample label and value cells, as well as exploring multiple clusterings using several subspaces of informative features. These ideas might contribute to improving the quality of the clustering process, which might result in better data extractions without a significant impact on efficiency.

CRediT authorship contribution statement

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

Acknowledgements

The authors of this article were supported by the Spanish R&D programme (PID2020-112540RB-C44, TIN2016-75394-R) and the Andalusian R&D programme, Spain (P18-RT-1060). Juan C. Roldán was partially supported by the Fulbright organisation, Spain and the University of Seville, Spain. We are thankful to Dr. Pedro A. Szekely and Dr. Craig Knoblock for their helpful discussions on the ideas that led to this article.

References

Alam, S., Dobbie, G., Koh, Y. S., Riddle, P., & Rehman, S. U. (2014). Research on particle swarm optimization based clustering. *Swarm and Evolutionary Computation*, *17*, 1–13. http://dx.doi.org/10.1016/j.swevo.2014.02.001.

Bhagavatula, C. S., Noraset, T., & Downey, D. (2015). TabEL: Entity linking in web tables. In *ISWC, Vol. 9366* (pp. 425–441). http://dx.doi.org/10.1007/978-3-319-25007-6_25.

Bong, C., & Rajeswari, M. (2011). Multi-objective nature-inspired clustering and classification techniques for image segmentation. *Applied Soft Computing*, *11*(4), 3271–3282. http://dx.doi.org/10.1016/j.asoc.2011.01.014.

Braunschweig, K., Thiele, M., & Lehner, W. (2015). From web tables to concepts. In *ER* (pp. 247–260). http://dx.doi.org/10.1007/978-3-319-25264-3_18.

Cafarella, M. J., Halevy, A. Y., Lee, H., Madhavan, J., Yu, C., Wang, D. Z., et al. (2018). Ten years of web tables. *VLDB*, *11*(12), 2140–2149. http://dx.doi.org/10.14778/3229863.3240492.

Cafarella, M. J., Halevy, A. Y., Zhang, Y., Wang, D. Z., & Wu, E. (2008). Uncovering the relational web. In *WebDB*.

Cava, W. G. L., Helmuth, T., Spector, L., & Moore, J. H. (2019). A probabilistic and multi-objective analysis of Lexicase selection and ϵ-Lexicase selection. *Evolutionary Computation*, *27*(3), 377–402. http://dx.doi.org/10.1162/evco_a_00224.

Chen, H., Tsai, S., & Tsai, J. (2000). Mining tables from large scale HTML texts. In *COLING* (pp. 166–172). http://dx.doi.org/10.3115/990820.990845.

Crestan, E., & Pantel, P. (2011). Web-scale table census and classification. In *WSDM* (pp. 545–554). http://dx.doi.org/10.1145/1935826.1935904.

Deng, Z., Choi, K., Jiang, Y., Wang, J., & Wang, S. (2016). A survey on soft subspace clustering. *Information Sciences*, *348*, 84–106. http://dx.doi.org/10.1016/j.ins.2016.01.101.

Embley, D. W., Seth, S. C., & Nagy, G. (2014). Transforming web tables to a relational database. In *ICPR* (pp. 2781–2786). http://dx.doi.org/10.1109/ICPR.2014.479.

Ferrara, E., de Meo, P., Fiumara, G., & Baumgartner, R. (2014). Web data extraction, applications and techniques. *Knowledge-Based Systems*, *70*, 301–323. http://dx.doi.org/10.1016/j.knosys.2014.07.007.

Figueiredo, L. N. L., Assis, G. T., & Ferreira, A. A. (2017). DERIN: a data extraction method based on rendering information and *n*-grams. *Information Processing and Management*, *53*(5), 1120–1138. http://dx.doi.org/10.1016/j.ipm.2017.04.007.

Figueiredo, E., Macedo, M., Siqueira, H. V., Santana, C. J., Gokhaled, A., & Bastos-Filhoa, C. J. (2019). Swarm intelligence for clustering. *Engineering Applications of Artificial Intelligence*, *82*, 313–329. http://dx.doi.org/10.1016/j.engappai.2019.04.007.

García, A. J., & Gómez-Flores, W. (2016). Automatic clustering using nature-inspired metaheuristics. *Applied Soft Computing*, *41*, 192–213. http://dx.doi.org/10.1016/j.asoc.2015.12.001.

Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., & Pollak, B. (2007). Towards domain-independent information extraction from web tables. In *WWW* (pp. 71–80). http://dx.doi.org/10.1145/1242572.1242583.

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

Jiménez, P., & Corchuelo, R. (2016a). On learning web information extraction rules with TANGO. *Information Systems*, *62*, 74–103. http://dx.doi.org/10.1016/j.is.2016.05.003.

Jiménez, P., & Corchuelo, R. (2016b). Roller: a novel approach to web information extraction. *Knowledge and Information Systems*, *49*(1), 197–241. http://dx.doi.org/10.1007/s10115-016-0921-4.

Jiménez, P., Roldán, J. C., Gallego, F. O., & Corchuelo, R. (2020). On the synthesis of metadata tags for HTML files. *Software: Practice and Experience*, *50*(12), 2169–2192. http://dx.doi.org/10.1002/spe.2886.

Jung, S., & Kwon, H. (2006). A scalable hybrid approach for extracting head components from web tables. *IEEE Transactions on Knowledge and Data Engineering*, *18*(2), 174–187. http://dx.doi.org/10.1109/TKDE.2006.19.

Kim, Y.-S., & Lee, K.-H. (2005). Detecting tables in web documents. *Engineering Applications of Artificial Intelligence*, *18*(6), 745–757. http://dx.doi.org/10.1016/j.engappai.2005.01.009.

Luna-Romera, J. M., García-Gutiérrez, J., Martínez-Ballesteros, M., & Riquelme-Santos, J. C. (2018). An approach to validity indices for clustering techniques in Big Data. *Progress in AI*, *7*(2), 81–94. http://dx.doi.org/10.1007/s13748-017-0135-3.

Martínez-Rodríguez, J., Hogan, A., & López-Arévalo, I. (2020). Information extraction meets the Semantic Web: A survey. *Semantic Web*, *11*(2), 255–335. http://dx.doi.org/10.3233/SW-180333.

Maulik, U., & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(12), 1650–1654. http://dx.doi.org/10.1109/TPAMI.2002.1114856.

Maulik, U., Bandyopadhyay, S., & Mukhopadhyay, A. (2011). *Multi-objective genetic algorithms for clustering*. Springer, ISBN: 978-3-642-16614-3.

Milošević, N., Gregson, C., Hernández, R., & Nenadic, G. (2016). Disentangling the structure of tables in scientific literature. In *NLDB* (pp. 162–174). http://dx.doi.org/10.1007/978-3-319-41754-7_14.

Nishida, K., Sadamitsu, K., Higashinaka, R., & Matsuo, Y. (2017). Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *AAAI* (pp. 168–174).

Oulabi, Y., & Bizer, C. (2019). Extending cross-domain knowledge bases with long tail entities using web table data. In *EDBT* (pp. 385–396). http://dx.doi.org/10.5441/002/edbt.2019.34.

Pimplikar, R., & Sarawagi, S. (2012). Answering table queries on the web using column keywords. *VLDB*, *5*, 908–919. http://dx.doi.org/10.14778/2336664.2336665.

Ritze, D., & Bizer, C. (2017). Matching web tables to DBpedia: A feature utility study. In *EDBT* (pp. 210–221). http://dx.doi.org/10.5441/002/edbt.2017.20.

Roldán, J. C., Jiménez, P., & Corchuelo, R. (2020). On extracting data from tables that are encoded using HTML. *Knowledge-Based Systems*, *190*, 105–157. http://dx.doi.org/10.1016/j.knosys.2019.105157.

Sheskin, D. J. (2011). *Handbook of parametric and nonparametric statistical procedures* (5th ed.). Chapman & Hall/CRC, ISBN: 978-1-439-85801-1.

Sim, K., Gopalkrishnan, V., Zimek, A., & Cong, G. (2013). A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*, *26*(2), 332–397. http://dx.doi.org/10.1007/s10618-012-0258-x.

Sleiman, H. A., & Corchuelo, R. (2013a). A survey on region extractors from web documents. *IEEE Transactions on Knowledge and Data Engineering*, *25*(9), 1960–1981. http://dx.doi.org/10.1109/TKDE.2012.135.

Sleiman, H. A., & Corchuelo, R. (2013b). TEX: an efficient and effective unsupervised web information extractor. *Knowledge-Based Systems*, *39*, 109–123. http://dx.doi.org/10.1016/j.knosys.2012.10.009.

Sleiman, H. A., & Corchuelo, R. (2014). A class of neural-network-based transducers for web information extraction. *Neurocomputing*, *135*, 61–68. http://dx.doi.org/10.1016/j.neucom.2013.05.057.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2013). A hybrid approach for extracting informative content from web pages. *Information Processing and Management*, *49*(4), 928–944. http://dx.doi.org/10.1016/j.ipm.2013.02.005.

Wu, X., Cao, C., Wang, Y., Fu, J., & Wang, S. (2016). Extracting knowledge from web tables based on DOM tree similarity. In *KSEM* (pp. 302–313). http://dx.doi.org/10.1007/978-3-319-47650-6_24.

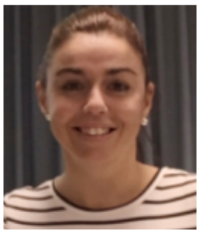
Xu, D., & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, *2*(2), 165–193. http://dx.doi.org/10.1007/s40745-015-0040-1.

Yang, Y., & Luk, W. (2002). A framework for web table mining. In *WIDM* (pp. 36–42). http://dx.doi.org/10.1145/584931.584940.

Yoshida, M., Torisawa, K., & Tsujii, J. (2001). A method to integrate tables of the world wide web. In *WDA* (pp. 31–34).

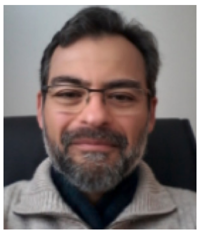
Zhang, Z. (2017). Effective and efficient semantic table interpretation using TableMiner. *Semantic Web*, *8*(6), 921–957. http://dx.doi.org/10.3233/SW-160242.

Zhang, S., & Balog, K. (2020). Web table extraction, retrieval, and augmentation. *ACM Transaction on Intelligent Systems and Technology*, *11*, 13:1–13:35. http://dx.doi.org/10.1145/3372117.



Patricia Jiménez is working as a lecturer for the University of Seville. Her research focuses on large-scale web data extraction, with an emphasis on methods to evaluate their performance regarding both efficiency and effectiveness.

Juan C. Roldán is working as a researcher for the University of Seville. His research focuses on large-scale web data extraction, with an emphasis on extracting data from the web tables as a mean to feed Data Science applications.



Rafael Corchuelo is working as a reader for the University of Seville. His research focuses on application and information integration, with an emphasis on web information extraction and social media analytics.