A coral-reef approach to extract information 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
Information extraction
Coral-reef optimisation
Feature selection
Clustering

A B S T R A C T

This article presents Coraline, which is a new table-understanding proposal. Its novelty lies in a coral-reef optimisation algorithm that addresses the problem of feature selection in synchrony with a clustering technique and some custom heuristics that help extract information in a totally unsupervised manner. Our experimental analysis was performed on a large collection of tables with a variety of layouts, encoding problems, and formatting alternatives. Coraline could achieve an F_1 score as high as 0.90 and took 7.07 CPU seconds per table, which improves on the best supervised proposal by 6.67% regarding effectiveness and 40.54% regarding efficiency; it also improves on the best unsupervised proposal by 11.11% regarding effectiveness while it remains very competitive regarding efficiency.

1. Introduction

Nowadays, there are billions of web sites in the World Wide Web, which makes it the most popular content-distribution platform and a major source of information. The information is presented using visualisation elements that include text, images, tables, or multi-media streams that aim at facilitating people understanding it without any further assistance. However, to really empower people with knowledge and enable them to make informed decisions, it is necessary to have automatic systems that allow to find, extract, and process that information effectively and efficiently.

HTML tables are a popular means to display information due to their compactness and conciseness [1–3]. It is not surprising then that they are considered one of the most significant information sources in the current Web [4]. The information in typical HTML tables can be leveraged in many application domains [2,4–8], e.g., knowledge management, information retrieval, web mining, summarisation, knowledge base construction, or question answering. Researchers commonly refer to the tables that are used to display information as relational tables, in contrast to non-relational tables that are used to position other elements on the screen.

Relational HTML tables are typically generated on user demand by filling-in a template with information that comes from a back-end database. The information records are typically placed in a grid in which the data-value cells are arranged horizontally and/or vertically and there are some headers that provide meta-data that helps people understand the semantics [9]. Unfortunately, the groups of cells that form a record or the relationships between both types of cells are not made explicit in HTML, which makes it difficult for a machine to understand their relational nature. There have been several initiatives to enrich HTML with semantics, e.g., RDFa, JSON-LD, Microdata, or Microformats. Unfortunately, a recent analysis of the 32.04 million domains in the November 2019 Common Crawl has revealed that only 11.92 million domains provide such semantic hints [10], which argues for a method to deal with the remaining 20.12 million domains.

Summing up, it is necessary to devise information extractors that can reverse-engineer relational tables to infer their structure and pull their information out in machine-understandable formats. The literature reports on many general proposals [11, 12], which do not seem appropriate for HTML tables [13] because they were not devised to make relational tables apart from non-relational tables, to make meta-data cells apart from data-value cells, or to infer their underlying relationships. This motivated many researchers to work on specific-purpose table-understanding proposals [3,7]. They revolve around a pipeline that encompasses locating the relational tables in the input HTML documents, identifying their cells, classifying their roles, and then extracting information records. The first two tasks and the last task are relatively easy to implement using the current technology; the third task is far more involved due to the many existing layouts, encodings problems, and formatting alternatives [3,7].

This article introduces Coraline, which provides a new means to implement the table-understanding pipeline. Its main contribution is its approach to identify the role of the cells, which simulates a reef in which corals compete to preserve their genome through as many generations as possible [14]. This meta-heuristic is used to find an informative subset of features in synchrony with a clustering technique and some custom heuristics that allow to group the cells into two clusters: one that contains the meta-data

cells and another that contains the data-value cells. Our analysis of the literature reveals that this approach is novel regarding information extraction and our experimentation confirms that it is very effective and efficient. Coraline improves on the best supervised proposal by 6.67% regarding effectiveness and 40.54% regarding efficiency; it improves on the best unsupervised proposal by 11.11% regarding effectiveness, which comes at the cost of some inefficiency that does not preclude it from being used in practice because it takes an average of 7.07 CPU seconds per table.

The rest of the article is organised as follows: Section 2 reviews the related work, Section 3 describes Coraline in depth, Section 4 reports on the results of our experimental analysis, and Section 5 concludes the article.

2. Related work

In this section, we first analyse the literature on information extraction and then provide an overall picture of coral reefs and how they inspired the meta-heuristic used in Coraline.

2.1. Information extraction

The literature provides many general-purpose proposals for information extraction that are based on analysing visual features [15], matching trees [16], text alignment [17,18], neural networks [19], learning first-order rules [20], and also inferring propositio-relational rules [21], just to mention a few. Unfortunately, they are not intended to identify relational tables, neither aim they to discover the relationships between their cells. This makes them inappropriate in our context [22] and motivated many researchers to work on specific-purpose table-understanding proposals [3,7].

Table-understanding proposals generally implement a pipeline with the following tasks: locating relational tables, identifying their cells, classifying the roles of the cells, and then extracting information records. We have analysed the most recent surveys on extracting information from HTML tables [3,7] and our conclusion is that only the third task poses challenges, yet, due to the many existing layouts, encoding problems, and formatting alternatives. We have identified three supervised approaches [5,23,24] and nine heuristic-based approaches [1,24–31] to implement this task, which we summarise below.

Yoshida et al. [23] used the Expectation Maximisation algorithm, which was fed with many tables to adjust a probability model according to a number of pre-defined table types that ease identifying the roles of the cells. Cafarella et al. [24] learnt a classifier from a training set that provides many cells with structural and content-based features plus additional user-provided annotations. Nishida et al. [5] devised the current state-of-the-art proposal, which relies on a recurrent neural network to encode the tokens in each cell as a 3-D volume to which a convolutional neural network is then applied to infer features that help classify the tables and identify the roles of their cells.

Chen et al. [25] devised a heuristic-based proposal that measures cell similarity on a per-row/column basis according to some pre-defined features; the first rows/columns that are more dissimilar to each other are assumed to have the meta-data cells. Yang and Luk [26] devised a proposal for numeric tables that assumes that data-value cells contain a number or a range of numbers with an optional measurement unit or strings like "N/A" or "nil". Kim and Lee [27] assumed that the top-most rows and the left-most columns contain meta-data cells as long as they meet some custom heuristics regarding spanning and pattern-based coherency. Jung and Kwon's [28] proposal seeks for regions with spanned cells, common background colours or fonts, empty cells, and cells whose contents match some patterns; they then use some heuristics to divide the input tables into two regions: the meta-data cells are then assumed to be on the top-most, left-most region and the data-value cells are assumed to be in the other region. Gatterbauer et al. [32] defined a number of common table types and used some heuristics that are based on style features; unfortunately, the authors did not disclose the exact heuristics, but revealed that their proposal needs to be improved in order to work more effectively in a domain-independent manner. Embley et al.'s [30] proposal looks for four so-called critical cells, namely: the first and fourth critical cells are set to the upper-left and to the bottom-right corners, 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; the region delimited by the first two critical cells is assumed to have the meta-data cells. Milošević et al. [1] restricted their attention to the tables in 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 meta-data cells; the cells in a full-spanned row and in subsequent rows that also have spanned cells, the cells that are encoded in a `thead` tag, or the cells that are in a row that is enclosed in horizontal lines are also considered meta-data cells.

Unfortunately, the previous proposals have some inherent drawbacks, namely: (a) Regarding the layouts, the proposals by Cafarella et al.'s [24], Braunschweig et al.'s [33] and Milošević et al.'s [1] cannot deal with vertical listings; the proposals by Cafarella et al. [24], Braunschweig et al. [33], Yoshida et al. [23] and Gatterbauer et al. [32] cannot deal with matrices; Wu et al.'s [2] proposal can work on any layouts, but the authors admitted that it may have difficulties to make vertical listings apart from matrices. (b) Regarding encoding problems, the proposals by Jung and Kwon [28], Milošević et al. [1], and Wu et al. [2] assume that meta-data and data-value cells are encoded using `th` and `td` tags, respectively, which is not generally true in the Web (in our experimental analysis, we found that 34.96% of the meta-data cells were encoded using `td` tags and 11.74% of the data-value cells were encoded using `th` tags); the proposals by Yoshida et al. [23], Cafarella et al. [24], and Braunschweig et al. [33] have problems when the row/column lengths are heterogeneous due to cells that are incorrectly spanned. (c) Regarding the formatting alternatives, Braunschweig et al. [33] do not deal with tables with multiple row/columns of meta-data cells; neither work well the proposals by Chen et al. [25], Yoshida et al. [23], Yang and Luk [26], Kim and Lee [27], Jung and Kwon [28], Embley et al. [30], Braunschweig et al. [33], and Milošević et al. [1] when there are not any meta-data cells. (d) Furthermore, Cafarella et al.'s [24] and Nishida et al.'s [5] proposals are supervised; unfortunately, assembling the training set is a time-consuming and an error-prone task that does not scale well to the Web. (e) Finally, Yang and Luk's [26] and Milošević et al.'s [1] proposals are domain specific and then not general-enough to deal with the variety of tables in the current Web.

Our proposal is superior in that it can work with all of the common layouts, pays special attention to common encoding problems and formatting alternatives, and it is completely unsupervised and domain agnostic.

2.2. Coral-reef optimisation

This section describes corals, reefs, and their biological life cycle. It provides a foundation that helps understand how we have mapped the HTML information extraction problem onto a coral-reef optimisation problem [34] that helps tell meta-data and data-value cells apart.

A coral consists of a calcium carbonate skeleton where a polyp lives. When the polyp dies, its skeleton remains and may be used by larvae to settle and grow. It is the accumulation of skeletons and polyps that forms a reef. Typical reefs have, literally, millions of corals that compete to preserve their genome across as many generations as possible.

Corals can reproduce asexually and/or sexually and their genome may undergo mutations. Asexual reproduction, aka. budding, happens when a coral clones itself as a larva. There are two types of sexual reproduction, namely: spawning and brooding. In both cases, a male coral releases a sperm that eventually fertilises an ovum released by a female coral. In the former case, the fertilisation occurs in the water, whereas in the latter case, it occurs inside the body of the female coral. The fertilisation process crosses the genome of the progenitors at two different points and also results in a larva. Larvae float in the water until they find a position in their reef where they can settle and become corals. Both larvae and corals perish continuously due to the following factors: on the one hand, the positions in the reef are limited, which means that many larvae weaken too much after several settling attempts and perish; on the other hand, predation and pollution threaten them all the time.

The previous biological eco-systems inspired an evolutionary approach that is known as Coral-Reef Optimisation [14]. Generally speaking, the idea is to represent the domain of a function using corals and then simulate their evolution to find the optimum. The corals can easily encode the search space using their genomes and the evolution process can quickly move through that space using the target function as a health score: the better the value of the function on a given coral, the more healthy it is and, consequently, the more likely to pass its genome on to the next generation.

The idea of using meta-heuristics to perform clustering has recently got some attention [35], particularly the idea of using coral-reef approaches [36]. On the one hand, meta-heuristics can naturally explore large and complex search spaces, which makes them appropriate for NP-hard problems like clustering [37]; on the other hand, coral-reef approaches have been applied to a variety of problems in different fields [14,34,36] and they have led to competitive and promising results. They can naturally deal with the myopia that takes other techniques to local optima thanks to the way that corals fight for space, along with the specific characteristics of coral reproduction [14]. However, to the best of our knowledge, no author has explored using a coral-reef approach to extracting information from HTML tables, which motivated us to explore this research niche and to exploit it in a new real-world scenario. Specifically, we address the problem of classifying the cells as meta-data or data-value cells by using a coral-reef approach in which the most informative cell features are selected and used to cluster the cells, which are then categorised using some simple heuristics. Our formulation got inspiration from Salcedo-Sanz et al.'s [14] original proposal; note that Tsai et al. [36] improved on the original formulation by exploring several search strategies using a map-reduce approach that is very appropriate to deal with big data, but that is not the case of typical HTML tables.

3. Our proposal

In this section, we describe our proposal to extract information from HTML tables. First, we present some preliminaries; next, we introduce our pipeline; finally, we focus on our coral-reef approach to classify the cells.

3.1. Preliminaries

Definition 1 (*Core Mathematical Concepts*)**.** A bag is a set in which there can be multiple instances of the same object. We use operators size, $\uplus$, and $\setminus$ to denote the size, union, and subtraction of bags, respectively. An n-dimensional vector V is an array of the form $(v_1, v_2, \ldots, v_n)$ $(n \geq 1)$. Given a vector V, its dimensionality is denoted as $\dim V$. Given a vector V and a natural number i, its ith component is denoted as $V[i]$ $(1 \leq i \leq \dim V)$. A matrix M is a vector of vectors $((v_{1,1}, v_{1,2}, \ldots, v_{1,m}), \ldots, (v_{n,1}, v_{n,2}, \ldots, v_{n,m}))$ $(n \geq 1, m \geq 1)$. Given a matrix M and two natural numbers i and j, its component at position (i, j) is denoted as $M[i, j]$. A random variable is a function that returns values whose probability density is characterised by means of a theoretical distribution. We restrict our attention to random variable $\mathcal{B}_\lambda$, which is distributed according to a Bernoulli with mean λ $(0.00 \leq \lambda \leq 1.00)$, and random variable $\mathcal{U}_{\lambda_1,\lambda_2}$, which is uniformly distributed in integer interval $[\lambda_1, \lambda_2]$ $(\lambda_1 \leq \lambda_2)$. □

Definition 2 (*Documents and Tables*)**.** A document is a file whose contents are encoded using the HTML mark-up language, which allows to represent it using a DOM tree. A table is a grid that 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. Every piece of HTML whose root is a node with a `table` tag is considered a table. Rows are encoded as DOM sub-trees with `tr` tags and cells are encoded as DOM sub-trees with `th` or `td` tags. Cells can be either meta-data cells, which provide semantic hints, and data-value cells, which provide data values. Depending on how the information is laid out in a table, it is common to classify them as horizontal listings, vertical listings, or matrices [38]. □

Definition 3 (*Information Records*)**.** The information in a table can be modelled as a record of the form $\{h_i : v_i\}_{i=1}^{n}$ $(n \geq 1)$, where h_i is a header and v_i is a data-value cell $(1 \leq i \leq n)$. A header is a group of meta-data cells that help interpret the components of an information record. We use term tuple to refer to the data-value cells in an information record. □

Definition 4 (*Features and Clusterings*)**.** A feature is a property of a cell. The features of a single cell are represented by means of a feature vector; the features of all of the cells in a table are represented by means of a feature matrix. A feature can be visual (which is computed from the rendering attributes), structural (which is computed from the DOM tree structural attributes), lexical or syntax (which are computed from the content attributes). Note that the attributes can be categorical, discrete, or continuous, which means that they must be transformed into real numbers before computing the corresponding features. A clustering is a binary matrix that classifies every cell in a table as either meta-data or data-value. □

Definition 5 (*Corals and Reefs*)**.** A coral is represented by means of a genome that is encoded as a vector of the form $(c_1, c_2, \ldots, c_n)$ $(n \geq 1)$ in which each component is a Boolean that indicates whether a feature is considered or not for clustering purposes. (As usual, we represent Booleans using zeros and ones.) A reef is represented as a vector in which each component is either a coral or a *null* value; the components of a reef are commonly referred to as positions. We use term larva to refer to a newborn; it becomes a coral when it settles in a reef. Given a feature matrix M and a coral C, we use $\text{health}(M, C)$ to denote a triplet (s, d, c) in which s, d, and c represent, respectively, the Silhouette, the Davies–Bouldin, and the Caliński–Harabasz scores of the clustering that results from projecting M onto the subspace of features encoded by coral C [39]. We use size R and occupation R to denote the number of positions and corals in reef R. □

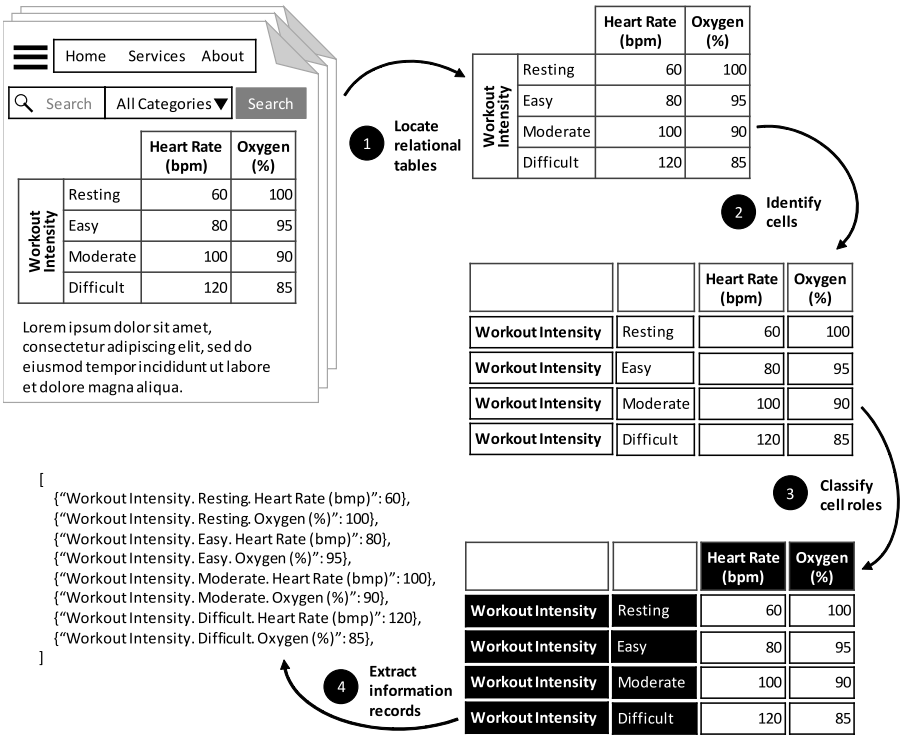

Fig. 1. Information extraction pipeline.

Definition 6 (*Hyper-parameters*)**.** Coraline requires the user to set the following hyper-parameters: *RSIZE* (the number of positions in the reef), *NGEN* (the number of generations to simulate), *MNSA* (the maximum number of settling attempts), *MUTP* (the probability that a coral mutates), *INITR* (the initial occupancy ratio), *SPAWR* (the spawning ratio), *BUDDR* (the budding ratio), and *PERIR* (the perishment ratio). Note that there is no need to introduce a hyper-parameter to model the brooding ratio since it would be trivially defined as $1.00 - SPAWR$. □

3.2. The pipeline

The pipeline is fed with a stream of HTML documents and produces a stream of information records. It performs the following tasks in sequence, cf. Fig. 1: locating relational tables, identifying their cells, classifying the roles of the cells, and then extracting information records.

Locating relational tables. This task works on the input documents and extracts their tables as follows: (a) It transforms the input document into a DOM tree using an HTML parser. (b) It computes the attributes of the DOM nodes by means of a custom script that is executed by a headless browser. (c) It locates the DOM nodes with a `table` tag using a CSS selector. (d) It discards the tables whose `width` or `height` attributes are set to `0px`, the tables whose `display` attribute is set to `none`, the tables with one row or column, and inner nested tables. (e) If a table or any of its ancestors in the DOM tree has attribute `dir` set to "rtl", then the table is flipped horizontally so that the first column is always the left-most column.

Identifying cells. This task works on the tables extracted by the previous task and makes their cells explicit: (a) It iterates over the rows of the input table and finds the cells by looking for nodes with `th` and `td` tags. (b) The spanned cells are replicated according to their `rowspan` or `colspan` attributes. (c) Rows/columns that are larger than 200 cells are cut to prevent unnecessary overhead with tables that are incorrectly encoded. (d) Rows/columns that are shorter than expected are padded using empty cells. (e) Rows/columns that consist exclusively of empty cells are removed (a cell is empty if it consists of blanks, dashes, or question marks). (f) Duplicated rows/columns are removed, except for the top-most and/or the left-most ones.

Classifying cell roles. This task works on the tables returned by the previous task and returns clusterings that tell their meta-data cells apart from their data-value cells as follows: (a) A feature matrix M is computed from the cells in the input table plus their deviations with respect to the cells in the same row, in the same column, and the remaining cells. (b) A clustering K is then computed from the feature matrix M using the approach that is described in the following subsection. (c) Clustering K is corrected according to the majority vote in each row or column. (d) The cells in the cluster that lies at the top-most rows and/or the left-most columns are classified as meta-data cells and the others are classified as data-value cells.

Extracting information records. This task works on pairs of feature matrices and clusterings (M, K) computed by the previous task and returns information records as follows: (a) It guesses the layout. This is straightforward if none of the clusters in K is empty: it is a horizontal listing if the majority of meta-data cells are at the top-most rows, it is a vertical listing if most of them are on the left-most columns, and, otherwise, it is a matrix. If one of the clusters is empty, the classification requires to compute the averages of the deviation features per row, per column, and per cell: it is a horizontal listing if the per-column average is the smallest one, it is a vertical listing if the per-row average is the smallest one, and, otherwise, it is a matrix. (b) It then creates the output information records according to the type of layout, namely: if it is a horizontal listing, then the headers are computed from the meta-data cells at the top of the table in a column-wise manner and the tuples are computed from the data-value cells in a row-wise manner; if it is a vertical listing, then the headers are computed from the meta-data cells on the left of the table in

```
method cluster(M) returns K
    ▷ Step 1: initialisation.
    R := random reef with ⌊INITR · RSIZE⌋ dim M[1, 1]-dimensional corals
    ▷ Step 2: evolution.
    loop NGEN times
        ▷▷ Step 2.1: budding.
        L1 := pick ⌊BUDDR · occupation R⌋ random corals from reef R
        ▷▷ Step 2.2: spawning.
        S := pick ⌊SPAWR · occupation R⌋ random corals from reef R
        L2 := fertilise(S, S)
        ▷▷ Step 2.3: brooding.
        B := pick the corals in reef R that are not in S
        L3 := fertilise(S, B)
        ▷▷ Step 2.4: settling.
        L := L1 ⊎ L2 ⊎ L3
        R := settle larvae in L using (M, R) in MNSA attempts at most
        ▷▷ Step 2.5: perishing.
        R := perish ⌊PERIR · occupation R⌋) random corals in R
    end
    ▷ Step 3: computing result.
    C := select the best coral in R
    M′ := project feature matrix M onto the sub-space encoded by C
    K := invoke k-means on feature matrix M′ with k = 2
end
```

Algorithm 1: Method to cluster cells.

a row-wise manner and the tuples are computed from the data-value cells in a column-wise manner; if it is a matrix, then each data-value cell is a tuple and its corresponding header is created from the meta-data cells on the corresponding left-most columns and top-most rows.

3.3. Clustering the cells

The key to identify the roles of the cells is to cluster them appropriately. Typically, the cluster at the top-most rows and/or the left-most columns contains the meta-data cells and the remaining cells are data-value cells. Algorithm 1 presents our method to perform such a clustering using a coral-reef optimisation approach. It works on a feature matrix M that represents a relational table and returns a clustering K.

The first step initialises the reef. This basically amounts to creating a vector with *RSIZE* positions in which $\lfloor INITR \cdot RSIZE \rfloor$ random positions are initialised with random corals. (Recall that *INITR* is a hyper-parameter that determines the ratio of positions that are occupied with corals in the initial population; recall, too, that *RSIZE* is a hyper-parameter that determines the number of positions in the reef). The initial corals are implemented as random Boolean vectors whose dimensionality is determined by the dimensionality of the vectors in the feature matrix. (Whether a component of the initial corals is set to true or false is determined randomly using a Bernoulli random variable with mean $\lambda = 0.50$.) Note that every cell in the input table is projected onto the same feature space, so any of the cells in the input feature matrix provides the exact dimensionality of the corals.

The second step evolves the reef according to hyper-parameter *NGEN*, which involves performing budding, performing spawning, performing brooding, settling the resulting larvae, and perishing some random corals.

Budding is the simplest kind of reproduction since any of the corals can clone itself into a larva. Thus the first bag of larvae L_1 is computed by picking $\lfloor BUDDR \cdot \text{occupation } R \rfloor$ random corals from reef R. (Recall that hyper-parameter *BUDDR* controls the ratio of corals that reproduce by budding.) Spawning and brooding are more involved since they require to select individuals to fertilise each other. The implementation relies on the *fertilise* method that is presented in Algorithm 2. This method gets two bags of corals A and B as input and outputs a bag of larvae L. It first initialises L to an empty bag and then loops as long as there are at least two different corals in bags A and B. In each iteration, it selects two different random corals, mutates them, and crosses them over to produce a newborn. Note that mutation consists of flipping any of the components of a coral and whether it happens or not depends on a random Bernoulli variable whose mean is controlled by means of hyper-parameter *MUTP*; crossing two corals over requires to compute two random crossing points at which their genomes are exchanged. The final operations in the loop update the resulting bag of larvae L and subtract the selected corals from the input sets. Thanks to the *fertilise* method, implementing sexual reproduction is straightforward: spawning amounts to initialising bag S with $\lfloor SPAWR \cdot \text{occupation } R \rfloor$ random corals and then invoking $fertilise(S, S)$ so that the selected corals fertilise each other and produce a new bag of larvae L_2; brooding amounts to initialising bag B with the corals in the reef that have not been selected for spawning and then using the latter to fertilise the former and produce a new bag of larvae L_3. (Recall that hyper-parameter *SPAWR* controls the ratio of corals that perform spawning; implicitly, the corals that do not perform spawning perform brooding.)

```
method fertilise(A, B) returns L
    L := ∅
    while A ≠ B ∧ A ≠ ∅ ∧ B ≠ ∅ do
        C1 := pick one random coral from A
        C1′ := if B_MUTP then mutate C1 else C1
        C1 := pick one random coral from A \ {C1}
        C2′ := if B_MUTP then mutate C2 else C2
        d := dim C1; p1 := U_{1,d−2}; p2 := U_{p1,d−1}
        C := crossover C1′ and C2′ at points p1 and p2
        L := L ⊎ {C}
        A := A \ {C1}
        B := B \ {C2}
    end
end
```

Algorithm 2: Method to simulate fertilisation.

The reproduction steps result in a bag L with new larvae, which must now settle. For each larva C, a set I with *MNSA* random positions in the reef are sampled (recall that *MNSA* is a hyper-parameter that controls the maximum number of settling attempts); larva C then settles at the first position in I that is either empty or contains a weaker larva. Simply put, larva C settles at position $\min\{i \in I \mid R[i] = null \vee \text{health}(M, C) > \text{health}(M, R[i])\}$; if not such a position exists, then the larva perishes. Note that computing the health of a larva or a coral requires projecting the input feature matrix M onto the subspace of features encoded by that larva or coral and then computing the Silhouette, the Davies–Bouldin, and the Caliński–Harabasz scores. We resorted to computing the clusters using the standard k-means procedure with $k = 2$ and we compared the health triplets using the Lexicase procedure [40].

The final sub-step of the evolution perishes $\lfloor PERIR \cdot \text{occupation } R \rfloor$ random corals in R, which simulates predation and pollution. (Recall that hyper-parameter *PERIR* controls the ratio of corals that perish.)

When the main loop finishes, the best coral in the reef is selected, that is, any of the corals with the maximum health, then projects the input feature matrix M onto the feature space encoded by that coral, and finally invokes the k-means algorithm onto the projection with $k = 2$.

4. Experimental analysis

In this section, we report on our experimental analysis.[1] First, we describe our setup, next, present the results, and, finally, analyse them and confirm our conclusions using a statistically sound method.

4.1. Setup

Our experiments were run on a Windows 10 computer with an AMD Ryzen 7 2700X processor and 16 GiB of DDR4 RAM memory. We used Python 3.7 as the programming language, BeautifulSoup 4.9.1 to parse the HTML documents and to work with the DOM trees, the Selenium 3.141.0 headless browser with Firefox 80.0.1 and Geckodriver 0.27.0 to compute the feature vectors, SciKit Learn 0.20.3 to perform basic clustering and to compute effectiveness measures, Pandas 0.24.2 to work with the datasets, and NumPy 1.16.3 to implement some vector and matrix operations. The statistical tests were performed using the SCMAMP 0.2.55 library.

We collected a repository with 2 544 HTML tables from the Wikipedia [41] and the Dresden Web Table Corpus (DWTC) [42]. We omitted the Wikipedia tables that are written using templates like `infobox`, `navbox`, or `sistersitebox` since extracting information from them can be done using simple rule-based extractors that are currently deployed at major knowledge bases. In order to make the repository as diverse as possible, we also sampled some random tables and replaced the ones with very common looks by new random tables that were more dissimilar. The tables were split into four batches that were annotated by four independent persons (the annotations consisted in labelling the tables with their layout, i.e., "horizontal listing", "vertical listing", or "matrix", and the cells with their roles, i.e., "meta-data" or "data-value"). The batches shared 300 tables whose annotations were used to compute Krippendorff's Alpha inter-agreement coefficient. The coefficient was 96.11%, which is high enough to consider this repository a good ground truth.

We compared Coraline to four well-known competitors by Yoshida et al. [23], Jung and Kwon [28], Embley et al. [30], and Nishida et al. [5]. The first three are unsupervised proposals and the last one corresponds to the state-of-the-art supervised proposal. We configured them using the guidelines provided by the authors. Unfortunately, Yoshida et al.'s [23] 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. Regarding Coraline, we used the guideline by Salcedo-Sanz et al. [34] as a starting point and then performed some grid search to adjust its hyper-parameters as follows: 100 positions in the reef ($RSIZE = 100$), 30 generations ($NGEN = 30$), a maximum of three attempts to settle larvae ($MNSA = 3$), a 5.00% probability of mutation ($MUTP = 0.05$), 50% ratio of initial occupation of the reef ($INITR = 0.50$), 70% ratio of spawners ($SPAWR = 0.70$, i.e., the ratio of brooders is 30%), 10% ratio of budders ($BUDDR = 0.10$), and 10% ratio of perishment ($PERIR = 0.10$).

[1] Coraline, the competitors, and the datasets are publicly available at Mendeley Data (http://dx.doi.org/10.17632/87gr74cr4r.3).

4.2. Results

Regarding the performance measures, we collected the usual ones, namely: precision, recall, the F_1 score, and prediction time. The confusion matrix was computed by considering the meta-data class as the positive one and the data-value class as the negative one. Precision was then defined as the ratio of meta-data cells that were correctly predicted (true positives) to the total number of predicted meta-data cells (true positives plus false positives); recall was defined as the ratio of meta-data cells that were correctly predicted (true positives) to the total number of actual meta-data cells (true positives plus false negatives). The F_1 score was computed as the two-harmonic mean of precision and recall. The prediction time 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 time required to learn the model was apportioned across all of the tables. The ground truth was 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.

Table 1 shows the raw experimental results. The rows provide the averaged results attained per layout and repository plus their grand averages. The first column shows the name of the repository and the second column reports on the layout with which the proposal is dealing; then come five complex columns that show the results regarding the proposals that were compared. The highlighted cells correspond to the best results.

Fig. 2 provides an overall picture using box and whisker plots. The dots in the vertical lines show the full distribution of results, the boxes enclose the second-to-third quartile ranges, and the horizontal lines within the boxes denote the medians.

4.3. Analysis

Regarding effectiveness, Coraline attains the best average precision, recall, and F_1 score with the tables in both repositories. Only the proposal by Nishida et al. [5] attains slightly better results with the horizontal listings from the Wikipedia repository and the vertical listings from the DWTC repository. The problem in these cases is that Coraline classified some horizontal and vertical listings as matrices when their first rows and columns were very different from the others. The difference is small, but Coraline is an unsupervised proposal whereas Nishida et al.'s [5] proposal is supervised; note, too, that the neural network in their approach is more inefficient than the coral-reef optimisation approach in our proposal. Yoshida et al.'s [23] proposal did not attain the best results since this technique is based on the assumption that the meta-data cells have common contents across different tables, which is not expected to be generally true in the Web. Furthermore, their approach is also based on the frequency of the terms, which are not very repetitive and may occur only once in the case of numeric cells. The effectiveness results achieved by Jung and Kwon's [28] proposal are very similar to the results achieved by Yoshida et al.'s [23] proposal, especially in the Wikipedia repository. In the DWTC repository, the results are worse. This happens because there are some data-value cells that are highlighted in the tables and they are wrongly classified as meta-data cells. Embley et al.'s [30] proposal seems to work better than the previous two, and it is sometimes better than Nishida et al.'s [5] proposal and sometimes not. However, we should note that it achieves very good results when dealing with matrices because it was originally intended to extract information from spreadsheets, but the tables in our repository do not typically have many repeated values and matrices are not the most frequent layout. This is likely the reason why Coraline is generally

Table 1
Raw experimental results.

Repository	Layout	Coraline				Yoshida et al. [23]				Jung and Kwon [28]				Embley et al. [30]				Nishida et al. [5]			
		P	R	F_1	T	P	R	F_1	T	P	R	F_1	T	P	R	F_1	T	P	R	F_1	T
Wikipedia	Horizontal lst.	0.86	0.92	0.88	7.90	0.72	0.78	0.74	0.38	0.70	0.75	0.72	< 0.01	0.63	0.79	0.69	< 0.01	0.91	0.95	0.92	11.96
	Vertical lst.	0.88	0.90	0.89	6.52	0.45	0.51	0.47	0.38	0.47	0.59	0.50	< 0.01	0.77	0.83	0.80	< 0.01	0.61	0.63	0.62	11.98
	Matrix	0.92	0.97	0.94	7.88	0.55	0.60	0.55	0.38	0.58	0.66	0.60	< 0.01	0.88	0.96	0.91	< 0.01	0.84	0.80	0.81	11.96
	Grand Average	0.89	0.93	0.90	7.43	0.57	0.63	0.59	0.38	0.58	0.67	0.61	< 0.01	0.76	0.86	0.80	< 0.01	0.79	0.79	0.78	11.97
DWTC	Horizontal lst.	0.82	0.83	0.82	7.08	0.64	0.69	0.65	0.22	0.49	0.58	0.52	< 0.01	0.65	0.70	0.65	< 0.01	0.71	0.72	0.71	11.82
	Vertical lst.	0.91	0.92	0.91	5.77	0.46	0.49	0.47	0.21	0.49	0.60	0.52	< 0.01	0.83	0.82	0.82	< 0.01	0.92	0.93	0.92	11.81
	Matrix	0.97	0.96	0.96	7.26	0.55	0.60	0.56	0.21	0.29	0.51	0.36	< 0.01	0.95	0.98	0.96	0.00	0.83	0.77	0.79	11.82
	Grand Average	0.90	0.90	0.90	6.70	0.55	0.59	0.56	0.21	0.42	0.56	0.47	< 0.01	0.81	0.83	0.81	0.00	0.82	0.81	0.81	11.82
Overall	Horizontal lst.	0.85	0.88	0.86	7.59	0.69	0.74	0.71	0.32	0.62	0.68	0.64	< 0.01	0.64	0.76	0.67	< 0.01	0.83	0.86	0.84	11.91
	Vertical lst.	0.91	0.91	0.91	5.88	0.46	0.50	0.47	0.24	0.32	0.52	0.38	< 0.01	0.82	0.82	0.82	< 0.01	0.87	0.88	0.88	11.83
	Matrix	0.93	0.96	0.94	7.74	0.55	0.60	0.55	0.34	0.56	0.64	0.58	< 0.01	0.89	0.97	0.92	< 0.01	0.83	0.79	0.81	11.92
	Grand Average	0.90	0.92	0.90	7.07	0.57	0.61	0.58	0.30	0.50	0.61	0.53	< 0.01	0.78	0.85	0.80	< 0.01	0.84	0.84	0.84	11.89

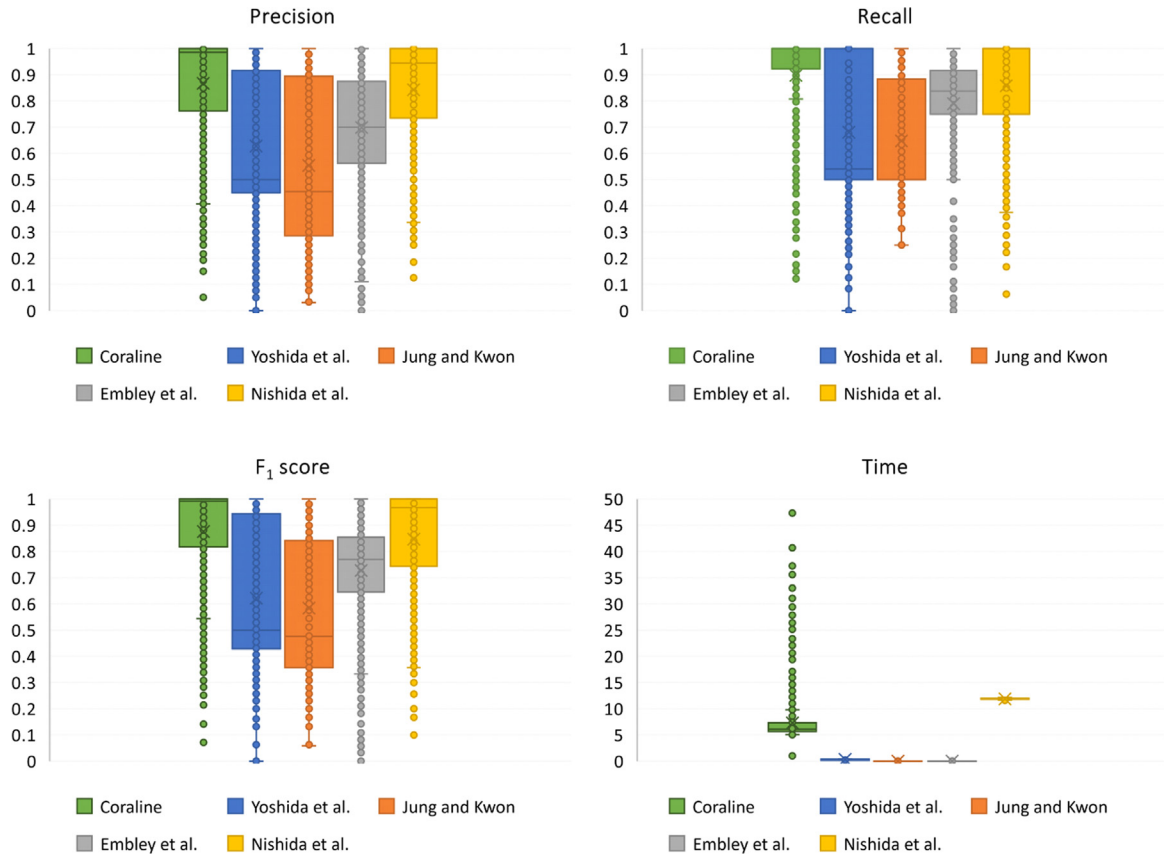

Fig. 2. Box and whisker plots.

better than Embley et al.'s [30] proposal, although there are a few cases in which its recall is slightly worse.

Regarding efficiency, Jung and Kwon's [28] and Embley et al.'s [30] proposals are the fastest, since they take an average time that is almost negligible. Note, however, that the efficiency comes at the cost of effectiveness, which does not seem an appropriate trade-off in our context. Yoshida et al.'s [23] proposal ranks next and it is followed by Coraline and the proposal by Nishida et al. [5]. Yoshida et al.'s [23] proposal seems to provide a balance between effectiveness and efficiency, but the improved effectiveness of Coraline and Nishida et al.'s [5] proposal justifies leaning towards them in our context. Note that the timings are worse, but good enough in practice.

The previous analysis provides an overall idea of which proposals rank the first and how they compare to each other. The box and whisker plots suggest that Coraline generally ranks the best regarding effectiveness and it is closely followed by Nishida et al.'s [5] proposal, and then comes Yoshida et al.'s [23], Embley et al.'s [30], and Jung and Kwon's [28] proposals. They also suggest that Coraline is better regarding efficiency than Nishida et al.'s [5] proposal, but worse than the other proposals. The previous conclusions are based on the fact that Coraline has the smallest quartile range and the highest median regarding every performance measure.

Table 2
Empirical ranks.

Proposal	P	R	F_1	T
Coraline	1.85	2.11	2.08	1.46
Nishida et al. [5]	2.39	2.38	2.53	1.54
Embley et al. [30]	3.45	3.37	3.09	3.00
Yoshida et al. [23]	3.50	3.51	3.49	4.06
Jung and Kwon [28]	3.81	3.63	3.81	4.94

To prove that our conclusions are statistically sound, we performed a statistical analysis at the standard confidence level ($\alpha = 0.05$) [43,44]. Table 2 shows the empirical ranks computed from the results. We conducted Iman–Davenport's test and got a p-value that was nearly zero in each case, which confirms that the results support the hypothesis that there are statistically significant-differences in rank amongst the proposals regarding the performance variables. We then compared the best-ranking proposal according to each performance variable with the others and got a p-value of 0.00 for each comparison, which confirms the hypothesis that the differences in rank are statistically significant at the standard confidence level.

Summing up: the analysis supports the conclusion that Coraline is 6.67% better regarding effectiveness and 40.54% better regarding efficiency than the current state of the art and it is 11.11% better regarding effectiveness than the second best unsupervised proposal.

5. Conclusions

This article introduces Coraline, a new table-understanding proposal that uses a coral-reef optimisation approach to make meta-data cells apart from data-value cells, which is the key to guessing the layout of the tables and interpreting their contents. The approach is used to select a subspace of informative features of the cells and cluster them in synchrony, which has proven very effective and efficient on a variety of tables with different layouts, encoding problems, and formatting alternatives. Our experimental study on a large repository of tables from the Wikipedia and the Dresden Web Table corpus confirms that Coraline outperforms the state of the art both regarding effectiveness and efficiency in a totally unsupervised manner.

Our research plans include exploring how to extend Coraline so that it can explore several clusterings using multiple subspaces of informative features. We guess that this might result in a more effective approach at the cost of some extra inefficiency that might be improved by exploring parallel programming techniques and semi-supervised clustering.

CRediT authorship contribution statement

Patricia Jiménez: Conceptualisation, Validation, Resources, Writing – original draft, Writing – review & editing, visualisation. **Juan C. Roldán:** Conceptualisation, Software, Resources, Data curation, Investigation. **Rafael Corchuelo:** Conceptualisation, Software, Resources, Formal analysis, Investigation, Writing – review & editing, Visualisation, Supervision, 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.

Acknowledgements

The authors were supported by the Spanish R&D programme through grant PID2020-112540RB-C44 (MCIN/AEI/10.13039/501100011033) as well as the Andalusian R&D programme through grant P18-RT-1060. The work by Juan C. Roldán was also supported by a PIF grant from the University of Seville, Spain and the Fulbright programme, Spain. Dinamic Area, S.L. provided an ideal industrial context and the machinery required to perform the experimental evaluation. The authors are very grateful to Drs. Szekely and Knoblock from the University of Southern California for our fruitful collaboration on some of the ideas that led to the results in this article.

References

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

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

[3] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation, ACM Trans. Intell. Syst. Technol. 11 (2020) 13:1–13:35, http://dx.doi.org/10.1145/3372117.

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

[5] 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, URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14396.

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

[7] J.C. Roldán, P. Jiménez, R. Corchuelo, On extracting data from tables that are encoded using HTML, Knowl. Based Syst. 190 (2020) 105–157, http://dx.doi.org/10.1016/j.knosys.2019.105157.

[8] X.L. Dong, H. Hajishirzi, C. Lockard, P. Shiralkar, Multi-modal information extraction from text, semi-structured, and tabular data on the Web, in: SIGKDD, 2020, pp. 3543–3544, http://dx.doi.org/10.1145/3394486.3406468.

[9] O.Y. Yuliana, C. Chang, A novel alignment algorithm for effective web data extraction from singleton-item pages, Appl. Intell. 48 (11) (2018) 4355–4370, http://dx.doi.org/10.1007/s10489-018-1208-0.

[10] C. Bizer, R. Meusel, A. Primpel, Web data commons: RDFa, microdata, embedded JSON-LD, and microformat data sets, Technical Report, University of Mannheim, 2019, URL http://webdatacommons.org/structureddata/2019-12/stats/stats.html.

[11] H.A. Sleiman, R. Corchuelo, A survey on region extractors from web documents, IEEE Trans. Knowl. Data Eng. 25 (9) (2013) 1960–1981, http://dx.doi.org/10.1109/tkde.2012.135.

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

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

[14] S. Salcedo-Sanz, J.D. Ser, I. Landa-Torres, S. Gil-Lopez, J.A. Portilla-Figueras, The coral-reef optimization algorithm: a novel meta-heuristic for efficiently solving optimization problems, Sci. World J. 2014 (739768) (2014) http://dx.doi.org/10.1155/2014/739768.

[15] W. Liu, X. Meng, W. Meng, ViDE: a vision-based approach for deep web data extraction, IEEE Trans. Knowl. Data Eng. 22 (3) (2010) 447–460, http://dx.doi.org/10.1109/tkde.2009.109.

[16] M. Kayed, C.-H. Chang, FiVaTech: page-level web data extraction from template pages, IEEE Trans. Knowl. Data Eng. 22 (2) (2010) 249–263, http://dx.doi.org/10.1109/tkde.2009.82.

[17] H.A. Sleiman, R. Corchuelo, TEX: an efficient and effective unsupervised web information extractor, Knowl.-Based Syst. 39 (2013) 109–123, http://dx.doi.org/10.1016/j.knosys.2012.10.009.

[18] H.A. Sleiman, R. Corchuelo, Trinity: on using trinary trees for unsupervised web data extraction, IEEE Trans. Knowl. Data Eng. 26 (6) (2014) 1544–1556, http://dx.doi.org/10.1109/tkde.2013.161.

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

[20] P. Jiménez, R. Corchuelo, On learning web information extraction rules with TANGO, Inf. Syst. 62 (2016) 74–103, http://dx.doi.org/10.1016/j.is.2016.05.003.

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

[22] M.J. Cafarella, A.Y. Halevy, D.Z. Wang, E. Wu, Y. Zhang, WebTables: exploring the power of tables on the web, VLDB 1 (1) (2008) 538–549, http://dx.doi.org/10.14778/1453856.1453916.

[23] M. Yoshida, K. Torisawa, J. Tsujii, A method to integrate tables of the World Wide Web, in: WDA, 2001, pp. 31–34, URL https://wda2001.csc.liv.ac.uk/Papers/13_yoshida_wda2001.pdf.

[24] M.J. Cafarella, A.Y. Halevy, Y. Zhang, D.Z. Wang, E. Wu, Uncovering the relational web, in: WebDB, 2008, pp. 1–6, URL http://webdb2008.como.polimi.it/images/stories/WebDB2008/paper30.pdf.

[25] H.-H. Chen, S.-C. Tsai, J.-H. Tsai, Mining tables from large scale HTML texts, in: COLING, 2000, pp. 166–172, http://dx.doi.org/10.3115/990820.990845.
[26] Y. Yang, W.-S. Luk, A framework for web table mining, in: WIDM, 2002, pp. 36–42, http://dx.doi.org/10.1145/584931.584940.
[27] Y.-S. Kim, K.-H. Lee, Detecting tables in web documents, Eng. Appl. AI 18 (6) (2005) 745–757, http://dx.doi.org/10.1016/j.engappai.2005.01.009.
[28] S. Jung, H. Kwon, A scalable hybrid approach for extracting head components from web tables, IEEE Trans. Knowl. Data Eng. 18 (2) (2006) 174–187, http://dx.doi.org/10.1109/tkde.2006.19.
[29] H. Elmeleegy, J. Madhavan, A.Y. Halevy, Harvesting relational tables from lists on the Web, VLDB 20 (2) (2011) 209–226, http://dx.doi.org/10.1007/s00778-011-0223-0.
[30] D.W. Embley, S.C. Seth, G. Nagy, Transforming web tables to a relational database, in: ICPR, 2014, pp. 2781–2786, http://dx.doi.org/10.1109/icpr.2014.479.
[31] X. Chu, Y. He, K. Chakrabarti, K. Ganjam, TEGRA: table extraction by global record alignment, in: SIGMOD, 2015, pp. 1713–1728, http://dx.doi.org/10.1145/2723372.2723725.
[32] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, B. Pollak, Towards domain-independent information extraction from web tables, in: WWW, 2007, pp. 71–80, http://dx.doi.org/10.1145/1242572.1242583.
[33] K. Braunschweig, M. Thiele, W. Lehner, From web tables to concepts, in: ER, 2015, pp. 247–260, http://dx.doi.org/10.1007/978-3-319-25264-3_18.
[34] S. Salcedo-Sanz, P. García-Díaz, J.A. Portilla-Figueras, J.D. Ser, S. Gil-Lopez, A coral-reef optimization algorithm for optimal mobile network deployment with electromagnetic pollution control criterion, Appl. Soft Comput. 24 (2014) 239–248, http://dx.doi.org/10.1016/j.asoc.2014.07.007.
[35] K.-C. Hu, C.-W. Tsai, M.-C. Chiang, A multiple-search multi-start framework for meta-heuristics for clustering problems, IEEE Access 8 (2020) 96173–96183, http://dx.doi.org/10.1109/access.2020.2994813.
[36] C. Tsai, W. Chang, Y. Wang, H. Chen, A high-performance parallel coral reef optimization for data clustering, Soft Comput. 23 (19) (2019) 9327–9340, http://dx.doi.org/10.1007/s00500-019-03950-3.
[37] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering, Mach. Learn. 75 (2) (2009) 245–248, http://dx.doi.org/10.1007/s10994-009-5103-0.
[38] E. Crestan, P. Pantel, Web-scale table census and classification, in: WSDM, 2011, pp. 545–554, http://dx.doi.org/10.1145/1935826.1935904.
[39] J.M. Luna-Romera, M. Martínez-Ballesteros, J. García-Gutiérrez, J.C. Riquelme, External clustering validity index based on the χ^2 statistical test, Inform. Sci. 487 (2019) 1–17, http://dx.doi.org/10.1016/j.ins.2019.02.046.
[40] W.G.L. Cava, T. Helmuth, L. Spector, J.H. Moore, A probabilistic and multi-objective analysis of Lexicase selection and ϵ-Lexicase selection, Evol. Comput. 27 (3) (2019) 377–402, http://dx.doi.org/10.1162/evco_a_00224.
[41] Wikipedia, Database download, 2020, URL https://en.wikipedia.org/wiki/Wikipedia:Database_download.
[42] J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, Top-k entity augmentation using consistent set covering, in: SSDBM, 2015, pp. 8:1–8:12, http://dx.doi.org/10.1145/2791347.2791353.
[43] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, fifth ed., Chapman & Hall/CRC, 2011.
[44] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple datasets" for all pair-wise comparisons, J. Mach. Learn. Res. 9 (89) (2008) 2677–2694, URL https://jmlr.org/papers/v9/garcia08a.html.