A hybrid quantum approach to leveraging data from HTML tables

Patricia Jiménez[1] · Juan C. Roldán[1] · Rafael Corchuelo[1]

Abstract

The Web provides many data that are encoded using HTML tables. This facilitates rendering them, but obfuscates their structure and makes it difficult for automated business processes to leverage them. This has motivated many authors to work on proposals to extract them as automatically as possible. In this article, we present a new unsupervised proposal that uses a hybrid approach in which a standard computer is used to perform pre- and post-processing tasks and a quantum computer is used to perform the core task: guessing whether the cells have labels or values. The problem is addressed using a clustering approach that is known to be NP using standard computers, but our proposal can solve it in polynomial time, which implies a significant performance improvement. It is novel in that it relies on an entropy-preservation metaphor that has proven to work very well on two large collections of real-world tables from the Wikipedia and the Dresden Web Table Corpus. Our experiments prove that our proposal can beat the state-of-the-art proposal in terms of both effectiveness and efficiency; the key difference is that our proposal is totally unsupervised, whereas the state-of-the-art proposal is supervised.

Keywords HTML tables · Data extraction · Quantum computing

1 Introduction

Automated business processes commonly integrate multiple data sources to produce more useful information than the sources can yield individually. We are interested in developing tools that help feed them with data that are provided by web sites across the world. Those data sources are particularly problematic in cases in which the data are encoded using HTML, since this encoding is intended to help render user-friendly documents, not to help machines

✉ Rafael Corchuelo
corchu@us.es

Patricia Jiménez
patriciajimenez@us.es

Juan C. Roldán
jcroldan@us.es

1 ETSI Informática, University of Sevilla, Avda. Reina Mercedes, s/n, Sevilla 41012, Spain

understand the data therein. Particularly, we focus on analysing data tables, which are HTML tables that are used to display data, not to position other HTML elements on the screen. Data tables are very popular according to many studies; for instance, Cafarella et al. [8] found 154 million data tables in a crawl with 14.10 billion tables, Crestan and Pantel [15] found 1.30 billion data tables in a crawl with 12 billion tables, Pimplikar and Sarawagi [53] found 25 million data tables in a crawl with 500 million HTML documents, and the recent Web Table Corpora initiative found 233 million data tables in a crawl with 1.78 billion HTML documents.

Unfortunately, it is not easy to use the data in a typical HTML table to feed an automated business process. The problem is that such data are not structured: they are encoded using a variety of HTML tags that facilitate rendering them, but obfuscate extracting them using programmatic procedures. HTML5 provides a number of tags that are intended to encode some semantics and facilitate the extraction, but they are biased towards encoding horizontal tables, they do not provide a means to delimit the data tuples or the headers, and they cannot make it explicit the relationship between the headers and the data in the tuples; furthermore, it is very common to find tables in which the tags are not used properly (for instance, in our experimental repositories, roughly 34.96% of the label cells and 11.74% of the value cells are incorrectly encoded.) Meta-data tags are intended to enhance the ability of plain HTML to encode the semantics of the data [35]. 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 tags [5], which means that there are roughly 20.12 million domains that do not facilitate at all extracting their data. Major knowledge bases have built-in data extractors that can deal with many HTML documents, but they are far from universal. For instance, Oulabi and Bizer [52] analysed the tables in a few domains that are well supported by DBpedia and extracted 206,690 data records with no matches in the knowledge base. And WikiData has a number of data extractors that can only work with tables that are generated using templates like infobox, navbox, or sistersitebox. It is then not surprising that many sites that provide data tables remain out of the reach of automated business processes [46].

The previous problems clearly argue for a method to extract data from HTML tables. In the literature, there are many general proposals to extract data from HTML documents [12,23,55,58,61]. Some of them are based on analysing visual features [47], text alignment [57,60], neural networks [59], matching trees [38], learning first-order rules [33], and also inferring propositio-relational rules [34], to mention a few. Unfortunately, they are not appropriate to extract the underlying relationships between the cells in a data table [9], which motivated many researchers to devise proposals that are specifically tailored to understand the intrinsic relational nature of HTML tables [14,20,31,48,54,70,71].

In this article, we present LuperQ, which is our proposal to feed automated business processes with data that are encoded using HTML tables. It deviates from the other proposals in the literature in that it relies on a quantum clustering approach that builds on an entropy-preservation metaphor that helps make the label cells apart from the value cells very effectively and efficiently, which is the cornerstone to understand the tables automatically. Our approach can find the optimum solution in polynomial time, which constitutes a huge speedup for a problem that is known to be NP using standard computers. We performed our experiments on two large repositories with tables from the Wikipedia and the Dresden Web Table Corpus. Our conclusion was that LuperQ can outperform the state-of-the-art proposal by Nishida et al. [51] in terms of both effectiveness and efficiency; the key here is that LuperQ can attain these results in a totally unsupervised manner, whereas the state-of-the-art proposal is supervised. The differences in performance were confirmed to be statistically significant using statistical hypothesis testing at the standard significance level.

The rest of the article is organised as follows: Sect. 2 describes and compares the related work; Sect. 3 presents the details of our proposal; Sect. 4 analyses its computational complexity; Sect. 5 reports on our experimental results; finally, Sect. 6 presents our conclusions and some future work.

2 Related work

In this section, we first summarise the literature on extracting data from HTML tables; then, we provide a short introduction to quantum computing; next, we summarise the literature on quantum clustering; finally, we discuss on our contributions.

2.1 Data extraction from HTML tables

Extracting data from HTML tables is a hot research topic. The two recent surveys by Roldán et al. [54] and Zhang and Balog [71] make it clear that there are a variety of approaches to the problem. Most of them are modelled as pipelines that are composed of the following tasks: locating the tables in the input documents, segmenting them into cells, discriminating the non-data tables, analysing the function of the cells, analysing the structure of the tables, and interpreting them. The functional analysis task is a cornerstone in this pipeline since the overall quality of the data extracted heavily depends on the ability of a proposal to make the label cells apart from the value cells.

Some of the proposals to implement the functional analysis task are very naive, namely: Braunschweig et al. [7] assumed that the label cells are located on the first row and Wu et al. [66] assumed that they are encoded using th tags only. Unfortunately, many real-world HTML tables are far more complex.

The literature provides better approaches. Chen et al. [13] devised a proposal that identifies the label cells by measuring their similarity according to some pre-defined features; Yoshida et al. [69] used the Expectation Maximisation method to learn the probability that a cell provides a label or a value using a large learning corpus; Yang and Luk [68] devised a proposal that is specifically tailored to work with numeric tables, since it exploits the fact that the data in such tables have well-known patterns; Kim and Lee [42] devised a proposal that first uses some heuristics that are based on the dimensions of the input table to determine the regions in which the label or the value cells are and then performs some pattern-based coherency checks; Jung and Kwon [36] developed another heuristic-based proposal that attempts to identify the regions with label or value cells by analysing some stylistic features; Gatterbauer et al. [25] used some visual features to match the structure of a table onto a number of common table types in which it is easy to identify the functions of the cells; Cafarella et al. [8] presented 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; Embley et al. [21] developed a method that relies on identifying some so-called critical cells that allow to divide the input table into four regions, some of which can index the others; Milošević et al. [49] developed a heuristic-based proposal to extract data from the tables in the PubMed Central repository; and Nishida et al. [51] devised the current state-of-the-art proposal, which uses a recurrent neural network with an embedding layer, a convolution layer, a filter layer, a fully-connected layer, and a SoftMax layer.

2.2 Quantum computing in a nutshell

Quantum computing is nowadays a key topic in computing research. Unfortunately, quantum computers are far from being sold at major retail stores, but the hype behind them is motivating many companies and researchers to become early adopters and to publish many introductory tutorials [37,41,43]. Basically, quantum computing is about taking advantage of some phenomena that occur at the (sub-)atomic levels to perform computations. In the last three decades, many physicists have been struggling to transform the idea into real-world computing devices [10,26]. It is now the time for computer scientists to leverage them to boost machine learning. In this field, it is common to address NP problems by searching a space of solutions using (meta-)heuristics that allow to find good-enough solutions that are not necessarily optimal. Quantum computers might be the key to find the optimal solutions very efficiently [18,65].

Describing some fundamental differences amongst classical and quantum computers may help understand quantum computing better. They both build on encoding data using strings of zeroes and ones. In classical computers, the zeros and ones are referred to as bits and they are implemented using voltages, current pulses, or latch circuits. In quantum computers, they are referred to as qubits and they are implemented using ions, electrons, or other (sub-)atomic particles; such particles oscillate between the two states (including many intermediate states) almost instantaneously unless they are somewhat constrained. Commonly, researchers say that unconstrained particles are in superposition and constrained particles are entangled. In classical computers, computations are performed by moving the bits through a circuit that consists of Boolean gates, e.g., “not”, “and”, “or”, or “xor”. In quantum computers, there are two mainstream models: the circuit model and the adiabatic model. The circuit model is similar to the electronic counterpart since it relies on circuits in which the qubits are entangled using quantum gates, e.g., “Pauli-X”, “Hadamard”, “Phase”, or “Toffoli”. The adiabatic model arranges the qubits in a network, e.g., a Chimera or a Pegasus network, and then uses annealing to perform computations. Fortunately, standard programmers may use high-level programming languages that are compiled into machine code that is interpreted by a processor that ultimately relies on Boolean gates. Unfortunately, quantum programmers must use a standard computer to encode the data and to set up a circuit or a network, then wait for the qubits to stabilise, and then use a standard computer to observe their state and to decode the results. Simply put: classical computers may be programmed at a very high abstraction level in which Boolean gates constitute just a theoretical foundation; in quantum computers, the qubits, the gates, or the networks constitute the highest abstraction level achieved so far.

As of the time of writing this article, it remains unclear which model is superior to the other or if they are equivalent. The quantum circuit model is attractive in that it is the closest to the electronic counterpart; unfortunately, the lack of quantum memory renders most current proposals of theoretical interest only. The adiabatic approach seems very appropriate to deal with problems that can be naturally expressed as optimisation problems and the current technology provides many more qubits than the circuit model; furthermore, there are some research results that allow to decompose arbitrary-size problems into smaller problems whose individual solutions can be computed using an adiabatic computer and then combined using a standard computer [6].

The key is that quantum computers work almost instantaneously. That is obviously a strong point in favour of quantum computing; unfortunately, the problems are manyfold, namely: the number of qubits is very limited (e.g., Google’s Bristlecone computer relies on the circuit model and provides 72 qubits, whereas D-Wave’s Advantage computer relies on the adiabatic model and provides 5000 qubits); developers must work at the qubit level because there are not any "quantum compilers" that can transform arbitrary algorithms that are expressed in high-level languages into quantum circuits or adiabatic networks; quantum computers are very sensitive to noise, which means that their current error rate is far above the error rate in standard computers; finally, quantum computers require to sample the solution to a problem many times in order to determine which the most-probable one is.

2.3 Approaches to quantum clustering

Regarding the circuit model, there are some well-known proposals by Aïmeur et al. [1], Wittek [64], and Kerenidis et al [40]. They leveraged some key ideas from algorithmic proposals like k-means, k-medians, or hierarchical clustering and attempted to implement them using quantum circuits. The key problem is to compute the distance between any two points in the input dataset, which has been addressed using oracles (associative memories), Grover's search circuit, or the improvement by Dürr and Høyer. Unfortunately, these solutions require quantum memory circuits, which are still on the design table. Giovannetti et al. [27] or Kerenidis and Prakash [39] have recently presented some proposals to implement quantum memory circuits that might be used to implement an oracle by pre-loading the distance matrix using a standard computer. Unfortunately, as far as we know, this kind of memory has not been implemented yet, which renders the previous proposals of theoretical interest only. Neither is it clear how floating point operations can be implemented using quantum circuits since the number of qubits required to implement a single double precision floating point number typically exceeds the total number of qubits that are available in many quantum computers. Chakraborty et al. [11] presented some building blocks that might be useful to implement a quantum version of k-means. Unfortunately, their proposal has many limitations: their formulation works for $k = 2$ only and it is not clear how it can be extended; they require a standard computer to compute the seed clusters; they can implement only an iteration of the k-means algorithm since current quantum circuits cannot loop; and it cannot deal with real data since their circuits work on positive integers only (neither is it clear how they react when a subtraction results in a negative integer). Kerenidis et al. [40] have recently presented another proposal to implement k-means using a quantum circuit. It can deal with real data, but it also requires a quantum memory circuit to store intermediate data. The authors also devised a proposal to implement such a circuit [39], but, as far as we know, it has not been materialised yet. They analysed the performance of their proposal in the case of so-called well-clusterable datasets, but how it works with other kinds of datasets is not known. (A dataset is well-clusterable if most of the data are close to the centroids, the centroids are sufficiently far from each other, and the clusters are approximately the same size.)

The adiabatic model seems to be more appropriate to address clustering problems. The reason is that clustering can be naturally modelled as an optimisation problem in which the goal is to maximise a fitness function that measures how compact and isolated the clusters are subject to restrictions like hard vs soft clustering, partitioning vs overlapping clustering, flat vs hierarchical clustering, or single-way vs multi-way clustering. The key difficulty is how to map clustering onto a problem that an adiabatic computer can solve. Currently, they can only deal with quadratic unconstrained binary optimisation problems (aka QUBO problems), using either spin variables, whose values are $+1$ or -1, or Boolean variables, whose values are 0 or 1. Several authors have devised different approaches to perform the mapping. Neukart et al. [50] first use a standard computer to generate a number of random seed clusters that are distributed homogeneously across the input dataset; the number of data in each seed cluster (ν) and the shift of each seed cluster with regard to the previous one (ϵ) are hyper-parameters that must be fine-tuned for each particular dataset. They require a total of $k\,d\ \nu$ qubits to cluster a dataset with d-dimensional data into k clusters; basically, they associate $d\ \nu$ qubits with each cluster and formulate a QUBO problem per datum in the input dataset; they assign the datum to the cluster that results in a larger number of qubits whose state is one. Bauckhage et al. [4] presented a proposal to cluster a dataset into two clusters of approximately the same size. They require the input dataset to be standardised so that its mean is zero. Their proposal builds on the fact that the k-means algorithm actually implements a number of heuristics whose goal is to find a good-enough minimum of the within-cluster scatter function; they realised that finding that minimum is equivalent to maximising the between-cluster scatter function, which sounds intuitive but requires a non-trivial mathematical proof. Assuming that $k = 2$ and the two clusters are similar in size, they introduced a clever trick to compute the QUBO problem without relying on explicit distance functions. Kumar et al. [44] presented two proposals. The first one uses one-hot encoding and can cluster a dataset into an arbitrary number of clusters; basically, it assigns an array of k Boolean variables to each datum; each variable determines whether the datum belongs to the corresponding cluster or not. The idea was to compute an energy matrix from the Euclidean distance matrix, but their encoding scheme requires to set n Lagrange multipliers whose exact values are not easy to compute. Unfortunately, the authors concluded that the quality of the clustering worsens as $n \gg k$, which is very common in practice. Their formulation is also very dependent on the floating point precision that can be implemented using the quantum computer, which is six bits in the current state of the art according to the authors. This is the reason why they developed an ad-hoc proposal for the common case in which $k = 2$. Wereszczyński et al. [62] presented another approach that targets finding two clusters only; unfortunately, it was only simulated on a standard computer using small datasets with 50 data only. Their proposal assigns a qubit to every datum and creates a QUBO problem in which data that are less than half the diameter of the dataset are assigned positive weights and data that are further apart are assigned negative weights.

There are a few more proposals that got inspiration from quantum mechanics. They rely on algorithms that are intended to be run on a standard computer, not a quantum computer. The goal is to map the clustering problem onto some phenomena that are formalised using Schrödinger's equation, which relies on a wave function that can be analysed using some complex, but well-known results in numerical calculus. Horn and Gottlieb [30] range amongst the pioneers in this field. They devised a quantum clustering algorithm that basically constructs a wave function from the input dataset and then searches for the clusters in its equi-potential regions. It relies on a hyper-parameter that determines how large the clusters can be. Unfortunately, the authors presented a solution for two-dimensional data only and simply sketched how to extend it to the multi-dimensional case; an inherent problem of this extension is that the evaluation of the potential function requires $O(n^2)$ time, where n denotes the number of data, independently from their dimensionality. Li et al. [45] followed up on Horn and Gottlieb's [30] proposal and devised a new one that applies Kernel Entropy Component Analysis [32] to the input dataset prior to clustering; they also devised a statistical approach to compute the potential function that is more effective and efficient than the original one, but still requires to compute the k-nearest neighbourhood of every datum, which is an inherently complex operation. Decheng et al. [16] presented an improvement that uses a new weighted distance that does not assume that all of the attributes of the input data are equally important. Unfortunately, the method to compute the weight heavily depends on the characteristics of the datasets, which hinders its practical applicability. Eslava et al. [22] presented an approach that improves on Horn and Gottlieb's [30] and Li et al's [45] proposals since they use a Bayesian framework that provides an objective measure of goodness-of-fit that helps them optimise their hyper-parameters in a totally unsupervised manner.

2.4 Discussion

Extracting data from HTML tables is an important problem nowadays. The sprout of data-hungry services that rely on machine learning is motivating the need for many datasets on a variety of topics. Unfortunately, many such datasets are available in HTML tables only, which argues for a solution to extract them automatically. In the literature, there are many proposals to extract data from HTML tables, but our analysis reveals that none of them has ever attempted to use a clustering approach to implement the functional analysis task, which we think is one of the cornerstones in the process.

We guess that many authors may have thought that the clustering approach would not result in an efficient solution to the problem. Assuming that the cells are somewhat projected onto d-dimensional feature vectors ($d \geq 1$), the size of the space that must be searched to find which of those features are actually relevant is $O(2^d)$; furthermore, the size of the space that must be searched to find the clusters is $\frac{1}{k!} \sum i = 0k(-1)^{k-i} \binom{k}{i} i^n$, where k denotes the number of clusters and n denotes the number of cells ($n \geq 1$) [67]. Finding the optimum in such a space is not feasible using standard computers; even in cases like ours in which $k = 2$. This has motivated many researchers to devise algorithms whose goal is to find good-enough clusterings that are not necessarily optimal. Furthermore, it is not trivial to determine which of the clusters correspond to which kind of cells; the clustering is a clue, but not the solution to the problem.

Quantum computing has changed the vision of NP problems since many of them have been proven to become polynomial or even constant-time using that technology. Clustering is one such problem. In the literature, there are many proposals: we discard the ones that rely on the circuit model because there are not any quantum memories available, which renders such proposals of theoretical interest only; we also discard the quantum-inspired proposals, because they are standard algorithms that were devised to run on standard computers, not quantum computers; thus, our focus is on the adiabatic model, which seems more appropriate to address clustering.

Our proposal differs from the existing ones, namely: it relies on an entropy preservation metaphor in which the input dataset is interpreted as a collection of particles, which deviates from the k-means-inspired approaches by Neukart et al. [50] or Bauckhage et al. [4]; it has proven to work well in a context in which the size of the clusters is clearly unbalanced, but the proposal by Bauckhage et al. [4] is biased towards finding clusters that are similar in size; it has proven to work well with datasets in which the number of data is much larger than the number of clusters, but the first proposal by Kumar et al. [44] worsens as the difference increases; it requires floating point computations during its preprocessing step only, which is carried out using a standard computer, whereas Kumar et al.'s [44] first proposal depends on the limited precision that can be achieved using the quantum computers; it can be easily generalised to an arbitrary number of clusters, although we only need two, whereas the proposals by Bauckhage et al. [4], Wereszczyński et al. [62], and the second proposal by Kumar et al. [44] can only find two clusters; furthermore, it has been tested on two large repositories of real-world tables, whereas Wereszczyński et al.'s [62] proposal was only simulated and tested on small datasets.

3 Our proposal

In this section, we first present some preliminaries and then describe the three steps in which we have grouped the tasks of the data extraction pipeline: pre-processing, identifying cell functions, and post-processing.

3.1 Preliminaries

Definition 1 (*Mathematical concepts*) A d-dimensional vector V is a tuple of the form $(v_1, v_2, \ldots, v_d)$ $(d \geq 1)$; 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 (p, q)-dimensional matrix M is a tuple of the form $((v_{1,1}, v_{1,2}, \ldots, v_{1,q}), \ldots, (v_{p,1}, v_{p,2}, \ldots, v_{p,q}))$ $(p \geq 1, q \geq 1)$; its dimensionality is denoted as $\dim M = (p, q)$. Given matrix M and two natural numbers i and j, its component at position (i, j) is denoted as $M[i, j]$ $(1 \leq i \leq p, 1 \leq j \leq q)$.

Definition 2 (*Documents, tables, 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 a grid that is used to display data or to position other elements on the screen. The former are called data tables and the latter are called non-data tables. Every piece of HTML whose root is a node with a table tag is considered a table. Rows are encoded using tr tags and cells are encoded using th or td tags. The function of a cell can be either label, which means that it provides a semantic hint, or value, which means that it provides a datum. Data tables are commonly laid out as horizontal listings, in which the label cells are at the top rows (if any) and the value cells are in the rows below, vertical listings, in which the label cells (if any) are on the leftmost columns and the value cells are in the columns to the right, and matrices, which have label cells both at the topmost rows and the leftmost columns and the value cells occupy the bottom-right area of the table. A data record is a map of the form $\{h_i : v_i\}_{i=1}^{k}$ $(k \geq 1)$ 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 in the same row/column to form a descriptor that provides a semantic hint for the corresponding value.

Definition 3 (*Features, clusterings*) A feature is a property of a cell. The features of a cell are represented by means of feature vectors; the features of the cells of a table are represented by means of a collection of feature vectors. The features can be categorised as intra-cell features or inter-cell features. An intra-cell feature is computed from the attributes of a single cell. (Table 1 provides the catalogue of attributes from which we computed useful intra-cell features in our experimental study.) It can be a visual feature, which is computed from the rendering, a structural feature, which is computed from the structure of the DOM tree, a lexical feature, which is computed from the contents of the cells, or a miscellaneous feature, which is computed from several sources. An inter-cell feature is computed from 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. A clustering is a Boolean matrix whose components indicate whether the corresponding cells in a table are label cells or value cells.

Definition 4 (*Quantum foundations*) A quantum system is a set of (sub-)atomic particles that interact with each other through couplers. (Note that is not generally possible to connect every two particles with a coupler due to physical constraints.) Both the particles and the couplers

Table 1 Taxonomy of attributes

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

may have some energy; the particles interact with each other by exchanging energy through the couplers. The energy level of a particle induces a property that can be probabilistically mapped onto a Boolean state, e.g., a spin or a spatial position. A particle is said to be in superposition or entangled depending on whether its state is not constrained or constrained by its interaction with other particles, respectively. The state of a system is determined by the state of its particles. A Hamiltonian is a function that models the total amount of energy of a quantum system in terms of its state. A process in a quantum system is said to be adiabatic if it does not transfer any energy or mass to the environment, that is: if it happens in total isolation. A quantum computer is a device that traps a number of particles, sets their initial states, controls their interactions through the couplers, and reads their final states.

Definition 5 (*Ising systems*) An Ising system is a quantum system whose Hamiltonian is modelled as follows:

$$\sum_{i=1}^{n} H[i]S[i] + \sum_{i,j=1|i<j}^{n,n} J[i,j]\,S[i]\,S[j],$$

where n denotes the number of particles, $H[i]$ denotes the energy of the ith particle, $J[i, j]$ denotes the energy of the coupler between the ith and the jth particle, and $S[i]$ and $S[j]$ are variables that denote their spin ($1 \leq i \leq n$, $1 \leq j \leq n$). The range of energy depends on the quantum computer used, so it is commonly normalised in real interval $[-1.00, +1.00]$. The spin variables range in set $\{-1, +1\}$.

For the sake of simplicity, this Hamiltonian is commonly rewritten as follows:

$$B^T\,E\,B,$$

where B denotes the following vector of Boolean variables:

$$B[i] = {}^1\!/\!_2\,(S[i] + 1)\ (1 \leq i \leq n),$$

and $E = diag(H) + J$ is the so-called energy matrix, where $diag(H)$ denotes an $n \times n$ diagonal matrix with the elements of vector H.

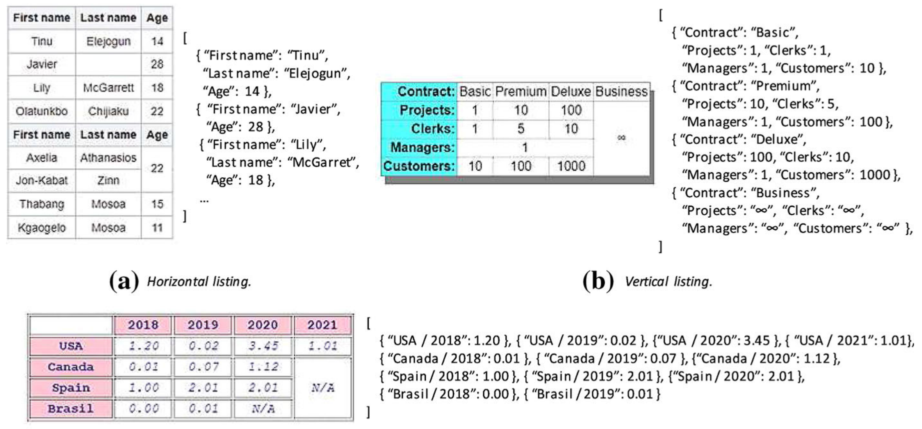
Definition 6 (*QUBO problems, adiabatic optimisation*) A QUBO problem is a quadratic unconstrained binary optimisation problem of the following form:

$$\underset{B}{\operatorname{argmin}}\, B^T\,E\,B$$

where E is an energy matrix and B is a vector of Boolean variables. Adiabatic optimisation is a quantum method that helps finding its optimal solutions. This is a well-known NP-hard problem using standard computers [50], but it can be addressed using adiabatic computers as follows: assume that there is a quantum computer that implements a system whose Hamiltonian is h; let h_t denote the exact value of the Hamiltonian at time t ($t \geq 0$); assume that the system has a particular Hamiltonian h^* whose state encodes the solution to the problem and that h_0 and h^* do not commute; then, the adiabatic theorem [28] states that there is a time s such that if the Hamiltonian of the system changes adiabatically according to equation

$$h_t = (1 - {}^t\!/\!_s)\,h_0 + {}^t\!/\!_s\,h^*,$$

then the system will remain in a state with Hamiltonian h^* after time s. That is, it suffices to wait for s units of time so that the system stabilises in a state that represents the solution to the problem. The value of s is known to be in the order of $O(e^{\alpha\,n\,\beta})$, where α and β are positive constants and n is the size of the problem. Determining whether constants α and β



First name	Last name	Age
Tinu	Elejogun	14
Javier		28
Lily	McGarrett	18
Olatunkbo	Chijiaku	22
First name	**Last name**	**Age**
Axelia	Athanasios	22
Jon-Kabat	Zinn	
Thabang	Mosoa	15
Kgaogelo	Mosoa	11

```
[
  { "Firstname": "Tinu",
    "Last name": "Elejogun",
    "Age": 14 },
  { "Firstname": "Javier",
    "Age": 28 },
  { "Firstname": "Lily",
    "Last name": "McGarret",
    "Age": 18 },
  ...
]
```

(a) *Horizontal listing.*

Contract:	Basic	Premium	Deluxe	Business
Projects:	1	10	100	∞
Clerks:	1	5	10	
Managers:		1		
Customers:	10	100	1000	

```
[
  { "Contract": "Basic",
    "Projects": 1, "Clerks": 1,
    "Managers": 1, "Customers": 10 },
  { "Contract": "Premium",
    "Projects": 10, "Clerks": 5,
    "Managers": 1, "Customers": 100 },
  { "Contract": "Deluxe",
    "Projects": 100, "Clerks": 10,
    "Managers": 1, "Customers": 1000 },
  { "Contract": "Business",
    "Projects": "∞", "Clerks": "∞",
    "Managers": "∞", "Customers": "∞" },
]
```

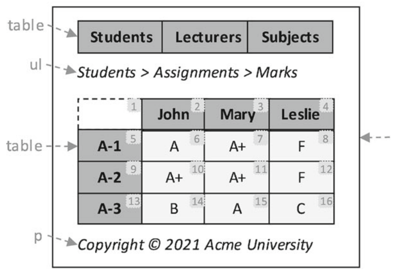
(b) *Vertical listing.*

	2018	2019	2020	2021
USA	1.20	0.02	3.45	1.01
Canada	0.01	0.07	1.12	N/A
Spain	1.00	2.01	2.01	
Brasil	0.00	0.01	N/A	

```
[
  { "USA / 2018": 1.20 }, { "USA / 2019": 0.02 }, {"USA / 2020": 3.45 }, { "USA / 2021": 1.01},
  { "Canada / 2018": 0.01 }, { "Canada / 2019": 0.07 }, {"Canada / 2020": 1.12 },
  { "Spain / 2018": 1.00 }, { "Spain / 2019": 2.01 }, {"Spain / 2020": 2.01 },
  { "Brasil / 2018": 0.00 }, { "Brasil / 2019": 0.01 }
]
```

(c) *Matrix.*

Fig. 1 Sample tables and data records



```
<html>
<head> ... </head>
<body>
  <table>
    <tr> <td>Students</td>  <td>Iecturers</td>  <td>Subjects</td>  </tr> ...
  </table>
  <ul> <li>Students</li> <li>Lecturers</li> <li>Marks</li> </ul>
  <table>
    <tr> <td/> <td>John</td>  <td>Mary</td>  <td>Leslie</td>  </tr>
    <tr>A-1<td/>  <td>A</td>  <td>A+</td>  <td>F</td>  </tr>...
  </table>
  <p>Copyright &copy; 2021 Acme University</p>
</body>
</html>
```

Fig. 2 Running example: sample document

are small enough to render the original problem tractable or not in a particular computer is a matter of experimentation [3].

Example 1 Figure 1 shows some real-world sample HTML tables with different layouts and their corresponding data records. Figure 2 shows a fictitious sample document that we use as a running example; the details regarding the CSS style used are omitted since they are cumbersome and provide little information to illustrate the proposal. The header of the document consists of a menu with options and a navigation bar; the body consists of a grid that shows the marks attained by several students (per column) on their assignments (per row); the footer consists of a copyright message. Note that the author of the document decided to encode both the menu and the grid using the table HTML tag, but the menu is a non-data table and the grid is a data table in this context. The grid is a matrix table because it has labels both on the first row and column. We have attached some tags to a few elements in the document to facilitate locating them in the HTML excerpt; we have also added a small number at the upper right corner of each cell in the grid to facilitate references. Realise that the encoding focuses on describing how the HTML elements must be rendered, not on helping a computer understand the data.

3.2 Step 1: pre-processing

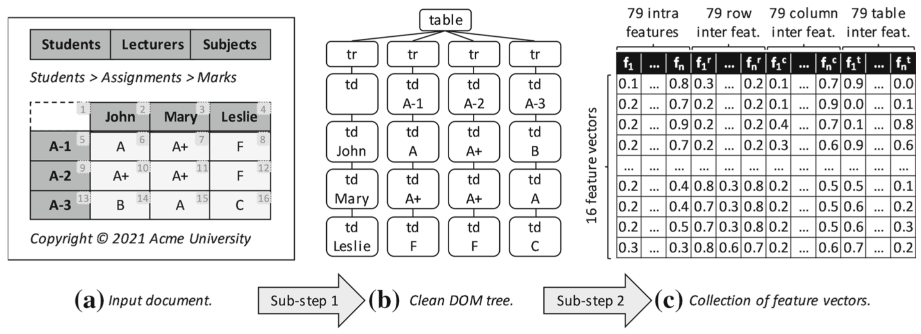
This step implements the location, the discrimination, and the segmentation tasks of the data extraction pipeline. It gets a document as input and returns its collection of data tables plus their corresponding collections of feature vectors.

The first sub-step computes the collection of tables in the input document, namely:

1. It loads the input document and transforms it into a DOM tree using an HTML parser. Next, it uses a headless browser to render the DOM tree on a virtual canvas and makes it explicit all of the attributes of the DOM nodes using a simple custom script. Finally, it uses a CSS selector to locate the DOM nodes with a table tag.
2. It then discards the tables whose width or height attributes are 0px or the tables whose display attribute is none because they are not visible to the user; it also discards the tables that have one single row/column because they are typically used to display listings that do not provide any data in tabular form; it further discards the tables that are nested within other tables because they are typically used to show menus, paginators, and other structures without any relevant data.
3. It then sets the maximum cell span to 200 cells, which helps avoid overhead when processing tables that are incorrectly encoded. The cells whose span is greater than one are replicated accordingly and their span is set to one. The rows that are shorter than the largest row are padded to the right using empty cells, which prevents outputting ragged tables. If the input 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 leftmost column. Duplicated rows or columns are removed, except for the topmost and the leftmost ones. If all of the cells in a row or a column are empty, then they are removed. (A cell is considered empty if it consists exclusively of blanks, dashes, question marks, or language-dependent symbols like "N/A".)

The second sub-step computes the collection of feature vectors that correspond to the cells of the input table. It iterates over the DOM nodes that correspond to the cells and computes their intra-cell features as the weighted average of their attributes and the intra-cell features of their children, where the weight is computed as the relative area of the bounding box in which each DOM node is rendered. We transform the attributes of the DOM nodes as follows: in the case of composite attributes, e.g., the colours, we split them into their components; in the case of enumerated attributes, e.g., the font, we use one-hot encoding; in the case of numeric attributes (be them original or split ones), we normalise their values in interval $[0.00, +1.00]$ using global min-max normalisation. This way, all of the attributes are numeric and range in the same interval, which facilitates computing the features. Then, we compute three inter-cell features that measure the deviations of the intra-cell features with respect to the intra-cell features of the cells in the same row, column, and table. The result is a collection of $(4\,\kappa)$-dimensional vectors, where κ denotes the number of intra-cell features.

Example 2 Figure 3 illustrates the pre-processing step. The input is the document in our running example. The first sub-step converts it into a DOM tree and cleans it: first, the document is parsed and the table elements are selected; second, the table element that represents the main menu is discarded because it consists of a single row, which is not common at all for data tables; third, the table element that represents the grid needs not any more cleaning because it does not have any excessively-spanned cells, the document is written using the common "lrt" direction, and no duplicated or empty rows/columns exists. The second sub-step transforms it into a collection of vectors that represent the cells in the



Students	Lecturers	Subjects

Students > Assignments > Marks

1	John 2	Mary 3	Leslie 4
A-1 5	A 6	A+ 7	F 8
A-2 9	A+ 10	A+ 11	F 12
A-3 13	B 14	A 15	C 16

Copyright © 2021 Acme University

79 intra features			79 row inter feat.			79 column inter feat.			79 table inter feat.		
f_1	...	f_n	f_1^r	...	f_n^r	f_1^c	...	f_n^c	f_1^t	...	f_n^t
0.1	...	0.8	0.3	...	0.2	0.1	...	0.7	0.9	...	0.0
0.2	...	0.7	0.2	...	0.2	0.1	...	0.9	0.0	...	0.1
0.2	...	0.9	0.2	...	0.2	0.4	...	0.7	0.1	...	0.8
0.2	...	0.7	0.2	...	0.2	0.3	...	0.6	0.9	...	0.6
...	...	...	...	...	...	...	...	...	...	...	...
0.2	...	0.4	0.8	0.3	0.8	0.2	...	0.5	0.5	...	0.1
0.2	...	0.4	0.7	0.3	0.8	0.2	...	0.5	0.6	...	0.2
0.2	...	0.5	0.7	0.3	0.8	0.2	...	0.5	0.6	...	0.3
0.3	...	0.3	0.8	0.6	0.7	0.2	...	0.6	0.7	...	0.2

Fig. 3 Running example: pre-processing

grid, in the order indicated by the small numbers at the upper-right corner of each cell. Unfortunately, this collection is far too large to be shown in a figure because it has 16 feature vectors with 316 dimensions each; thus, we decided to illustrate how it looks only. The first group of columns refer to the intra-cell features that are computed from the attributes of the DOM tree, cf. Table 1; basically, the attributes are decomposed and transformed into numeric ones, if necessary, and then normalised to $[0.00, +1.00]$. The following three groups of columns refer to the inter-cell features that measure the deviation of the previous features with respect to features of the cells in the same row, column, and table, respectively.

3.3 Step 2: identifying cell functions

This step implements the functional analysis task in the pipeline. It works on the tables and the collections of feature vectors returned by the previous step as follows: it first clusters the cells and then uses some heuristics to determine the function of the cells in each cluster.

3.3.1 Clustering the cells

Our idea is to map the clustering problem using an entropy-preservation metaphor. It is well-known that the entropy of the Universe increases monotonically. The net effect is that the Universe expands because its particles tend to go further apart as time goes by (thus increasing the entropy) unless some energy is used to keep them together (thus preserving the entropy). Our idea is to interpret a dataset as a collection of particles in a multi-dimensional Universe. For these particles to be stable in that space, some energy must be injected in the system so that its total amount of entropy remains constant; otherwise, the particles would tend to move apart and the entropy would increase. The idea is then to assume that the cells in a table are particles that are represented using their feature vectors; if the table "does not disintegrate", it is because there is some energy that keeps the entropy constant. The goal is then to find a clustering that requires a minimum amount of energy to keep the entropy constant.

The first sub-step maps the clustering problem onto a standard QUBO problem of the following form:

$$\underset{B}{\operatorname{argmin}}\ B^T\ E\ B,$$

where B is a Boolean variable vector and E is the energy matrix.

First, we have to define the Boolean variables. We introduce a matrix K with $n \times 2$ Boolean variables such that variable $K[i, k]$ denotes whether the ith cell belongs to cluster k or not and n denotes the number of cells ($1 \leq i \leq n, k \in \{1, 2\}$). The relationship between the variables in K and the variables in B are the following:

$$\begin{aligned} B[i] &= K[(i-1) \bmod n + 1, \lceil i/n \rceil] \text{ and} \\ K[i, k] &= B[i + n\,(k-1)]. \end{aligned}$$

Although the formulation is a bit involved, it is conceptually simple since the goal is to map the two-dimensional structure of matrix K onto the one-dimensional structure of vector B and vice versa. (Note that this idea can be easily extended to an arbitrary number of clusters; we focus on two clusters because we have two kinds of cells only.)

Now, we have to define the energy matrix. We know that $E[i, j]$ is the coefficient that corresponds to variables $B[i]$ and $B[j]$ in the QUBO problem ($1 \leq i, j \leq 2n$). We also know that variable $B[i]$ corresponds to variable $K[p, q]$ and variable $B[j]$ corresponds to variable $K[u, v]$, where

$$\begin{aligned} p &= (i-1) \bmod n + 1, \\ q &= \lceil i/n \rceil, \\ u &= (j-1) \bmod n + 1, \text{ and} \\ v &= \lceil j/n \rceil. \end{aligned}$$

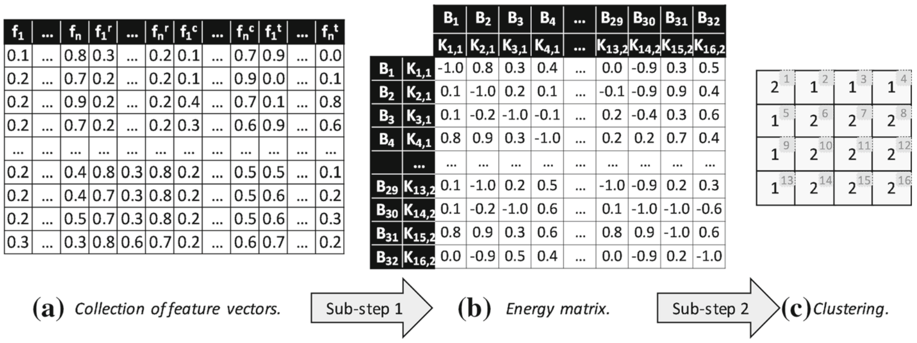
We now need to introduce two ancillary functions, namely: a distance function $\mathcal{D}$, which works on two indices that represent two cells and returns the distance between their corresponding feature vectors, and an energy function $\mathcal{E}$, which works on a distance value and returns the amount of energy required to keep two particles at that distance so that the entropy is preserved. Without any loss of generality, we assume that function $\mathcal{D}$ is normalised to interval $[0.00, +1.00]$, where the lower bound indicates the distance between a cell and itself and the upper bound indicates the maximum distance between any two cells in the same table; we also assume that function $\mathcal{E}$ is normalised to interval $[-1.00, +1.00]$, where the lower bound represents the amount of energy required to keep the farthest particles apart and the upper bound represents the amount of energy required to keep the closest particles together.

There are several cases regarding the energy matrix:

Case $p = u$ and $q = v$: in this case, $B[i]$ and $B[j]$ refer to the same cell and the same cluster. It makes sense to set $E[i, j] = -1.00$ so that the system is biased towards setting $B[i] = B[j] = 1$, which helps minimise the Hamiltonian; that is, the system is biased towards setting $K[p, q] = K[u, v] = 1$ so that both cells are assigned to the same cluster.

Case $p = u$ and $q \neq v$: in this case, $B[i]$ and $B[j]$ refer to the same cell, but different clusters. It makes sense to set $E[i, j] = +1.00$ so that the system is biased towards setting $B[i] = 0$ or $B[j] = 0$, which helps minimise the Hamiltonian; that is, the system is biased towards setting $K[p, q] = 0$ or $K[u, v] = 0$, which prevents the same cell from being assigned to two different clusters. Note that this case works co-ordinately with the previous one, which seeks to assign each cell to a cluster.

Case $p \neq u$ and $q = v$: in this case, $B[i]$ and $B[j]$ refer to different cells in the same cluster. It makes sense to set $E[i, j] = -\mathcal{E}(\mathcal{D}(p, u))$ so that the system is biased towards setting $B[i] = 1$, $B[j] = 1$, or both depending


Sub-step 1
Sub-step 2

f_1	...	f_n	f_1^r	...	f_n^r	f_1^c	...	f_n^c	f_1^t	...	f_n^t
0.1	...	0.8	0.3	...	0.2	0.1	...	0.7	0.9	...	0.0
0.2	...	0.7	0.2	...	0.2	0.1	...	0.9	0.0	...	0.1
0.2	...	0.9	0.2	...	0.2	0.4	...	0.7	0.1	...	0.8
0.2	...	0.7	0.2	...	0.2	0.3	...	0.6	0.9	...	0.6
...	...	...	...	...	...	...	...	...	...	...	...
0.2	...	0.4	0.8	0.3	0.8	0.2	...	0.5	0.5	...	0.1
0.2	...	0.4	0.7	0.3	0.8	0.2	...	0.5	0.6	...	0.2
0.2	...	0.5	0.7	0.3	0.8	0.2	...	0.5	0.6	...	0.3
0.3	...	0.3	0.8	0.6	0.7	0.2	...	0.6	0.7	...	0.2

(a) *Collection of feature vectors.*

		B_1	B_2	B_3	B_4	...	B_{29}	B_{30}	B_{31}	B_{32}
		$K_{1,1}$	$K_{2,1}$	$K_{3,1}$	$K_{4,1}$	...	$K_{13,2}$	$K_{14,2}$	$K_{15,2}$	$K_{16,2}$
B_1	$K_{1,1}$	-1.0	0.8	0.3	0.4	...	0.0	-0.9	0.3	0.5
B_2	$K_{2,1}$	0.1	-1.0	0.2	0.1	...	-0.1	-0.9	0.9	0.4
B_3	$K_{3,1}$	0.1	-0.2	-1.0	-0.1	...	0.2	-0.4	0.3	0.6
B_4	$K_{4,1}$	0.8	0.9	0.3	-1.0	...	0.2	0.2	0.7	0.4
	...	...	...	...	...	...	...	...	...	...
B_{29}	$K_{13,2}$	0.1	-1.0	0.2	0.5	...	-1.0	-0.9	0.2	0.3
B_{30}	$K_{14,2}$	0.1	-0.2	-1.0	0.6	...	0.1	-1.0	-1.0	-0.6
B_{31}	$K_{15,2}$	0.8	0.9	0.3	0.6	...	0.8	0.9	-1.0	0.6
B_{32}	$K_{16,2}$	0.0	-0.9	0.5	0.4	...	0.0	-0.9	0.2	-1.0

(b) *Energy matrix.*

2^{1}	1^{2}	1^{3}	1^{4}
1^{5}	2^{6}	2^{7}	2^{8}
1^{9}	2^{10}	2^{11}	2^{12}
1^{13}	2^{14}	2^{15}	2^{16}

(c) *Clustering.*

Fig. 4 Running example: clustering cells

on how distant the cells are. Note that the maximum energy level returned by function $\mathcal{E}$ is $+1.00$, which happens when two cells have the same feature vectors and their distance is then 0.00; such cells should clearly be in the same cluster, which is biased by setting $E[i, j]$ to the minimum possible energy level. Note, too, that the minimum energy level returned by $\mathcal{E}$ is -1.00, which corresponds to the farthest apart cells; such cells should clearly not be in the same cluster, which is biased by setting $E[i, j]$ to the maximum possible energy level. The energy level that corresponds to the other pairs of cells depends completely on the model provided by function $\mathcal{E}$.

Case $p \neq u$ and $q \neq v$: in this case, $B[i]$ and $B[j]$ refer to different cells in different clusters. It is the complementary of the previous case, so it makes sense to set $E[i, j] = -\mathcal{E}(\mathcal{D}(p, u))$.

Note that the exact formulation regarding how the energy matrix is computed depends completely on the definitions of the distance function $\mathcal{D}$ and the energy function $\mathcal{E}$. In the literature, there are a variety of choices to implement both functions [2,17,29] and none of them is clearly superior to the others in every case. Thus, we defer making a decision to the experimental analysis section.

The second sub-step uses a quantum computer to find the values of the B variables that minimise $B^T\ E\ B$. Once the values of the B variables are computed, it is very simple to map them onto the corresponding values of the K variables that determine the cluster to which each cell belongs. Hopefully, this step will result in a perfect clustering almost instantaneously, which is the key to attain the best possible effectiveness very efficiently.

Example 3 Figure 4 illustrates the previous procedure. The first sub-step consists in transforming the collection of feature vectors that was computed previously into a QUBO problem that helps compute a clustering of the input cells. That requires to introduce 32 Boolean variables of the form $K[i, k]$ because we have 16 cells and need to group them into two clusters; intuitively, variable $K[i, k]$ must be one in cases in which the ith cell belongs to cluster k and zero otherwise ($1 \leq i \leq 16, k \in \{1, 2\}$). The K variables are very intuitive, but their two-dimensional structure makes it impossible to use them in a QUBO problem unless they are mapped onto a one-dimensional vector. That is the role of the 32 Boolean variables of the form $B[j]$, for $1 \leq j \leq 2n$; intuitively, variables $B[1], B[2], \ldots, B[16]$ correspond to variables $K[1, 1], K[2, 1], \ldots, K[16, 1]$ and variables $B[17], B[18], \ldots, B[32]$ correspond to variables $K[1, 2], K[2, 2], \ldots, K[16, 2]$. Once the variables are mapped, it is relatively easy to compute the corresponding energy matrix by finding the cells to which each component of the matrix refers to, computing their distances, and the associated energy level. The energy matrix is a 32×32 real matrix whose components range in interval $[-1.00, +1.00]$, where the lower bound corresponds to the closest cells and the upper bound corresponds to the most distant cells in terms of their feature vectors. The second sub-step finds the values of the B variables using a quantum computer, then maps them onto the corresponding K variables, and outputs a clustering, which is a Boolean matrix that indicates the cluster to which each cell belongs.

3.3.2 Finding the function of each cluster

The idea is to use some heuristics that help determine which cluster has the label cells and which one has the data cells.

The first sub-step consists in removing the noise in the clustering that was computed before. Our experiments prove that the clusterings are not generally perfect because they are totally independent from the actual layout of the input tables. We decided to include the row and the column of the cells as additional intra-cell features, which definitely helped in the process, but it is not generally expected to produce perfect clusterings because there are cases in which a few label or data cells have features that deviate largely from the other cells of the same kind. Thus, it is necessary to resort to a heuristic to remove the noise from the resulting clusterings and the majority vote has proven to work very well in our context. That is, if there is a subset of cells with a given function in a row/column in which the majority of cells have the opposite function, we then correct the function of that minority of cells. In case of ties, the majority vote around the cells is used. If the ties persists, then a random choice is made. This approach proved to correct the noise very well.

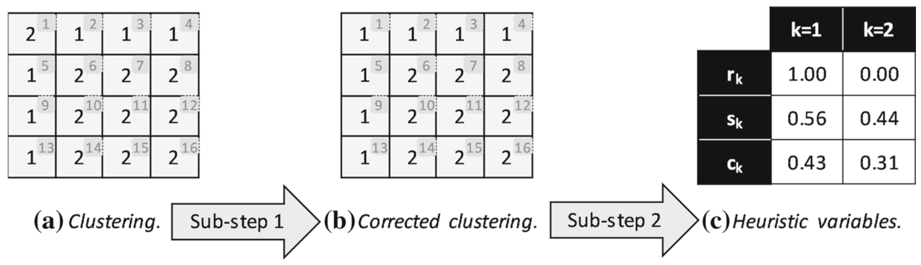
The second sub-step determines which cluster corresponds to label cells and which one corresponds to value cells. We use a heuristic that builds on the following variables:

$$\begin{aligned} r_k &= {}^1\!/_2\,({}^{a_k}\!/_n + {}^{b_k}\!/_m), \\ s_k &= 1 - {}^{|K_k|}\!/_{m\,n}, \text{ and} \\ c_k &= 1 - \sum_{d \in D_k} {}^{d}\!/_{(|K_k| \max D_k)}, \end{aligned}$$

where $k \in \{1, 2\}$, m and n denote the number of rows and columns, a_k and b_k denote the number of cells in the first row or column that are in cluster K_k, and D_k is a set with the distance of the cells in cluster K_k to the top-left position in the table. Set D_k is defined as follows: $D_k = \{\sqrt{(p-1)^2 + (q-1)^2} \mid \langle p, q\rangle \in K_k\}$, where notation $\langle p, q\rangle \in K_k$ gets the indices of the cells from the feature vectors in cluster K_k.

Variable r_k measures the ratio of cells in cluster K_k that are in the first row and column; thus, the higher r_k, the higher the chances that cluster K_k consists of label cells since such cells are typically placed in the first row or column ($k \in \{1, 2\}$). Variable s_k measures the one-complement of the relative size of cluster K_k; thus, the higher s_k, the higher the chances that cluster K_k consists of label cells since these cells are typically a minority ($k \in \{1, 2\}$); finally, variable c_k measures the one-complement of the average normalised distance of cluster K_k to the top-left corner of a table; thus, the smaller c_k, the higher the chances that cluster K_k consists of label cells since such cells are typically near the top-left corner of the tables ($k \in \{1, 2\}$).

Our heuristic is now straightforward using the previous formulation: we assume that the label cells are in cluster K_1 and the value cells in cluster K_2 if ${}^1\!/_3\,(r_1 + s_1 + c_1) \geq$



	k=1	k=2
r_k	1.00	0.00
s_k	0.56	0.44
c_k	0.43	0.31

Fig. 5 Running example: finding functions

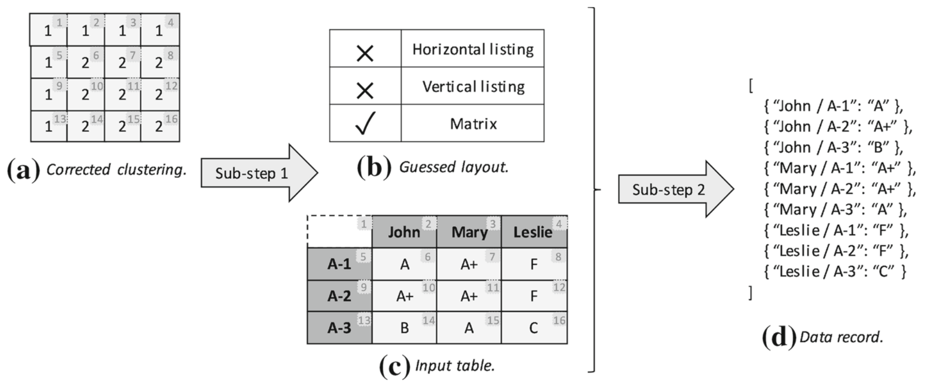
$1/3\ (r_2 + s_2 + c_2)$; otherwise, we assume that the value cells are in cluster K_1 and the label cells are in cluster K_2.

Example 4 Figure 5 illustrates how the function of the clusters is found. The first sub-step cleans the input clustering. Note that it identifies the empty cell at the upper left corner as belonging to cluster K_2. This cell deviates largely from the others both in style and contents, but the majority of cells in its row and column were identified as belonging to cluster K_1. Thus, it makes sense to correct the clustering and flag that cell as belonging to cluster K_1. The second sub-step computes the r_k, s_k, and c_k heuristic variables ($k \in \{1, 2\}$) in order to determine the functions of the cells in each cluster. The r_k variables measure the proportion of cells in each cluster that are on the first row and/or column ($k \in \{1, 2\}$); realise that $r_1 = 1.00$ and $r_2 = 0.00$ because all of the cells of cluster K_1 are in the first row and column, which provides a strong indication that the cells in cluster K_1 are label cells. The s_k variables measure the one-complement of the relative size of the clusters ($k \in \{1, 2\}$); realise that $s_1 = 0.56$ and $s_2 = 0.44$, which indicates that cluster K_1 has the minority of cells and is thus likely to contain the label cells. The c_k variables somewhat measure the distance of cells to the top-left corner ($k \in \{1, 2\}$); realise that $c_1 = 0.43$ and $c_2 = 0.31$, which again indicates that cluster K_1 is the closest to the upper-left corner and then more likely to have label cells. Summing up: the three heuristic variables indicate that cluster K_1 is likely to provide the label cells and cluster K_2 is then likely to provide the data cells.

3.4 Step 3: post-processing

This step implements the structural analysis and the interpretation tasks in the pipeline. It works on an input table, the collection of feature vectors computed in the first step, and the clustering computed in the second step; it returns a set of records that provide the data in the input table in a structured format that is amenable for further processing.

The first sub-step classifies the input table as a horizontal listing, a vertical listing, or a matrix. This is simple in the case in which none of the clusters in the input clustering is empty; it is a bit more involved in the other case. Sometimes, the input table does not have any label cells because the semantics are implicit in the context or the data themselves. These tables result in a clustering in which one of the clusters is empty. In such cases, we guess the orientation according to the group of inter-cell features whose average deviation is the smallest one, namely: it is horizontally-oriented if the average of the per-column inter-cell features is the smallest one; it is vertically-oriented, if the average of the per-row inter-cell features is the smallest one; otherwise, it is both horizontally- and vertically-oriented if the smallest average corresponds to the per-table inter-cell features. Our proposal is to add an



	John	Mary	Leslie
A-1	A	A+	F
A-2	A+	A+	F
A-3	B	A	C

```
[
  { "John / A-1": "A" },
  { "John / A-2": "A+" },
  { "John / A-3": "B" },
  { "Mary / A-1": "A+" },
  { "Mary / A-2": "A+" },
  { "Mary / A-3": "A" },
  { "Leslie / A-1": "F" },
  { "Leslie / A-2": "F" },
  { "Leslie / A-3": "C" }
]
```

Fig. 6 Running example: post-processing

artificial row and/or column of computer-generated label cells before the first actual row and/or column in the input table. Classifying the input table is now straightforward: 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; if both the topmost and the leftmost columns have label cells, then it is a matrix.

The second sub-step consists in creating the data records themselves. If the input table is a horizontal/vertical listing, then the label cells in the first few rows/columns index the data on a per-column/per-row basis. In this case, the records are generated per row/column by catenating the contents of the label cells vertically/horizontally to form a header and mapping them onto the contents of the corresponding value cells. In cases in which the catenation of the labels results in two identical headers, we disambiguate them by means of a sequential index; in cases in which two consecutive labels are the same, one of them is ignored since this is typically the result of replicating a spanned cell. 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 as we mentioned before.

Example 5 Figure 6 illustrates how the records of the original table are computed. The first sub-step takes the input clustering and guesses that the input table is a matrix because it has label cells in both the first row and column. The second sub-step then creates a data record in which the headers provided by the label cells are mapped onto the values provided by the corresponding data cells. Note that we used the JSON notation to represent the data records and a slash to separate the headers, but this is completely customisable in our implementation.

4 Complexity analysis

In this section, we analyse the complexity of our proposal. Our goal is not to characterise its exact time complexity, but an upper bound that makes it clear that it can break the NP complexity that is inherent to finding an optimal clustering using a standard computer. For instance: we use the number of DOM nodes in a document as an upper bound to the number of cells in any of its tables, which is perfectly valid, but very likely far above the actual figure. We first present three supporting lemmata and then the theorem that analyses the complexity of LuperQ; there is a corollary that instantiates the previous complexity result in the context of the adiabatic quantum computer used in our experimentation. Below, we use ν to denote the number of DOM nodes in the input document and κ to denote the number of intra-cell features; α and β denote two constants that depend on the adiabatic computer used.

Lemma 1 (Step 1: Pre-processing) *This step does not require more than $O(\nu\,\kappa)$ time to process an input document.*

Proof The pre-processing step involves the following operations: a) locating the tables: this involves parsing, rendering, and selecting the table nodes; these operations can be implemented using industrial components that are not expected to require more than $O(\nu)$ time. b) Discriminating the tables: this requires to check a few conditions on a subset of DOM nodes; thus, this operation may not require more than $O(\nu)$ time. c) Segmenting the tables, which requires to perform some simple operations on their cells; then, this operation may not require more than $O(\nu)$ time. d) Computing the features; this operation requires to normalise the attributes (which may not require more than $O(\nu\,\kappa)$ time), to compute the intra-cell features (which may not require more than $O(\nu\,\kappa)$ time), and to compute the inter-cell features (which may not require more than $O(\nu\,\kappa)$ time to compute the intra-cell feature averages plus $O(3\,\nu\,\kappa)$ time to compute the deviations). Summing up: the pre-processing step may not require more than $O(3\,\nu + 3\,\nu\,\kappa + 3\,\nu\,\kappa) \subseteq O(\nu\,\kappa)$ time. □

Lemma 2 (Step 2: Identifying cell functions) *This step does not require more than $O(\nu^2\,\kappa + e^{\alpha\nu\beta})$ time to process a table.*

Proof This step involves the following operations: a) clustering the cells: this operation requires to compute the distance between every two cells in the input table and the corresponding energy to preserve the entropy of the dataset; since there cannot be more than ν cells, then this operation may not require more than $O(\nu^2\,\kappa)$ time (note that the time to compute the distance amongst any two κ-dimensional vectors is not expected to require more than $O(\kappa)$ time, but computing the associated energy level involves computing a real-valued function on a distance only, which may be safely assumed to require $O(1)$ time); finding the optimal clustering requires a time in the order of $O(e^{\alpha\nu\beta})$ [28], where α and β are constants that depend on the quantum computer used. b) Correcting the clustering: this operation may not require more than $O(\nu)$ time to compute the majority vote per row/column plus $O(\nu)$ time to correct the function of each cell. c) Finding the function of each cluster: this time is dominated by the computation of variables c_i, which requires to iterate over every pair of cells; that is, it may not require more than $O(\nu^2)$ time. Summing up: identifying cell functions may not require more than $O(\nu^2\,\kappa + e^{\alpha\nu\beta} + 2\,\nu + \nu^2) \subseteq O(\nu^2\,\kappa + e^{\alpha\nu\beta})$ time.

□

Lemma 3 (Step 3: Post-processing) *This step does not require more than $O(\nu\,\kappa)$ time to process a table.*

Proof This step involves the following operations: a) Classifying the input table: if none of the clusters identified is empty, then the operation must determine if the label cells are arranged at the topmost rows, the leftmost columns or both, which may not require more than $O(\nu)$ time; otherwise, it must check the coherency of the rows and columns, which may not require more than $O(3\,\nu\,\kappa)$ time to iterate over the cells and compute the average deviation of their features; since $O(\nu) \subseteq O(3\,\nu\,\kappa)$, then classifying the input table is not expected to take more than $O(3\,\nu\,\kappa)$ time. b) Computing the records: this operation iterates over the cells and creates the records by associating headers to the value cells, which may not require more than $O(\nu)$ time. Summing up: the post-processing step may not require more than $O(3\,\nu\,\kappa + \nu) \subseteq O(\nu\,\kappa)$ time. □

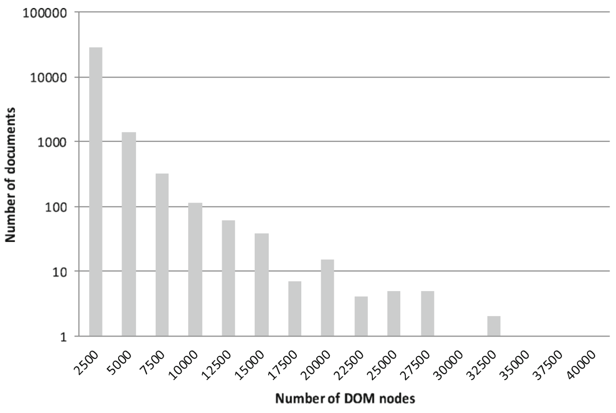


Fig. 7 Distribution of number of DOM nodes

Theorem 1 (Complexity of LuperQ) *Our proposal does not require more than* $O(\nu^2 + e^{\alpha\nu\beta})$ *time to process a table.*

Proof According to the previous lemmata, LuperQ does not take more than $O(\nu\,\kappa + \nu^2\,\kappa + e^{\alpha\nu\beta} + \nu\,\kappa)$ time to process a table. Realise that ν is generally expected to be larger than κ since typical HTML documents have thousands of DOM nodes, whereas the total number of features is comparatively smaller. Figure 7 shows the distribution of the number of DOM nodes in the HTML documents in our experimental repositories; the average is 973.97 DOM nodes, with a minimum of 6 nodes and a maximum of 39,552; note that the number of features is a constant in our experiments, cf. Table 1. Thus, the previous upper bound is a subset of $O(\nu + \nu^2 + e^{\alpha\nu\beta} + \nu) \subseteq O(\nu^2 + e^{\alpha\nu\beta})$. Recall that α and β are constants that depend on the quantum computer used [28], so the actual time complexity depends completely on it. □

Corollary 1 (Our implementation) *We implemented LuperQ on a hybrid system: we used a standard computer to run the pre-processing step, the computation of the QUBO problem, and the post-processing step; we used an adiabatic quantum computer to solve the QUBO problem. The quantum computer was a D-Wave 2000Q system, which allows to configure the annealing time from 1 up to 2000* μ*s. We used the default value of 20* μ*s, as suggested by the manufacturer, which means that* $e^{\alpha\nu\beta} \leq 20\ \mu s$*. Unfortunately, this does not imply that we can solve every clustering problem in a maximum of 20* μ*s because there are other factors that have an impact on the total time. These factors include the time required to set the computer up, which usually requires some milliseconds, and the time that the computer must be idle in between two consecutive problems, which is in the order of the annealing time. Furthermore, the number of qubits available is 2048 and they are organised in a Chimera network that allows for a maximum of 65 fully-connected qubits. This implies that every QUBO problem must be mapped onto the available qubits taking their physical connections into account; furthermore, some large problems must be partitioned into smaller problems whose solutions must be combined. The previous problems can be addressed using a standard computer thanks to D-Wave's qsolver algorithm [6]. There is not a complexity analysis available, but the results of the performance studies that were carried out on complex, large synthetic problems show that the delay ranges in the order of seconds. Our experiments confirm the previous results since most of the problems were mapped, partitioned, and solved in a matter of seconds* *or less. Summing up: we can safely conclude that $O(v^2)$ is a sensible characterisation of the upper limit to the amount of time that LuperQ requires to process a table when it is implemented using the D-Wave 2000Q system.*

Note 1 In the previous discussion, we intentionally omitted the time that a problem must wait for D-Wave's computer to become available. The D-Wave 2000Q system can only work on a problem at a time, which means that the problems that are submitted to it must wait in a queue until the previous problems are solved. In our experience, the waiting time ranged from an instant to up to 10–15 s. It was not considered because it is not inherent to our proposal, but something that derives from the fact that the computer is shared by many scientists around the world.

5 Experimental analysis

In this section, we first describe our experimental setup, next describe how we configured our proposal and the competitors, then present and analyse the results of our empirical comparison, and, finally, report on the results attained using other clustering approaches. The experimental repositories as well as the implementation of our proposal and the competitors are available at http://dx.doi.org/10.17632/g4tz8m5p6k.2 so that other scientists can repeat our results and work on new proposals. The tool that we used to create the ground truth is available at http://tomatera.tdg-seville.info.

5.1 Experimental setup

We implemented our proposal using Python 3.7.6 and many ancillary components. We used BeautifulSoup 4.9.0 to parse the HTML documents; we computed their attributes by rendering them on a virtual canvas using the Selenium 3.141.0 headless browser with Firefox 80.0.1 and Geckodriver 0.27.0. We used SciKit Learn 0.24.3 to leverage the implementation of some common distance functions and to compute effectiveness measures, Pandas 1.0.3 to implement the datasets, and NumPy 1.18.2 to implement some vector and matrix operations. The Ocean 2.2.0 toolkit was used to map QUBO problems onto the quantum computer used. The statistical tests were performed using the SCMAMP 0.2.55 library from the R project.

We confronted it with the proposals by Yoshida et al. [69], Jung and Kwon [36], and Embley et al. [21], which are well-known unsupervised techniques, as well as Nishida et al.'s [51] proposal, which is the most advanced supervised proposal of which we are aware. The unsupervised proposals rely on algorithmic approaches that build on the Expectation Maximisation method, custom heuristics, or a custom search algorithm, respectively; the supervised one relies on a deep neural network.

The experiments were run on a hybrid system that consisted of a standard computer and a quantum computer. The former was a commodity Windows 10 computer that was equipped with an AMD Ryzen 7 2700X processor with eight two-threaded cores and 16 GiB of DDR4 RAM memory; the latter was a D-Wave 2000Q computer that provides 2048 qubits that are connected using a Chimera network.

We assembled two experimental repositories with data tables from the Wikipedia [63] and the Dresden Web Table Corpus (DWTC) [19]. They were annotated by four independent judges who achieved a Krippendorff Alpha coefficient as high as 96.11% on a random subset of 400 tables. The high degree of inter-agreement makes the previous repositories a good ground truth to perform experimentation. The Wikipedia repository provides a total of 1353 tables that are distributed as follows: 1089 horizontal listings, 75 vertical listings, and 189 matrices; they have 90 717 value cells and 8734 label cells. The Dresden repository provides a total of 1191 tables that are distributed as follows: 709 horizontal listings, 424 vertical listings, and 58 matrices; they have 49 820 value cells and 6294 label cells.

We computed the F_1 score and the Accuracy score as the effectiveness measures and the prediction time as the efficiency measure. The F_1 score was computed as the one-harmonic mean of precision and recall. (Precision was computed as the ratio of the number of cells that are correctly predicted to be label/value cells to the total number of cells that are predicted as label/value cells; recall was computed as the ratio of the number of cells that are correctly predicted to be label/value cells to the actual number of label/value cells). We computed the F_1 score at the class level and globally; in the former case, the score was computed regarding the value cells and the label cells independently and they were then averaged independently, too; in the latter case, it was computed by averaging all of the previous results using the number of cells of each kind as their relative weights. In both cases, the goal was to measure the extent to which the imbalance of value cells (the majority kind of cells) and label cells (the minority kind of cells) may have an impact on the results. We also measured the Accuracy score, which is defined as the ratio of true positives plus true negatives to the total number of cells. Note that it was computed globally since it coincides with the per-class score in the case of binary problems like ours in which there are only two kinds of data. Both measures provide a good overview of how good a proposal is at classifying cells properly: the F_1 score provides a view in terms of precision and recall, whereas the Accuracy score provides a global view in terms of how good it is at not producing wrong predictions.

We collected the prediction time as the efficiency measure. It was computed as the average number of CPU seconds required to predict the function of the cells in a table; note that CPU time was used instead of wall time because it is far more stable across different experimentations and discards the IO time required to load the datasets or the time consumed by other concurrent processes. In the case of the supervised proposal, the time required to learn the model was apportioned across all of the tables.

We used the well-known threefold cross validation method to strengthen the validity of our results. The experimental repositories were partitioned into three equal-size random splits, all of which were used to compute the performance measures; the other two were used for learning purposes or ignored depending on the proposal. That is: all of the experimental data were used for learning purposes (when necessary) and for validation.

To confirm that the experimental ranks computed were sound, we performed hypothesis testing [24,56], namely: we first computed the empirical ranking for each performance measure; we then performed Hommel's test to compare the best-ranking proposal according to the empirical ranking to the others. The differences in rank are assumed to be statistically significant if the resulting p-value is smaller than the standard significance level ($\alpha = 0.05$) and not significant otherwise.

5.2 Configuring the proposals

Our proposal has two hyper-parameters whose values must be set before it is confronted with other proposals, namely: the function used to compute the distance between the cells and the function used to transform the distances into energy levels.

The literature provides many proposals to implement the distance function [17]. We selected the distances that can be applied to multi-dimensional real data for which Scikit-

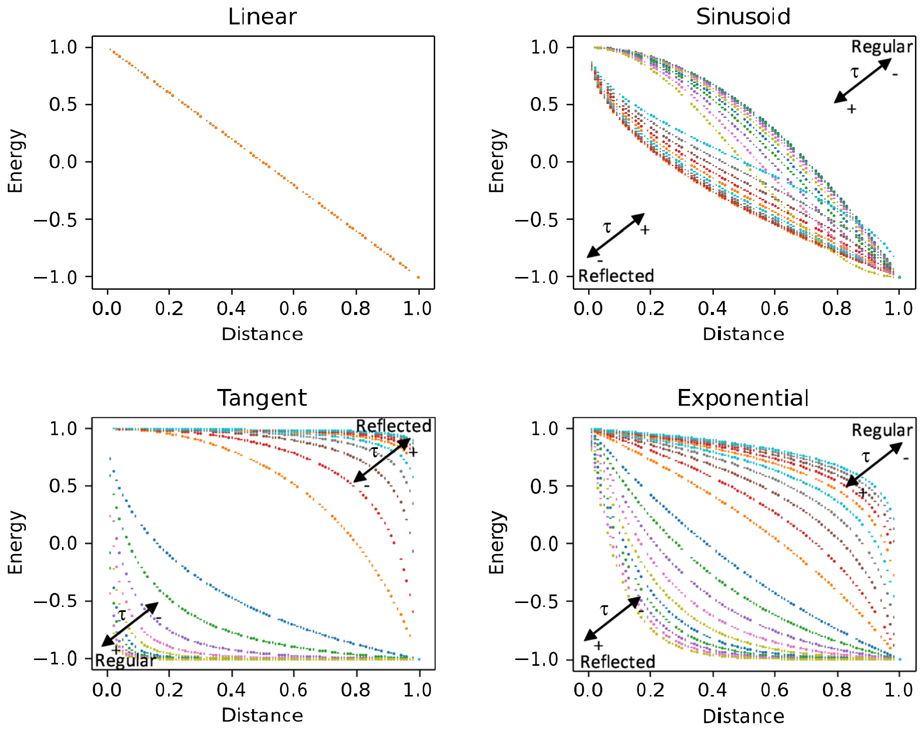


Fig. 8 Energy functions

Learn provides an implementation, namely: Euclidean, Standardized Euclidean, Squared Euclidean, Correlation, L1, L2, Cosine, Bray-Curtis, Chebyshev, Manhattan, and Canberra.

The literature also provides many proposals to implement the energy function [2,29]. We experimented with the following ones:

$$\begin{aligned}
\text{linear}_\tau(d) &= 1.00 - 2.00\,d,\\
\text{sinusoid}_\tau(d) &= \cos(\tau\,\pi\,d),\\
\text{tangent}_\tau(d) &= 1.00 - \tanh(\tau\,\sqrt{d}),\text{ and}\\
\text{exponential}_\tau(d) &= 2.00 - \cosh(-\tau\,\sqrt[2\tau]{d})
\end{aligned}$$

In the previous formulation, τ denotes a parameter that allows to compute the specific functions in each family (The linear case is a baseline in which τ is ignored). We experimented with both the regular formulations above and their reflections. The results of the functions were normalised to interval $[-1.00, +1.00]$ using the following formula:

$$\mathcal{N}(f, d) = 2.00\,\frac{f(d) - f(1.00)}{f(0.00) - f(1.00)} - 1.00,$$

where f denotes an energy function and d denotes a distance. Figure 8 illustrates the normalised energy functions. The illustration makes it explicit which bunch of curves correspond to the regular or reflected functions and also whether the inner curves correspond to increasing values of the τ parameter or vice versa.

Table 2 Grid search (energy function)

Energy	Tau	F1	Time	Energy	Tau	F1	Time
Regular linear	N/A	0.33	0.70	Reflected linear	N/A	0.67	0.75
Regular sinusoid	0.10	0.39	0.68	Reflected sinusoid	0.10	0.70	0.75
	0.20	0.36	0.72		0.20	0.71	0.78
	0.30	0.36	0.71		0.30	0.71	0.78
	0.40	0.36	0.72		0.40	0.71	0.78
	0.50	0.36	0.72		0.50	0.71	0.77
	0.60	0.34	0.70		0.60	0.83	0.72
	0.70	0.34	0.70		**0.70**	**0.83**	**0.71**
	0.80	0.33	0.70		0.80	0.79	0.73
	0.90	0.36	0.71		0.90	0.77	0.73
	1.00	0.37	0.70		1.00	0.74	0.73
Regular tangent	1.00	0.33	0.69	Reflected tangent	1.00	0.70	0.73
	2.00	0.32	0.73		2.00	–	–
	3.00	0.32	0.73		3.00	–	–
	4.00	0.32	0.73		4.00	–	–
	5.00	0.31	0.73		5.00	–	–
	6.00	0.27	0.72		6.00	–	–
	7.00	0.27	0.73		7.00	–	–
	8.00	0.27	0.72		8.00	–	–
	9.00	0.26	0.72		9.00	–	–
	10.00	0.26	0.71		10.00	–	–
Regular exponential	1.00	0.33	0.69	Reflected exponential	1.00	0.70	0.76
	2.00	0.33	0.71		2.00	0.70	0.85
	3.00	0.33	0.70		3.00	0.70	0.84
	4.00	0.33	0.69		4.00	0.70	0.84
	5.00	0.33	0.70		5.00	0.70	0.84
	6.00	0.32	0.71		6.00	–	–
	7.00	0.32	0.72		7.00	–	–
	8.00	0.32	0.73		8.00	–	–
	9.00	0.32	0.72		9.00	–	–
	10.00	0.32	0.73		10.00	–	–

We randomly selected 100 tables from our repositories and performed a stratified grid search on the hyper-parameter space. We first set the distance function to the standard Euclidean distance and experimented with many different energy functions, cf. Table 2. The best results in terms of the F_1 score were attained using the reflected sinusoid with parameter $\tau = 0.60$ or $\tau = 0.70$; we took the second choice because the time was smaller in that case. Then, we experimented with the distance functions, cf. Table 3. The best results in terms of the F_1 score were attained using the correlation distance function.

We configured the competitors using the guidelines provided by the authors [21,36,51,69]. Unfortunately, Yoshida et al.'s [69] guideline was incomplete, so we made some decisions that are in accordance with the common practices in the literature, namely: we initialised the

Table 3 Grid search (distance function)

Distance	F_1	Time
Euclidean	0.83	0.74
Squared Euclidean	0.80	0.75
Correlation	**0.84**	**0.75**
L1	0.76	0.73
L2	0.76	0.75
Cosine	0.74	0.74
Bray-Curtis	0.77	0.72
Chebyshev	0.79	0.75
Manhattan	0.80	0.77
Canberra	0.78	0.77

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.

5.3 Empirical comparison

Figure 9 shows our experimental results using boxplots that help understand their distribution; note that the chart regarding the prediction times is in 10-logarithmic scale because this measure ranges in very different intervals depending on the proposal. Table 4 shows the results of our statistical analysis using tables with the following columns: the first column provides the name of a proposal; the second column presents its empirical rank; the third column shows the average of several performance measures plus/minus their corresponding standard deviations; the last column shows the p-values computed by Hommel's test when comparing the best ranking proposal to the others (the cells in boldface indicate significant differences in rank; the cells with “N/A” correspond to the comparison of the best-ranking proposal to itself).

Regarding the F_1 score on the value cells, LuperQ attains the highest average and median and has the smallest and highest inter-quartile range; it is closely followed by the proposals by Nishida et al. [51], Jung and Kwon [36], Yoshida et al. [69], and Embley et al. [21]. All of the proposals attain relatively high F_1 scores since the first quartile is above 0.65 and the median is above 0.80 in all cases. This is not surprising at all because value cells constitute the majority of cells in our experimental repositories. It is regarding its ability to identify label cells that LuperQ shines when compared to the other proposals. All of them attain F_1 scores in the full range, but LuperQ is the one with the highest average and median and the smallest and highest inter-quartile range; note that 75% of the distribution of the F_1 score on the label cells is above 0.75 (the first quartile and the median coincides in this distribution) and the average is a bit above 0.80. It is followed by the proposals by Nishida et al. [51], Embley et al. [21], Jung and Kwon [36], and Yoshida et al. [69]. Regarding the global F_1 score LuperQ seems to rank the first one since it attains the highest average and median and has the smallest and highest inter-quartile range; it is followed by the proposals by Nishida et al. [51], Embley et al. [21], Jung and Kwon [36], and Yoshida et al. [69]. Note that the Accuracy score reflects the imbalance much better. According to it, LuperQ ranks the first because it has the highest average and median and the smallest and highest inter-quartile range again; it is closely followed by the proposal by Nishida et al. [51]; then come the

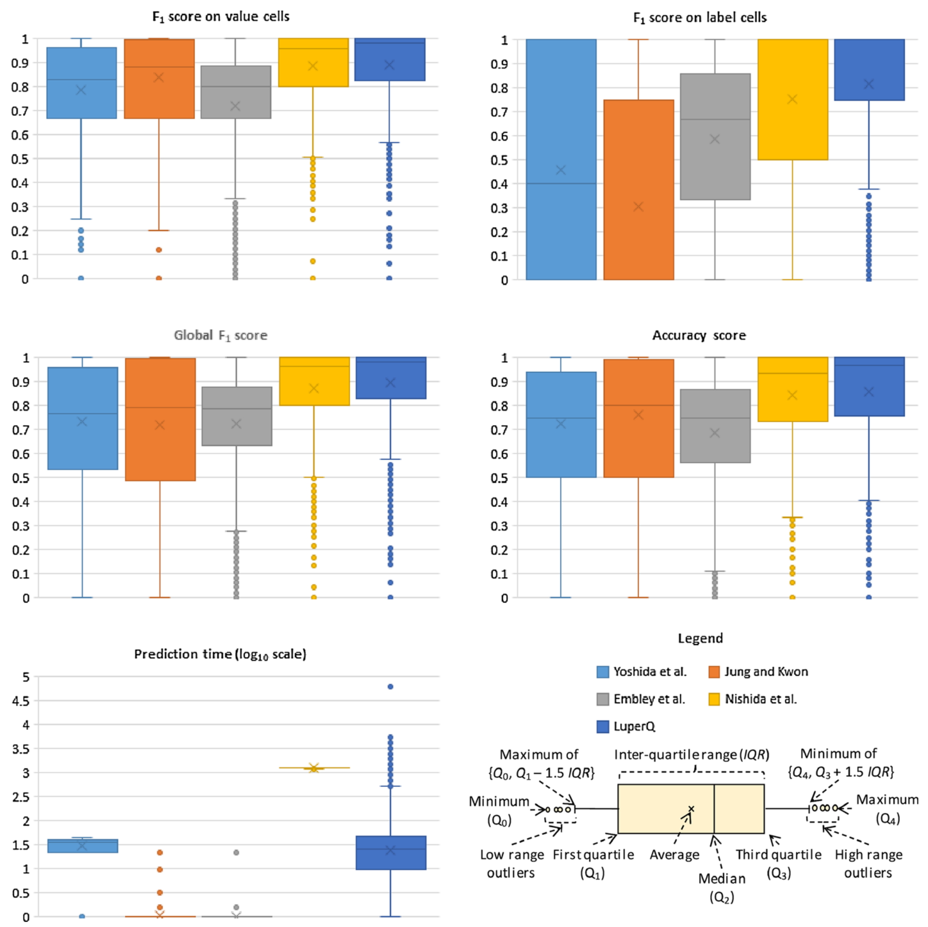


Fig. 9 Comparison of competitors

proposals by Embley et al. [21] and Jung and Kwon [36]; finally, comes the proposal by Yoshida et al. [69], which has the smallest average and median and the largest inter-quartile range.

Regarding the prediction time, Embley et al.'s [21] and Jung and Kwon's [36] proposals are clearly the fastest ones since they were able to process all of the tables in our experimental repositories in a few hundredths of a CPU second. Unfortunately, their effectiveness is not very good, which means that their extra efficiency is not so appealing in a real-world context. LuperQ ranks at the third position and it is followed by Yoshida et al.'s [69] proposal and Nishida et al.'s [51] proposal, which is clearly the most inefficient one due to its deep neural approach.

The previous intuitive conclusions regarding performance were confirmed by our statistical analysis. Note that LuperQ attains the first position in the empirical ranks regarding the effectiveness measures. In the case of the class-level F_1 scores, the p-values returned by Hommel's test confirm that the differences in rank are statistically significant when comparing LuperQ to every other competitor, but Nishida et al.'s [51] proposal. In the case of the global F_1 score or the Accuracy score, the p-value computed by Hommel's test is nearly zero in

Table 4 Statistical analysis of competitors

Competitor	Rank	F_1 (Value)	Hommel	Competitor	Rank	F_1 (Label)	Hommel
Yoshida et al.	3.38	0.79 ± 0.20	**0.00E+00**	Yoshida et al.	2.92	0.79 ± 0.20	**0.00E+00**
Jung and Kwon	3.04	0.84 ± 0.16	**0.00E+00**	Jung and Kwon	4.10	0.31 ± 0.43	**0.00E+00**
Embley et al.	3.84	0.72 ± 0.24	**0.00E+00**	Embley et al.	3.58	0.59 ± 0.32	**0.00E+00**
Nishida et al.	2.40	0.89 ± 0.15	1.28E-01	Nishida et al.	2.23	0.89 ± 0.16	4.78E-01
LuperQ	2.33	0.89 ± 0.16	N/A	LuperQ	2.16	0.82 ± 0.33	N/A
Competitor	Rank	F_1 (Global)	Hommel	Competitor	Rank	Accuracy	Hommel
Yoshida et al.	3.33	0.74 ± 0.24	**0.00E+00**	Yoshida et al.	3.35	0.73 ± 0.23	**0.00E+00**
Jung and Kwon	3.38	0.72 ± 0.27	**0.00E+00**	Jung and Kwon	3.18	0.77 ± 0.22	**0.00E+00**
Embley et al.	3.66	0.73 ± 0.23	**0.00E+00**	Embley et al.	3.77	0.69 ± 0.25	**0.00E+00**
Nishida et al.	2.39	0.88 ± 0.18	**2.49E-04**	Nishida et al.	2.43	0.85 ± 0.19	**5.86E-04**
LuperQ	2.25	0.90 ± 0.16	N/A	LuperQ	2.27	0.86 ± 0.20	N/A
Competitor	Rank	Pred. Time	Hommel				
Yoshida et al.	3.76	0.32 ± 0.10	**0.00E+00**				
Jung and Kwon	1.55	0.01 ± 0.01	**0.00E+00**				
Embley et al.	1.46	0.01 ± 0.01	N/A				
Nishida et al.	4.99	12.41 ± 0.16	**0.00E+00**				
LuperQ	3.25	1.07 ± 13.78	**0.00E+00**				

all cases, which is a strong indication that the differences in rank are statistically significant. Therefore, the conclusion supported by the statistical analysis is that LuperQ ranks the first regarding effectiveness, even though its results are statistically indistinguishable from the proposal by Nishida et al. [51] at the class level. This result is very important since it proves that LuperQ can be as effective as the state-of-the-art proposal, but in a totally unsupervised manner. Regarding the efficiency measure, note that Hommel's test returns a zero p-value for every comparison, which means that the differences in rank are then statistically significant.

Clearly, all of the proposals have difficulties to identify label cells; generally speaking, the greater the imbalance, the more difficulties to identify them properly. Yoshida et al.'s [69] proposal seems to be the one that has more difficulties to deal the imbalance. It did not attain very good results because it builds on the assumption that the label cells usually have the same common contents across different tables, which is a difficult-to-meet assumption when working at Web scale; furthermore, its frequency-based approach also has problems when processing cells with numeric contents, since many of them occur only once in the datasets. Neither was Jung and Kwon's [36] proposal very good at dealing with the imbalance. It seems to have trouble with tables in which the user highlights some cells that provide data using a style that is similar to the style of the label cells. That was somewhat common in our experimental repositories because we have found out that many authors use the th HTML tag to highlight some data cells. Embley et al.'s [21] proposal is very fast, but has many problems when dealing with listings; we found that tables with no repeated contents are interpreted as matrices with one row and one column of headers, which is not usually true. The proposal was devised to work in the context of spreadsheets, where we assume that matrices are far more common than in the Web. Nishida et al.'s [51] proposal achieves very good results because it is supervised and learns patterns from a training set in which a person must provide as many tables as possible with annotations regarding the function of their cells; unfortunately, it tends to get in trouble when it is confronted with a table that deviates largely from the tables in the training set. It can deal with label cells much better than the previous proposals, but the imbalance of the datasets still has a negative impact on its results. We found out that the cases in which LuperQ cannot identify them properly correspond to tables in which the difference amongst the label and the data cells is very vague, many of which are matrices from which it is not easy to extract data unless one can really understand their semantics. In some cases, the problem was that the ratio of label to data cells was heavily skewed in large tables with less than 1% label cells. We also found out that LuperQ sometimes had trouble with horizontal listings in which the first columns are somewhat highlighted; in such cases, it tends to mistake horizontal listings for matrices. We profiled LuperQ and we found that the cases in which it requires more than 1.00 second correspond to cases in which the algorithm to map the QUBO problems onto the quantum computer had difficulties to find the optimum mapping [6].

5.4 Replacing the clusterer

We set up several variants of LuperQ in which we replaced the component that implements our quantum clustering approach by the classical k-means algorithm with the mini-batch extension to deal with large datasets, as well as the quantum approaches by Neukart et al. [50], Bauckhage et al. [4], Kumar et al. [44], and Wereszczyński et al. [62]. Figure 10 and Table 5 summarise our results.

Regarding the F_1 score on the value cells, LuperQ attains the highest average value and median and it has the smallest inter-quartile range, which indicates that it ranks the first. It

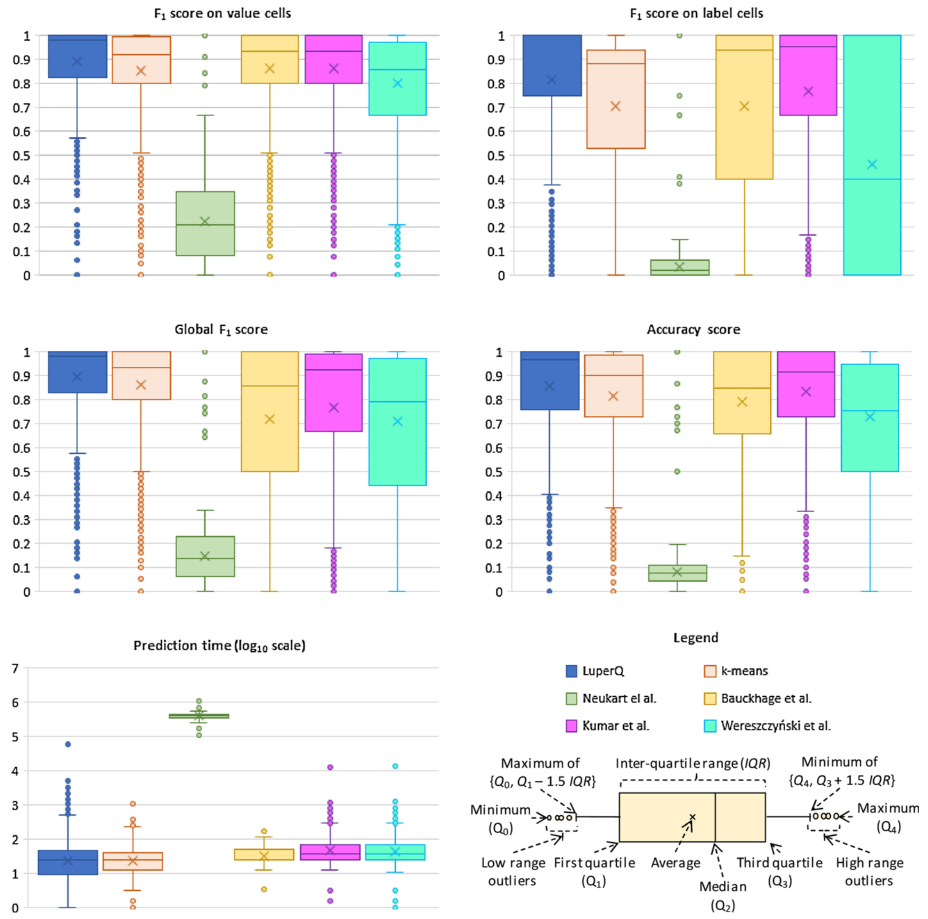


Fig. 10 Comparison using other clusterers

is followed by the variant that uses k-means, Bauckhage et al.'s [4] clusterer, and Kumar et al.'s [44] clusterer; then comes the variant that uses Wereszczyński et al.s [62] clusterer; the variant that uses Neukart et al.'s [50] seems to clearly deviate from the others since it attains the worst average and median and has the lowest inter-quartile range. The results regarding the F_1 score on the label cells makes the variants further apart: the results regarding LuperQ are very similar in both cases because it has proven to deal well with the imbalance of value and label cells. The other variants are clearly worse at identifying the label cells since they are the minority in our experimental repositories. The global F_1 score makes the differences between LuperQ and its variants very clear in the case of the variants that use quantum approaches to cluster the cells since LuperQ attains the highest average and median score and has the smallest and highest inter-quartile range; it is followed by the variant that uses k-means. The Accuracy score also makes a clear difference between LuperQ and the variants that use Neukart et al.'s [50] or Wereszczyński et al.'s [62] clusterers; it is followed by the variants that use k-means, Kumar et al.'s [44] clusterer, and Bauckhage et al.'s [4] clusterers.

Regarding the prediction time, LuperQ and its variants seem to behave very similarly, except for the variant that uses Neukart et al.'s [50] clusterer. That result was not surprising

Table 5 Statistical analysis of other clusterers

Clusterer	Rank	F_1(Value)	Hommel	Clusterer	Rank	F_1(Label)	Hommel
LuperQ	2.53	0.89 ± 0.16	N/A	LuperQ	2.29	0.82 ± 0.33	N/A
k-Means	3.05	0.85 ± 0.18	**0.00E+00**	k-Means	3.76	0.71 ± 0.33	**0.00E+00**
Neukart et al.	5.91	0.22 ± 0.16	**0.00E+00**	Neukart et al.	5.40	0.03 ± 0.05	**0.00E+00**
Bauckhage et al.	3.18	0.83 ± 0.20	**0.00E+00**	Bauckhage et al.	2.91	0.71 ± 0.35	**0.00E+00**
Kumar at al.	2.91	0.86 ± 0.18	**0.00E+00**	Kumar at al.	2.80	0.77 ± 0.33	**0.00E+00**
Wereszczyński et al.	3.42	0.80 ± 0.19	**0.00E+00**	Wereszczyński et al.	3.84	0.46 ± 0.44	**0.00E+00**
Competitor	Rank	F_1(Global)	Hommel	Competitor	Rank	Accuracy	Hommel
LuperQ	2.57	0.86 ± 0.20	N/A	LuperQ	2.49	0.86 ± 0.20	N/A
k-Means	3.12	0.82 ± 0.21	**0.00E+00**	k-Means	3.03	0.82 ± 0.21	**0.00E+00**
Neukart et al.	5.84	0.15 ± 0.10	**0.00E+00**	Neukart et al.	5.97	0.08 ± 0.06	**0.00E+00**
Bauckhage et al.	3.15	0.79 ± 0.22	**0.00E+00**	Bauckhage et al.	3.13	0.79 ± 0.22	**0.00E+00**
Kumar at al.	3.27	0.77 ± 0.32	**0.00E+00**	Kumar at al.	2.93	0.83 ± 0.20	**0.00E+00**
Wereszczyński et al.	3.05	0.71 ± 0.27	**0.00E+00**	Wereszczyński et al.	3.46	0.73 ± 0.23	**0.00E+00**
Clusterer	Rank	Pred. Time	Hommel				
LuperQ	2.34	1.37 ± 0.61	4.14E-01				
k-Means	1.73	1.37 ± 0.43	N/A				
Neukart et al.	6.00	5.6 ± 0.07	**0.00E+00**				
Bauckhage et al.	4.07	1.69 ± 0.41	**0.00E+00**				
Kumar at al.	3.52	1.66 ± 0.38	**0.00E+00**				
Wereszczyński et al.	3.34	1.63 ± 0.41	**0.00E+00**				

because all of the quantum proposals take approximately the same time to prepare and solve the underlying QUBO problems; the reason why the variant that uses Neukart et al.'s [50] clusterer deviates significantly is that it requires to prepare and solve a different QUBO problem for each datum in the input dataset, that is: each cell is clustered independently from the others by solving its own QUBO problem, which results in extra effort as the number of cells in a table increases. Unfortunately, the extra effort was not worth neither in terms of effectiveness nor efficiency.

The previous conclusions were clearly confirmed by our statistical analysis. Note that Hommel's test returns a zero p-value regarding the comparisons that involve effectiveness measures, which means that the experimental data support the hypothesis that LuperQ ranks at the first position and the differences are statistically significant. It also returns zero p-values for the comparisons regarding the prediction time, except for the case of the comparison with the variant that uses k-means. Summing up: the variant that used LuperQ's original approach to clustering seems to attain the best effectiveness results without degrading efficiency when compared to a variant that builds solely on classical computing algorithms. It is expected that it can clearly beat it in future, when quantum computers provide enough qubits to deal with larger problems than they can do today.

6 Conclusions

This article presents LuperQ, which is a new proposal to extract data from HTML tables. It differentiates from the existing ones in that it relies on an adiabatic quantum clustering approach to identify the function of the cells. This approach has proven to be very effective and efficient in practice and supports the idea that adiabatic quantum computing is a technology that helps solve some classical NP problems in polynomial time. Whether a particular problem can attain such a tremendous speedup or not depends completely on the problem and requires experimentation. Our empirical analysis makes it clear that extracting data from HTML tables is one of the problems that may benefit from this speedup. Future work includes researching how to use a quantum computer to select the most informative features automatically. In our experimentation, we used all of the cells in the input tables and all of the features, but we realised that some of them do not actually contribute to the results in some cases; detecting them automatically might help make our proposal more effective and efficient. Generalising our clustering approach to deal with arbitrary datasets and distributing large problems across several quantum computers will also be paid much attention.

Acknowledgements This work was supported by the Spanish and the Andalusian R&D programmes through Grants TIN2016-75394-R, PID2020-112540RB-C44 (MCIN/AEI/10.13039/501100011033), and P18-RT-1060 (FEDER funds from the EU). The standard computer used to perform the experimentation was partially supported by Dinamic Area, S.L. and the quantum computer was fully supported by D-Wave Systems, Inc.

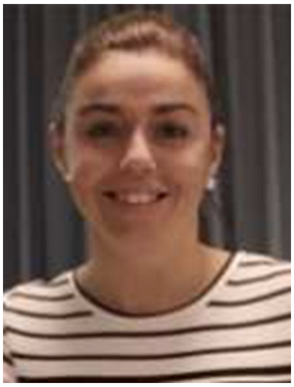
References

1. Aïmeur E, Brassard G, Gambs S (2013) Quantum speed-up for unsupervised learning. Mach Learn 90(2):261–287
2. Banasiak J, Joel LO, Shindin S (2019) Discrete growth-decay-fragmentation equation: well-posedness and long-term dynamics. J Evol Equ 19(2019):771–802
3. Bapst V, Foini L, Krzakala F, Semerjian G, Zamponi F (2013) The quantum adiabatic algorithm applied to random optimization problems. Phys Rep 523(2013):127–205

4. Bauckhage C, Brito E, Cvejoski K, Ojeda C, Sifa R, Wrobel S (2017) Ising models for binary clustering via adiabatic quantum computing. In: EMMCVPR, vol 10746, pp 3–17
5. Bizer C, Meusel R, Primpel A (2019) Web Data Commons: RDFa, Microdata, embedded JSON-LD, and Microformat data sets. Technical report, University of Mannheim. http://webdatacommons.org/structureddata/2019-12/stats/stats.html
6. Booth M, Reinhardt SP, Roy A (2017) Partitioning optimization problems for hybrid classical/quantum execution. Technical report, D-Wave, Inc
7. Braunschweig K, Thiele M, Lehner W (2015) From web tables to concepts: a semantic normalization approach. In: ER, pp 247–260
8. Cafarella MJ, Halevy AY, Zhang Y, Wang DZ, Wu E (2008) Uncovering the relational Web. In: WebDB
9. Cafarella MJ, Halevy AY, Lee H, Madhavan J, Yu C, Wang DZ, Wu E (2018) Ten years of web tables. In: VLDB, vol 11, pp 2140–2149
10. Castelvecchi D (2017) Quantum computers ready to leap out of the lab in 2017. Nature 541(7635):9–10
11. Chakraborty S, Halder S, Kundu S (2016) Design and analysis of a quantum circuit to cluster a set of data points. Adv Signal Process 4(2):7–12
12. Chang C-H, Kayed M, Girgis MR, Shaalan KF (2006) A survey of web information extraction systems. IEEE Trans Knowl Data Eng 18(10):1411–1428
13. Chen H, Tsai S, Tsai J (2000) Mining tables from large scale HTML texts. In: COLING, pp 166–172
14. Costa-Silva A, Jorge AM, Torgo L (2006) Design of an end-to-end method to extract information from tables. IJDAR 8(2–3):144–171
15. Crestan E, Pantel P (2011) Web-scale table census and classification. In: WSDM, pp 545–554
16. Decheng F, Jona S, Panga C, Donga W, Wond CJ (2018) Improved quantum clustering analysis based on the weighted distance and its application. Heliyon 4:1–20
17. Deza MM, Deza E (2016) Encyclopedia of distances, 4th edn. Springer
18. Dunjko V, Taylor JM, Briegel HJ (2016) Quantum-enhanced machine learning. Phys Rev Lett 117(130501):1–6
19. Eberius J, Thiele M, Braunschweig K, Lehner W (2015) Top-*k* entity augmentation using consistent set covering. In: SSDBM, pp 8(1–8), p 12
20. Embley DW, Hurst M, Lopresti DP, Nagy G (2006) Table-processing paradigms: a research survey. IJDAR 8(2–3):66–86
21. Embley DW, Seth SC, Nagy G (2014) Transforming web tables to a relational database. In: ICPR, pp 2781–2786
22. Eslava RVC, Lisboa PJG, Ortega-Martorell S, Jarman IH, Martín-Guerrero JD (2020) Probabilistic quantum clustering. Knowl Based Syst 194:105567
23. Ferrara E, de Meo P, Fiumara G, Baumgartner R (2014) Web data extraction, applications and techniques: a survey. Knowl Based Syst 70:301–323
24. García S, Herrera F (2008) An extension on "Statistical comparisons of classifiers over multiple datasets" for all pair-wise comparisons. J Mach Learn Res 9:2677–2694
25. Gatterbauer W, Bohunsky P, Herzog M, Krüpl B, Pollak B (2007) Towards domain-independent information extraction from web tables. In: WWW, pp 71–80
26. Gibney E (2017) D-Wave upgrade: how scientists are using the world's most controversial quantum computer. Nature 541(7638):447–448
27. Giovannetti V, Lloyd S, Maccone L (2008) Architectures for a quantum random access memory. Phys Rev A 78(5):1–9
28. Griffiths DJ (2004) Introduction to quantum mechanics, 2nd edn. Pearson Prentice Hall
29. Hobbie RK, Roth BJ (2007) Exponential growth and decay. In: Intermediate physics for medicine and biology. Springer, pp 31–47
30. Horn D, Gottlieb A (2002) Algorithm for data clustering in pattern recognition problems based on quantum mechanics. Phys Rev Lett 88(1):1–4
31. Hurst M (2001) Layout and language: challenges for table understanding on the Web. In: WDA
32. Jenssen R (2010) Kernel entropy component analysis. IEEE Trans Pattern Anal Mach Intell 32(5):847–860
33. Jiménez P, Corchuelo R (2016) On learning web information extraction rules with TANGO. Inf Syst 62:74–103
34. Jiménez P, Corchuelo R (2016) Roller: a novel approach to web information extraction. Knowl Int Syst 49(1):197–241
35. Jiménez P, Roldán JC, Gallego FO, Corchuelo R (2020) On the synthesis of metadata tags for HTML files. Softw Pract Exp 50:2169–2192
36. Jung S, Kwon H (2006) A scalable hybrid approach for extracting head components from web tables. IEEE Trans Knowl Data Eng 18(2):174–187
37. Kasirajan V (2021) Fundamentals of quantum computing. Springer

38. Kayed M, Chang C-H (2010) FiVaTech: page-level web data extraction from template pages. IEEE Trans Knowl Data Eng 22(2):249–263
39. Kerenidis I, Prakash A (2017) Quantum recommendation systems. In: ITCS, vol 67, pp 49:1–49:21
40. Kerenidis I, Landman J, Luongo A, Prakash A (2019) q-means: a quantum algorithm for unsupervised machine learning. In: NIPS, pp 4136–4146
41. Kietzmann J, Demetis DS, Eriksson T, Dabirian A (2021) Hello quantum! How quantum computing will change the world. IEEE IT Profess 23(4):106–111
42. Kim Y-S, Lee K-H (2005) Detecting tables in web documents. Eng Appl AI 18(6):745–757
43. Knight W (2018) Serious quantum computers are finally here. MIT Technology Review
44. Kumar V, Bass G, Tomlin C, Dulny J (2018) Quantum annealing for combinatorial clustering. Quantum Inf Process 17(2):39
45. Li Y, Wang Y, Wang Y, Jiao L, Liu Y (2016) Quantum clustering using kernel entropy component analysis. Neurocomputing 202:36–48
46. Limaye G, Sarawagi S, Chakrabarti S (2010) Annotating and searching web tables using entities, types, and relationships. VLDB 3:1338–1347
47. Liu W, Meng X, Meng W (2010) ViDE: a vision-based approach for deep web data extraction. IEEE Trans Knowl Data Eng 22(3):447–460
48. Lopresti DP, Nagy G (2000) A tabular survey of automated table processing. In: GREC, pp 93–120
49. Milošević N, Gregson C, Hernández R, Nenadic G (2016) Disentangling the structure of tables in scientific literature. In: NLDB, pp 162–174
50. Neukart F, Compostella G, Seidel C, von Dollen D, Yarkoni S, Parney B (2017) Traffic flow optimization using a quantum annealer. Front ICT 20:66
51. Nishida K, Sadamitsu K, Higashinaka R, Matsuo Y (2017) Understanding the semantic structures of tables with a hybrid deep neural network architecture. I:n AAAI, pp 168–174
52. Oulabi Y, Bizer C (2019) Extending cross-domain knowledge bases with long tail entities using web table data. In: EDBT, pp 385–396
53. Pimplikar R, Sarawagi S (2012) Answering table queries on the Web using column keywords. VLDB 5:908–919
54. Roldán JC, Jiménez P, Corchuelo R (2020) On extracting data from tables that are encoded using HTML. Knowl Based Syst 190:105157
55. Sarawagi S (2008) Information extraction. Found Trends Databases 1(3):261–377
56. Sheskin DJ (2020) Handbook of parametric and nonparametric statistical procedures, 5th edn. Chapman & Hall/CRC Press
57. Sleiman HA, Corchuelo R (2013) TEX: an efficient and effective unsupervised web information extractor. Knowl Based Syst 39:109–123
58. Sleiman HA, Corchuelo R (2013) A survey on region extractors from web documents. IEEE Trans Knowl Data Eng 25(9):1960–1981
59. Sleiman HA, Corchuelo R (2014) A class of neural-network-based transducers for web information extraction. Neurocomputing 135:61–68
60. Sleiman HA, Corchuelo R (2014) Trinity: on using trinary trees for unsupervised web data extraction. IEEE Trans Knowl Data Eng 26(6):1544–1556
61. Turmo J, Ageno A, Català N (2006) Adaptive information extraction. ACM Comput Surv 38(2):66
62. Wereszczyński K, Michalczuk A, Josiński H, Polański A (2018) Quantum computing for clustering big datasets. In: IEEE applications of electromagnetics in modern techniques and medicine, pp 276–280
63. Wikipedia. Wikipedia download (2020)
64. Wittek P (2014) Clustering structure and quantum computing. In: Quantum machine learning. Elsevier, pp 99–107
65. Wittek P (2016) Quantum machine learning. Academic Press
66. Wu X, Cao C, Wang Y, Fu J, Wang S (2016) Extracting knowledge from web tables based on DOM tree similarity. In: KSEM, vol 9983, pp 302–313
67. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193
68. Yang Y, Luk W (2002) A framework for web table mining. In: WIDM, pp 36–42
69. Yoshida M, Torisawa K, Tsujii J (2001) A method to integrate tables of the World Wide Web. In: WDA, pp 31–34
70. Zanibbi R, Blostein D, Cordy JR (2004) A survey of table recognition. IJDAR 7(1):1–16
71. Zhang S, Balog K (2020) Web table extraction, retrieval, and augmentation: a survey. ACM Trans Intell Syst Technol 11:13:1-13:35

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 HTML tables as a means to feed Data Science applications.



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.

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.