From Sets of Good Redescriptions to Good Sets of Redescriptions

Janis Kalofolias[1], Esther Galbrun[2] and Pauli Miettinen[1]

[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany;
[2]Inria Nancy – Grand Est, Nancy, France

Abstract. Redescription mining aims at finding pairs of queries over data variables that describe roughly the same set of observations. These redescriptions can be used to obtain different views on the same set of entities. So far, redescription mining methods have aimed at listing all redescriptions supported by the data. Such an approach can result in many redundant redescriptions and hinder the user's ability to understand the overall characteristics of the data.

In this work, we present an approach to identify and remove the redundant redescriptions, that is, an approach to move from a set of good redescriptions to a good set of redescriptions. We measure the redundancy of a redescription using a framework inspired by the concept of subjective interestingness based on maximum-entropy distributions as proposed by De Bie in 2011. Redescriptions, however, generate specific requirements on the framework, and our solution differs significantly from the existing ones. Notably, our approach can handle disjunctions and conjunctions in the queries, whereas the existing approaches are limited only to conjunctive queries. Our framework can also handle data with Boolean, nominal, or real-valued data, possibly containing missing values, making it applicable to a wide variety of data sets. Our experiments show that our framework can efficiently reduce the redundancy even on large data sets.

Keywords: Data mining; Redescription mining; Pattern selection; Maximum-entropy; Subjective interestingness

1. Introduction

Redescription mining is a data mining task that aims at finding alternative characterisations of (roughly) the same objects. The motivation behind this is simple and intuitive: if some objects can be described in alternative ways,

Received 16 Mar 2017
Revised 20 Sep 2017
Accepted 26 Nov 2017

then they form a particularly coherent group. Furthermore, identifying such alternative descriptions tells us something about the properties appearing in such synonymous characterisations.

Take for instance the case of areas of the globe, for which records of observed animal species and also climatic information are available. Areas that host particular species and share a particular climatic profile form a coherent group, closely related to the concept of an environmental niche in ecology. As another example, in medicine, identifying groups of patients who share similar profiles might help relate genetic traits, disease symptoms, and treatment outcomes. The same idea can also be of interest in other fields, including ethnography, sociology, and chemistry.

Like many other data mining tasks, redescription mining is subject to the issue of pattern explosion, wherein a large number of results are returned by the mining algorithm—many of them slight variations of one another. As a result, we face the challenge of identifying an interesting non-redundant subset, worth further inspection.

The problem we address in this paper is that of pattern selection. Considering a data set that consists of a pair of matrices over the same objects, and given a collection of redescriptions that has been mined from this data set, we want to select a subset of redescriptions that is informative and non-redundant. Note that we are not introducing any new pattern formalism nor any new algorithm for mining redescriptions. Instead, we focus on the problem of ranking and filtering redescriptions in post-processing.

One approach to evaluating the interestingness of patterns, which we adopt here, is to use them to construct a statistical model of the data. A pattern can be seen as an observation from the data and can be evaluated against the current model. If the current model already accounts for the observation, that is, the pattern does not provide any new information about the data, it is considered redundant and can be discarded. Otherwise, the pattern is deemed interesting and is integrated into the model, thereby increasing its quality.

We propose an iterative approach. At each step, the candidate redescriptions are evaluated against a model that incorporates the current knowledge available about the data, namely the information contained in the redescriptions selected so far. Using our proposed score, we can measure the novel information each candidate redescription contains with respect to the current state. We then rank these redescriptions and select the most informative one to be incorporated into the model. Next, the candidate scores are updated to reflect the new model and we proceed with selecting the best one. This process is iterated until no additional information can be incorporated into the model, that is, until only redundant candidate redescriptions are left.

Our modelling techniques rely on the maximum entropy distribution and the framework of subjective interestingness, as proposed by De Bie (2011). In brief, we maintain a maximum entropy distribution conditioned on all of the already-seen redescriptions to model what is already known, and consider redundant all redescriptions that have a high likelihood under this distribution. We emphasise, however, that our model differs from that of De Bie (2011) and subsequent work (e.g. Kontonasios, Vreeken and De Bie, 2013) in significant ways, as the patterns

we are studying—namely redescriptions—are more intricate than the patterns studied in prior work.[1]

The approach presented here rests on two main lines of work. *Redescription mining* on the one hand provides us with a rich pattern language, while on the other hand *maximum entropy modelling* provides us with well-grounded selection principles. We cover the basics of these two fields in Section 2, while general related work is discussed in Section 6. Our main model is discussed in Section 3, and is extended to handle missing values in Section 4. We present our algorithms for calculating the model and ranking the redescriptions in Section 5. An empirical evaluation is provided in Section 7.

2. Background

In this section, we give the basic definitions of redescription mining and maximum entropy modelling. Further information on redescription mining can be found, for instance, from the work of Galbrun and Miettinen (2018); maximum entropy modelling, especially as applicable to data analysis, is discussed by De Bie (2011).

2.1. Redescription mining

In the formulation of redescription mining considered here, the input data consist of entities with two sets of characterising variables, thus forming a data set with two sides. We generally refer to these two sides as the left- and right-hand sides, and represent them using two matrices $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$, respectively. The columns of these matrices correspond to the sets of variables $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$. The domains of the variables are denoted by $\mathcal{V}_{\mathbf{L}}$ and $\mathcal{V}_{\mathbf{R}}$, respectively, and they can be either continuous or discrete. The rows of the matrices correspond to the entities. The set of entities characterised by the two sides is denoted by E, and so both matrices have $|E|$ rows. The value of $D(i,j)$ is the value of variable $v_j \in V$ for entity $\boldsymbol{d}_i \in E$.

Truth assignments are then defined over the variables by requiring them to take values within a subset of their domain. We denote these truth assignments using the notation $[a \leq v \leq b]$, except for Boolean variables where the truth assignment $[v = \texttt{True}]$ is denoted simply as v. These truth assignments and their negations can be combined into logical statements using the Boolean operators $\wedge$ (and) and $\vee$ (or).

The output of redescription mining consists of a collection of query pairs, the redescriptions, of the form $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$, where the queries $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$ are *logical statements* over the variables in $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$, respectively.

For instance, the following pair of queries

$$q_{\mathbf{L}} = \text{greater white-toothed shrew} \wedge \text{Egyptian mongoose}$$
$$q_{\mathbf{R}} = \big([15.6 \leq t_8^- \leq 19] \wedge [1.62 \leq p_8 \leq 7.44] \wedge [66.2 \leq p_{12} \leq 137]\big) \vee [13.9 \leq t_3^{\sim} \leq 14.3]$$

constitutes a redescription. This redescription was extracted from the Bio data

[1] The present work is an extended version of our earlier conference publication (Kalofolias, Galbrun and Miettinen, 2016).

set, which we use in our experiments. This data set comes from the ecological domain and characterises geographic areas in terms of the fauna that inhabits them and of their climatic profile. More details about this data set can be found in Section 7. The left-hand side query is a simple conjunction of Boolean variables and states that both the greater white-toothed shrew and the Egyptian mongoose are present. The right-hand side query is more complex and requires the values of the temperatures of March ($t_3^{\sim}$) and August (t_8^{-}) as well as the precipitation of August (p_8) and December (p_{12}) to take value within the specified ranges.

This query language, combining Boolean and numerical variables into conjunctions and disjunctions, is at once rich enough to capture patterns that are more expressive than purely Boolean conjunctions, but still constrained enough that the queries remain interpretable and that we are able to devise algorithms to find them. Since we aim to evaluate the interestingness of the patterns by using them to build a statistical model of the data, we need to design a modelling approach that is able to accommodate this expressive query language. In other words, we need a flexible approach with strong modelling power.

The *support* of a query q is the subset of entities for which the query holds true, that is, $\text{supp}(q) = \{\boldsymbol{d}_i : q \text{ is true for } \boldsymbol{d}_i \in D\}$. The support of a redescription R is the set of entities that satisfy both logical statements $\text{supp}(R) = \text{supp}(q_{\mathbf{L}}) \cap \text{supp}(q_{\mathbf{R}})$. The set of variables over which q is expressed is denoted as $\text{vars}(q)$.

In essence, a redescription provides alternative characterisations for a set of entities; the similarity of the support of the queries that constitute it is therefore a core property of a redescription, which we refer to as its *accuracy*. Following the established practice in the redescription mining literature (see, e.g. Ramakrishnan, Kumar, Mishra, Potts and Helm, 2004; Parida and Ramakrishnan, 2005; Galbrun and Miettinen, 2012*a*; Zinchenko, Galbrun and Miettinen, 2015; Mihelčić and Šmuc, 2016; Galbrun and Miettinen, 2018), we measure the accuracy using the Jaccard coefficient: given a redescription R, its accuracy is the Jaccard coefficient of its support,

$$\text{J}(R) = \text{J}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \frac{|\text{supp}(q_{\mathbf{L}}) \cap \text{supp}(q_{\mathbf{R}})|}{|\text{supp}(q_{\mathbf{L}}) \cup \text{supp}(q_{\mathbf{R}})|} \ .$$

Using the Jaccard coefficient to measure the accuracy comes with many known benefits (see, e.g. Galbrun and Miettinen, 2018). Accuracy, however, is a *local* feature of the redescription in that it informs on the quality of an individual redescription, but does not help in finding a good *set* of redescriptions. Therefore, we need to look at the variables and values at play in the logical statements of the redescriptions. We will use these logical statements to constrain a maximum entropy distribution over possible data sets, allowing us to evaluate the probability of observing a certain redescription given those we already know to be present.

Most interesting in a redescription is the association between the two queries, and the entities that satisfy both of them. For this reason, our main objective here is to model the conjunction of the two queries that make up a redescription and the intersection of their supports, that is, the support of the redescription. Henceforth, we will treat a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$ as the logical statement $s = q_{\mathbf{L}} \wedge q_{\mathbf{R}}$, with its associated support $\text{supp}(s) = \text{supp}(q_{\mathbf{L}}) \cap \text{supp}(q_{\mathbf{R}}) = \text{supp}(R)$, evaluated over the data set D resulting from the concatenation of $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$; similarly, the variables in $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$ are pooled together to form V. Thus, our data set D is a matrix with $N = |V|$ columns and $M = |E|$ rows. At first, we assume

that the data set is complete. We will explain how our model can be extended to handle missing values in Section 4.

Note that our ranking procedure is applied in post-processing, given a set of redescriptions obtained with a mining algorithm. Our procedure is thus dependent on the ability of the mining algorithm to return a sufficiently diverse set of redescriptions, among which our algorithm will then identify the most interesting and non-redundant ones.

2.2. Maximum entropy modelling

Let $\mathcal{P}$ be a set of probability distributions over $\mathcal{V}$. That is, for every $p \in \mathcal{P}$ and every $\boldsymbol{x} \in \mathcal{V}$, $p(\boldsymbol{x}) \geq 0$ and $\int_{\mathcal{V}} p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1$. The distribution $p \in \mathcal{P}$ is the *maximum entropy distribution* subject to the constraints $\{(f_c, \pi_c) : c \in C\}$ if it is the solution to the following program:

$$\max_{p \in \mathcal{P}} - \int_{\mathcal{V}} p(\boldsymbol{x}) \log p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{1}$$

$$\text{subject to} \quad \int_{\mathcal{V}} p(\boldsymbol{x}) f_c(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \pi_c \qquad \text{for all } c \in C. \tag{2}$$

In the framework of this paper, the constraints (f_c, π_c) encode our current knowledge of the data, obtained from the already-seen redescriptions (see Section 3). Before presenting our specific model, however, we will briefly discuss the general approach of finding the maximum-entropy distribution.

The objective function of maximum entropy is strictly concave and continuous. We can therefore use standard tools from constrained variational optimisation in order to obtain the solution

$$p_\lambda(\boldsymbol{r}) = \frac{1}{Z_{\lambda_c}} \exp\left(\sum_{c \in C} \lambda_c f_c(\boldsymbol{r})\right), \tag{3}$$

where λ_c are the Lagrange multipliers for the constraints and Z_{λ_c} is the *partition function*

$$Z_{\lambda_c} = \int_{\mathcal{V}} \exp\left(\sum_{c \in C} \lambda_c f_c(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\,. \tag{4}$$

The role of the latter is to ensure proper normalisation of the distribution p_λ.

Under the condition that the constraints of (2) are satisfiable, strong duality holds, and we can thus compute the Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_c)_{c \in C}$ by solving the convex dual of the problem in Equations (1) and (2),

$$\min_{\lambda_c \geq 0,\, c \in C} \max_{p \in \mathcal{P}} - \int_{\mathcal{V}} p(\boldsymbol{x}) \log p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} - \sum_{c \in C} \lambda_c \left(\int_{\mathcal{V}} p(\boldsymbol{x}) f_c(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} - \pi_c\right). \tag{5}$$

Solving the corresponding Euler–Lagrange equation, the dual objective can be maximised with respect to p, which yields the new objective of the dual problem

$$q(\boldsymbol{\lambda}) = \log Z_{\boldsymbol{\lambda}} - \sum_{c \in C} \lambda_c \pi_c\,. \tag{6}$$

The objective $q(\boldsymbol{\lambda})$ of the dual problem is convex, which allows for the optimal

vector of the Lagrange multipliers $\boldsymbol{\lambda}^*$ to be computed as the points where the gradient of q vanishes.

The gradient of q is

$$\frac{\partial q}{\partial \lambda_{c'}} = \frac{1}{Z_{\lambda_c}} \int_{\mathcal{V}} \exp\left(\sum_{c \in C} \lambda_c f_c(\boldsymbol{r})\right) f_{c'}(\boldsymbol{r})\, \mathrm{d}\boldsymbol{r} - \pi_{c'} \ , \tag{7}$$

and the points where it vanishes are the ones that satisfy the conditions

$$\mathbb{E}_p \left[f_c(\boldsymbol{r})\right] = \pi_c \qquad \text{for all } c \in C \ , \tag{8}$$

that is, exactly the constraints that were enforced in the original problem formulation (2).

3. Theory

We now introduce our probabilistic models for the data. First we explain how to model the values occurring within an arbitrary row while accounting for the presence of a given set of statements, using the maximum entropy principle. Then we explain how to represent an entire data set based on this row model, using a mixture model that comes in two variants.

3.1. Modelling rows

In this section, our aim is to identify the maximum entropy distribution over the values that a database row may assume while satisfying a set of constraints, each with a given probability.

In the following, we consider a database row to be an assignment of values to each variable of the data set. Let $\boldsymbol{r} \in \mathcal{V}$ be a random vector of such value assignments, where $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_N$ is the Cartesian product of the domains of the variables. Further let S be the set of constraints, where each constraint $s \in S$ takes the form of a logical statement, which must be satisfied with a given validity probability $\pi_s \in [0, 1]$. These statements encode prior knowledge on the specific row and are assumed to be satisfied by a nonempty set of distributions over $\mathcal{V}$. Among those distributions, we want to identify the one with maximum entropy $p^{\mathcal{M}}_{\texttt{ROW}}(\boldsymbol{r} \mid S)$, which we will simply denote as $p_{\texttt{ROW}}(\boldsymbol{r})$, when the set of constraints S and the model $\mathcal{M}$ are clear from the context.

Since the constraints are logical statements, we can adapt the derivation of Section 2.2 by substituting the constraint function of Equation (2) with

$$f_s(\boldsymbol{r}) = \chi_s(\boldsymbol{r}) \ , \tag{9}$$

where $\chi_s(\boldsymbol{r})$ is the characteristic function of statement s, that is, $\chi_s(\boldsymbol{r}) = 1$ if s holds true on $\boldsymbol{r}$ and $\chi_s(\boldsymbol{r}) = 0$ otherwise. As a result, the optimality conditions of Equation (8) now take the form

$$\mathbb{E}_{\mathcal{M}} \left[\chi_s(\boldsymbol{r})\right] = \pi_s \quad \text{for all } s \in S \ , \tag{10}$$

while the solution and partition functions of Equations (3) and (4) become,

respectively,

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \frac{1}{Z_{\boldsymbol{\lambda}}} \exp\left(\sum_{s\in S} \lambda_s \chi_s(\boldsymbol{r})\right) \tag{11}$$

and

$$Z_{\boldsymbol{\lambda}} = \int_{\mathcal{V}} \exp\left(\sum_{s\in S} \lambda_s \chi_s(\boldsymbol{r})\right) \mathrm{d}\boldsymbol{r} \ . \tag{12}$$

In practice, however, although the quantity in the exponent of (11) is trivial to compute for a given row $\boldsymbol{r}$, the partition function Z_{λ_c} is much more complex. Indeed, the integral of (4) runs over all the possible values that the random row $\boldsymbol{r}$ may assume. In addition, computing the integral itself is difficult, since the general formulation of the maximum entropy problem does not accept an analytical solution, making numerical approximation methods necessary (cf. Burden and Faires, 2011). Numerical methods are also necessary in computing the Lagrange multipliers and require multiple evaluations of the value or the derivatives of the dual objective function of Equation (6). In the rest of this section we mitigate these issues by exploiting intrinsic properties of our models.

Factorising the distribution. In its general form, the integration in the partition function (12) runs over the entire domain $\mathcal{V}$. However, each logical statement s only involves a small subset of variables $\mathrm{vars}(s) \subseteq V$.

This observation allows for a simplification of the computation by aggregating the statements into groups, such that each group involves as few variables as possible and can be computed independently. When the variables involved in computing the statements of a group do not overlap with the variables involved in another group, we can split the sums within the exponents of Equations (11) and (12) into corresponding groups.

Formally, we define a partitioning $\mathcal{K}$ of the variables in V such that for all $s \in S$ there exists a $K \in \mathcal{K}$ with $\mathrm{vars}(s) \subseteq K$, and denote by $S_K = \{s \in S : \mathrm{vars}(s) \subseteq K\}$ the subset of statements that only contain variables in K. Now the sets S_K form a partitioning of S and we can split the sums over these partitions

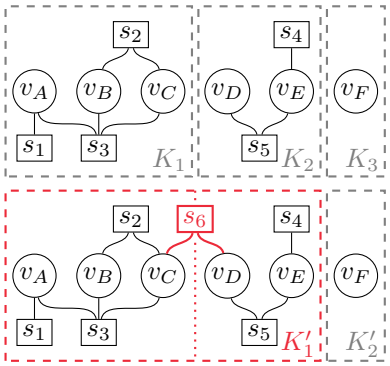
$$\exp\left(\sum_{s\in S} \lambda_s \chi_s(\boldsymbol{r})\right) = \prod_{K\in\mathcal{K}} \exp\left(\sum_{s\in S_K} \lambda_s \chi_s(\boldsymbol{r})\right) .$$

The integral in the partition function can similarly be split into a product of sub-integrals, each over the domain $\mathcal{V}_K$ of all variables in K. Thus, we obtain

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \prod_{K\in\mathcal{K}} p_{\boldsymbol{\lambda}_K}(\boldsymbol{r}) \ , \tag{13}$$

which is a product of sub-probabilities defined over independent subsets of the original variables

$$\begin{aligned} p_{\boldsymbol{\lambda}_K}(\boldsymbol{r}) &= \frac{1}{Z_K(\boldsymbol{\lambda})} \exp\left(\sum_{s\in S_K} \lambda_s \chi_s(\boldsymbol{r})\right) \\ Z_K(\boldsymbol{\lambda}) &= \int_{\mathcal{V}_K} \exp\left(\sum_{s\in S_K} \lambda_s \chi_s(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x} \ , \end{aligned} \tag{14}$$



$$
\begin{aligned}
v_A, v_B, v_C &\in \{0,1\} \\
v_D &\in [10,50] \\
v_E &\in [0,5] \\
v_F &\in \{0,1\}
\end{aligned}
$$

$$
\begin{aligned}
s_1 &= v_A \\
s_2 &= v_B \wedge v_C \\
s_3 &= v_A \vee v_B \vee v_C \\
s_4 &= [2 < v_E] \\
s_5 &= [20 < v_D] \vee [2 < v_E] \\
s_6 &= v_C \vee \neg [v_D < 30]
\end{aligned}
$$

Figure 1. Clustering of the six data variables $v_A, \ldots, v_F$ of a row model with five logical statements $s_1, \ldots, s_5$ (top) and after the addition of a sixth statement s_6 (bottom).

and the normalisation can be performed per each $Z_K(\boldsymbol{\lambda})$ term. To maximise computational efficiency, we aim for the finest partitioning possible.

Example 1. Consider a data set over six variables, $\{v_A, \ldots, v_F\}$, and a collection of five logical statements over these variables $S = \{s_1, \ldots, s_5\}$, as shown in Figure 1 (right). In this case, the finest partitioning of the variables, so that all statements are completely contained within one cluster, consists of the 3 groups $K_1 = \{v_A, v_B, v_C\}$, $K_2 = \{v_D, v_E\}$, and $K_3 = \{v_F\}$, as shown in Figure 1 (top left). Suppose that at this point the constraint $s_6 = v_C \vee \neg[v_D < 30]$ is added. To fulfil the containment requirement, we now need to merge K_1 and K_2, resulting in the clustering shown in Figure 1 (bottom left).

To further speed-up the computation, we can also perform a re-parametrisation (or factorisation) of the distribution, for instance, using the Junction Tree algorithm (Jensen and Jensen, 1994).

Quantising the domain. A further source of complexity in Equation (14) results from integrating over the domain of the variable set. Since the real-valued variables are constrained through the interval notation, we can quantise these variables into discrete bins aligned with these intervals, in order to reduce the integral into an equivalent summation.

We achieve this by consulting the truth table for the statements in S, to group together the regions of the domain where the same combination of statements are satisfied. Formally, for an arbitrary ordering $s_1, \ldots, s_\sigma$ of all statements in S, we define a truth vector $(t_1, \ldots, t_\sigma) \in \{0,1\}^\sigma$, assigning a value to the truth of each statement. Each such vector defines a region of the variable domain over which the truth of each statement matches the respective element of the truth vector

$$T_{t_1 \cdots t_\sigma} = \{\boldsymbol{r} \in \mathcal{V} : \chi_{s_i}(\boldsymbol{r}) = t_i \text{ for all } s_i \in S\} \ .$$

All these regions together form a partitioning of the domain, denoted as $\mathcal{T}$. Note that unfeasible truth assignments correspond to empty regions and can be safely ignored. Using a similar subscript notation, we denote the combinations of Lagrange multipliers as $\zeta_{t_1 \cdots t_\sigma}(\boldsymbol{\lambda}) = \lambda_1 \cdot t_1 + \cdots + \lambda_\sigma \cdot t_\sigma$.

Table 1. Domain quantisation for the statements $\{s_1, s_2, s_3\}$ over the Boolean variables $\{v_A, v_B, v_C\}$ from K_1. Statements truth table (left) and accumulated terms (right).

v_A	v_B	v_C	χ_{s_1}	χ_{s_2}	χ_{s_3}	$T_{\boldsymbol{t}}$
0	0	0	0	0	0	T_{000}
1	0	0	1	0	1	T_{101}
0	1	0	0	0	1	T_{001}
1	1	0	1	0	1	T_{101}
0	0	1	0	0	1	T_{001}
1	0	1	1	0	1	T_{101}
0	1	1	0	1	1	T_{011}
1	1	1	1	1	1	T_{111}

$T_{\boldsymbol{t}}$	$\lvert T_{\boldsymbol{t}}\rvert$	$\zeta_{\boldsymbol{t}}$
T_{000}	1	0
T_{001}	2	λ_3
T_{011}	1	$\lambda_2 + \lambda_3$
T_{101}	3	$\lambda_1 + \lambda_3$
T_{111}	1	$\lambda_1 + \lambda_2 + \lambda_3$

Table 2. Domain quantisation for the statements $\{s_4, s_5\}$ over the real-valued variables $\{v_D, v_E\}$ from K_2. From statement truths to region probabilities $P(Q) := \int_Q p(\boldsymbol{x}) d\boldsymbol{x}$.

Q	v_D	v_E	χ_{s_4}	χ_{s_5}	$T_{\boldsymbol{t}}$	$\lvert T_{\boldsymbol{t}}\rvert$	$\zeta_{\boldsymbol{t}}$	$p(\boldsymbol{x})$	$P(Q)$
Q_1	$[10, 20]$	$[0, 2]$	0	0	T_{00}	$10 \cdot 2$	0	$1/40$	0.5
Q_2	$[20, 50]$	$[0, 2]$	0	1	T_{01}	$30 \cdot 2$	λ_5	$1/(40 \cdot 15)$	0.1
Q_3	$[10, 20]$	$[2, 5]$	1	1	T_{11}	$10 \cdot 3 + 30 \cdot 3$	$\lambda_4 + \lambda_5$	$2/(40 \cdot 15)$	0.1
Q_4	$[20, 50]$	$[2, 5]$	1	1					0.3

We can now express Equations (11) and (12) as

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \frac{1}{Z(\boldsymbol{\lambda})} \exp\left(\zeta_{\boldsymbol{\tau}(\boldsymbol{r})}(\boldsymbol{\lambda})\right) , \tag{15}$$

$$Z(\boldsymbol{\lambda}) = \sum_{T_{\boldsymbol{t}} \in \mathcal{T}} \int_{T_{\boldsymbol{t}}} \exp\left(\zeta_{\boldsymbol{t}}(\boldsymbol{\lambda})\right) \mathrm{d}\boldsymbol{x} = \sum_{T_{\boldsymbol{t}} \in \mathcal{T}} |T_{\boldsymbol{t}}(\boldsymbol{\lambda})| \exp\left(\zeta_{\boldsymbol{t}}(\boldsymbol{\lambda})\right) , \tag{16}$$

where $\boldsymbol{\tau}(\boldsymbol{r}) = (\chi_{s_1}(\boldsymbol{r}), \ldots, \chi_{s_\sigma}(\boldsymbol{r}))$ is the truth vector of the statements computed at the value assignment $\boldsymbol{r}$.

Example 2. Going back to Example 1, consider the computation of the factor associated to group $K_1 = \{v_A, v_B, v_C\}$, involving statements s_1, s_2 and s_3 (see Figure 1). The truth table of these statements is shown in Table 1 (left). The terms can be gathered based on the satisfiability of the constraints in the just 5 different combinations of Table 1 (right). The partition function of Equation (16) can then be written with one term per combination, as

$$Z_{K_1}(\boldsymbol{\lambda}) = 1 \cdot e^{\zeta_{000}} + 2 \cdot \mathrm{e}^{\zeta_{001}} + 1 \cdot \mathrm{e}^{\zeta_{011}} + 3 \cdot \mathrm{e}^{\zeta_{101}} + 1 \cdot \mathrm{e}^{\zeta_{111}} .$$

In this small example, the number of terms has been reduced from $2^3 = 8$ to 5. As the number of variables rises, this simplification can become more dramatic.

In the more general case involving non-Boolean variables, the boundaries delimiting the regions of the domain can be easily identified from the thresholds of the intervals.

Example 3. Consider now the computation of the factor associated to group $K_2 = \{v_D, v_E\}$, involving the statements s_4 and s_5 of Figure 1, respectively;

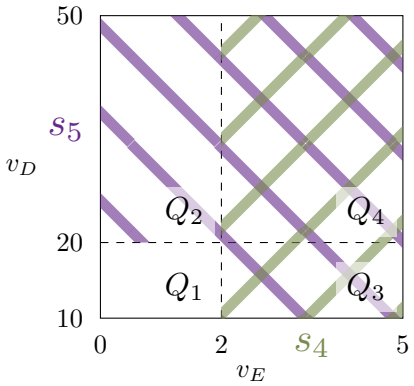


Figure 2. Regions of the domains of v_D and v_E.

we assume satisfiability probabilities $\pi_{s_4} = 0.4$ and $\pi_{s_5} = 0.5$, and use domains $v_D \in [10, 50]$ and $v_E \in [0, 5]$.

The literals involved are $[2 < v_E]$ and $[20 < v_D]$, so the relevant thresholds are $(0, 2, 5)$ for v_E and $(10, 20, 50)$ for v_D, respectively. The resulting quantisation is depicted in Figure 2, where each of the four blocks represents one region of the domain, denoted as $Q_1, \dots, Q_4$. For each region, the satisfiabilities of both statements are shown in Table 2. This combination determines the partition $T_{\boldsymbol{t}}$ to which the region is assigned. The measure $|T_{\boldsymbol{t}}|$ of a partition is simply the sum of the areas of the regions it consists of. Using these partitions and the corresponding combinations of Lagrange multipliers, we can write the partition function as

$$\begin{aligned} Z_{K_2}(\boldsymbol{\lambda}) &= \int_{T_{00}} e^{\zeta_{00}(\boldsymbol{\lambda})}\,\mathrm{d}\boldsymbol{x} + \int_{T_{01}} e^{\zeta_{01}(\boldsymbol{\lambda})}\,\mathrm{d}\boldsymbol{x} + \int_{T_{11}} e^{\zeta_{11}(\boldsymbol{\lambda})}\,\mathrm{d}\boldsymbol{x} \\ &= |T_{00}|\,\mathrm{e}^{0} \quad + |T_{01}|\,\mathrm{e}^{\lambda_5} \quad + |T_{11}|\,\mathrm{e}^{\lambda_4+\lambda_5} \\ &= 20 \quad + 60\,\mathrm{e}^{\lambda_5} \quad + 120\,\mathrm{e}^{\lambda_4+\lambda_5}\ . \end{aligned}$$

From the optimality conditions of Equation (8) it follows that

$$\begin{cases} \left(120\,\mathrm{e}^{\lambda_4+\lambda_5}\right)/Z_{K_2} = & \pi_{s_4} = & 0.4 \\ \left(60\,\mathrm{e}^{\lambda_5} + 120\,\mathrm{e}^{\lambda_4+\lambda_5}\right)/Z_{K_2} = & \pi_{s_5} = & 0.5\ , \end{cases}$$

which can be solved analytically to obtain $\mathrm{e}^{\lambda_4^*} = 2$, $\mathrm{e}^{\lambda_5^*} = 1/15$, and $Z_{K_2} = 40$, giving the probability measure

$$p_{\boldsymbol{\lambda}^*}(\boldsymbol{r}) = 1/40 \cdot 2^{\chi_{s_4}(\boldsymbol{r})} \cdot 1/15^{\chi_{s_5}(\boldsymbol{r})}\ .$$

The case of certain satisfiability. A special case which allows the problem to be simplified consists of sets of statements such that all the statements are certain (i.e. $\pi_s = 1$). Indeed, we can show that the maximum entropy distribution incorporating a set of statements S all with $\pi_s = 1$ is

$$p(\boldsymbol{r}) = \begin{cases} 1/Z & \text{if } \boldsymbol{r} \text{ satisfies all } s \in S \\ 0 & \text{otherwise}\ , \end{cases} \tag{17}$$

where $Z = |T_{\mathbf{1}}|$ is the measure of the subset $T_{\mathbf{1}}$, which consists of the regions where all the statements in S are satisfied.

3.2. Modelling a data set

Having presented our row model in Section 3.1, we now proceed to describe the means to assemble several such row models into a model for an entire data set. Our database models take as input an original data set D and a collection of logical statements S mined from this data set.

We consider the data set as consisting of multiple entities, each encoded as a database row. In the development of our models we assume that these entities are independent, and therefore the order of the rows within the database can be disregarded. On the other hand, we do not assume that these entities are identically distributed and we allow for a different subset of statements to apply as prior knowledge on different entities. In other words, we allow each entity to follow a *local distribution*, that may differ from that of the other entities. Each of these distributions corresponds to a row model constrained with a particular subset of statements. Thus, we compute the overall probability for the data set by combining these local distributions.

Formally, let $\boldsymbol{d}_i$ be the values observed in the data set D on the i-th row and $\boldsymbol{r}_i$ the corresponding random variable, both taking values over the domain of the row values $\mathcal{V}$. We denote as κ_q the event that a statement q is satisfied on a data row $\boldsymbol{d}$. One of our goals is to make a decision about the validity of the event, a decision that we represent by the function $\hat{\phi}(q, \boldsymbol{d})$. At this point, knowing $\boldsymbol{d}$ deterministically defines the outcome, which matches the characteristic function of the statement on $\boldsymbol{d}$, that is,

$$\hat{\phi}(q, \boldsymbol{d}) = \chi_q(\boldsymbol{d}) \ . \tag{18}$$

We denote as S_i the subset of statements to be applied to the i-th row model as constraints ($S_i \subset S$). This subset contains those statements from S that are satisfied on the observed row $\boldsymbol{d}_i$, formally defined as

$$S_i \coloneqq \left\{ s \in S : \hat{\phi}_i(s, \boldsymbol{d}_i) = 1 \right\} \ . \tag{19}$$

Note that this formulation will later allow us to extend inference for the case of missing values in Section 4, making the usefulness of the $\hat{\phi}$ function more obvious.

Using the row model introduced in the previous section, henceforth denoted by $p_{\texttt{ROW}}$, the probability of the row values $\boldsymbol{r}_i$ can now be defined as

$$\mathbb{P}\left(\boldsymbol{r}_i \mid D, S\right) = \mathbb{P}\left(\boldsymbol{r}_i \mid \boldsymbol{d}_i, S\right) = p_{\texttt{ROW}}(\boldsymbol{r}_i \mid S_i) \ . \tag{20}$$

Given the row probabilities, we now aim to combine them into a probability over all possible values of a data set.

A naïve approach that satisfies the assumption that rows follow a local distribution would be to model the data as a bag-of-rows, resulting in a database probability that degenerates into a product of row probabilities. However, this imposes the very strict assumption that all rows are independent and has the major practical drawback that the final probability depends too much on the individual probability of each row. Consider, for instance, two queries q_1 and q_2 with the same validities on all but one row, on which q_1 is twice as likely as q_2. Under the bag-of-rows model the data set probability of q_1 is double that of q_2.

Instead, our models compute the total probability as a weighted average of the probabilities obtained from the relevant row models. We present two variants for the overall model, which differ in the entities that are considered relevant for computing the overall probability. The first model variant considers all entities in the data set. In other words, it takes into account all the row models, and we therefore refer to it as the `MEALL` model. The second model variant, which we call the `MEBLK` model, only considers the entities containing the query statement, that is, it only takes into account the row models associated to entities that satisfy the statement being evaluated.

Intuitively, evaluating the occurrence probability of a statement can be thought of as first selecting a relevant entity, then computing the probability that the statement occurs on that entity. The entity must be selected with a probability that reflects its representativity in the data set, which can motivate the use of a non-uniform entity-selecting distribution in certain cases. For instance, in a data set where the entities correspond to different geographic areas, recording, for example, species occurrences or voting outcomes, it might make sense to weight entities based on the size of the region. Likewise, with entities representing time spans of varying duration, one might want to select them with probabilities proportional to the time they span. However, unless more information is available, the uniform distribution over the relevant entities constitutes an obvious choice of entity-selecting distribution for this purpose, which we adopt here. This evaluation procedure can be statistically formalised as a mixture model, where the mixture coefficient determines the probability of selecting a particular entity and thus its associated row model. The graphical representation of both `MEALL` and `MEBLK` models is presented in Figure 3 (excluding the elements in blue).

Each of the M plates represents a variable $\boldsymbol{r}_i$ depending on S and $\boldsymbol{d}_i$ through S_i, as explained in Equation (19). The mixing coefficient ρ acts as a prior probability over the row indices: it formally specifies which row models should be mixed together to obtain the final distribution over the values of $\boldsymbol{r}$, a single row summarising the entire data set. To enforce the selection behaviour $\boldsymbol{r} = \boldsymbol{r}_\rho$, the row selecting function can be expressed as

$$\mathbb{P}\left(\boldsymbol{r} \mid \boldsymbol{r}_1, \ldots, \boldsymbol{r}_M, \rho\right) \coloneqq \delta\left(\boldsymbol{r} - \boldsymbol{r}_\rho\right) \ , \tag{21}$$

where $\delta\left(\cdot\right)$ is the Dirac delta function.[2] The row prior ρ is the uniform distribution over the row indices.

For a fixed row vector $\boldsymbol{r}$, the satisfiability of a statement q can be asserted deterministically, and we have

$$\mathbb{P}\left(\kappa_q \mid q, \boldsymbol{r}\right) = \chi_q(\boldsymbol{r}) \ , \tag{22}$$

where κ_q is the event that statement q is satisfied by the value assignment $\boldsymbol{r}$.

[2] The Dirac delta, which is the continuous equivalent of the Kronecker delta, is a generalised function that assumes an infinite mass when its argument is zero, in our case effectively ensuring that only the case of $\boldsymbol{r} = \boldsymbol{r}_\rho$ is possible.

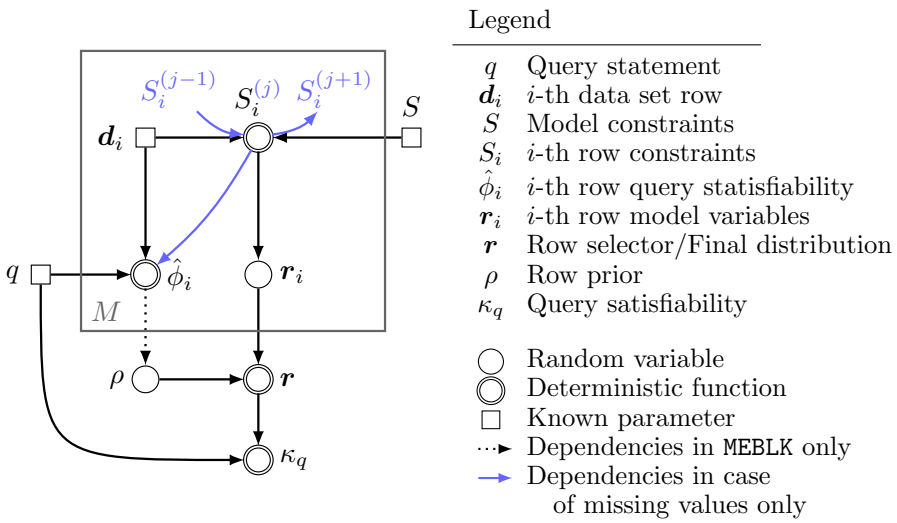


Figure 3. Graphical representation of the extended models MEALL (solid lines) and MEBLK (all lines). The elements extending the models in the case of missing values, explained in Section 4, are depicted in blue.

Now the posterior distribution of MEALL becomes

$$\mathbb{P}(\kappa_q \mid q, D, S) = \sum_{\rho=1}^{M} \mathbb{P}(\rho) \int_{\mathcal{V},\mathcal{V}_1,\ldots,\mathcal{V}_M} \mathbb{P}(\boldsymbol{r} \mid \boldsymbol{r}_1, \ldots, \boldsymbol{r}_M, \rho) \times \mathbb{P}(\kappa_q \mid q, \boldsymbol{r}) \times \prod_{i=1}^{M} \mathbb{P}(\boldsymbol{r}_i \mid D, S)\, \mathrm{d}\boldsymbol{r}\, \mathrm{d}\boldsymbol{r}_1 \cdots \mathrm{d}\boldsymbol{r}_M \;, \tag{23}$$

which can be simplified by substituting the specific distributions to

$$p_{\texttt{ALL}}(\kappa_q \mid q, D, S) = \frac{1}{M} \sum_{i=1}^{M} p_{\texttt{ROW}}(\kappa_q \mid q, S_i) \;, \tag{24}$$

with $p_{\texttt{ROW}}(\kappa_q \mid q, S_i) \coloneqq \mathbb{E}[\chi_q(\boldsymbol{r}_i) \mid S_i]$ being the probability that the query statement q is satisfied on row $\boldsymbol{r}_i$.

The quantity in Equation (24) is essentially an average over all the probabilities assigned by the rows. If we change the row prior distribution to be uniform over the rows which support the query statement q exclusively,

$$\mathbb{P}(\rho \mid \boldsymbol{d}_1, \ldots, \boldsymbol{d}_M, q) \coloneqq \frac{\hat{\phi}_i(q, \boldsymbol{d}_\rho)}{\sum_{i=1}^{M} \hat{\phi}_i(q, \boldsymbol{d}_i)} \;, \tag{25}$$

we obtain the MEBLK model, represented by the additional dotted dependencies in Figure 3.

The key difference between our two models, MEALL and MEBLK, lies in the different subsets of entities considered when evaluating a query statement, which manifests in the use of a different mixing coefficient ρ.

In concrete terms, MEALL evaluates the probability of the query statement

being satisfied in the data set overall, while MEBLK evaluates the probability of the query statement being satisfied on the rows where it was observed to hold in the original data set. As a result, MEALL possesses the potentially useful property that the occurrence probability of a statement and that of its negation sum to unity, $p_{\texttt{ALL}}(\kappa_q \mid q, \cdot) = 1 - p_{\texttt{ALL}}(\kappa_{\neg q} \mid q, \cdot)$, which does not generally hold under MEBLK. On the other hand, MEBLK offers a more intuitive interpretation of the satisfiability probability, in accordance with the framework of De Bie (2011). Also, once a statement has been incorporated into MEBLK, the query probability of that same statement is certain. That is, if the query statement has been previously added to the model, under MEBLK it will be assigned an occurrence probability of one and deemed unsurprising.

In addition, MEALL is better suited to comparing queries without penalising support size, while MEBLK favours greater support sizes. As a simple example illustrating this effect, consider three Boolean conjunctive statements s, q_1, and q_2 such that $\text{supp}(q_1) > \text{supp}(q_2)$ and s is contained into both q_1 and q_2. Imagine that s has been previously incorporated into the model and we now want to evaluate q_1 and q_2. MEALL assigns the same occurrence probability to both statements, whereas MEBLK favours the one with the larger support as more surprising.

Finally, MEBLK yields qualitatively better results as well as generally lower computational complexity, as evidenced by the experiments of Section 7.

4. Handling Missing Values

Data sets are oftentimes incomplete, in the sense that the values for some entries are indicated as missing. In this section, we explain how our models are extended to take into account the presence of missing values in the data sets.

First, note that in the case of nominal variables, a naive way to handle missing values is to consider them as an extra category. By adopting such an approach, our model as presented so far can already be used to handle missing values. An obvious downside of this approach, however, is that it cannot be applied to real-valued variables. Another and more crucial downside is that using a distinct but deterministic value does not capture the true interpretation of a missing value: the absence of information. This absence has to be accounted for probabilistically, reflecting the implied uncertainty. In this section, we explain how we extend our model to do just that.

4.1. Inferring statement satisfiability

A key component of computing the model of Section 3.2 is the ability to evaluate the satisfiability of a statement on a specific row of the data set. For rows containing no missing values, the probability of this event equals the characteristic function of the statement, as stated in Equation (22). If, however, a row in the database contains entries with missing values, this is no longer true. Instead, a verdict can only be made probabilistically, using the known part as observations and inferring the missing one from the beliefs dictated by the model.

Recall that $\boldsymbol{d}$ denotes the N-dimensional vector containing the values of a data set row. Some entries in this vector might be missing, and it is helpful to distinguish the missing values from the known ones. Hence, we overload the notation and let $\boldsymbol{d} = (\boldsymbol{d}_{\text{k}}, \boldsymbol{r}_{\mu})$, where $\boldsymbol{d}_{\text{k}}$ is the vector of values for the known

variables and $\boldsymbol{r}_\mu$ is the random variable that corresponds to the missing values. In other words, if we use $\boldsymbol{\mu} \subseteq \{1, \ldots, N\}$ to index the variables whose value is missing, we have $\boldsymbol{r}_\mu = (r_i)_{i \in \boldsymbol{\mu}}$ taking values in the domain $\mathcal{V}_{\boldsymbol{\mu}} = \times_{i \in \boldsymbol{\mu}} \mathcal{V}_i$.

In this section, we are interested in the event that a given statement q is satisfied on a random row $\boldsymbol{r}$ sampled according to probability $p_{\mathcal{M}}$, in the presence of evidence from a database row $\boldsymbol{d}$. We denote this event as κ_q, and write the associated probability as

$$\begin{aligned} \mathbb{P}(\kappa_q \mid q, \boldsymbol{d}, \mathcal{M}) &= \mathbb{E}_{\mathcal{M}}[\chi_q(\boldsymbol{r}) \mid \boldsymbol{d}] = \mathbb{E}_{\mathcal{M}}[\chi_q(\boldsymbol{r}_\mu) \mid \boldsymbol{d}_{\mathrm{k}}] \\ &= \int_{\mathcal{V}_{\boldsymbol{\mu}}} \chi_q((\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}}))\, p_{\mathcal{M}}((\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}}))\, \mathrm{d}\boldsymbol{r}_\mu \ , \end{aligned} \tag{26}$$

where from now on we omit the implied dependency on the model $\mathcal{M}$.

Note that the probability is conditioned on both the model and the statement to emphasise that we are only interested in whether or not the statement holds true, rather than in the model prior or the choice of the statement itself.

In the case of fully known data $\boldsymbol{d} = (\boldsymbol{d}_{\mathrm{k}}, \emptyset)$, as expected, this probability simplifies into the characteristic function

$$\mathbb{P}(\kappa_q \mid q, \boldsymbol{d}) = \chi_q(\boldsymbol{d}_{\mathrm{k}}) \ . \tag{27}$$

Using Equation (17), the probability density assigned to $\boldsymbol{r}$ can be written as

$$p_{\mathcal{M}}(\boldsymbol{r}) = \frac{1}{Z} \prod_{s \in S} \chi_s(\boldsymbol{r}), \tag{28}$$

where the product runs over the characteristic functions $\chi_s(\boldsymbol{r})$ of all constraint statements that are embedded in the model. Then, to compute the satisfiability probability of statement q over row $\boldsymbol{d} = (\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}})$, we take the expectation over the random part $\boldsymbol{r}_\mu$. Expressing the model probability as per Equation (28) yields

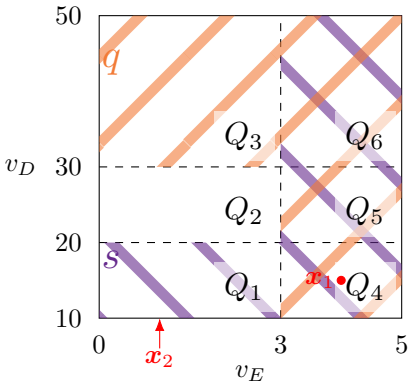
$$\mathbb{P}(\kappa_q \mid q, \boldsymbol{d}) = \frac{1}{Z} \int_{\mathcal{V}_{\boldsymbol{\mu}}} \chi_q((\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}})) \prod_{s \in S} \chi_s((\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}}))\, \mathrm{d}\boldsymbol{r}_{\boldsymbol{\mu}} \ . \tag{29}$$

As in Section 3.1, we can use domain quantisation to compute the integrals in Equation (29) more efficiently. In other words, we exploit the fact that the statements in $S \cup \{q\}$ partition the domain into regions within which the probabilistic density is constant.

Given the statements in S and a row $(\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}})$, we consider the regions of the domain $\mathcal{V}$ that at once contain the known values $\boldsymbol{d}_{\mathrm{k}}$ and satisfy the statements in S. We denote the size of these regions by $w_S((\boldsymbol{d}_{\mathrm{k}}, \boldsymbol{r}_{\boldsymbol{\mu}}))$. Notice that $w_{S \cup \{q\}}(\cdot)$ bounds both $w_S(\cdot)$ and $w_{\{q\}}(\cdot)$ from below. We can now rewrite Equation (29) as

$$\mathbb{P}(\kappa_q \mid q, \boldsymbol{d}) = \frac{1}{Z} w_{S \cup \{q\}}(\boldsymbol{d}_{\mathrm{k}}) \ . \tag{30}$$

Example 4. Consider the two real-valued variables v_D and v_E from our previous examples, with domains respectively $[10, 50]$ and $[0, 5]$. Assume that the current model contains the single constraint statement $s = [v_D < 20] \vee [3 < v_E]$. Our task is to assess the probability that the query statement $q = [30 < v_D] \vee [3 < v_E]$ is satisfied on the data value combinations $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ listed on the right-hand side of Figure 4. Each value combination consists of a pair of values, one for



$s = [v_D < 20] \vee [3 < v_E]$
$q = [30 < v_D] \vee [3 < v_E]$

	$\boldsymbol{x}_1$ $(15, 4)$	$\boldsymbol{x}_2$ $(\varnothing, 1)$	$\boldsymbol{x}_3$ $(\varnothing, \varnothing)$
$w_{\{s\}}(\boldsymbol{x})$	30	30	110
$w_{\{q\}}(\boldsymbol{x})$	20	60	140
$w_{\{s,q\}}(\boldsymbol{x})$	20	0	80

Figure 4. Partitions of the variable domains and region sizes for three different value combinations; s and q are the constraint and query statements respectively.

v_D and one for v_E, respectively, with $\varnothing$ denoting a missing value. The resulting domain partitions are shown on Figure 4.

When computing the satisfiability probability for fully known value combinations only one region per combination needs to be considered: the region where the corresponding data point lies. Hence, for the complete data value $\boldsymbol{x}_1$ we only need to consult region Q_4. For the data value $\boldsymbol{x}_2$, the set of regions $\{Q_1, Q_2, Q_3\}$ must be consulted. Since both of its entries are missing, the entire domain must be consulted, when computing the satisfiability probability of $\boldsymbol{x}_3$.

4.2. Selecting constraining statements

The presence of missing values introduces uncertainty when evaluating whether a statement holds on particular data row, and therefore we need to redefine the decision function $\hat{\phi}$. In fact, we can no longer use the characteristic functions to select the subset of statements that constrain the model associated to a given data row, as in Equations (18) and (19).

Instead, for each data set row $\boldsymbol{d}_i$ and its associated set of constraining statements S_i, we decide whether statement s or its complement is satisfied on the row depending which of these two events is more likely. Then, if the statement is more likely to be satisfied than its complement, we embed it in the row model, associated to a certain probability $\pi_s = 1$. Essentially, selecting the model constraints in this way corresponds to employing the maximum likelihood criterion in order to decide between the complementary events "s is satisfied on $\boldsymbol{d}_i$" and "$\neg\, s$ is satisfied on $\boldsymbol{d}_i$".

To avoid dependency on the partition function Z, we base the decision on the likelihood ratio between the occurrence of the statement and the occurrence of its complement, defined as

$$\mathrm{lr}(s, \boldsymbol{d}_i, S_i) \coloneqq \frac{\mathbb{P}\left(\kappa_s \mid s, \boldsymbol{d}_i, S_i\right)}{\mathbb{P}\left(\kappa_{\neg\, s} \mid s, \boldsymbol{d}_i, S_i\right)} = \frac{w_{S_i \cup \{s\}}(\boldsymbol{d}_i)}{w_S(\boldsymbol{d}_i) - w_{S_i \cup \{s\}}(\boldsymbol{d}_i)}\,, \tag{31}$$

where the last equality uses the formula for the probabilities introduced in Equation (30), as well as the fact that $\mathbb{P}\left(\kappa_s \mid s, \boldsymbol{d}_i, S_i\right) = 1 - \mathbb{P}\left(\kappa_{\neg\, s} \mid s, \boldsymbol{d}_i, S_i\right)$.

Finally, we define $\hat{\phi}$ to be the function that returns a hard decision based on the likelihoods ratio between the events κ_s and $\neg\,\kappa_s$,

$$\hat{\phi}(s, \boldsymbol{d}_i, S_i) \coloneqq \begin{cases} 1 & \text{if } \mathrm{lr}(s, \boldsymbol{d}_i, S_i) > 1, \text{ and} \\ 0 & \text{otherwise}\,, \end{cases} \tag{32}$$

which we use as a replacement for the characteristic function $\chi_s(\boldsymbol{d})$ for selecting the constraining statements.

Notice that the model is built by incorporating statements iteratively. Hence, when considering whether to embed a statement s, the decision relies on the likelihood ratio computed based on the statements previously added into the model. To reflect this, we consider the set of constraints to be ordered, making it effectively a sequence $S = (s_1, \ldots, s_n)$. Let us denote $S_i^{(j)}$ the set of constraints applied on the i-th row at stage j. It can be defined recursively as follows

$$S_i^{(0)} \coloneqq \emptyset\,,$$
$$S_i^{(j+1)} \coloneqq \begin{cases} S_i^{(j)} \cup \{s_j\} & \text{if } \hat{\phi}(s_j, \boldsymbol{d}_i, S_i^{(j)}) = 1 \text{ and} \\ S_i^{(j)} & \text{otherwise}\,, \end{cases}$$

because a constraint's satisfiability probability can only increase once the constraint has been added to the model.

The extended models are depicted in the graphical model of Figure 3 (including the elements in blue).

5. Algorithms

The models described in Section 3 can be used for different tasks, such as generating synthetic data sets and ranking patterns. We focus on the latter, which in turn involves two main operations: (i) training the model by incorporating new information in the form of patterns and (ii) querying the model, i.e. evaluating the satisfiability probability for a pattern. In this section, we present the algorithmic procedures for carrying out these two main operations in practice.

Recall that the patterns considered here consist of logical statements (redescriptions) and their supporting rows.

5.1. Training the model

As explained in the previous section, our models are mixtures of row models. Each of the row models is maintained in a factorised form where different factors involve disjoint sets of variables that do not interact in any statements, so as to allow independent computations. In addition, any given factor is shared by only a subset of the row models.

Our models are maintained as a set of factors $\mathcal{F}$. Each factor f contains a set of statements S_f and applies to a subset of rows I_f. Accordingly, it is represented as the pair $f = (S_f, I_f)$. Overloading the notation, we denote by $\mathrm{vars}(f)$ the set of data variables associated with f, that is, $\mathrm{vars}(f) = \bigcup_{s \in S_f} \mathrm{vars}(s)$. For any given factor f, $\mathrm{vars}(f)$ and I_f define a tile in the data set, and

$$J_i = \{f \in \mathcal{F} : \mathrm{vars}(f) \cap \mathrm{vars}(s) \neq \emptyset,\ i \in I_f\} \tag{33}$$

Algorithm 1: TrainModel

input : model $\mathcal{F}$, new statement s
output : updated model $\mathcal{F}$

1 $\mathcal{J} \leftarrow \{J_i : i \in \text{supp}(s)\}$; // See Equation (33)
2 **foreach** $J \in \mathcal{J}$ **do**
3 $\quad I_J \leftarrow \bigcap_{f \in J} I_f$; // Collect cluster rows
4 $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \left\{ \left(\{s\} \cup \bigcup_{f \in J} S_f, I_J \right) \right\}$; // Add new factor
5 $\quad$ **foreach** $f \in J$ **do** // Update overlapping factors
6 $\quad\quad I_f \leftarrow I_f \setminus I_J$; // Update rows
7 $\quad\quad$ **if** $I_f \neq \emptyset$ **then**
8 $\quad\quad\quad \mathcal{F} \leftarrow \mathcal{F} \setminus \cup\{f\}$; // Delete
9 **return** $\mathcal{F}$;

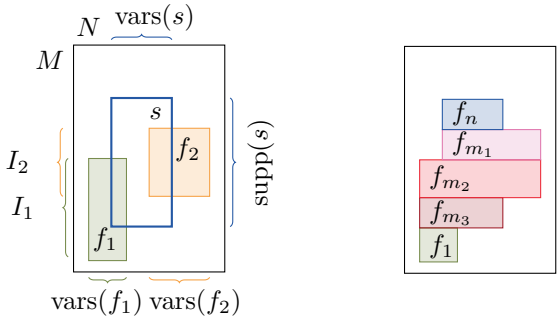


Figure 5. From two existing factors and a new statement (left) to the updated model containing five factors (right).

is the set of all factors which overlap with a statement s and contain row i in their tile. Note that factor tiles do not overlap, due to the way they are constructed. For any given statement s, vars(s) and supp(s) also define a tile in the data set.

Training the model entails incrementally incorporating new statements into it. At each step, the task of the main training procedure is to update the model factors to incorporate the newly appended statement, while preventing factors from overlapping, so that their independence is preserved.

The pseudo-code for this procedure, TrainModel, is presented in Algorithm 1, and works as follows. When adding a new statement s to the model, we form the collection $\mathcal{J}$ of the sets of existing factors that overlap with s as per Equation (33) (line 1). Then, for each cluster J we create a new factor and add it to the model (line 4). The new factor applies exactly to the rows of the cluster (line 3) and contains all statements from the existing overlapping factors, in addition to s. We also update the set of rows to which the existing overlapping factors apply (line 6), deleting the factor altogether if it no longer applies to any row (line 8).

In the example shown in Figure 5, the model contains two factors, f_1 and f_2. A new statement s is to be added, triggering the creation of several new factors. The factors f_{m_1}, f_{m_3} and f_{m_2} are created by merging the rows where s overlaps respectively with f_1, with f_2, and with both, while factor f_n is created for the rows that did not overlap with either (corresponding in Algorithm 1 to the case

Algorithm 2: QUERYMODELROW

```
   input  : model F, statement s
   output : probability p_ROW(s | D; F)
1  F_s ← {f ∈ F : vars(f) ∩ vars(s) ≠ ∅};
2  (w, v) ← QUANTISATION(s, F_s);
3  Z_s ← Σ_i w_i;
4  p ← (v ⊗ w)/Z_s ;                          // ⊗ is the element-wise product
5  p ← MARGINALISEANDNORMALISE(s, F_s, p);
6  return p;
```

of an empty row cluster $J = \emptyset$). The updated f_1 remains, although it now covers fewer rows, whereas f_2 has been deleted.

Note that the type of the domain of the variables only affects TRAINMODEL implicitly, through the satisfiability assessment $\chi_s(\cdot)$ of the newly appended statement s over the data set rows.

5.2. Querying the model

We now turn to the procedure that allows us to query a model. That is, given a model that has been trained as explained above, the original data set and a pattern extracted from the data set, the aim is to evaluate the satisfiability probability of the pattern.

The main building block in computing this probability is the evaluation of the satisfiability probabilities over single rows, essentially the probabilities predicted by the row models.

Querying a row model. Each row model is parameterised by the subsets of statements that occur on the corresponding data row. The satisfiability probability $p_{\texttt{ROW}}(s \mid D; \mathcal{F})$ of a statement for a given row model is computed by QUERYMODELROW as shown in Algorithm 2 and works as follows.

At first, we collect in $\mathcal{F}_s$ the factors of the model that share variables with statement s. Next, the domain of the variables can be partitioned into regions over which the value of the characteristic function of s is constant, by syntactically parsing the statement and collecting all the involved thresholds (cf. Section 3.1). Since, by construction, the value of the characteristic function of s is the same in all points of a region, we can compactly represent the result of this quantisation by a pair of vectors $(\boldsymbol{w}, \boldsymbol{v})$, where $\boldsymbol{w}_i$ measures the area of region i, and $\boldsymbol{v}_i$ indicates whether s is satisfied on that region (line 2). Using this information, we are able to compute the probabilities for the different regions satisfying the statement (lines 3–4). Finally, the probability of each region of the domain is appropriately re-weighted with respect to the different factors in $\mathcal{F}_s$ (line 5). More specifically, the statement s together with the subset $\mathcal{F}_s$ form a Junction Tree (Barber, 2012), and we may therefore employ the Message Passing steps of the Junction tree algorithm to efficiently re-scale the initial probabilities. In this way, the effect of each overlapping factor is taken into account, while the corresponding constraints are respected, to finally yield the probability of statement s.

Note that the complexity of the querying procedure is dominated by the computation of the partition function (lines 2–3) and is exponential with respect to the number of variables involved. This computation yields vectors $\boldsymbol{w}$ and $\boldsymbol{v}$ of size exponential with respect to the number of variables appearing in the

Algorithm 3: RankPatterns

```
   input  : data set D, set of statements S with selected statement s_0
   output : ordered list of statements O
1  F ← TrainModel(∅, s_0);
2  O ← (s_0); S ← S \ {s_0};
3  while S ≠ ∅ do
4  |  s* ← s ∈ S, minimising QueryModel(F, D, s);
5  |  F ← TrainModel(F, s*);
6  |  O ← (O, s*);
7  |  S ← S \ {s*};
8  return O;
```

model $\mathcal{F}$ and statement s. Indeed, in the case of Boolean variables, the problem of computing the partition function Z_s (line 3) can be trivially reduced to the #SAT problem (counting the number of satisfying assignments of a Boolean expression) and vice-versa. Since the latter is a known #P-complete problem, so is the former. Additionally, Boolean variables constitute a subset of the query language that we support, so our problem is at least #P-hard. Furthermore, it can be shown that querying the maximum entropy probability of a statement is PP-hard (Tatti, 2006), even for queries without disjunctions, so that in general, approximations of the partition function are also intractable. Interestingly, while an exact evaluation would still need an exponential evaluation time, the partition function could be approximated more efficiently in the case of largely uneven quantisations of the variable domains. Such an approximation scheme could evaluate the validity on the domain partitions, starting from the largest ones and progressively tightening an upper and lower bound on the partition function value, until a sufficiently tight interval is obtained.

Putting the rows together. Now that we know how to compute the probabilities for individual rows, we need to combine them together. The two models, `MEBLK` and `MEALL`, offer two alternatives for doing so. The overall probability returned by `MEALL` is simply the average of $p_{\texttt{ROW}}(s \mid D; \mathcal{F}_i)$ over all rows i in the data set, while `MEBLK` only averages over rows that satisfy the statement under evaluation, that is, over $\mathrm{supp}(s)$.

5.3. A ranking scheme

Combining the two main operations of training and evaluation explained above, the procedure for ranking patterns is presented in Algorithm 3.

This procedure takes as input a data set and a collection of statements, with one of them designated to initialise the model training (line 1). This method then iteratively constructs a ranking of all the statements. In each step, the model is queried to identify the statement with lowest predicted probability (line 4), which is essentially the statement whose observation is most surprising at that point. This statement is incorporated into the model (line 5), appended to the list of results and removed from the set of candidates (lines 6 and 7). The procedure iterates until all statements have been ranked.

6. Related Work

Redescription mining. Redescription mining was introduced by Ramakrishnan et al. (2004) for Boolean data. Later, Zaki and Ramakrishnan (2005) and Parida and Ramakrishnan (2005) further developed the theory of Boolean redescription mining, Galbrun and Miettinen (2012*a*) extended it to real-valued data, and Galbrun and Kimmig (2014) studied redescription mining in relational data.

Various algorithms have been proposed for mining the redescriptions. These can be divided into methods based on decision-tree induction (Ramakrishnan et al., 2004; Zinchenko et al., 2015) and methods that grow the queries using a greedy heuristic (Gallo, Miettinen and Mannila, 2008; Galbrun and Miettinen, 2012*a*) (see also Galbrun and Miettinen, 2018). In addition, some methods have also been proposed to visualise and interact with the redescriptions (Galbrun and Miettinen, 2012*b*; Galbrun and Miettinen, 2014; Mihelčić and Šmuc, 2016)

Redescription mining is related to rule mining techniques that aim at discovering association rules (Agrawal and Srikant, 1994) or subgroups (Novak, Lavrač and Webb, 2009), for instance, and to classification and clustering techniques including subspace clustering (Kröger and Zimek, 2009; Agrawal, Gehrke, Gunopulos and Raghavan, 1998) and multi-view clustering (Bickel and Scheffer, 2004), among others. However, its goal of finding descriptive, interpretable patterns across several data sets is a distinguishing feature.

Maximum entropy models. The maximum entropy principle, formalised by Jaynes (2003) and based on his earlier work (Jaynes, 1982), states that the distribution that has the maximum entropy, subject to given constraints, does not add any bias beyond what is assumed in these constraints.

It is a versatile principle which has been applied from ecological modelling (Phillips, Anderson and Schapire, 2006) to natural language processing (Berger, Pietra and Pietra, 1996), and has been used to construct a condensed representation of the data set to be used for query approximation (Mannila, Pavlov and Smyth, 1999; Pavlov, Mannila and Smyth, 2003) or to compute degrees of belief (Grove, Halpern and Koller, 1992), to name but a few examples.

De Bie (2011) proposed to use maximum entropy models to evaluate the *subjective interestingness* of patterns and in the present work we follow his general idea. In another line of work, Tatti and Vreeken (2011) proposed an approach based on maximum entropy and Kullback–Leibler divergence for comparing sets of noisy tiles mined from Boolean data sets and a maximum entropy model to iteratively discover biclusters over multiple related data sets was later presented (Wu, Vreeken, Tatti and Ramakrishnan, 2014).

Related modelling approaches include the use of Markov Random Fields (Wang and Parthasarathy, 2006) and the minimum description length (MDL) principle (Vreeken and van Leeuwen, 2011; van Leeuwen and Galbrun, 2015), but do not offer a similar flexibility and modelling power to accommodate an expressive query language.

Similarly to most existing approaches (Jaroszewicz and Simovici, 2002; Tatti, 2008), we apply ranking and filtering as a post-processing, meaning that we require as an input a collection of candidate patterns. Mampaey, Vreeken and Tatti (2012) proposed a method for finding interesting itemsets which mines candidates on the fly.

Pattern-based maximum entropy models. De Bie (2011) proposed a maxi-

mum entropy model to identify interesting tiles in Boolean data sets while taking into account assumptions on the expected row and column sums, that is, with prior information given as the parameters of the Rasch model (Rasch, 1960). Kontonasios, Vreeken and De Bie (2011) extended this approach to real valued data, with prior information taking the form of means, variances and histograms for the rows and columns. An iterative ranking scheme (Kontonasios et al., 2013), like ours, was then derived from this static model, which was also further extended to increasingly complex priors and pattern types (Kontonasios and De Bie, 2012; Kontonasios and De Bie, 2015).

Concurrently, Mampaey, Tatti and Vreeken (2011) presented a maximum entropy model, named `MTV`, for iteratively mining interesting itemsets from a Boolean data set, using as prior knowledge the frequencies of individual items.

The most obvious difference between these models is on the type of prior information and patterns that they handle. Some of these works (e.g. De Bie, 2011; Mampaey et al., 2011) are limited to binary tiles (or itemsets). These can be represented as monotone, conjunctive Boolean queries, and comprise only a subset of the patterns we can handle.

The method of Kontonasios et al. (2011) handles numerical data, like ours, but it deals with tile-like blocks. That is, their patterns are defined by specifying row and column indices, then looking at the distribution of values. Our patterns, redescriptions, are instead defined by specifying value ranges for some variables, then selecting the rows which satisfy these conditions. Our domain quantisation approach shares similarities with the maintenance of tiles histograms. We start with a coarse histogram and progressively adjust the bins based on the thresholds appearing in the predicates of the constraining statements.

To the best of our knowledge, we are the first to deal with arbitrary logical patterns, including disjunctions and negations, beyond the typical conjunctions.

A further distinction is between approaches which model rows of the data set individually (Tatti, 2008; Mampaey et al., 2011; Mampaey et al., 2012) versus those which consider the data set as a whole (De Bie, 2011). The `MTV` model corresponds to a special case of our row model (see Section 3.1) for non-certain validities. We enforce each pattern on the relevant subset of entities of the data set, whereas in their model all patterns are applied on the same, single row. In other words, they assume that the entities of the data set are identically distributed. In this respect, our approach is similar to that of De Bie (2011), which assumes that the distribution of tiles might differ from one another.

Another important difference with the `MTV` model is that we do not incorporate the absence of a pattern as information into either of our models. This is one more reason why the information contained in our models deviates from simple pattern frequencies.

Despite the similarities between our models and the original model of De Bie (2011), there is no straightforward way to compare them. In particular, De Bie (2011) employs an information-content criterion that cannot be transferred to the case of redescriptions, like other works in the literature (Kontonasios et al., 2011; Kontonasios et al., 2013). The numerical extension of the last two models are also further unsuitable as competitors, since they cannot handle the requirements on value ranges specified in the predicates of redescriptions.

7. Experimental Evaluation

In this section, we present experiments to investigate the behaviour and performance of our algorithms and compare our two models `MEALL` and `MEBLK`.

We implemented our algorithm[3] using MATLAB for the high level procedures and C++ for the core operations. All experiments were run on a cluster with 16 cores (at 2.4 GHz and with 48 GB of memory). The sets of redescriptions for the real-world data were mined in advance using the ReReMi algorithm (Galbrun and Miettinen, 2012*a*) in the Siren redescription mining interface[4] (Galbrun and Miettinen, 2012*b*; Galbrun and Miettinen, 2014).

Among existing redescription mining algorithms, we selected the ReReMi algorithm (Galbrun and Miettinen, 2012*a*). This choice is motivated by several properties of this algorithm: (i) it offers a rich query language that provides hard instances for the maximum-entropy calculations, (ii) it can handle missing values, (iii) it is rather susceptible to redundancy in the result set.

We proceed with a series of experiments on synthetic data sets, before moving on to real-world data.

7.1. Evaluation on synthetic data sets

Our goal in this series of experiments with synthetic data sets is to shed light on the different aspects that impact the complexity of the computations in a controlled experimental setting. Our focus here is on the quantitative evaluation of the performance of the algorithms and our primary measure in these experiments is therefore the wall-clock time for training and querying the model.

Note, however, that our objective here is not to present a highly efficient and fully optimised algorithm, but to demonstrate the feasibility of the approach. The evaluation on synthetic data is therefore designed to compare empirically the impact of the various aspects of the data and the queries on the runtimes.

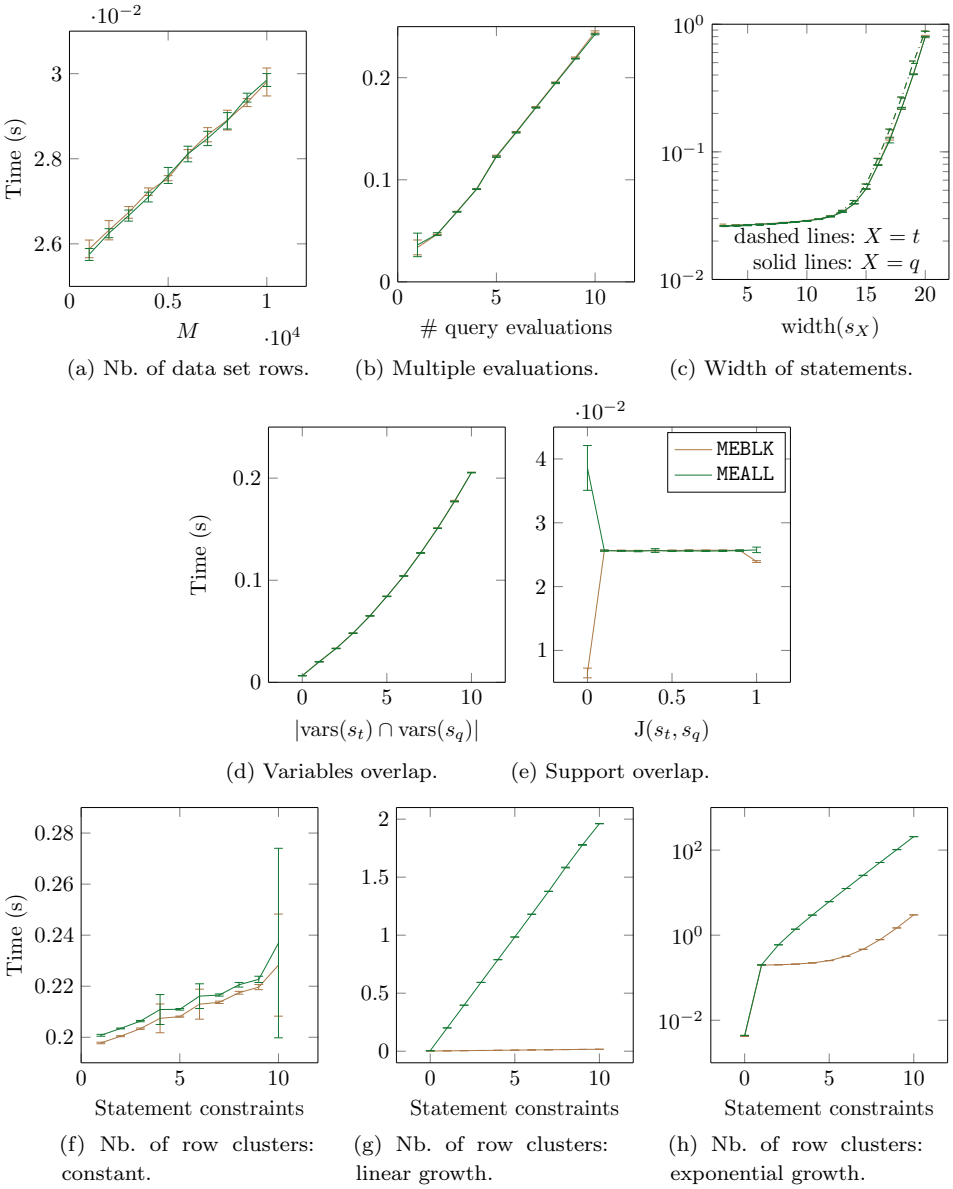
We start by considering the simplest case of a model trained with a single Boolean statement and queried with another Boolean statement. That is, we first train the model with statement s_t, then query the trained model to evaluate the occurrence probability of statement s_q.

We call *width* of a statement the number of distinct variables it contains, that is, $\text{width}(s_t) = |\text{vars}(s_t)|$, while the *overlap* between two statements is simply the number of variables they have in common, that is, $\text{over}(s_t, s_q) = |\text{vars}(s_t) \cap \text{vars}(s_q)|$. We can fully control these parameters by using logical conjunctions and choosing suitable sets of variables for our statements.

Our base case is as follows. We let $s_t = v_A \wedge v_B \wedge v_C$ and $s_q = v_B \wedge v_C \wedge v_D$, so that $\text{width}(s_t) = \text{width}(s_q) = 3$ and $\text{over}(s_t, s_q) = 2$. We fix the number of rows in our data set to $M = 1000$, of which 70% satisfy both s_t and s_q while the rest are evenly distributed over the remaining 3 truth combinations.

Starting from this base case, we can study the impact of different parameters in turn, by repeating the training and querying of our model while varying a chosen parameter of the problem. For each configuration, we record the total runtimes for training and querying. The results are reported in Figure 6. The

[3] The source code is available at http://siren.mpi-inf.mpg.de/max-ent/.
[4] http://siren.gforge.inria.fr, accessed 13 December 2017



(a) Nb. of data set rows. (b) Multiple evaluations. (c) Width of statements.

(d) Variables overlap. (e) Support overlap.

(f) Nb. of row clusters: constant. (g) Nb. of row clusters: linear growth. (h) Nb. of row clusters: exponential growth.

Figure 6. Experimental results on synthetic data sets: runtimes for training and querying averaged over 10 repetitions. The width of the error bars equals twice the standard deviation.

markers and the error bars indicate respectively the mean runtimes and twice the standard deviation for 10 repetitions of each configuration.

The first parameter we consider is the number of rows in the data set (Figure 6a), and the second parameter is the number of query evaluations, that is,

we train the model once with s_t, before repeatedly querying it with the same statement s_q (Figure 6b). In both cases, the behaviour of both models is linear.

Next, we study the impact of the shape of the statements by fixing the width of one statement to 3, while increasing that of the other, always keeping their overlap constant. The models show the same exponential increase of time in both cases (Figure 6c). Indeed, in both cases factors of equal sizes need to be evaluated.

Let us now study the impact of interactions between the statements, in terms of shared variables and of shared rows. The impact of varying variable overlap, as measured by $\mathrm{over}(s_t, s_q)$, with constant widths $\mathrm{width}(s_t) = \mathrm{width}(s_q) = 11$, is shown in Figure 6d. On the other hand, the impact of varying row overlap, as measured by $\mathrm{J}(s_t, s_q)$, is shown in Figure 6e. The latter shows further evidence that the number of different value combinations matters, rather than the number of rows in the overlap. Indeed, since the algorithm processes rows clustered according to the combination of factors they participate in, the number of such clusters in the data set has a major impact on the runtimes.

To further investigate the complexity arising from the structure of the data set through the number of clusters it induces, we modify our setup as follows. Starting with an empty model, in each iteration i we add a different statement s_i and then query the updated model with the same statement s_q. All statements are defined over 10 Boolean variables. The query statement is a simple conjunction $s_q = v_1 \wedge \ldots \wedge v_{10}$, whereas the training statement added at the i-th iteration has the form $s_i = v_i \wedge \left(\left(\bigvee_{k \neq i} v_k \right) \vee \neg \left(\bigvee_{k \neq i} v_k \right) \right)$. The training statements s_i all have $\mathrm{width}(s_i) = 10$ and $\mathrm{over}(s_i, s_q) = 10$, but have all distinct truth tables.

Next, we create 3 different data sets, each with $M = 1024$ rows. The first one only contains rows from $\{\mathbf{0}, \mathbf{1}\}$, that is, with all of their values equal, so that at most 2 clusters will be created, regardless of the number of training statements added to the model. The second data set contains rows from the standard basis $\{\boldsymbol{e}_k : k = 1, \ldots, 10\}$, so that the number of clusters increases by one with each added training statement. Finally, the third data set contains rows with all the possible Boolean combinations of the 10 variables, so that the number of clusters grows exponentially, reaching 2^i at the i-th iteration and thus representing the worst case scenario.

The runtimes for each iteration on these three data sets are reported in Figures 6f–h, respectively. We observe that the runtime for the `MEALL` model closely follows the number of clusters, since they all have to be evaluated. On the other hand, the `MEBLK` model only evaluates the clusters that support s_q. The small additional overhead for the `MEBLK` model in the last case is due to the training phase, which still needs to track all clusters.

Finally, we look at the impact of the presence of missing values on the runtime. As a starting point, we use a subset of the DBLP data set, a fully Boolean data set which we present in more details in Section 7.2. We generate synthetic variants with different densities of missing values by deleting some entries at random. More specifically, we select a fraction α of the rows from the data set then replace a fraction β of the values on these rows by the missing value token. We mined redescriptions from the complete data set and let our models rank these redescriptions with each data set variant. The runtimes for ranking 20 queries while keeping $\alpha = 1$, that is, spreading the missing values over the entire data set, but for different values of β, that is, varying the ratio of deleted values, is shown in Figure 7 (left). Vice-versa, in Figure 7 (right) we fixed the ratio of

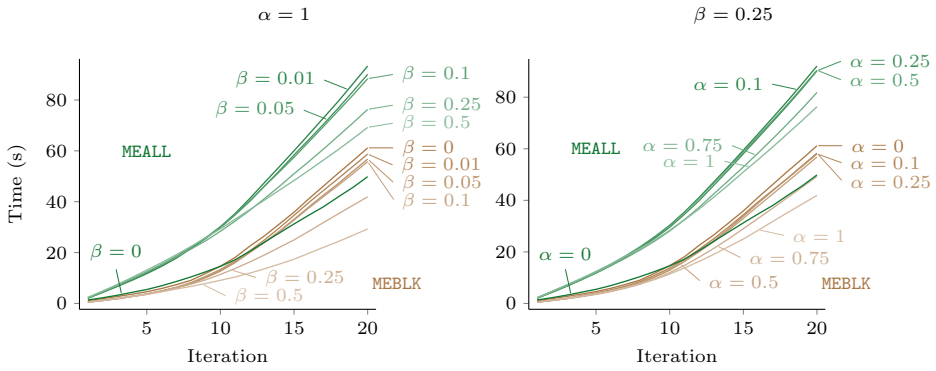


Figure 7. Runtimes for ranking 20 queries from a data set with various levels of missing values.

deleted values but vary the ratio of rows affected. Note that the cases where $\alpha = 0$ or $\beta = 0$ correspond to using the complete data set.

We observe that the presence of missing values results in longer runs with `MEALL`. As the amount missing values increases, the runtime decreases, but never reaches the runtimes on the complete data. The effect on `MEBLK` is similar, in the sense that more missing values result in shorter runs, but without the initial cost of having any missing entries at all.

7.2. Evaluation on real-world data sets

After this systematic investigation of the performance of our algorithms and models on synthetic data sets, we now present results from real-world data for a qualitative assessment.

Our goal with these experiments is to show that the approach is applicable to real-world data and that the proposed approach provides added value, in the sense that the rankings obtained with this advanced mechanism cannot be matched with some straightforward ranking criterion, such as accuracy or coverage.

We first provide quantitative evidence that the obtained rankings differ from those obtained with other methods. Then, we show top ranked redescriptions from real-world examples; these serve as subjective evidence that the selected redescriptions indeed provide a more nuanced representation, as well as a better insight into the data than, for instance, simply considering the most accurate redescriptions.

Data sets. We use four data sets, coming from four different domains and having different properties, in terms of their sizes and of the types of variables they involve.

First, the DBLP data set is extracted from the popular computer science bibliography.[5] The entities are researchers and one side records major conferences where they published, while the other side records the co-authorship graph ($|E| = 2345$, $|V_{\mathbf{L}}| = 19$, and $|V_{\mathbf{R}}| = 2345$; both sides are Boolean).

[5] http://www.informatik.uni-trier.de/~ley/db/, accessed 13 December 2017

Our second data set, Bio, comes from the domain of ecology. The entities represent geographic areas of Europe, the left-hand side records the presence of various mammals species (Mitchell-Jones et al., 1999), while the right-hand side consists of bioclimatic variables, that is, monthly average rainfall and monthly average, minimum, and maximum temperatures (Hijmans, Cameron, Parra, Jones and Jarvis, 2005) ($|E| = 2575$, $|V_{\mathbf{L}}| = 194$, and $|V_{\mathbf{R}}| = 48$; the species records are Boolean and the climate variables real-valued).

Our third data set, Cover,[6] also comes from the domain of ecology. The entities represent geographic areas of a national forest in Colorado, USA. The wilderness area, soil type, and cover type constitute the right-hand side variables, while other, topographic variables, such as elevation and slope, are on the left-hand side ($|E| = 581012$, $|V_{\mathbf{L}}| = 10$, and $|V_{\mathbf{R}}| = 45$; all variables on the right are Boolean except for the cover type which is nominal, and all those on the left are real-valued).

Finally, we consider a fourth data set, which contains missing values. The EthnoAtlas data set combines ethnographic information about the cultural norms and practices of various societies around the world, assembled from several sources (Murdock, 1967; Gray, 1999) and made available online,[7] with the climatic profile of the regions where they live ($|E| = 1267$, $|V_{\mathbf{L}}| = 90$, and $|V_{\mathbf{R}}| = 23$; all variables on the right are real-valued except for the major habitat type which is nominal). We use different variants of this data set, one where the ethnographic variables are considered as real-valued variables and one where they are considered as nominal, that is, consisting of unordered values. We obtain further variants with different amounts of missing values by disregarding variables that contain more than 50 ($|V_{\mathbf{L}}| = 75$) and more than 25 ($|V_{\mathbf{L}}| = 49$) percent of missing values respectively. We use indices R and N to denote the real-valued and nominal variants, respectively, combined, when relevant, with indices .5 and .25 for the variants with fewer missing values.

Runtimes. The RankPatterns algorithm was run to rank sets of redescriptions of size $|S| = 100$, 60, and 230, extracted from DBLP, Bio, and Cover, respectively. For each data set, Figure 8 depicts the average time required to query the occurrence probability for each of the remaining candidate redescriptions, during each iteration of the ranking algorithm, ignoring the negligible time required for training the model.

These plots show steep rises in the query times that correspond to iterations where the last training redescription overlaps with the largest factor in the model, which must therefore be extended. As explained in Section 3, our algorithms use factorisation and quantisation to reduce the computational cost of evaluating occurrence probabilities. For this reason, the runtime complexity is dominated by the size of the largest factor in terms of value combinations, which may grow arbitrarily large as the degree of overlap increases. This effect is exacerbated in the case of real-valued variables due to the quantisation becoming finer following some model updates.

Fortunately, statements involving variables that have occurred in earlier selected statements are typically assigned higher probabilities. This means that such overlapping redescriptions are generally deemed uninteresting and pushed

[6] https://archive.ics.uci.edu/ml/datasets/Covertype, accessed 13 December 2017
[7] http://intersci.ss.uci.edu/wiki/index.php/Ethnographic_Atlas#Rdata_format_version_of_Ethnographic_Atlas, accessed 13 December 2017

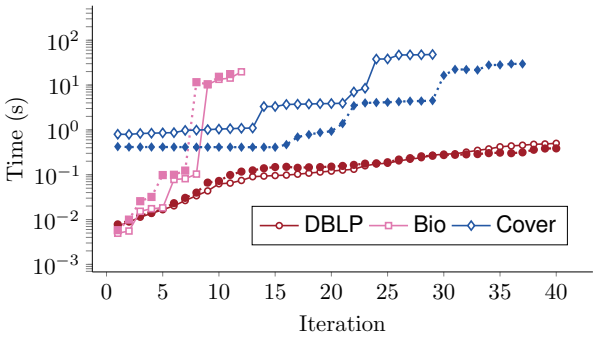


Figure 8. Experimental results on real world data sets: mean runtime per query for MEALL (solid lines) and MEBLK (dashed lines).

Table 3. Ranking conjunctive redescriptions from DBLP with MEALL, MEBLK, and MTV. Step 4.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	MEALL $p(R)$	MEBLK $p(R)$
COLT ∧ ICML	R. Schapire	—	—
EDBT ∧ PODS	A. Silberschatz	—	—
FOCS ∧ SODA	N. Alon	—	—
ICDM ∧ SDM	P. Yu	—	—
VLDB ∧ ICDT	H. Garcia-Molina	.1250	.1250
ICDT ∧ SIGMOD	S. Abiteboul	.1250	.1250
UAI ∧ ICML	D. Koller	.1259	.1484
SDM ∧ KDD	J. Han	.1261	.2083
ICDT ∧ PODS	S. Abiteboul	.1265	.1331
COLT ∧ ICML	M. Kearns	.1277	.2500
COLT ∧ ICML	A. Blum	.1277	.2054
SDM ∧ KDD	P. Yu	.1284	.4674
ICDM ∧ SDM	H. Wang	.1284	.4250
PKDD ∧ SDM	P. Yu	.1284	.3977
FOCS ∧ COLT	Y. Mansour	.1292	.1667
COLT ∧ ICML	R. Schapire	.1313	1.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	MTV $p(R)$ (10^{-2})
COLT ∧ ICML	R. Schapire	—
ICDM ∧ SDM	H. Wang	—
PKDD ∧ SDM	P. Yu	—
UAI ∧ ICML	D. Koller	—
SDM ∧ KDD	J. Han	.0243
COLT ∧ ICML	M. Kearns	.0253
COLT ∧ ICML	A. Blum	.0506
ICDT ∧ PODS	S. Abiteboul	.0521
EDBT ∧ PODS	A. Silberschatz	.0733
FOCS ∧ COLT	Y. Mansour	.0897
ICDT ∧ SIGMOD	S. Abiteboul	.0982
SDM ∧ KDD	P. Yu	.1079
VLDB ∧ ICDT	H. Garcia-Molina	.1194
ICDM ∧ SDM	P. Yu	.1210
FOCS ∧ SODA	N. Alon	.3209
COLT ∧ ICML	R. Schapire	.7251

lower in the ranking, with the beneficial consequence of delaying the formation of larger factors to later iterations of the algorithm.

Comparison to the MTV model. In this section we compare the ranking obtained with our models to the ranking obtained with the model of Mampaey et al. (2011), denoted as MTV.

The MTV algorithm originally mines interesting itemsets. In this comparison, we consider only the DBLP data set, which is fully Boolean, and focus on the subset of redescriptions that consist purely of conjunctions, to accommodate the more restricted pattern language of MTV. Furthermore, we bypassed the tile mining scheme proposed by the authors, using the model to rank the collection of pre-mined redescriptions.

The rankings produced by our models and by MTV after the first four redescriptions have been ranked are shown in Table 3, together with the probabilities assigned at that stage to the remaining redescriptions.

All three models are initialised by first incorporating the most accurate redescription. Note that up to the stage shown here, the top redescriptions

selected by MEALL and MEBLK are identical, but the probabilities assigned to the remaining redrescriptions differ. In particular, the duplicate of the first redescription is assigned a probability of 1 by MEBLK, but a probability of 0.1313 by MEALL. Still, it is considered uninteresting under all models.

Overall, the MTV model assigns much lower probabilities than our models and its updates tend to be more conservative. Note that the MTV model takes into account the frequencies of individual variables as prior information, allowing to build rankings in a more informed manner than our models where ties are broken based on accuracy. Incorporating such information into our model is an interesting avenue for future work.

We observe that our models tend to strongly push down redescriptions that share variables with previously selected redescriptions, MEBLK even more so than MEALL, so that similar redescriptions are ranked far away from each other. This behaviour is desirable since as soon as one redescription is reported, interest on similar ones quickly drops. As a result, redescriptions ranked at the top by our models clearly constitute a more diverse set than those selected by MTV. Indeed, the first four redescriptions selected by our models cover the fields of machine learning, databases, theoretical computer science, and data mining, respectively, compared to only machine learning and data mining for MTV.

Ability to generalise. Next, we consider the ability of the redescriptions to generalise. More specifically, we study whether the redescriptions that are ranked as the most interesting by our models behave differently in terms of their ability to generalise as compared to lower ranked ones. To this aim, we follow the procedure introduced by Zinchenko et al. (2015). That is, we split the entities of the data set into five subsets, mine redescriptions while holding out one subset, then assess the accuracy of the obtained redescriptions on the full original data set. For the DBLP data set, entities were split at random between training set and hold-out set. For Bio, we sampled longitudinal stripes, so as to take into account the north–south trends in climate data. For each redescription, we compute the ratio between its accuracy in the training set and overall.

The reason why we consider the accuracy in the overall data rather than only in the hold-out set is that, in contrast to the models validated in a typical cross-fold setup, our redescriptions constitute *local* models of the data. More specifically, the accuracy of a redescription only considers the entities that are in the supports of either of the queries. If none of the entities in the hold-out set is included in either of the supports, the redescription's accuracy in the hold-out set is undefined.

Here, our aim is not to evaluate whether the models hold everywhere but, instead, where they hold, whether they do so with varying accuracies. Table 4 shows the average of this ratio for redescriptions ranked among the top 10 % as compared to the rest of the redescriptions, for both data sets and both models. The ability to generalise of the top ranked redescriptions appears to be very similar to that of the rest of the redescriptions, meaning that selecting the top-ranked redescriptions per our model does not harm the ability of the results to generalise and that the rank of a redescription is not directly related to its ability to generalise.

Coverage. In this experiment, we compare the rankings obtained with our models to other methods for ranking the redescriptions. More precisely, we look at how the set of entities is covered by the top redescriptions when picked

Table 4. Average accuracy ratio between testing and training sets for the top ranked redescriptions versus the rest.

Data set	Model	J ratio top	J ratio rest
DBLP	MEBLK	0.733 (42 values)	0.707 (425 values)
DBLP	MEALL	0.733 (42 values)	0.707 (425 values)
Bio	MEBLK	0.839 (37 values)	0.910 (403 values)
Bio	MEALL	0.819 (33 values)	0.911 (407 values)

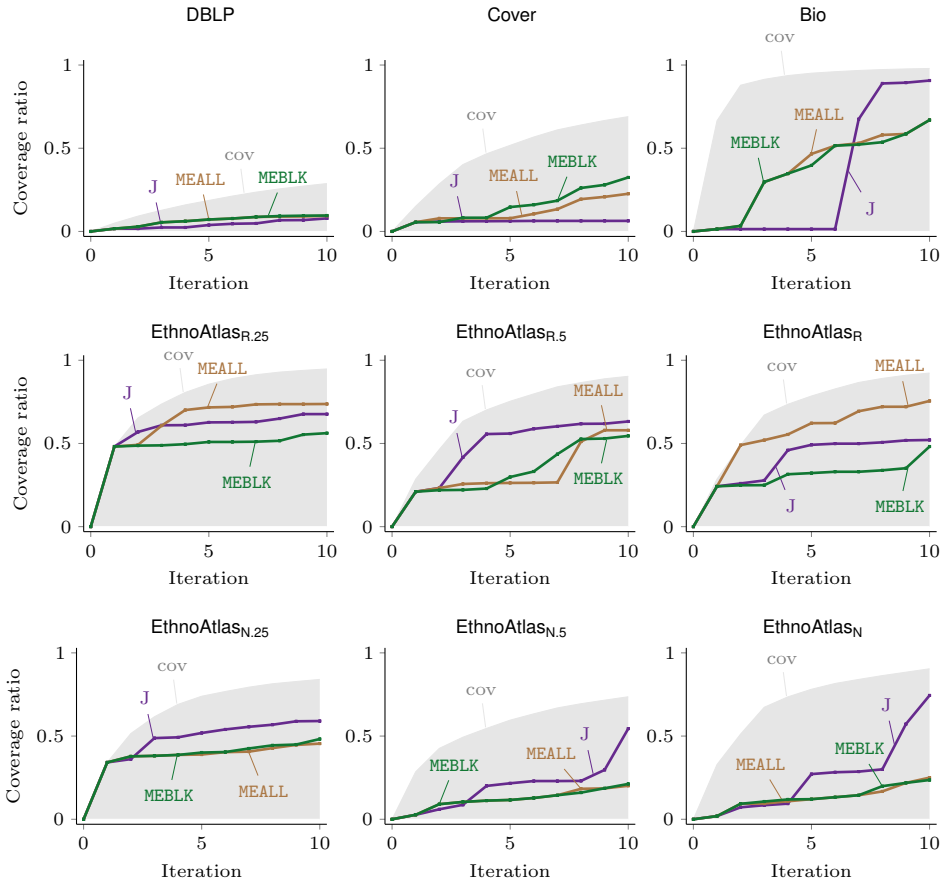


Figure 9. Cumulative ratio of entities covered by the redescriptions when selected according to various rankings.

according to different rankings, as a means to assess how similar these rankings are.

Figure 9 shows the cumulative ratio of entities that are covered by the ten first redescriptions for each ranking. Beside the rankings obtained with our MEALL and MEBLK models, we consider two further rankings. First, we rank the redescriptions in order of decreasing accuracy, denoted by J in Figure 9. Second, we rank the redescriptions by selecting at each step the redescription that covers the most

currently uncovered entities. Clearly, this latter ranking provides the upper-bound for the achievable coverage, and it is denoted by the grey shading in Figure 9.

One further criterion for ranking would be the p-values computed for each redescription, indicating how likely it would be that two independent queries with the observed marginals have as large an intersection as the redescription support. However, this is not useful in practice, due to the fact that most of these p-values are essentially zero by construction.

We observe that `MEALL` either yields a clearly better coverage than `MEBLK` or is on par with it. When it comes to comparing our models to ordering the redescriptions by accuracy, the best coverage depends very much on which data set and what number of redescriptions we consider. This shows that the surprisingness does not reduce to coverage. On first thought, one might guess that the most surprising redescription per our models is the one that covers most of the not-yet-covered observations, but as these results show, that is not the case.

This experiment clearly shows that the rankings obtained with our approach are not equivalent to using a straightforward criterion such as accuracy or coverage.

Example rankings. Finally, we turn to a more qualitative assessment of our proposed ranking procedure.

Our first example illustrates the iterative ranking process on the DBLP data set, showing the evolution of the top-ranked redescriptions over a few steps in Table 5.

Using the ReReMi algorithm, we mined redescriptions from this data set, which were then sorted by accuracy and fed to the ranking algorithm. In the first step, the top redescription is added to the model, the probabilities for the other redescriptions are computed and the list is sorted by increasing occurrence probability, since higher probabilities are associated to less surprising and thus less interesting redescriptions. The second redescription is then added to the model, the probabilities recomputed and the ranking updated. The iterations continue until the entire list has been processed.

Table 5 shows how the redundant redescriptions are pushed away from the top of the list. For example, after adding the first redescription, all other redescriptions with SDM on the left-hand side are pushed out of the table. Notice, however, that *some* overlap is possible, if the redescriptions are otherwise surprising enough: for example, in the 7-th step, ICML appears in the right-hand side of two redescriptions that have already been included in the model.

In this example, the redescriptions typically represent fields of computer sciences, by associating major conferences and key researchers from these fields. Three fields are represented among the most accurate redescriptions (Step 0, Table 5 top left), namely (i) data mining, with for instance the SIAM International Conference on Data Mining (SDM) and researchers including Phillip S. Yu and Jiawei Han, (ii) databases, with the International Conference on Database Theory (IDBT), Renée J. Miller and Serge Abiteboul, as well as (iii) machine learning, with the International Conference on Machine Learning (ICML), the Conference on Learning Theory (COLT), and Robert E. Shapire. Redescriptions selected by our ranking procedure (Step 7, Table 5 bottom right) better represent the diversity of fields in computer science. At the same time, they often combine several conferences to delimit more specific subfields.

Next, we consider redescriptions mined from the Bio data set. In Table 6 we

Table 5. Ranking DBLP with MEBLK. Steps 0, 1, 6 and 7.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J(R)
SDM	P. Yu ∨ V. Kumar	.198
SDM	P. Yu ∨ V. Kumar	.198
COLT ∧ ICML	R. Schapire	.175
COLT ∧ ICML	R. Schapire	.175
COLT	P. Bartlett ∨ M. Kearns	.173
COLT	P. Bartlett ∨ A. Blum	.172
SDM	J. Han ∨ V. Kumar	.166
ICDT	R. Miller ∨ S. Abiteboul	.164
SDM	V. Kumar ∨ H. Wang	.164
COLT	C. Smith ∨ Y. Mansour	.162

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P. Yu ∨ V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	.125
FOCS ∧ SODA	N. Alon	.125
UAI ∧ ICML	D. Koller	.125
COLT ∧ ICML	R. Schapire	.125
COLT ∧ ICML	M. Kearns	.125
COLT ∧ ICML	R. Schapire	.125
ICDT ∧ PODS	S. Abiteboul	.125
VLDB ∧ ICDT	H. Garcia-Molina	.125
COLT ∧ ICML	A. Blum	.125

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P. Yu ∨ V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	—
FOCS ∧ SODA	N. Alon	—
UAI ∧ ICML	D. Koller	—
VLDB ∧ ICDT	H. Garcia-Molina	—
COLT ∧ ICML	M. Kearns	.133
COLT ∧ ICML	A. Blum	.134
FOCS ∧ COLT	Y. Mansour	.143
COLT ∧ ICML	R. Schapire	.147
COLT ∧ ICML	R. Schapire	.147

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P. Yu ∨ V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	—
FOCS ∧ SODA	N. Alon	—
UAI ∧ ICML	D. Koller	—
VLDB ∧ ICDT	H. Garcia-Molina	—
COLT ∧ ICML	M. Kearns	—
ICDT ∧ SIGMOD	S. Abiteboul	.153
ICDT ∧ PODS	S. Abiteboul	.153
FOCS ∧ COLT	Y. Mansour	.161
SDM ∧ KDD	J. Han	.215

show the top of the list of redescriptions, sorted by accuracy, which were fed to the ranking procedure. The top of the output list is shown in Table 7.

In this example, the redescriptions characterise geographic areas based on the mammals species that inhabit them on one hand ($q_{\mathbf{L}}$) and based on the climatic profile, on the other hand ($q_{\mathbf{R}}$). More specifically, the right-hand side variables contain the minimum, maximum, and average monthly temperatures, and total monthly precipitations at the different locations. For instance, variables $t_5^{\sim}$, t_{10}^{+}, p_8 denote the average temperature of May, the maximum temperature of October, and the precipitation of August, respectively.

When ranking the redescriptions by accuracy, we note that the first six all characterise the habitat of the polar bear in terms of temperature. While the conditions are different, with the first climate query fixing the range of the average temperature in May, the second fixing the range of the average temperature in March, and so on, they all impose a similar requirement: low temperatures. Hence, while very accurate, those redescriptions are also highly redundant, and of limited interest when taken together. Instead, the top redescriptions selected by our algorithm exhibit a much greater diversity, involving various species and climate conditions and having support of different sizes spreading over various geographic areas. While they are optimal neither in terms of accuracy nor in terms of entity coverage, these redescriptions are still of fairly high quality when considered individually, while they form a set of good quality when considered together.

8. Conclusions

Thus far, redescription mining was mostly focused on the problem of finding good redescriptions, ignoring the more global problem of finding a good *set* of redescriptions. In this work, we have approached this latter problem from the point of view of maximum-entropy distributions: a redescription is non-redundant if and only if it has a low likelihood under the maximum-entropy distribution,

Table 6. Redescriptions from Bio ranked by accuracy.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$
(1) polar bear	$[-7.07 \leq t_5^{\sim} \leq -3.38]$
(2) polar bear	$[-16.7 \leq t_3^{\sim} \leq -11.5]$
(3) polar bear	$[-4.5 \leq t_{10}^{+} \leq -1]$
(4) polar bear	$[1 \leq t_9^{+} \leq 3.5]$
(5) polar bear	$[-9.6 \leq t_4^{+} \leq -5.6]$
(6) polar bear	$[-11.9 \leq t_3^{+} \leq -7.3]$
(7) b. vole ∨ n. r. vole ∨ s. mouse ∨ h. seal	$[10.9 \leq t_8^{+} \leq 29.9] \wedge [-9.2 \leq t_{12}^{+} \leq 12.8] \wedge [34.7 \leq p_6] \wedge [47.6 \leq p_8]$
(8) w. mouse	$(([2.9 \leq t_3^{+}] \vee [9.7 \leq t_7^{+} \leq 13.2]) \wedge [-3.26 \leq t_{11}^{\sim} \leq 15.9]) \vee [5.81 \leq t_6^{\sim} \leq 5.88]$
(9) w. mouse ∨ h. seal ∨ A. noctule	$[-0.8 \leq t_2^{+}] \wedge [-0.141 \leq t_{10}^{\sim} \leq 19.6] \wedge [26.6 \leq p_4]$

Table 7. Redescriptions from Bio ranked with MEALL.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$
(1) polar bear	$[-7.07 \leq t_5^{\sim} \leq -3.38]$
(2) g. w. shrew ∧ E. mongoose	$([15.6 \leq t_8^{-} \leq 19] \wedge [1.62 \leq p_8 \leq 7.44] \wedge [66.2 \leq p_{12} \leq 137]) \vee [13.9 \leq t_3^{\sim} \leq 14.3]$
(3) w. mouse ∧ N. bat ∧ E. p. shrew	$([3.2 \leq t_3^{+} \leq 14.5] \wedge [17.3 \leq t_8^{+} \leq 25.2] \wedge [14.9 \leq t_9^{+} \leq 22.8]) \vee [19.6 \leq t_7^{\sim} \leq 19.9]$
(4) wolverine	$([7.2 \leq t_9^{+} \leq 11.7] \wedge [-11. \leq t_3^{\sim} \leq -5.37] \wedge [63.1 \leq p_7 \leq 106]) \vee [-3.43 \leq t_{11}^{\sim} \leq -3.34]$
(5) h. mouse ∧ E. mole	$[-0.3 \leq t_4^{-} \leq 8.7] \wedge [19.4 \leq t_8^{+} \leq 27.2] \wedge [45.4 \leq p_6] \wedge [48.8 \leq p_8 \leq 126]$
(6) w. lemming	$(([-4.3 \leq t_{11}^{+} \leq 1.6] \wedge [3.29 \leq t_5^{\sim} \leq 9.75]) \vee [-6.8 \leq t_3^{-} \leq -6.8]) \wedge [21.9 \leq p_3 \leq 72.2]$
(7) N. lemming	$(([-21.8 \leq t_2^{-} \leq -8.7] \wedge [12.5 \leq t_8^{+} \leq 16.6]) \vee [5.41 \leq t_5^{\sim} \leq 5.43]) \wedge [59 \leq p_8 \leq 166]$
(8) E. p. vole ∨ R. muntjac	$[-0.8 \leq t_2^{+} \leq 9] \wedge [10.9 \leq t_4^{+} \leq 17.5] \wedge [17.7 \leq t_9^{+} \leq 24.4] \wedge [43.3 \leq p_7]$
(9) L. shrew	$(([-9.7 \leq t_2^{+} \leq -4.7] \vee [4.44 \leq t_{10}^{\sim} \leq 4.52]) \wedge [41.7 \leq p_6 \leq 68.3]) \vee [-4.39 \leq t_{12}^{\sim} \leq -4.32]$

conditioned on the redescriptions seen previously. Thus, our approach fits into the general framework of subjective interestingness as proposed by De Bie (2011).

Working with redescriptions comes with its own challenges, though. Most notably, restricting redescriptions only to conjunctive queries – as is (implicitly) done in the existing work on subjective interestingness – severely limits the usability of the approach, and hence we extended our maximum-entropy model to handle also disjunctive queries. Many real-world data sets contain missing values, and our model can also handle them.

Another significant difference with the existing line of work on subjective interestingness is that we do not enforce any a priori conditions on the data (such as row and column marginals). With heterogeneous variable types (some variables are Boolean, others nominal, and yet others real-valued), these kinds of a priori conditions become less intuitive. Incorporating them to our model is also not straightforward, and we leave it as a topic for future work.

One potential extension of the current model is to take the amount of missing values into account: an observation with more missing values should be less trustworthy than a more completely observed one. In our model, this could be achieved via the row prior ρ, but a proper use of this approach would require us

to know the model for the missing values (e.g. whether they appear uniformly or in bursts).

Finally, another natural direction for future work is to develop methods that can directly mine the most surprising redescription given the current model; the work presented in this paper can only be applied as a post-processing step. Designing a scalable method for directly mining the most surprising redescriptions is not straightforward. For example, it not clear how to effectively employ the quantisation and partition approaches presented in this work when one has to consider arbitrary statements.

References

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998), 'Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications', *SIGMOD Record* **27**(2), 94–105.

Agrawal, R. and Srikant, R. (1994), Fast Algorithms for Mining Association Rules in Large Databases, *in* 'Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94)', pp. 487–499.

Barber, D. (2012), *Bayesian Reasoning and Machine Learning*, Cambridge University Press.

Berger, A. L., Pietra, V. J. D. and Pietra, S. A. D. (1996), 'A Maximum Entropy Approach to Natural Language Processing', *Computational Linguistics* **22**(1), 39–71.

Bickel, S. and Scheffer, T. (2004), Multi-View Clustering, *in* 'Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04)', pp. 19–26.

Burden, R. L. and Faires, J. D. (2011), *Numerical Analysis*, 9 edn, Brooks/Cole.

De Bie, T. (2011), 'Maximum Entropy Models and Subjective Interestingness: An Application to Tiles in Binary Databases', *Data Mining and Knowledge Discovery* **23**(3), 407–446.

Galbrun, E. and Kimmig, A. (2014), 'Finding Relational Redescriptions', *Machine Learning* **96**(3), 225–248.

Galbrun, E. and Miettinen, P. (2012*a*), 'From Black and White to Full Color: Extending Redescription Mining Outside the Boolean World', *Statistical Analysis and Data Mining* **5**(4), 284–303.

Galbrun, E. and Miettinen, P. (2012*b*), Siren: An interactive tool for mining and visualizing geospatial redescriptions, *in* 'Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)', pp. 1544–1547.

Galbrun, E. and Miettinen, P. (2014), Interactive Redescription Mining, *in* 'Proceedings of the 2016 International Conference on Management of Data (SIGMOD'14)', pp. 1079–1082.

Galbrun, E. and Miettinen, P. (2018), *Redescription Mining*, Springer, Cham.

Gallo, A., Miettinen, P. and Mannila, H. (2008), Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining, *in* 'Proceedings of the 8th SIAM International Conference on Data Mining (SDM'08)', pp. 334–345.

Gray, J. P. (1999), 'A corrected ethnographic atlas', *World Cultures* **10**(1), 24–85.

Grove, A. J., Halpern, J. Y. and Koller, D. (1992), Random worlds and maximum entropy, *in* 'Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS'92)', pp. 22–33.

Hijmans, R. J., Cameron, S. E., Parra, L. J., Jones, P. G. and Jarvis, A. (2005), 'Very High Resolution Interpolated Climate Surfaces for Global Land Areas', *International Journal of Climatology* **25**, 1965–1978.

Jaroszewicz, S. and Simovici, D. A. (2002), Pruning Redundant Association Rules using Maximum Entropy Principle, *in* 'Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'02)', pp. 135–147.

Jaynes, E. (1982), 'On the rationale of maximum-entropy methods', *Proceedings of the IEEE* **70**(9), 939–952.

Jaynes, E. T. (2003), 'Probability Theory: The Logic of Science', *Cambridge university press* **10**, 33.

Jensen, F. V. and Jensen, F. (1994), Optimal Junction Trees, *in* 'Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI'94)', pp. 360–366.

Kalofolias, J., Galbrun, E. and Miettinen, P. (2016), From Sets of Good Redescriptions to Good Sets of Redescriptions, *in* 'Proceedings of the 16th IEEE International Conference on Data Mining (ICDM'16)', pp. 211–220.

Kontonasios, K.-N. and De Bie, T. (2012), Formalizing Complex Prior Information to Quantify Subjective Interestingness of Frequent Pattern Sets, *in* 'Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis (IDA'12)', pp. 161–171.

Kontonasios, K.-N. and De Bie, T. (2015), 'Subjectively interesting alternative clusterings', *Machine Learning* **98**(1-2), 31–56.

Kontonasios, K.-N., Vreeken, J. and De Bie, T. (2011), Maximum Entropy Modelling for Assessing Results on Real-Valued Data, *in* 'Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'1)', pp. 350–359.

Kontonasios, K.-N., Vreeken, J. and De Bie, T. (2013), Maximum Entropy Models for Iteratively Identifying Subjectively Interesting Structure in Real-Valued Data, *in* 'Proceedings of the 2013 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD'13)', pp. 256–271.

Kröger, P. and Zimek, A. (2009), Subspace Clustering Techniques, *in* L. Liu and M. T. Özsu, eds, 'Encyclopedia of Database Systems', Springer, pp. 2873–2875.

Mampaey, M., Tatti, N. and Vreeken, J. (2011), Tell Me What I Need To Know: Succinctly Summarizing Data with Itemsets, *in* 'Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)', pp. 573–581.

Mampaey, M., Vreeken, J. and Tatti, N. (2012), 'Summarizing Data Succinctly with the Most Informative Itemsets', *ACM Transactions on Knowledge Discovery from Data* **6**(4), 16:1–16:42.

Mannila, H., Pavlov, D. and Smyth, P. (1999), Prediction with Local Patterns Using Cross-entropy, *in* 'Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)', pp. 357–361.

Mihelčić, M. and Šmuc, T. (2016), InterSet: Interactive Redescription Set Exploration, *in* 'Proceedings of the 19th International Conference on Discovery Science (DS'16)', pp. 35–50.

Mitchell-Jones, A. J. et al. (1999), *The Atlas of European Mammals*, Academic Press.

Murdock, G. P. (1967), 'Ethnographic atlas: a summary', *Ethnology* **6**(2), 109–236.

Novak, P. K., Lavrač, N. and Webb, G. I. (2009), 'Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining', *The Journal of Machine Learning Research* **10**, 377–403.

Parida, L. and Ramakrishnan, N. (2005), Redescription Mining: Structure Theory and Algorithms, *in* 'Proceedings of the 20th National Conference on Artificial Intelligence and the 7th Innovative Applications of Artificial Intelligence Conference (AAAI'05)', pp. 837–844.

Pavlov, D., Mannila, H. and Smyth, P. (2003), 'Beyond independence: Probabilistic models for query approximation on binary transaction data', *IEEE Transactions on Knowledge and Data Engineering* **15**(6), 1409–1421.

Phillips, S. J., Anderson, R. P. and Schapire, R. E. (2006), 'Maximum Entropy Modeling of Species Geographic Distributions', *Ecological Modelling* **190**(3), 231–259.

Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M. and Helm, R. F. (2004), Turning CARTwheels: an Alternating Algorithm for Mining Redescriptions, *in* 'Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)', pp. 266–275.

Rasch, G. (1960), 'Probabilistic models for some intelligence and achievement tests', *Copenhagen: Danish Institute for Educational Research* .

Tatti, N. (2006), 'Computational complexity of queries based on itemsets', *Information Processing Letters* **98**(5), 183–187.

Tatti, N. (2008), 'Maximum entropy based significance of itemsets', *Knowledge and Information Systems* **17**(1), 57–77.

Tatti, N. and Vreeken, J. (2011), Comparing apples and oranges, *in* 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 398–413.

van Leeuwen, M. and Galbrun, E. (2015), 'Association discovery in two-view data', *IEEE Transactions on Knowledge and Data Engineering* **27**(12), 3190–3202.

Vreeken, J. and van Leeuwen, M. (2011), 'KRIMP: Mining itemsets that compress', *Data Mining and Knowledge Discovery* **23**(1), 169–214.

Wang, C. and Parthasarathy, S. (2006), Summarizing Itemset Patterns Using Probabilistic Models, *in* 'Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)', pp. 730–735.

Wu, H., Vreeken, J., Tatti, N. and Ramakrishnan, N. (2014), 'Uncovering the plot: detecting surprising coalitions of entities in multi-relational schemas', *Data Mining and Knowledge Discovery* **28**(5-6), 1398–1428.

Zaki, M. J. and Ramakrishnan, N. (2005), Reasoning About Sets Using Redescription Mining, *in*

'Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)', pp. 364–373.

Zinchenko, T., Galbrun, E. and Miettinen, P. (2015), Mining predictive redescriptions with trees, *in* 'IEEE International Conference on Data Mining Workshops', pp. 1672–1675.