Associating odor qualities to their molecular properties with redescription mining

Guillaume Bosc

June 19, 2014

Date Performed:	February 3, 2014 - June 27, 2014
Laboratory:	LIRIS UMR CNRS 5205, INSA de Lyon, France
Teams:	Data Mining and Machine Learning (DM2L) and Base de Données (BD)
INSA instructor:	Mehdi Kaytoue
DM2L instructor:	Jean-François Boulicaut
Project Members:	Fabien De Marchi, Mehdi Kaytoue, Marc Plantevit and Moustafa Bensafi

Abstract

The scope of this report covers the perception of odors. The olfaction is an area that neuro-scientists studied for several decades, but the opportunities are still potentially numerous. The causal relations between the physicochemical properties of a molecule and its olfactory qualities are not established yet, although their existence has been proven, but could have applications in many domains such as perfumery or food industry. To tackle this problem, we propose in this work to introduce redescription mining, a new discipline of pattern mining only studied for a decade. It allows describing a same set of objects from different data sources, possibly heterogeneous. The contributions are twofold: first, we proceed to the running of an existing algorithm of redescription mining over olfactory datasets to understand redescription mining principles and detect its limits thanks to an applicative point of view. Second, we remark that the choice of two points of views in which redescriptions are mined is too rigid. We hence propose a method that avoids the necessity of being provided several different data sources on the same set of objects. For that matter we introduce a proposition to handle hierarchies over attributes which allows to automatically splits the attributes according to distances constraints. Then, experiments are realized and discussed to be aware of the efficiency of this approach.

Contents

1 Introduction

Olfaction corresponds to the ability to perceive odors. This perception of odors is the result of a complex phenomenon having as starting point a molecule with specific characteristics which comes to be associated with receptors in the nasal cavity, causing the emission of a signal transmitted through channels until to the brain that makes us feel a smell [[28]–[22]]. However in olfaction, the perception could not be predicted on the basis of the physicochemical properties of the molecule yet, whereas in vision or hearing, changes in colors or sounds perception could be demonstrated by the change in light wave lengths for the first one or the change in frequency of the sound waves for the last one. Although scientists already know that these are the physicochemical characteristics, or molecular attributes, which determine the fragrant odor percept, no rules has been advanced yet [[30]–[17]–[15]]. However, as the number of dimensions that describe the molecules is very large (up to several thousands), the existing analytic methods are ineffective. Added to this an heterogeneity in the types of dimensions, with 2D or 3D representations of molecules, multiplies even more the size of the search space. The objective of the neuro-scientists is to establish clear rules that exist in this relationship "chemical structure/odor".

The main task of my internship is about a transverse project in LIRIS with the Data Mining and Machine Learning (DM2L) and DataBases (DB) teams and in collaboration with the Centre de Recherche en Neurosciences de Lyon (CRNL). The objectives of this project is to bring out the existing links between the physicochemical properties of a molecule and its olfactory qualities.

To tackle this problem, we introduce the redescription mining which corresponds to a shift-of-vocabulary for describing a set of objects [[27]–[24]–[33]–[13]–[11]]. In fact, redescription mining allows to switch from a description of a set of objects to another description which is not a logic equivalent one of the first one. Redescription mining could be used to match different points of view of a set of objects: for example it could be interesting that a set of attributes in a specific vocabulary is equivalent to a set of attributes of another specific vocabulary with regard to a set of objects. Thus, we can expect to match a graph structure of a molecule (the first vocabulary) for example to its smell qualities (the second vocabulary). Different approaches have been studied: some of them proceed a complete search [[27]–[24]–[33]], others restrict the expressiveness of the language for the descriptions [[33]–[13]], or also some are able to handle several types for the attributes (Boolean, numeric, categorical for example) [11]. But the main objective is to describe from a different, and possibly heterogeneous, data sources a more or less same set of objects (using for example the Jaccard coefficient).

The contributions of this internship are twofold: first, we experiment an existing algorithm of redescription mining, namely the ReReMi algorithm, on different datasets provided by the neuro-scientists of the CRNL. These experiments leads us to discuss about the quality of the results with the neuro-scientists and to identify the principles of redescription mining but also its limits. Second, this report presents a proposition to handle a hierarchy on the descriptors (or attributes) of the dataset. Redescription mining requires

to have several different data sources describing the same set of objects, where each data source proposes a different description of the set of objects from the other. We decided to study another approach, requiring a single data source, which automatically splits the descriptors of the data source according to the hierarchy. Thus, arranging the descriptors of a data source in a hierarchical structure, the split into several views could be realized on-the-fly providing distances constraints into the hierarchy. This approach leads to the development of a new algorithm.

Note also that, during the beginning of this second internship in the LIRIS research laboratory, I realize the valuing of the results of my first internship by participating in the writing of a publication at the international European Conference on Artificial Intelligence 2014 (ECAI'14) which corresponds to an improvement of the method to mine balanced pattern from games interactions [4] (and the associated research report for more information [5]). This first work leads to the publication of two articles: the first one at the workshop of the international conference MLSA@ECML/PKK 2013, however this article was not published in proceedings, but a report of research of 11 pages is available [3] ; the second article was published in the french national conference Extraction et Gestion de Connaissances 2014 (EGC'14) and I presented it during the conference in Rennes, France in January 2014 [2].

The rest of this report is organized as follows. The first section gives the definitions and the main notions of redescription mining necessary to understand the remainder of this report. Then I present the state of art and the existing approaches for redescription mining. In the section 4, I discuss the results of a set of experiments with an existing algorithm on the olfactory databases provided by the CRNL. The section 5 develops the main contribution of this report, namely the use of a hierarchy on the attributes of the data source.

2 A general introduction to redescription mining

Given two different sources of data, redescription mining aims to describe a same set of objects in each database. This section proposes a synthesis of its formalisms used in the different approaches to study redescription mining [[27]–[24]–[33]–[13]–[11]](these approaches are discussed in the Section 3). Redescription mining starts from different sources of data about the same set of objects called the transaction database. An established separation, or partition called as views, of the attributes is required. A partition of the attributes often corresponds to a source of data in the transaction database. Then, the aim is to describe a same subset of objects by two different queries over two different partitions of attributes of this transaction database.

Def.1: Transaction database. Let $\mathcal{O}$ be a set of *objects* and $\mathcal{A}$ a set of *attributes* which are the properties of the objects. We denote by $|X|$ the cardinality of the set X. The attributes could be of various types: Boolean, nominal, real-valued attributes, ... We denote by $\mathcal{V}$ the set of different views involved in the set of attributes $\mathcal{A}$. A view

corresponds to a coherent group of attributes (e.g. political view, or geographical view for attributes which describe states). We propose a function which maps each attribute to its corresponding view (or partition) to which it belongs: $view : \mathcal{A} \rightarrow \mathcal{V}$. This function proceeds to the partition of the set of attributes $\mathcal{A}$. Note that, for all attributes $a \in \mathcal{A}$ it exists a unique view $v \in \mathcal{V}$ such that $view(a) = v$. Thus, the dataset is specified, without ambiguity, by these three parameters $\mathcal{D} = (\mathcal{O}, \mathcal{A}, view)$.

Running example. We propose here the running example we will use throughout this section. In this example, we consider that the set of objects $\mathcal{O}$ represents a set of nine different countries: $\mathcal{O} = \{Canada, Chile, China, France, Great\ Britain, Mexico, Mozambique, Russia, United\ States\ of\ America\}$. The attributes which specify the characterizing proprieties of these countries are $\mathcal{A} = \{$*South Hemisphere, Atlantic Ocean, Pacific Ocean, Continent, Area, Elevation, History of Communism, History of Colonialism, Political Regime, Population*$\}$. Note that these attributes are not all of the same type: *South Hemisphere, Atlantic Ocean, Pacific Ocean, History of Communism* and *History of Colonialism* are Boolean attributes, *Continent* and *Political Regime* are nominal attributes and *Area, Elevation* and *Population* are real-valued attributes. This dataset involved two different views, a geographical view and a more political view, i.e. $\mathcal{V} = \{Geographical, Political\}$. Indeed, the attributes *South Hemisphere, Atlantic Ocean, Pacific Ocean, Continent, Area* and *Elevation* are related to the geographical view whereas the others are related to the political view. Thus, for instance, $view(Continent) = Geographical$, and $view($*History of Colonialism*$) = Political$. The representation of this dataset is given by two databases which are presented in the Figure 2. The top database ($\mathcal{D}_1$) corresponds to the *Geographical* view and the bottom database ($\mathcal{D}_2$) to the *Political* view.

Now that we have introduced a transaction database corresponding to several views, we introduce the notion of query. A query can be understood as a description of a subset of objects with predicates over attributes domains, and the goal of redescription mining is to find couples of queries (q_L, q_R) that satisfy a set of constraints.

Def.2: Queries. We defined a query as a logical statement expressed over attributes in $\mathcal{A}$. A query is conjunctions or disjunctions of literals, which could be a predicate or the negation of a predicate. Formally, the general form of a query is as follows:

```
<query>   ⟶  (<query>) ∧ (<query>)
<query>   ⟶  (<query>) ∨ (<query>)
<query>   ⟶  <literal>
<literal> ⟶  <predicate>
<literal> ⟶  ¬ <predicate>
```

Note that a `<predicate>` is the assignment of a value or a range within the domain of an attribute. We denote the domain of an attribute by $dom(a)$, where a is an attribute in $\mathcal{A}$. The domain of an attribute could be of various types:

- If the attribute a is *Boolean*: $dom(a) = \{true, false\}$, and the possible associated predicates are either $[a = true]$ or $[a = false]$.
- If the attribute a is *Nominal*: $dom(a) = \{value_1, value_2, ..., value_k\}$, and the possible associated predicates are subsets of $dom(a)$, i.e., $\mathcal{P}$ is a predicate if $dom_{\mathcal{P}}(a) \subseteq dom(a)$, where $dom_{\mathcal{P}}(a)$ is the specified range of the predicate $\mathcal{P}$ for the attribute a. Thus, for instance a predicate $\mathcal{P}$ over a nominal attribute a could be $[a \in dom_{\mathcal{P}}(a)]$.
- If the attribute a is *Real-Valued*: $dom(a) = [a, b] \subseteq \mathbb{R}$, and the possible associated predicates are subsets of $dom(a)$, i.e., $\mathcal{P}$ is a predicate if $dom_{\mathcal{P}}(a) \subseteq dom(a)$, i.e., $dom_{\mathcal{P}}(a) = \bigcup_k [a_k, b_k]$, where $\forall k, [a_k, b_k] \subseteq [a, b]$ and $\forall k \neq l, [a_k, b_k] \cap [a_l, b_l] = \emptyset$. Thus, for instance a predicate $\mathcal{P}$ over a real-valued attribute a could be $[a \in dom_{\mathcal{P}}(a)]$.

We overload the notation of an object $o \in \mathcal{O}$ calling that $o(a) \in dom(a)$ is the value taken by the object o for the attribute $a \in \mathcal{A}$. A predicate $\mathcal{P}$ over an attribute $a \in \mathcal{A}$ is said satisfied by an object $o \in \mathcal{O}$ if the condition within the predicate holds *true* when we replace the variable of the attribute a by the specific value of the object o for the attribute a, i.e. $o(a)$. Formally, a predicate $\mathcal{P}$ is satisfied if $o(a) \in dom_{\mathcal{P}}(a)$. Thus, we call the set of objects that satisfied a predicate $\mathcal{P}$ over an attribute a the *support* of $\mathcal{P}$ and we denote it as $supp(\mathcal{P}) = \{o \in \mathcal{O} \mid o(a) \in dom_{\mathcal{P}}(a)\}$. A query is said satisfied if the logic formula associated is satisfied, regarding to the satisfaction of predicate we presented earlier and the Boolean logic. We extend the notation denoting $supp(q)$ the support of a query q. We denote the set of attributes involved in a query by $attr(q) \subseteq \mathcal{A}$.

Example. Considering the running example presented previously and its associated databases in the Figure 2. First, let's consider Boolean attributes. A predicate over the attribute *South Hemisphere* in the top database, related to the geographical view, could be $\mathcal{P}_1 : [South\ Hemisphere = true]$, thus, $supp(\mathcal{P}_1) = \{Chile,\ Mexico\}$. Then, considering nominal attributes, a predicate over the *Continent* attribute could be $\mathcal{P}_2 : [Continent \in \{North\ America,\ South\ America\}]$, which is satisfied by the countries which belong to either the North America *or* the South America, with the meaning of the logical Boolean operator "$\vee$". Then, the set of countries that satisfy this predicate is $supp(\mathcal{P}_2) = \{Canada,\ Chile,\ Mexico,\ United\ States\ of\ America\ \}$. Furthermore, with real-valued attributes, $\mathcal{P}_3 : [Area \in [0.77, 2]\ ]$ is a predicate over the real-valued attribute *Area*. It corresponds to the set of countries $supp(\mathcal{P}_3) = \{Chile,\ Mexico,\ Mozambique\ \}$. Note that the values of lower and upper bound could be different without changing the set of countries that satisfy the predicate ([16]).

A query is so a conjunction or a disjunction of predicates. The predicates within a query could be of different types. For example, given the database presented in the Figure 2, the query $q = [South\ Hemisphere = false] \wedge ([Elevation \in [5500, 6500]\ ] \vee \neg[Continent \in \{Europe\}])$. This query aims to get countries which belong to the north hemisphere and which do not belong to the European continent or which elevation is between 5500 and

6500 meters. This query is satisfied by the following set of country $supp(q) = \{Canada, China, Mexico, Russia, United States\}$.

A query can be understood as a pattern, i.e. a description which covers a set of objects. A redescription is a pair of two queries that are syntactically different but that covers almost the same set of objects. The quality of a redescription depends then on how many objects they do both cover (the intersection): the more the better.

We present other constraints that a redescription has to verify to be interesting (output by any algorithm that mines them).

Def.3: Redescription. Formally, a redescription is denoted by a pair of queries (q_L, q_R), where the both queries are related to two disjoint set of views (and thus to two disjoint set of attributes), i.e. $view(q_L) \cap view(q_R) = \emptyset$, where we overload the *view* function to a query. Furthermore, $supp(q_L) \sim supp(q_R)$, with $\sim$ is a binary similarity relation. A redescription (q_L, q_R) has to satisfy the set of constraints $\mathcal{C}$.

Constraint 1 : Frequency. In order to mine redescriptions, it is not conceivable and feasible to extract all possible redescriptions: we need to mine only frequent redescriptions. The frequent redescriptions are obtained by fixing a minimum support threshold *minsupp*. A redescription is said frequent if the cardinality of the support of its both queries are greater or equal to the minimum support threshold *minsupp*. Mining only frequent redescriptions is a constraint we use to our redescription problem. Formally, a redescription (q_L, q_R) is frequent if: $c_1(q_L, q_R) \equiv (|supp(q_L)| \geq minsupp) \wedge (|supp(q_R)| \geq minsupp)$.

Constraint 2 : Similarity. The equivalence between two different queries from different databases does not have to be strict: indeed, if it requires that the set of objects that satisfy the both queries are the same, the number of redescriptions may be too low. To tackle this problem, most of studies about redescription mining used *Jaccard's coefficient*. This coefficient permits to know how two queries are equivalent, i.e., if their supports are more or less the same. For two sets of objects A and B the *Jaccard's coefficient* is computed by:

$$\mathcal{J}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Thus, one can easily remark that $\mathcal{J}(A, B) = 1$ if and only if $A = B$. Given a minimum threshold over the *Jaccard's coefficient minsim*, one could be able to mine approximate redescriptions: $c_2(q_L, q_R) \equiv \mathcal{J}(supp(q_L), supp(q_R)) \geq minsim$, for q_L and q_R two queries.

Constraint 3 : Significance. Another important requirement for the redescriptions which have been extracted is that they should be statistically significant. Indeed, let us consider a predicate over an attribute which is satisfied for more than 90% of the objects. Its support will be high but the information it brings seems to be quite poor. To prevent this problem, one can measure the significance of a redescription computing its *p-value*.

Let $p_L = \frac{|supp(q_L)|}{|\mathcal{O}|}$ (resp. $p_R = \frac{|supp(q_R)|}{|\mathcal{O}|}$) be the probability of q_L (resp. q_R), the *p-value* between the two queries q_L and q_R is given by:

$$pval(q_L, q_R) = \sum_{k=|supp(q_L)\cap supp(q_R)|}^{|\mathcal{O}|} \binom{|\mathcal{O}|}{k} (p_L p_R)^k (1 - p_L p_R)^{|\mathcal{O}|-k}$$

The higher the value of $pval(q_L, q_R)$ is, the less significant the redescription (q_L, q_R) is. To mine statistically significant redescriptions, one could impose a maximal threshold *maxpval* on the *p-values* in order to extract only redescriptions (q_L, q_R) such that $c_3(q_L, q_R) \equiv pval(q_L, q_R) \leq maxpval$. This condition is another constraint we have to satisfy to solve our redescription problem.

Problem of redescription mining. Thus, the problem of redescription mining is to extract all redescriptions from a dataset that satisfy the set of constraints $\mathcal{C}$, i.e. the redescriptions (q_L, q_R), such that $c_i(q_L, q_R) = true, \forall i \in \{1, 2, 3\}$.

Example. Given the two databases in the Figure 2 corresponding to the two views of the dataset presented previously. Let $minsupp = 4$ and $minsim = 0.65$. The support of the query: $q_1 = [Area \in [1.5, 20]\ ] \wedge [Pacific\ Ocean = true]$ is $supp(q_1) = \{Canada,\ China,\ Mexico,\ Russia,\ United\ Stated\}$, so this query is frequent and could belong to a possible redescription since $|supp(q_1)| = 5 \geq minsupp$. Likewise, the query $q_2 = [Population \in [100, 1500]\ ] \wedge [Political\ Regimen \in [Republic]\ ]$ over the second view is frequent ($supp(q_2) = \{China,\ Mexico,\ Mozambique,\ Russia,\ United\ Stated\}$). Thus, these two queries over the two different views are a redescription of the set of objects $\{China,\ Mexico,\ Russia,\ United\ Stated\}$, since the *Jaccard's coefficient* between the support of each query is greater than $misim$, $\mathcal{J}(supp(q_1), supp(q_2)) = 0.67$. Note that, in this example we only consider Boolean data, but it is important to highlight that, for real-valued data, the importance for the choice of the lower and the upper bound. Indeed, they have a great influence on the result of the redescriptions. They can improve the quality of a redescription if they are smartly chosen. But on the contrary they can reduce considerably the number and the quality of the set of redescriptions extracted.

3 Redescription mining practices

Redescription mining is still in the wake of its development. Few people have been interested in redescription mining, but their approaches are quite various: the first attempt to study redescription mining used a supervised classification [7] approach with two different decision trees which leaves are matching, the next approach is related to pattern mining [14] and try to mine redescriptions thanks to minimal patterns and generators. The last approach consists in using methods from subgroups discovery [[18]–[23]] with a beam-search principle. In this section, we try to present the line of thought of these studies. The summary table, namely Table 3, presents all characteristics of the different approaches, allowing to be aware of the advantages and limits of them.

3.1 A supervised classification approach

Redescription mining was first introduced by Ramakrishnan and al. [27]. The authors present here the definitions and the concepts related to redescription mining. The main first definitions, notions and preliminaries approaches of redescription mining are presented in [24], which seems to be an attempt to completely formalize this new area and to propose several theoretical properties.

Ramakrishnan and al. also propose the first algorithm able to mine these redescriptions, namely CARTwheels, which uses two structures of tree where the leaves are common for the both ([27] - [19]). In fact, this structure of tree is used to process a kind of classification. The CARTwheels is an alternating algorithm, because at each step of the algorithm one of the two trees is fixed and the other one is grown to match it. It extends the notion of classification and regression trees (CART) [6] which provides a powerful approach in terms of accuracy and efficiency of induction for pattern classification and data mining. The first tree (resp. the second tree), say the top tree (resp. bottom tree) is based on the set of attributes of the first (resp. second) database. Note that this algorithm is restricted by the type of data: only Boolean data is supported.

The Figure 3 presents two transaction databases which the sets of attributes are disjoint. The set of objects is $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5\}$. Each view contains four attributes, $\{X_1, X_2, X_3, X_4\}$ for the first view represented in the transaction database $\mathcal{D}_1$, and $\{Y_1, Y_2, Y_3, Y_4\}$ for $\mathcal{D}_2$. The first step of this algorithm consists in building a new database, choosing among one of the two transaction databases and adding a label column obtained thanks to the other transaction database. For example, choosing the transaction database $\mathcal{D}_1$ for the label column, we obtain the following database in the Figure 4. Note that, to select the attribute which become the label, or the class, we consider the first attribute for which the object is true: for the object o_2 for example, the attributes X_1 and X_3 are *true* but only the X_1 attribute is chosen for the label. Thus, the order of objects in the database impacts the result of the algorithm. Moreover, it is possible that an attribute does not appear in the label column. Then, using any of the impurity measures (entropy, Gini index, etc...) the top tree could be grown, see the Figure 5. Note that the 3-partition induced by the tree is not exactly the same that the 3-partition presented in the original dataset, only the query associated to X_2 corresponds exactly with the query associated to $\neg Y_3 \wedge \neg Y_1$, according to the label database in the Figure 4. Moreover, all possible paths of this tree correspond to, from the left most path to right most one: $Y_3 \wedge Y_2$, $(Y_3 \wedge \neg Y_2) \cup (\neg Y_3 \wedge Y_1)$ and $\neg Y_3 \wedge \neg Y_1$. Then this obtained partition become the starting point for the next iteration. The top tree is now fixed and the bottom tree can be grown. We build the new label database using the obtained partition and the transaction database $\mathcal{D}_2$ for the label column, see the Figure 6 (a). The bottom tree which is obtained at the end of this second step of the algorithm is presented in the Figure 7 at the center. During the new alternation (Step 3), the bottom tree is fixed and the obtained partition at the second step is used to build the new label database which the label column uses attributes from the transaction database $\mathcal{D}_1$. This label database is given in the Figure 6 (b). The

Figure 7 presents the results of the first three step of the algorithm. Note that the colored arrows correspond to matching queries.

3.2 A pattern mining approach

This approach is the extension of the previous one. The authors (appoximatively the same than for the paper about Cartwheels algorithm) continue their work, focusing a bit more on pattern mining. As the number of redescriptions extracted thanks to the previous algorithm CARTwheels being potentially too high, a new approach is thus analyzed, using the notion of minimal, generator (see [21]) and non redundant redescriptions [33]. Observe that this approach is related with frequent itemset mining, which is an area very studied by Zaki at this time, which could explain why the authors, including Zaki, have chosen this method. Thanks to this approach, the number of redescriptions which could be extracted is reduced. To present this work, some definitions and notions are required. Note that this work is also restricted to Boolean data, and the queries which composed the redescriptions are only obtained by conjunctions of true predicates over attributes. Thus, all queries in this subsection are in the form of $q_i = [A_1 = true] \wedge [A_2 = true] \wedge ... \wedge [A_n = true]$.

Super-query. Let q_1 and q_2 be two queries, such that q_1 and q_2 are only conjunctions of *true* predicates, i.e., for all predicates P_i in the queries, they are in the form of $P_i : [A_i = true]$, where A_i is the attribute over which the predicate acts. The query q_2 is said the super-query of the query q_1, and we denote $q_2 \sqsupseteq q_1$, if all predicates in q_1 are in q_2. Thus q_1 is said the sub-query of q_2.

Closed query. Let q_1 be a query formed by *true* predicates of a transaction database. q_1 is said closed if there exists no super-query $q_2 \sqsupseteq q_1$ such that $supp(q_2) = supp(q_1)$ and $q_1 \neq q_2$.

Generator. Let q_1 and q_2 be two queries of a transaction database. q_1 is a generator of q_2 if q_2 is a super-query of q_1, $q_2 \sqsupseteq q_1$, and $supp(q_2) = supp(q_1)$.

Conditional redescription. $(q_L, q_R)|q_C$ is a conditional redescription if (i) $attr(q_L) \neq \emptyset$ and $attr(q_R) \neq \emptyset$, (ii) $(q_L \cap q_R) = (q_L \cap q_C) = (q_R \cap q_C) = \emptyset$ and (iii) $supp(q_L \cup q_C) = supp(q_R \cup q_V)$. It means that q_L and q_R describe the same set of objects with respect to the condition q_C.

Minimal conditional redescription. $(q_L, q_R)|q_C$ is a minimal conditional redescription (or a non redundant redescription) if and only if there does not exists another redescription $(q'_L, q'_R)|q'_C$ such that $q'_i \sqsubseteq q_i$ for i in $\{L, R, C\}$.

Algorithm. The algorithm which has been developed to mine all non redundant redescription processes in three main steps: first it builds the lattice of closed queries from

the translation database, then it computes the minimal generators of the closed queries and after it keeps the non redundant redescription from the minimal generators.

Step 1: Constructing closed queries lattice. To construct the closed queries lattice, the authors decided to extend the Charm algorithm [32]. The approach they used is that when a closed query q is newly found, they determine the set of all closed super-queries S of q, and the minimal elements of S become the direct super-queries of q in the lattice structure. The Figure 9 presents the lattice structure which is obtained after executing the algorithm from the transaction database given in the Figure 8. Note that for readability, we replace a true predicate $[A_i = true]$ by $[A_i]$ since all predicate in this subsection are true predicate. This lattice presents together the closed query, the support involved and the minimum generator of this support.

Step 2: Computing minimal generators. Given the closed queries lattice which has been obtained at the previous step, the algorithm computes the minimal generators of each nodes of the closed queries lattice. We denote the set of minimal generators as $\mathcal{M}(q)$ for a closed query q. A minimal generator g of a closed query q is a minimal query which is a sub-query of q but not a sub-query of any of a direct closed sub-query of q in the closed queries. We denote $Q' = \{q^1, q^2, ..., q^k\}$ the set of all direct closed sub-queries of q in the closed lattice. Let $\Delta^i = attr(q) - attr(q^i)$ be the set of attributes which are present in the query q but not in its sub-query q^i. A query g is called a hitting query of Δ if and only if $\forall i \in [1, k], attr(g) \cap \Delta^i \neq \emptyset$. Furthermore, a query g is called a minimal hitting query if there does not exist another hitting g' such that $g' \sqsubset g$. The set of minimal generators of a closed query q, $\mathcal{M}(q)$, is the same as the set of minimal hitting queries of Δ.

For instance, consider the closed query $q = [d_1] \wedge [d_2] \wedge [d_3] \wedge [d_4] \wedge [d_5] \wedge [d_6] \wedge [d_7]$. Its set of direct closed queries is $\{[d_2] \wedge [d_3] \wedge [d_4], [d_1] \wedge [d_2] \wedge [d_3] \wedge [d_5] \wedge [d_6] \wedge [d_7], [d_1] \wedge [d_2] \wedge [d_4] \wedge [d_5] \wedge [d_6]\}$. Thus, $\Delta = \{d_1d_5d_6d_7, d_4, d_3d_7\}$. The set of hitting queries is: $\{[d_1] \wedge [d_3] \wedge [d_4], [d_1] \wedge [d_4] \wedge [d_7], [d_3] \wedge [d_4] \wedge [d_5], [d_4] \wedge [d_5] \wedge [d_7], [d_3] \wedge [d_4] \wedge [d_6], [d_4] \wedge [d_6] \wedge [d_7], [d_3] \wedge [d_4] \wedge [d_7], [d_4] \wedge [d_7]\}$. And then, the set minimal generators of q is $\mathcal{M}(q) = \{[d_1] \wedge [d_3] \wedge [d_4], [d_3] \wedge [d_4] \wedge [d_5], [d_3] \wedge [d_4] \wedge [d_6], [d_4] \wedge [d_7]\}$. The set of minimal generators of all closed queries are given in the Figure 9.

Step 3: Non redundant redescriptions set. Given the set of closed queries and their set of minimal generators. For each distinct pair g_1 and $g_2 \in \mathcal{M}(q)$, where q is a closed query, the algorithm generates the following redescription $(g_1 - (g_1 \cap g_2), g_2 - (g_1 \cap g_2)) \mid (g_1 \cap g_2)$.

All minimal non redundant redescriptions are given in the table of the Figure 10. Observe that a conditional redescription is obtained when the both minimal generators in a pair are non-disjoint, whereas an unconditional redescription is found when these minimal generators are disjoint.

3.3 Subgroup discovery approaches

The subgroup discovery [[18]–[31]] is a data mining technique which attempts to discover the subgroups of a given set of objects that are statistically more interesting with respect to a target attribute, i.e. they are the largest and their distribution according to the target attribute is the most unusual statistically. It is within this context that this work [13] has been realized. It corresponds to a new branch of the redescription mining studies. This approach is mainly focused on frequent itemset mining methods such as the Apriori principle. It explains how to mine redescriptions from queries in general form: a query could use conjunction, disjunction or negation operator, in any order. However, this work is still focused on Boolean data. This study presents two different algorithms to reach to extract redescriptions containing this kind of queries.

The Greedy algorithm. The Greedy algorithm performs a level-wise search combining with a greedy approach to prune uninteresting redescription and thus to reduce the search space. The first step consists of finding the initial redescriptions (the both queries of a redescription are only composed by one predicate) by an exhaustive search. Once all initial redescriptions have been found, only the best ones that satisfy the constraints are kept in the remaining of the algorithm. The second step tries to extend the initial redescriptions by adding a predicate alternatively to both queries. For each possible attribute, the algorithm tries to extend a query by the four possibilities: the conjunction or the disjunction of the positive predicate or the negation of the predicate. Once the algorithm has tries all possibilities for all attributes for all queries of the initial redescription, it selects only the new best extended redescriptions. If the maximum number of predicates per query is not reached and there still has attributes to add, the algorithm starts again at the step 2 using, this time, the extended redescription.

This approach is non optimal: the greedy approach allows to reduce the search space but in return, the algorithm does not assume that the set of redescriptions is the optimal. Some redescriptions could have been missed because their branch did not be explored since their Jaccard coefficient was to low. Indeed, there is no anti-monotonic property in this case, as it could be for many frequent itemset mining algorithms (such as Apriori).

The MID algorithm. The Mining Interesting Descriptors of subgroups algorithm (MID) is quite different from the Greedy algorithm. Whereas the Greedy algorithm starts with redescriptions composed by two queries of only one predicate, the MID algorithm starts with a conjunction of positive predicates which correspond to the set of frequent closed itemsets sorted by their p-value for all databases.

For each database, or view, the algorithm computes the set of all frequent closed itemsets. Each frequent closed itemset has to satisfy the conditions imposed by the thresholds. These itemsets form queries which correspond to a conjunction of the true predicates associated to each attribute of the itemset. This set constitutes the initial set of one part of the redescriptions, denoted as R_i^0, where i indicates that it referred to the i^{th} view. Then, the algorithm adds the negations of these initial queries to R_i^0 if they satisfy the threshold of the p-value. Then, for each view i, the algorithm will extend the set R_i^0 by creating

the next set R_i^1 that contains the queries in R_i^0 and others queries that are obtained by associating two different queries of R_i^0, either thanks to a conjunction or a disjunction. If these new queries satisfy the threshold of p-value they are added to R_i^1. The algorithm continues doing the same for R_i^j and stops when the number of predicates per query (K) is reached. The last step of the algorithm consists of forming all possible pairs of queries in R_i^{K-1} and testing if they satisfy the threshold of Jaccard coefficient to get the set of redescriptions.

Note that the two algorithms are quite different in their approach. Indeed, the first one get the set of redescriptions which the both queries are at most composed by K predicates whereas the last one, the MID algorithm, aims to get the set of redescriptions which the both queries are at most composed by K itemsets. Moreover, observe that the thresholds over the Jaccard coefficient is not used to prune the search space in the MID algorithm. This algorithm aims to prune the search space by selecting only interesting patterns: the association of frequent closed items.

The work realized by Galbrun is various. But, one of the main contributions of her work is that she extended the redescription mining from Boolean data to categorical and real-valued attributes. In fact, the previous works only consider the Boolean data, so to use this method to obtain a set of redescriptions, the user had to discretize the data if it was possible. The algorithm she developed is named ReReMi (see [11]). She also implements a tool which allows to visualize and interact with redescriptions from geospatial data, called Siren (see [12] and [10]). Another important contribution is the introduction of relational redescription mining in order to mine for structurally different connection patterns between two same objects in a relational dataset (see [9]). All her contributions are available in her thesis [8].

The ReReMi algorithm. The previous works are only able to mine redescriptions from Boolean data. The contribution of Galbrun is so to extend the type of data to categorical and real-valued [11] thanks to the ReReMi algorithm. This algorithm proceeds in different steps: (i) generate the initial pairs, (ii) extend these pairs, (iii) filter the results. Observe that, this algorithm limits the expressiveness of queries to linearly parsable queries, i.e. queries that can be parsed in linear order, without trees. This trick allows to reduce very significantly the search space but in return the redescriptions are more restricted but, maybe, more easily to interpret. The approach used in this algorithm is a strategy similar to beam-search.

Generate the initial pairs. The ReReMi algorithm starts by generating and evaluating all possible initial redescriptions, i.e. pairs of queries containing only one predicate. Then, only the k_p-best redescriptions are kept in the remaining of the algorithm.

Extend pairs. Once the k_p (at most) initial redescriptions have been computed, the ReReMi algorithm performs its extension step which consists in adding a new predicate to either the first query or the second query of each initial redescription. The new predicate

added could be linked either by the *and* or the *or* operator. Note that the predicate could be preceded by the *negation* operator. Thus, when the algorithm tries to extend a redescription, for each available attribute, it evaluates four possibilities and keeps those which accuracy is better than the current best ones redescriptions that respect the set of constraints C. The ReReMi algorithm maintains a list of at most k_i redescriptions which corresponds to the best redescriptions computed at this time. To deal with the real-valued attributes, the algorithm still proceeds with the *on-the-fly bucketing* approach which allows to chose the lower and the upper bounds among the different possibilities.

Filter the redescriptions. The last step of the ReReMi algorithm consist of leaving out the redescriptions which do not respect any constraints of the set of constraints C. Some constraints could be verify during the process of the algorithm but some of them could only be verify at the end of the algorithm.

4 Experiments: Application to olfaction

Once the related works have been studied, we want to test the approach of Galbrun to mine redescriptions with its algorithm ReReMi on our different datasets of the OlfaMining project. As a reminder, the OlfaMining project is a transverse work between the DM2L and the DB teams of Liris and in collaboration with the CRNL. The aim is to establish some links between the physicochemical properties of molecules and their olfactory qualities. For this, we are provided three different datasets which contain several molecules with some physicochemical properties and their olfactory qualities. So, the aim of this section is to run the algorithm ReReMi on these datasets in order to first see the quality of the results, and then to be aware of the drawbacks of this approach.

4.1 Data

In the context of the OlfaMining project, three different datasets are provided. Each of them has different characteristics in term of number of molecules that are described, number of physicochemical properties used and the different olfactory properties involved. See Table 1 which sum up the characteristics of the three different databases on which we run our experiments. The first database is called the Arctender's database [1]. This database is one of the first olfactory database which has been established. In this database, each molecule is associated to 2.88 qualities on average. A molecule is either associated to a quality or not: this is a Boolean association.

The second database we use is the Boelens database. The Boelens database as quite different characteristics from the first one. There are less molecules that are described (only 263) and many more physicochemical properties (4,886). The association between a molecule and an olfactory quality is no longer a Boolean association but now each molecule has a rate from 0 to 9 for each olfactory quality. A rate of 0 means that the molecule has not this olfactory quality but on the contrary, a rate of 9 means that this molecule smells

completely this olfactory quality. The rate is obtained by many evaluations given by several experts.

The last database is the Dravnieks database. This database contains many more olfactory qualities than the two other databases: it involves 146 different olfactory qualities. However, the association between a molecule and an olfactory quality is this time obtains by a rate from 0 to 100 given by 100 experts. The lower the rate is the less the molecule smells this olfactory quality, and on the contrary the higher the rate is, the more the molecule smells this olfactory quality.

Dataset	# molecules	# physicochemical properties	# olfactory qualities
Arctender	1,689	1,704	74
Boelens	263	4,885	30
Dravnieks	138	4,885	146

Table 1: Characteristics of the three databases.

Moreover, an expert give us a selection of 82 physicochemical properties which are probably sufficient to obtain good results and thus reduce considerably the number of physicochemical properties. Note that for the first dataset, Arctender, there are not the 82 attributes but only 42.

4.2 Algorithm and parameters

We choose to run these preliminaries experiments with the ReReMi algorithm which is able to mine redescriptions from numerical and nominal attributes. Note that the physicochemical properties of the molecules are real-valued attributes. The experiments are split in several parts. The objective is to test the influence of the different parameters to understand the principle but also the limits. We derived 4 different datasets from the 3 initial datasets: (i) the Arctender database with the selection of 42 attributes, (ii) the Arctender database with a selection of 243 physicochemical properties obtained thanks to a selection of non-correlated attributes, (ii) the Boelens database with the selection of the 82 attributes and (iv) the Dravnieks dataset with the selection of 82 physicochemical properties.

For each dataset we run 9 different types of experiment in which some attributes have been changed. For each of these 9 different types of experiments we launch 7 executions that vary the minimum relative support threshold from 0.01% to 0.5% (0.01, 0.05, 0.1, 0.2, 0.3, 0.4 and 0.5). There are 252 executions which have been launched (4 datasets, 9 types, 7 experiments). The details of the 9 different types of experiment are:

- Type 1: The queries are restricted to be conjunctions of positive predicates.
- Type 2: The queries are restricted to be conjunctions of positive or negative predicates.
- Type 3: The queries are restricted to be conjunctions or disjunctions of positive predicates.
- Type 4: The queries are restricted to be conjunctions or disjunctions of positive or negative predicates.

- Type 5: The queries are restricted to be conjunctions or disjunctions of positive or negative predicates (note that the conjunctions could be present in the both sides of a redescription, contrary to the previous types in which only 1 side of the redescription at most could contain conjunctions).
- Type 6: The same conditions that for the type 5 excepted that there is no filter of redundancy.
- Type 7: The same conditions that for the type 5 excepted that there are at most 2 predicates for the query of the olfactory qualities views and at most 10 predicates in the query which belongs to the physicochemical properties view.
- Type 8: The same conditions that for the type 5 excepted that there are at most 3 predicates for the query of the olfactory qualities views and at most 10 predicates in the query which belongs to the physicochemical properties view.
- Type 9: The same conditions that for the type 7 excepted that the 500 best initial redescriptions are kept in at the end of the step of searching initial pairs (instead of 100 previously).

4.3 Results

This section will discuss the results which have been obtained. On one hand, we will present the quantitative results which deal with the run-time, the number of redescriptions mined for example and on the other hand we will analyze the qualitative results trying to interpret the results and their actionability thanks to the knowledge of the experts. Note that the Type 6 is uninteresting because it leads to the same result than the Type 5.

4.3.1 Quantitative results

In this section we consider the run time of the ReReMi algorithm over our different datasets, but also the number of redescriptions which are output by the algorithm. The Figures 11-12-13-14 sum up these aspects for the four different datasets. Experiments are performed on a 2.6 GHz Intel Xeon X5650 with 16 GB main memory running Linux. Concerning the run time, we see clearly that the more general the language of the queries is, the more important the run time is. Indeed, the algorithm as to test all the possibilities during the extension steps. For example, the run time is up to ten times greater for the experiments of type 9 than for those of type 1 where the language is limited to conjunction of positive predicate over attributes. Moreover, the number of attributes which constitute the different views impacts directly the value of the run time, that is understandable because the number of possibilities to extend a query is thus more important. Note that the run time is really influenced by the the time to compute the initial pairs (the first step of the ReReMi algorithm). All the possibilities are explored and in the cases of the datasets (ii), (iii) and (iv) the both views are numerical attributes so the algorithm as to test several possible combinations for the intervals, whereas for the first dataset (i), the olfactory qualities are boolean attributes that allows to perform a more efficient search for the initial pairs. That is why in the Figure 14 the run time curve seems to be a constant. So, the beam-search approach seems not adapted for huge data because the first step of

full search requires too much time and memory. Observe that for the second dataset, the experiments related to the Type 9 could be realized because the run time is too high (more than 60 hours).

Concerning the number of redescriptions obtained as output, we could surmise that the higher the initial threshold of the Jaccard coefficient is, the less numerous the redescriptions are as output. However, this observation confirms that the beam-search approach is quite efficient because we do not miss interesting redescriptions which the initial pair has a low Jaccard coefficient. Note also that the more general the language is, the more numerous the redescriptions are.

4.3.2 Qualitative results

The aim of this set of experiments is to establish a link between the physicochemical properties and the olfactory qualities. It had only be proved that it exists a correlation between the olfactory hedonism (i.e. the fact that a molecule smells good or reeks) and physicochemical properties. However, these studies did not mention which properties are correlated to which olfactory qualities because the approaches they used were the Principal Component Analysis (PCA). Redescription mining could then be very useful to characterize the correspondence between the physicochemical properties and the olfactory qualities. Observe that redescription mining goes further because it expresses an equivalence relation between the two views even if neuro-scientists seem rather interested in the implication of the physicochemical properties in the resulting odor of the molecule.

The accuracy. In these experiments we use the Jaccard coefficient to measure the similarity between two queries. The Figures 15 - 16 - 17 - 18 give the distribution of the similarity of the output results according to their support. Once again, though these figures we see that the more general the language is, the more accurate the redescriptions are. For example, see the Figure 17, the accuracy of the experiments of Types 5 - 7 - 8 - 9 are in general better than the accuracy of the previous Types. Note that the output redescriptions of the different datasets have not the same accuracy. This could explain that the attributes of the views of physicochemical properties have to be carefully chosen and the olfactory qualities have to be well classified. All datasets on which we experiment use a different principle to classify the olfactory qualities of a molecule. Moreover, there are often some experts that evaluate the molecules, but the olfactory is a domain quite complex: people do not feel the odors identically.

The confidence. The figures 19 - 20 - 21 - 22 are the representation of the possible association rules that could have been mined from the datasets. Not that these figures represent the confidence of the association rules of the form $Physicochemical\ properties \longrightarrow Olfactory\ qualities$. The confidence between a query A and a query B is computed as:

$$confidence(A \longrightarrow B) = \frac{|supp(X) \cap supp(Y)|}{|supp(X)|}$$

Thus, the confidence is always greater or equal than the Jaccard coefficient. Note that we do not display the top-k rules but only the confidence of the top-k redescriptions: it is possible that there exists rules that are better than those that we display.

A problem of the results of these experiments is that the redescriptions with a high similarity (i.e. a Jaccard coefficient close to 1) always correspond to redescriptions which support is almost all molecules of the datasets. In fact, for the experiments of Type 9 for example, you see that for the redescriptions with the highest support, the accuracy is often close to 1. Similarly, the redescription which the cardinality of the support is low, the Jaccard coefficient is often high. To avoid this kind of problem, a parameter has been provided to remove redescriptions which the cardinality of their support is higher than a given percentage.

The redescriptions. The redescriptions obtained are quite numerous. For example, thanks to this algorithm we could have extracted this kind of redescriptions: r_1 = (Vanilin, [19.403 ≤ Mv ≤ 19.5106] OR [1.267 ≤ VE2_X ≤ 1.292] AND [11.574 ≤ Mp ≤ 14.625] AND [1.511 ≤ IC3 ≤ 3.461] OR [3.342 ≤ VR3_X ≤ 3.342] AND [10.0 ≤ D/Dtr11] AND [2.949 ≤ SpPosLog_H2 ≤ 4.385]). This redescription r_1 means that the molecules which smell Vanilin are approximately the same that those which satisfy the query over the physicochemical properties. Remember that, the language is linearly parsable. This redescription, for example is a result from the dataset (ii) with the parameter set of the Type 8. It has a support of 18: it means that the two queries of the redescription hold for 18 molecules of the dataset. The Jaccard coefficient is 0.7 and the confidence of the association rule from the query over physicochemical properties to the query of the olfactory qualities equals 1: it means that all molecules which satisfy the query over physicochemical properties are molecules which smell the Vanilin. This redescription, for example is very interesting according to neuroscientists because it is discriminating for the Vanilin olfactory quality. Note that I met a neuro-scientist during a half-day to discuss about the results I obtained. He found the results very interesting but he said that he would need a chemist to go further. We also discuss about the parameters we have to fix and he told me to increase the number of predicates in the queries to observe the behavior of the redescriptions.

In the different experiments, we limit the number of predicates for the olfactory qualities query at 2 or 3 whereas we allow up to 10 predicates for the query over the physicochemical properties. The results are quite interesting, but the neuroscientists suggested to improve considerably the number of predicates of the query over the physicochemical properties. Thus, we proceed a new set of experiments in which we improve this number of predicates. For each dataset we derive from the Type 7 of the previous experiments several new types, denoted as Type 71, Type 72, up to Type 79 at most. For the first dataset, only 6 new types have been experimented, and for the other we have established 9 types. The Table 2 gives the different maximum number of predicates for the query over physicochemical properties used for the several types for each dataset.

When we observe the different results, we see that the redescriptions are not pushed to grow up to the maximum number of predicates of the query over physicochemical prop-

Datasets	T. 71	T. 72	T. 73	T. 74	T. 75	T. 76	T. 77	T. 78	T. 79
(i)	15	20	25	30	35	40	None	None	None
(ii)	15	20	25	30	40	50	100	150	200
(iii)	15	20	25	30	40	50	60	70	80
(iv)	15	20	25	30	40	50	60	70	80

Table 2: Maximum number of predicates of the query over physicochemical properties of the new Types (denoted as T.).

erties. In fact, almost none of redescriptions have their right query composed by this maximum number of predicates (see Figure 27). The reasons of this auto-limit could be due to the principle of the algorithm: given a current redescription which satisfy the conditions, the algorithm try to extend it by adding a predicate at one of the two queries, however, if none extension improve the Jaccard coefficient, the algorithm does not extend the current redescription and output it. Remark that, an interesting improvement could be to allow some extensions of queries which do not improve the Jaccard coefficient of the current redescription in order to explore deeper the search space and to leave from a potential local optimum.

The results, in term of accuracy, are presented in the Figures 23–24–25–26. We see that the Jaccard coefficient of the redescriptions does not change from a given Type to another one. The results are not considerably better when we allow the queries to have a higher number of predicates. This could be due to the expressiveness of the language used in the algorithm. In fact, remember that, the ReReMi algorithm is based on a linearly parsable language for the queries. Thus the last predicate which extends a query has a direct impact on the support: it produces an union or an intersection between the support of this predicate and the support of the current query. This impact is too high to tune the redescriptions.

Moreover, the neuro-scientist would like to launch other experiments that only consider the conjunctions of true predicates (Type 1) with more predicates by queries. In fact, the associated language becomes very weak but could be easily interpretable. The results of these experiments are not detailed in this report but they have the same behavior than the results of the previous paragraph.

5 Enhancing redescription mining using hierarchies

The main contribution of this report is to generalize the notion of views detailed before. The redescription mining task needs a dataset which is composed by at least two views already given in the dataset: it requires this information or an expert has to explicitly split the dataset in several views thanks to his knowledge. The principle of our contribution is to work out a method which could be called a *views free* approach, i.e. the split between the attributes that describe the set of objects is not given before the execution but during the execution. The split is realized on the fly given a redescription and it is not necessary the same split for all redescriptions. The expressiveness of the redescription is thus

improved and the diversity of the redescriptions that could be obtain is larger. Indeed, some attributes within a single view could be more or less close from others but in the classical approach it is not taken into account. This principle rely on a hierarchy structure on the set of attributes and a distance measure between nodes into the hierarchy. Thus, two different sets of attributes which used to belong to the same view but which are quite distant from each other could now, thanks to the hierarchy, form a redescription.

5.1 Preliminaries

The contribution of this report is to switch from the views to a hierarchy structure method.

Hierarchy. A hierarchy, denoted as $\mathcal{H}(\mathcal{A}, \preceq)$, is a mathematical structure that represents a set $\mathcal{A}$ provided with a partial order (an antisymmetric preorder) relation $\preceq$. A hierarchy is also called a partially order set (poset) with a unique root. A partial order is a binary relation, denoted as $\preceq$, over a set $\mathcal{A}$ which the following assertions hold:

Reflexivity:	$a \preceq a$, with $a \in \mathcal{A}$
Antisymmetry:	if $a \preceq b$ and $b \preceq a$, then $a = b$, with $a, b \in \mathcal{A}$
Transitivity:	if $a \preceq b$ and $b \preceq c$, then $a \preceq c$, with $a, b, c \in \mathcal{A}$

Since this is a partial order, in the hierarchy, two elements a and b in $\mathcal{A}$ are not necessary comparable. We say that a and b are comparable if $a \preceq b$ or $b \preceq a$. In the case of $a \preceq b$ we say that b precedes a (a is a descendant of b) and a succeeds b (b is an ancestor of a). Moreover, b is said a direct ancestor of a if b is an ancestor of a and there does not exist another element $c \in \mathcal{A}$ ($c \neq b$ and $c \neq a$) such that $a \preceq c$ and $c \preceq b$. If a and b are comparable ($a \preceq b$ for example) we denote by $l(a, b)$ the length of the shortest path between a and b. Formally:

$l(a, b) = min(1 + l(c, b))$	where c is a direct ancestor of a
$l(a, b) = 1$	if b is a direct ancestor of a

Note that a direct ancestor b of a is an ancestor of a which length to a equals 1. We say that $c \in \mathcal{A}$ is a common ancestor of the elements a and b in $\mathcal{A}$ if $a \preceq c$ and $b \preceq c$. Observe that if $a \preceq b$, b is a common ancestor of a and b since $b \preceq b$. It could exist several common ancestors of two elements, but we say that those which have the minimum length with a and b are the direct common ancestors (they could be multiple).

Example. To illustrate these different notions related to the hierarchy, let us look at the Figure 1. This hierarchy is composed by 8 elements. Each element is represented by a node in the Figure 1. Note that for more readability, we assimilate a node with the element it represents. The arrow between two nodes refers to the fact that the two elements represented by the two nodes are comparable. For example, the arrow from the node A to the node B means that the element A is comparable to the element$B$ and we have $A \preceq B$.

Thus A is a descendant of B and B is a ancestor of A, and even a direct ancestor. The elements related to the nodes C and G are incomparable because there does not exist a list of arrow which could link one of the nodes to the other one. Their common ancestor is the element A, and it is their direct common ancestor. The length between A and G is $l(A, G) = 2$. The set of ancestors of the element related to the node H in the hierarchy is composed by $\{A, B, D, F\}$, and the elements B and F are its direct ancestors.

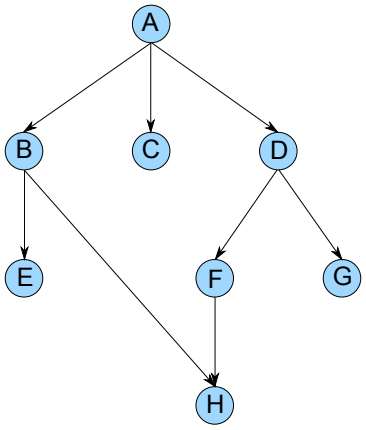


Figure 1: Example of hierarchy.

5.2 Problem

The aim of redescription is clear: characterize a group of objects by at least two different descriptions. Thanks to our approach we just need a hierarchy over the set of attributes. The aim is then to extract the best redescriptions which queries are far enough from each other and the attributes into each queries close enough. Formally, the objective of the hierarchical redescription mining is to obtain the $k-$best redescriptions (q_L, q_R) such that:

(1) $attr(q_L) \cap attr(q_R) = \emptyset$
(2) $d(attr(q_L), attr(q_R)) \geq d_{intra}$
(3) $d'(attr(q_L)), d'(attr(q_R)) \leq d_{inter}$
(4) $|supp(q_L)|, |sup(q_R)| \geq minsupp$
(5) $|supp(q_L)|, |sup(q_R)| \leq maxsupp$
(6) $sim(q_L), sim(q_R) \geq minsim$
(7) $pval(q_L, q_R) \leq maxpval$

The first constraint is to restrict the attributes in redescription: since we are not provide with several datasets, and an attribute has to be present at most once, we have to avoid multiple occurrences in the both queries of a redescription. The constraints (2) and (3) are related to the distance measure used. The attributes within a query have to be close enough from each other and the attributes of two queries of a redescription have to be far

enough from each other (see section 5.3 in which we discuss the possibilities and the choice of distance). Note that the d' distance corresponds to $d'(A) = maxd(a_i, a_j)\ for\ a_i, a_j \in A$, where A is a set of nodes. The thresholds d_{inter} and d_{intra} are fixed by the user. We discuss later about another approach with top-k redescriptions and multi-objectives optimization. The both constraints (4) and (5) prune uninteresting redescriptions where queries have a too low or too high cardinality of their support. The user has to specify a minimum support threshold to avoid too under-represented queries or over-represented ones. The constraint (6) consists in selecting only redescriptions which similarity between the two queries is higher than a fixed threshold $minsim$. We discuss later about which similarity we should use (see section 5.4). Note that the constraint (7) is related to the p-value criteria. Observe that in this approach we first only focus on queries q_L and q_R obtained by the conjunction of Boolean predicates. The improvements of the language and of the types could be realized later but it is not specially the aim of this approach.

Remark Note that the first constraint (1) for the redescriptions is to obtain an empty intersection between the attributes involved in the both queries. However, this constraint may be not really necessary: in fact, it depends on the data we will use. In some cases, a non-empty intersection between the attributes could be interesting, or at least not be an uninteresting result. On the other hand, the empty intersection between this two sets of attributes could result implicitly from the second constraint and the distance measure used. In fact, using an appropriate distance measure and fixing the d_{intra} parameter with a value high enough, the empty intersection could be imply.

5.3 Distance

The choice of the distance measure is one of the most important point. Indeed, as we remove redescriptions that do not respect the conditions over the distance, an inappropriate distance could have very uninteresting results, avoiding actionable redescriptions and keeping bad ones. The following of this section presents some distance measures that could be applied in a hierarchy structure, and then it explains how to aggregate to measure the distance between two sets of nodes.

5.3.1 Distance between two nodes

The distance between two nodes could be measure thanks to several measures, more or less intuitive.

Naive distance measure. The naive distance is a classical measure in a hierarchy structure, introduced initially for a tree structure. It is based on a naive intuition: the distance between two nodes of a hierarchy is the sum of the distances between each node to their direct common ancestor. However, the main problem of this distance measure is that it is to general and too simple: it only considers the distance with the direct common ancestor of the two attributes. The formal definition is:

$$d_{naive}(a_i, a_j) = l(a_i, a_{i\diamond j}) + l(a_j, a_{i\diamond j})$$

For example, the naive distance between the nodes E and F in the Figure 1 is : $d_naive(E, F) = l(E, A) + l(F, A) = 2 + 2 = 4$, and $d_{naive}(E, H) = l(E, B) + l(H, B) = 1 + 1 = 2$.

Advanced distance measure. To improve the previous measure, an option is to take into account the position of the direct common ancestor in the hierarchy. The closer the direct common ancestor is, the greater the distance between the two attributes is. Thus, the advanced distance measure could be expressed as follows:

$$d_{advanced}(a_i, a_j) = \frac{l(a_i, a_{i\diamond j}) + l(a_j, a_{i\diamond j})}{1 + l(a_{i\diamond j}, a_{root})}$$

This is the measure we used in the remainder of this report. Note that a_{root} is the root node of the hierarchy. Moreover, the advanced measure is not a distance as the mathematicians have defined it because the triangle inequality does not hold. For example, in the hierarchy of the Figure 1, the advanced distance between the nodes E and F is : $d_{advanced}(E, F) = \frac{l(E,A)+l(F,A)}{1+l(A,A)} = \frac{2+2}{1+0} = 4$, and $d_{advanced}(E, H) = \frac{l(E,B)+l(H,B)}{1+l(B,A)} = \frac{1+1}{1+1} = 1$. Note that with the advanced distance, E and H are closer than for the naive distance because their common ancestor is deeper in the hierarchy.

5.3.2 Distance between two sets of nodes

Once we defined the distance measure between two nodes, it is required to extend it to sets of nodes. Different options could be chosen: the maximum of the distances between one node in the first set and another node in the last set, the minimum of the distances, the mean of the distances, or even a harder aggregation method. However, considering our constraints on the distances measure, we chose two different aggregation methods : one for the constraint (2) and another one for the constraint (3). For the first one, we chose that the minimum of the distances, because the constraint (2) is related to the separation of attributes of the two automatic views, so two sets of nodes are separated enough if the minimum distance between two nodes of each set are far enough. For the constraint (3), as we say previously, we use the d' distance which corresponds in fact to $d'(A) = d(A, A)$ where the aggregation method is the maximum of the distances in order to have each node close to each other in the same query.

For example, let us consider the Figure 1, the distance between the sets of nodes $S_1 = \{B, E\}$ and $S_2 = \{F, G, H\}$ equals the minimum of the distances between a node in S_1 and another node in S_2, i.e. $d(S_1, S_2) = d_{advanced}(B, H) = \frac{0+1}{1+1} = \frac{1}{2}$. And the distance $d(S_2, S_2) = d'(S_2) = d_{advanced}(G, H) = \frac{1+2}{1+1} = \frac{3}{2}$ corresponds to the maximum of the distances between nodes of S_2.

5.4 Similarity

As we see previously in this report, the similarity is a turning point in redescription mining because it is the parameter that accepts or not a redescription as result. All the

previous approaches in the state of art used the Jaccard coefficient to measure the similarity of the both queries of a redescription. We observe that, in some cases, this measure could be too strict and interesting redescriptions could be ignored because their Jaccard coefficient is too low. However in our approach we do not switch to another similarity measure, because the Jaccard coefficient gives a idea about the interest of a redescription, it is easily computable and it will be easier to compare our results to the existing approaches.

5.5 Algorithm

To handle this new approach, I developed an algorithm which is able to deal with such a hierarchy on the attributes of the database. I developed a first version of this algorithm that could be improved in the next versions by reducing the number of analyzed attributes in the hierarchy. This first version is more assimilate to a prototype of the algorithm. The principle is based on those of the two last described approaches in the section 3. The input of this algorithm corresponds to the name of the file containing the hierarchy at the XML format, the name of the transaction file, the maximum and minimum distance thresholds, the maximum and minimum support thresholds and the minimum similarity threshold. In this first version, the algorithm does not handle the p-value of the redescription but it could be simply added as a post-process. It performs two main steps to mine redescriptions : (i) the first one finds the k-best initial redescriptions, and (ii) the second one has to extend this k-best redescriptions while the similarity measure is improved. The formal principle of the algorithm is given in the Algorithm 1.

The first step of the algorithm is performed thanks to a depth-first approach in the hierarchy: the algorithm keeps a node in the hierarchy if the constraint on the support holds. Once it selects such a node, the algorithm will try to find a second node which corresponds to the predicate of the second query. To do that, it proceeds again a depth-first approach and verify all constraints (support, distance, and similarity), if they hold a new redescription is created formed by the predicate of the first node for the first query and the predicated associated to the second node for the last query of the redescription. If this redescription takes part into the k-best redescriptions till then, it is added in the set of the k-best redescription (and a lower redescription could be removed from this set), otherwise this redescription is not added and the algorithm continues to explore the search space. The algorithm searches the second node (for the predicate of the second query of the redescription) into the set of nodes that are not been analyzed during the first search for the first node (for the predicate of the first query of the redescription). Despite this reduction of the search space for the search of the second predicate, the theoretical complexity of this first step is $O(n^2)$, where n is the number of attributes (or nodes in the hierarchy), since if all attributes satisfies the constraints, for the first node of the hierarchy we have to analyze n nodes, then $n-1$, $n-2$, ... ,1 which equals to $\frac{n\times(n-1)}{2}$.

The second step starts from the at most k-best initial redescriptions obtained at the previous step. For each one, it tries to extend it by adding a predicate in one of the both queries of the redescription, by analyzing all the nodes of the hierarchy by a depth-first approach. If the similarity of a extended redescription is higher than the initial one, the best extended redescription replaces the initial redescription, and the algorithm tries to

extend once again until no better extended redescription has been found. Thus, for each step of extension, the complexity is $O(n)$ with n is the number of nodes. But for a given redescription there could be at most n step (that corresponds to the cases where all the attributes are in one of the two queries of the redescriptions). Thus, the complexity of all extension steps for all the k-best redescriptions is $O(k \times n^2)$.

This approach could be considered as naive and may be easily improved but we face a much more complicated problem than it seems. In fact, there exists no monotonic properties : the support does not verify a monotonic properties in the hierarchy (the support of the parents of a node is not necessary greater or equal to the support of this node), the distance could not be used to prune the search space because of the possibility of having several parents, and the similarity is clearly not monotonic in the hierarchy. However some improvements could be realized: during the extension step of the algorithm, one could save the set of nodes that respect the constraint (3) for each query of each k-best redescriptions. The algorithm would only analyze these sets of nodes for each query to extend it. Nevertheless, this could reduce the run-time of the algorithm but it certainly increase the need of memory space.

5.6 Experiments

Once the algorithm has been developed, the objective is now to know its efficiency for both quantitative and qualitative aspects. For that, we retrieve new datasets to experiment with a hierarchy, and we proceed to several experiments with different parameters to understand the behavior of the algorithm.

5.6.1 Data

As the datasets provided by the CRNL do not include a hierarchy over the physico-chemical properties and odors qualities (the neuro-scientists could not manage to establish this hierarchy yet), we use a dataset taken from the website of Amazon[1] to test our generic approach on another application domain. The dataset[20] is available online. It contains the ratings of several products by customers. Amazon allows its consumers to evaluate products by giving a rate between 1 and 5. Moreover, Amazon provides a hierarchy on its set of products which has been used in our experiments. Thus the objectives of this experiment is to perform a recommendation system on different kind of products to advise customers. The initial dataset contains 519 781 products which take part into the hierarchy composed by 49 732 nodes (for example, Book - Science Fiction Book - ...) There are 6 526 253 ratings by 1 555 569 different customers. However we use a reduced dataset in order to the algorithm could achieve the run: in the reduced dataset we consider only the ratings since 2003, now there are 1 929 512 ratings by 511 261 different customers. The number of products and of nodes in the hierarchy do not change. Moreover, the ratings have been discretized in three categories : the ratings 1 and 2 are considered as ratings of dissatisfaction, the rating 3 as neutral and the ratings 4 and 5 as ratings of satisfaction. In

[1] http://www.amazon.com

this dataset all products are the leaves of the hierarchy, and only the products are rated so we use an aggregation method to provide ratings to nodes in the hierarchy: thus for each node we compute the number of satisfaction, neutral and dissatisfaction ratings for each customer. Then we say that the support of a node (or a product) equals the set of customers whose satisfaction ratings for this node is greater than the sum of neutral and dissatisfaction ratings. The Figure 28 and Figure 29 give some other precision about the characteristics of this dataset.

5.6.2 Results

The results of these experiments are not conclusive. In fact the dataset, even if it has been reduced, is too huge and not well adapted for this algorithm. The algorithm does not manage to proceed entirely the run: it requires to much resources (memory and time). It could be easily understandable comparing with the *ReReMi* algorithm: its run-time is up to more than 2 days for some experiments considering a dataset that contains nearly 300 attributes and 1,689 molecules maximum. But the Amazon dataset contains more than 550,000 attributes (which correspond to the nodes and products into the hierarchy) and more than 500,000 customers in the reduced dataset. Even if our algorithm is less complete than the ReReMi algorithm (just conjunction and Boolean data), handling such a dataset is not possible with this first version of our algorithm. In fact, there is a gap between the potential of the algorithm and the objectives of this dataset: our algorithm is not design to deal with massive and heterogeneous data. Indeed, the algorithm performs a beam-search approach that executes a complete exploration at the first level to detect the best redescriptions, but this complete exploration on this dataset requires too much resources (without pruning methods, it has to performs n^2 tests with n is the number attributes). Moreover, reducing this Amazon dataset could not permit to obtain results because if we select only tens of nodes in the hierarchy, the probability that same subset of tens of customers have rated these nodes is quite low. So the constraints on support will not hold and the results will not mean anything.

However, to test a little the algorithm, I created a small dataset based on the Amazon dataset: I select 27 existing products, and I consider the minimal hierarchy that contains these products. It leads to a hierarchy with 403 nodes (including the 27 products). I do not consider the existing ratings on these products but I randomly provide 136 ratings, respecting the initial distribution of the Amazon dataset, on the 27 products for 10 created virtual customers (the distribution of number products rated by a customer is no longer respected, due to the last remark of the previous paragraph). I run different experiments changing parameters of the algorithm. The run-times are very low (less than 3 seconds in mean), and the number of redescriptions often equals the maximum tolerated number of the beam-search approach, namely 100. Even if the results could not be interpreted, as the ratings have been created randomly, we can observe some phenomenons: the similarity of many initial redescriptions equals 1, and so they could not be extended. Moreover, some attributes in the hierarchy could not be interpreted in a redescription: such as the node "Directors" which is included in some redescriptions but it does not have any interpretation. Furthermore, the language we use is too strict, when a predicate is added to a query, as

the support of the redescription decreases, the constraint on frequency could not be held.

Now that we have identified the main locks, we can propose some solutions that could solved these problems. First of all, the beam-search method is not adapted for massive data: it then comes with an initial choice of patterns that serve as seeds for exploration. Various approaches can be considered, such as a step of full search on a weak language, which result is analyzed to produce these heuristic exploration seeds of the search space of a much more expressive language. Secondly, all nodes within the hierarchy in the dataset have to be meaningful in a redescription, a cleaning of the hierarchy could be required to increase the actionability of the redescriptions. Then, a parameter-free seems to be another important improvement to do: in fact, we could not evaluate a redescription only on its similarity measure, we should also take into account the support and the distance. This kind of methods require a multi-objectives optimization, and is linked to the skylines notions in pattern mining [[26]–[29]–[25]]. Thus, the user is no longer obliged to fix all parameters, the algorithm will select the redescriptions that best satisfy the conditions. Finally, as we discuss previously, the similarity measure is an important point, and a more specific study has to be performed to know which one is the most appropriate.

6 Conclusion

The aim of this report was the characterization of odors from several datasets in order to establish the existing links between physicochemical properties of a molecule and its olfactory qualities. For that, I studied redescription mining to understand the main notions. I proceeded a large set of experiments on datasets provided by the CRNL, which allows me to be aware of the possible improvements of the previous works. The meetings with the neuro-scientists allow me to be aware of their needs and about the changes we have to perform in the approach. Thus, I formulate a proposition to avoid the use of pre-established views (or partition of attributes of the dataset) following a hierarchy based approach on the set of attributes. Thus, it is no longer necessary to have several data sources describing a same set of objects, but now, a single dataset is required, and the method leads to an automatic split of the set of attributes to mine redescriptions. This approach has been first theoretically presented and then some experiments have been realized. The results show some problems on the approach we developed but also on the hierarchy we used.

Several locks have been identified and improvements have been formulated to be explore in the future work. In fact, a beam-search approach is not adapted within a massive and heterogeneous data context: the complete exploration during the first step requires too much resources which is the same problem in subgroups discovery [[18]–[31]]. An approach using well-chosen seeds at the first step could lead to a granular exploration based on heuristics. Moreover, the approach developed here could be switch to a free-parameter methods avoiding the necessity of fixing the several thresholds thanks to multi-objectives approaches. Thus the perspectives of future work are quite various and could fit into a paradigm of distributed mining to consider the issue of big data.

References

[1] S. Arctander. *Perfume and flavor chemicals:(aroma chemicals)*, volume 2. Allured Publishing Corporation, 1969.

[2] G. Bosc, M. Kaytoue, C. Raïssi, and J.-F. Boulicaut. Fouille de motifs séquentiels pour l'élicitation de stratégies à partir de traces d'interactions entre agents en compétition. In C. Reynaud, A. Martin, and R. Quiniou, editors, *Extraction et Gestion des Connaissances*, volume E-26 of *RNTI*, pages 359–370. Hermann-Éditions, 2014. [Communication personnaly presented].

[3] G. Bosc, M. Kaytoue, C. Rassi, and J.-F. Boulicaut. Strategic Pattern Discovery in RTS-games for E-Sport with Sequential Pattern Mining. In *Machine Learning and Data Mining for Sports Analytics MLSA 2013 co-located with ECML/PKDD, Praga, Czech Republic*, Sept. 2013. No published proceedings. The paper is available as RR-LIRIS-2013-012, 11 pages.

[4] G. Bosc, M. Kaytoue, C. Rassi, J.-F. Boulicaut, and P. Tan. Mining Balanced Patterns in Real-Time Strategy Games. In T. Schaub, editor, *21st Europen Conference on Artificial Intelligence (ECAI'14)*, Aug. 2014. Accepted as short paper, 2 pages.

[5] G. Bosc, M. Kaytoue, C. Rassi, J.-F. Boulicaut, and P. Tan. Mining Balanced Patterns in Real-Time Strategy Games. Technical Report RR-LIRIS-2014-007, LIRIS UMR 5205, May 2014. An extended version and supplementary materials corresponding to the results to appear in ECAI procedings (ECAI 2014; http://www.ecai2014.org/), 17 pages.

[6] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[7] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[8] E. Galbrun. *Methods for Redescription Mining*. PhD thesis, University of Helsinki, Finland, 2013.

[9] E. Galbrun and A. Kimmig. Towards finding relational redescriptions. In J.-G. Ganascia, P. Lenca, and J.-M. Petit, editors, *Discovery Science*, volume 7569 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2012.

[10] E. Galbrun and P. Miettinen. A case of visual and interactive data analysis: Geospatial redescription mining. In *Instant Interactive Data Mining Workshop at ECML-PKDD*, volume 12, 2012.

[11] E. Galbrun and P. Miettinen. From black and white to full color: extending redescription mining outside the boolean world. *Statistical Analysis and Data Mining*, 5(4):284–303, 2012.

[12] E. Galbrun and P. Miettinen. Siren: an interactive tool for mining and visualizing geospatial redescriptions. In Q. Yang, D. Agarwal, and J. Pei, editors, *KDD*, pages 1544–1547. ACM, 2012.

[13] A. Gallo, P. Miettinen, and H. Mannila. Finding subgroups having several descriptions: Algorithms for redescription mining. In *SDM*, pages 334–345. SIAM, 2008.

[14] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[15] K. Kaeppler and F. Mueller. Odor classification: a review of factors influencing perception-based odor arrangements. *Chemical senses*, 38(3):189–209, 2013.

[16] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Revisiting numerical pattern mining with formal concept analysis. In T. Walsh, editor, *IJCAI*, pages 1342–1347. IJCAI/AAAI, 2011.

[17] R. M. Khan, C.-H. Luk, A. Flinker, A. Aggarwal, H. Lapid, R. Haddad, and N. Sobel. Predicting odor pleasantness from odorant structure: pleasantness as a reflection of the physical world. *The Journal of Neuroscience*, 27(37):10015–10023, 2007.

[18] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in knowledge discovery and data mining*, pages 249–271. American Association for Artificial Intelligence, 1996.

[19] D. Kumar. *Redescription mining: Algorithms and applications in bioinformatics*. PhD thesis, Department of Computer Science Virginia Tech, Blacksburg, VA 24061, 2007.

[20] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

[21] J. Li, H. Li, L. Wong, J. Pei, and G. Dong. Minimum description length principle: Generators are preferable to closed patterns. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 409. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[22] U. J. Meierhenrich, J. Golebiowski, X. Fernandez, and D. Cabrol-Bass. De la molécule à lodeur. *Lactualité chimique*, (289):29, 2005.

[23] P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10:377–403, 2009.

[24] L. Parida and N. Ramakrishnan. Redescription mining: Structure theory and algorithms. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 837–844. AAAI Press / The MIT Press, 2005.

[25] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang. Towards multidimensional subspace skyline analysis. *ACM Transactions on Database Systems (TODS)*, 31(4):1335–1381, 2006.

[26] C. Raïssi, J. Pei, and T. Kister. Computing closed skycubes. *Proceedings of the VLDB Endowment*, 3(1-2):838–847, 2010.

[27] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm. Turning cartwheels: an alternating algorithm for mining redescriptions. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *KDD*, pages 266–275. ACM, 2004.

[28] C. Sezille and M. Bensafi. De la molécule au percept. *Biofutur*, (346):24–26, 2013.

[29] A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux. Mining dominant patterns in the sky. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 655–664. IEEE, 2011.

[30] L. Turin. A method for the calculation of odor character from molecular structure. *Journal of theoretical biology*, 216(3):367–385, 2002.

[31] S. Wrobel, J. Komorowski, and J. Z. (eds. An algorithm for multi-relational discovery of subgroups. pages 78–87. Springer, 1997.

[32] M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *SDM*, pages 457–473. SIAM, 2002.

[33] M. J. Zaki and N. Ramakrishnan. Reasoning about sets using redescription mining. In R. Grossman, R. J. Bayardo, and K. P. Bennett, editors, *KDD*, pages 364–373. ACM, 2005.

Annex

id	[1] South Hemisp.	[2] Atlantic Ocean	[3] Pacific Ocean	[4] Continent	[5] Area ($10^9 km^2$)	[6] Elev (m)
CA	*false*	*true*	*true*	{North America}	9.98	5959
CL	*true*	*true*	*true*	{South America}	0.76	6893
CN	*false*	*false*	*true*	{Asia}	9.71	8850
FR	*false*	*true*	*false*	{Europe}	0.64	4810
GB	*false*	*true*	*false*	{Europe}	0.24	1343
MX	*false*	*true*	*true*	{North America}	1.96	5636
MZ	*true*	*false*	*false*	{Africa}	0.79	2436
RU	*false*	*false*	*true*	{Asia, Europe}	17.10	5642
US	*false*	*true*	*true*	{North America}	9.63	6194

(a) $\mathcal{D}_1$ related to the *Geographical* view

id	[1] History of Communism	[2] History of colonialism	[3] Political Regime	[4] Population ($10^6 hab.$)
CA	*false*	*false*	*Monarchy*	33.476
CL	*false*	*false*	*Republic*	16.572
CN	*true*	*false*	*Republic*	1353.821
FR	*false*	*true*	*Republic*	65.350
GB	*false*	*true*	*Monarchy*	63.181
MX	*false*	*false*	*Republic*	115.296
MZ	*true*	*false*	*Republic*	22.894
RU	*true*	*false*	*Republic*	143.300
US	*false*	*false*	*Republic*	315.550

(b) $\mathcal{D}_2$ related to the *Political* view

Figure 2: Transaction databases.

tid	X_1	X_2	X_3	X_4
o_1	$false$	$false$	$false$	$true$
o_2	$true$	$false$	$true$	$false$
o_3	$true$	$true$	$false$	$false$
o_4	$false$	$true$	$true$	$false$
o_5	$false$	$false$	$false$	$true$

(a) $\mathcal{D}_1$

tid	Y_1	Y_2	Y_3	Y_4
o_1	$true$	$false$	$false$	$true$
o_2	$true$	$true$	$false$	$true$
o_3	$false$	$true$	$true$	$false$
o_4	$false$	$true$	$false$	$false$
o_5	$false$	$false$	$true$	$true$

(b) $\mathcal{D}_2$

Figure 3: Transaction databases.

Paper	Nb of databases	Form of queries	Principle	Characteristics	Data Type
[27]	2	DNF	Decision tree, greedy	Deterministic, Exact	Boolean
[24]	1	negation, and, or	Bi-clustering, closed, generators	Deterministic, Exact, Jaccard	Boolean
[33]	1	negation, and	minimal, generators	Deterministic, Exact	Boolean
[13]	n	negation, and, or	Beam-Search	Deterministic, Jaccard, greedy, support	Boolean
[11]	2	negation, and, or, linearly parsable queries	Beam-Search	Jaccard, support	Boolean, numerical, categorical

Table 3: Summary table.

tid	Y_1	Y_2	Y_3	Y_4	class
o_1	$true$	$false$	$false$	$true$	X_4
o_2	$true$	$true$	$false$	$true$	X_1
o_3	$false$	$true$	$true$	$false$	X_1
o_4	$false$	$true$	$false$	$false$	X_2
o_5	$false$	$false$	$true$	$true$	X_4

Figure 4: The label database (Step 1).

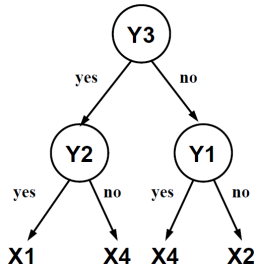


Figure 5: The top tree.

tid	X_1	X_2	X_3	X_4	class
o_1	$false$	$false$	$false$	$true$	$(Y_3 \wedge \neg Y_2) \cup (\neg Y_3 \wedge Y_1)$
o_2	$true$	$false$	$true$	$false$	$(Y_3 \wedge \neg Y_2) \cup (\neg Y_3 \wedge Y_1)$
o_3	$true$	$true$	$false$	$false$	$Y_3 \wedge Y_2$
o_4	$false$	$true$	$true$	$false$	$\neg Y_3 \wedge \neg Y_1$
o_5	$false$	$false$	$false$	$true$	$(Y_3 \wedge \neg Y_2) \cup (\neg Y_3 \wedge Y_1)$

(a) Step 2

tid	Y_1	Y_2	Y_3	Y_4	class
o_1	$true$	$false$	$false$	$true$	$(X_3 \wedge X_1) \cup (X_4 \wedge \neg X_3)$
o_2	$true$	$true$	$false$	$true$	$(X_3 \wedge X_1) \cup (X_4 \wedge \neg X_3)$
o_3	$false$	$true$	$true$	$false$	$\neg X_4 \wedge \neg X_3$
o_4	$false$	$true$	$false$	$false$	$X_3 \wedge \neg X_1$
o_5	$false$	$false$	$true$	$true$	$(X_3 \wedge X_1) \cup (X_4 \wedge \neg X_3)$

(b) Step 3

Figure 6: Label databases (Step 2 and 3).

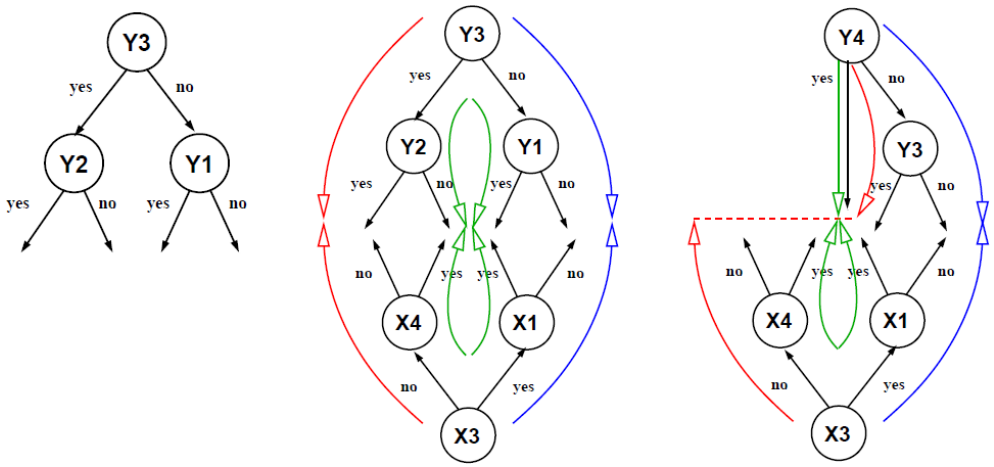


Figure 7: Alternating tree growing in the CARTwheels algorithm.

tid	d_1	d_2	d_3	d_4	d_5	d_6	d_7
o_1	$true$	$true$	$false$	$true$	$true$	$true$	$false$
o_2	$false$	$true$	$true$	$false$	$true$	$false$	$true$
o_3	$true$	$true$	$false$	$true$	$true$	$true$	$false$
o_4	$true$	$true$	$true$	$false$	$true$	$true$	$true$
o_5	$true$	$true$	$true$	$true$	$true$	$true$	$true$
o_6	$false$	$true$	$true$	$true$	$false$	$false$	$false$

Figure 8: Transaction database.

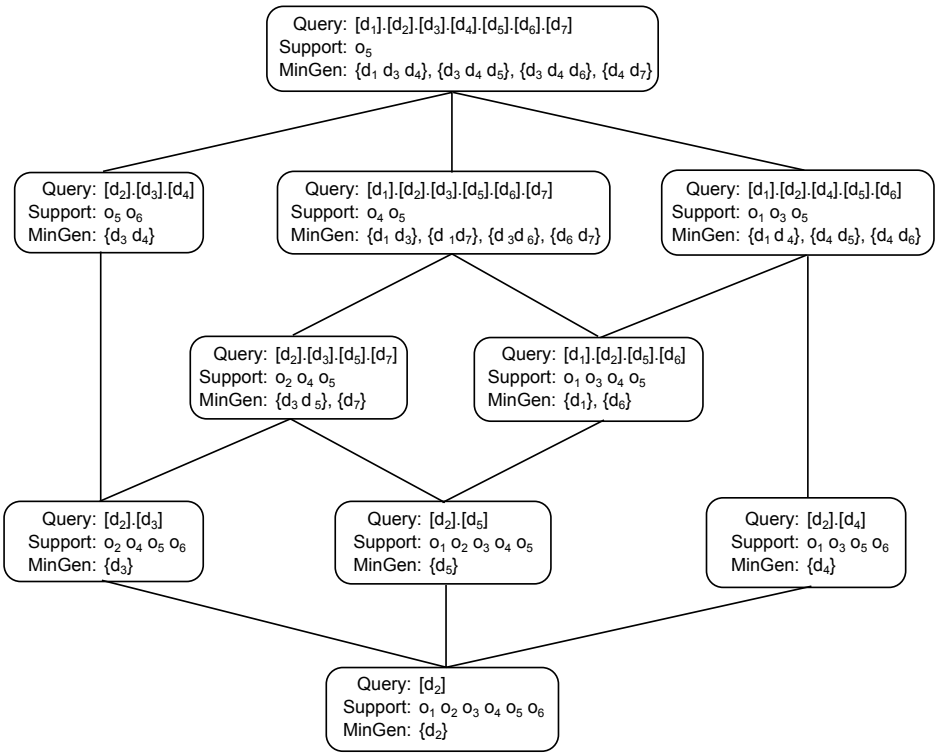
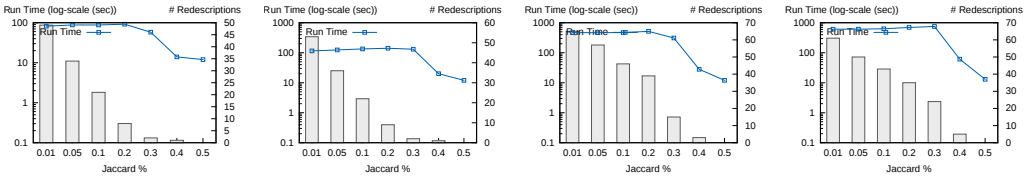


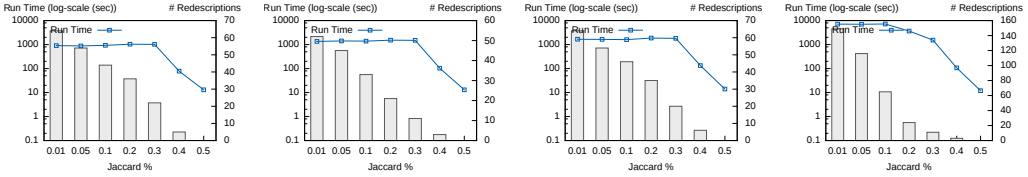
Figure 9: The closed queries lattice.

Redescription	Support
$([d_1], [d_6])$	$\{o_1, o_3, o_4, o_5\}$
$([d_3] \wedge [d_5], [d_7])$	$\{o_2, o_4, o_5\}$
$([d_1], [d_5]) \mid [d_4]$ $([d_5], [d_6]) \mid [d_4]$ $([d_1], [d_6]) \mid [d_4]$	$\{o_1, o_3, o_5\}$
$([d_1] \wedge [d_3], [d_6] \wedge [d_7])$ $([d_1] \wedge [d_7], [d_3] \wedge [d_6])$ $([d_3], [d_7]) \mid [d_6]$ $([d_3], [d_7]) \mid [d_1]$ $([d_1], [d_6]) \mid [d_3]$ $([d_1], [d_6]) \mid [d_7]$	$\{o_4, o_5\}$
$([d_1], [d_5]) \mid [d_3] \wedge [d_4]$ $([d_1], [d_6]) \mid [d_3] \wedge [d_4]$ $([d_5], [d_6]) \mid [d_3] \wedge [d_4]$ $([d_1] \wedge [d_3], [d_7]) \mid [d_4]$ $([d_3] \wedge [d_6], [d_7]) \mid [d_4]$ $([d_3] \wedge [d_5], [d_7]) \mid [d_4]$	$\{o_5\}$

Figure 10: Non redundant redescriptions.

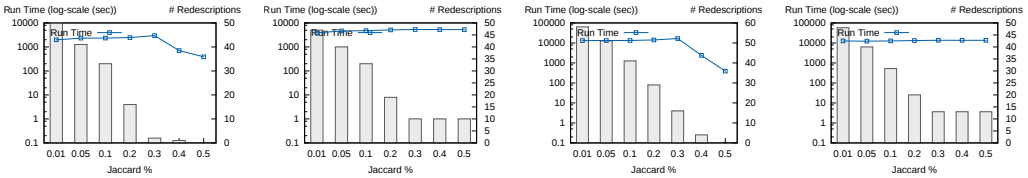


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

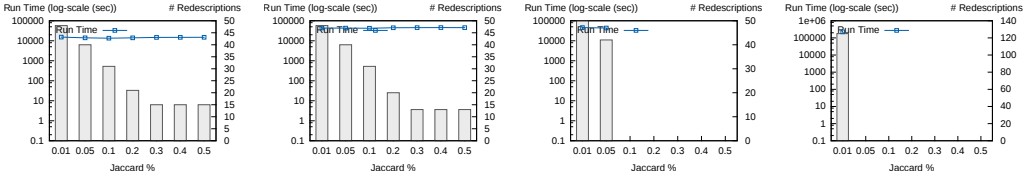


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 11: Run time and number of patterns for the first dataset (i).

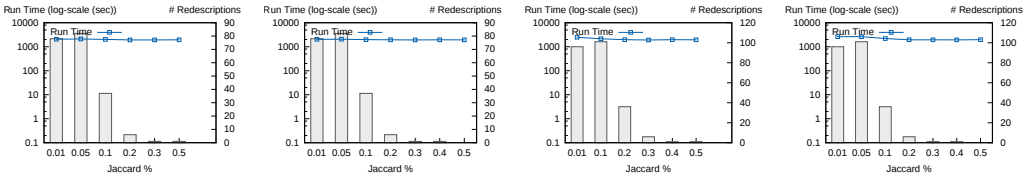


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

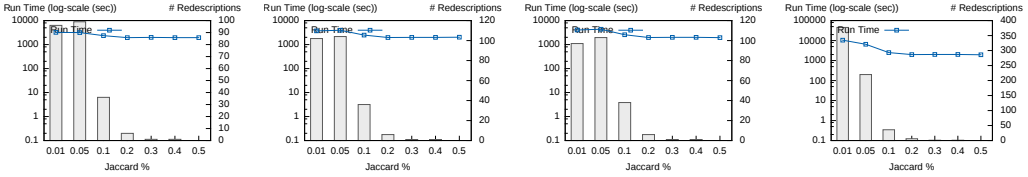


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 12: Run time and number of patterns for the second dataset (ii).

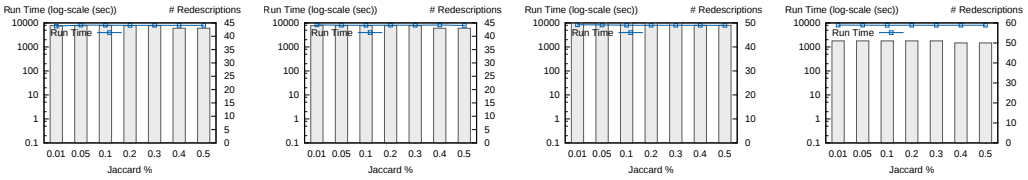


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

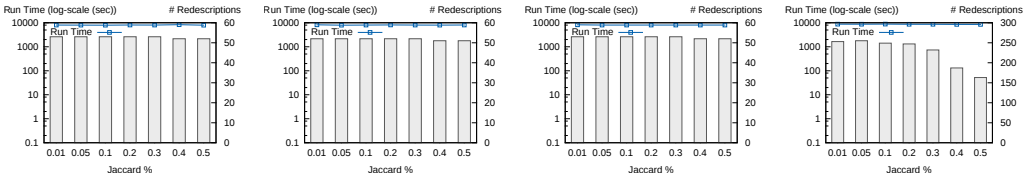


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 13: Run time and number of patterns for the third dataset (iii).

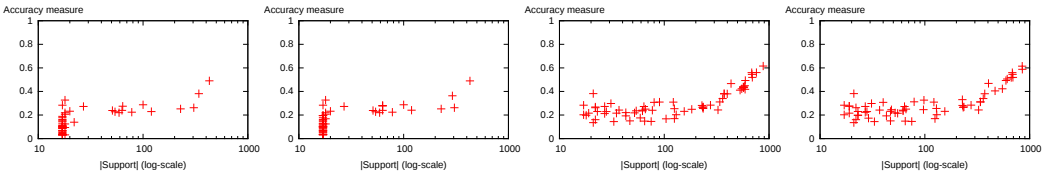


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

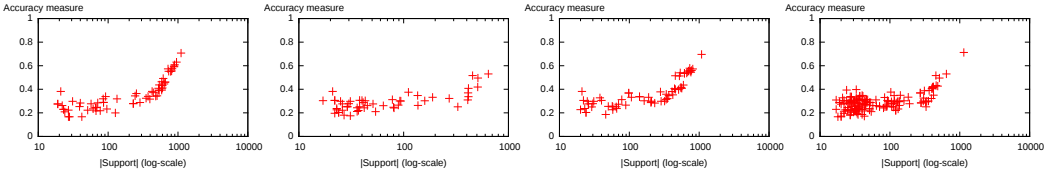


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 14: Run time and number of patterns for the fourth dataset (iv).

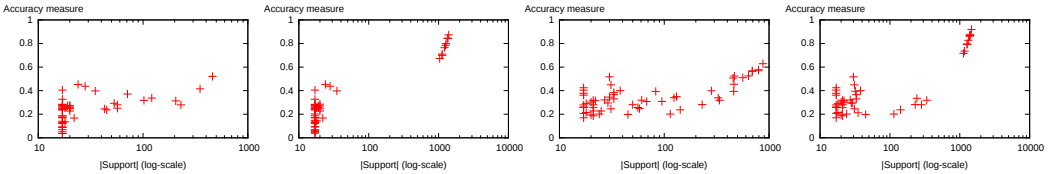


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

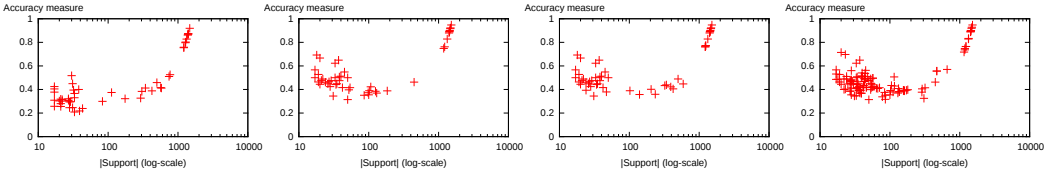


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 15: Accuracy for the first dataset (i).

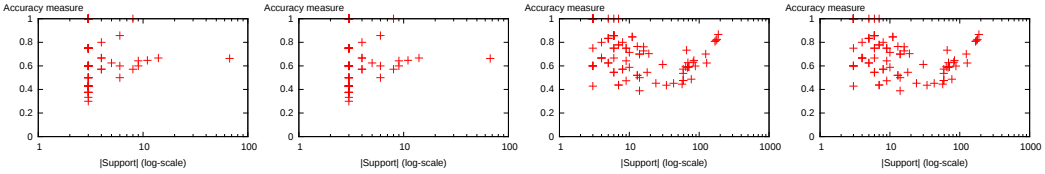


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

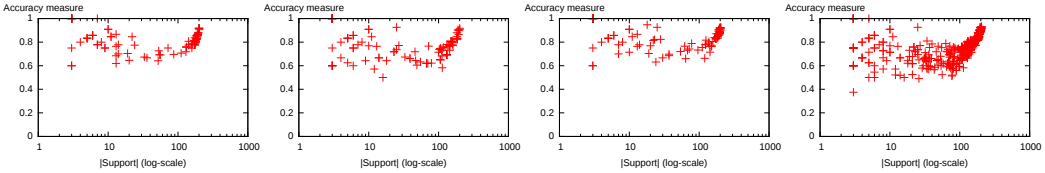


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 16: Accuracy for the second dataset (ii).

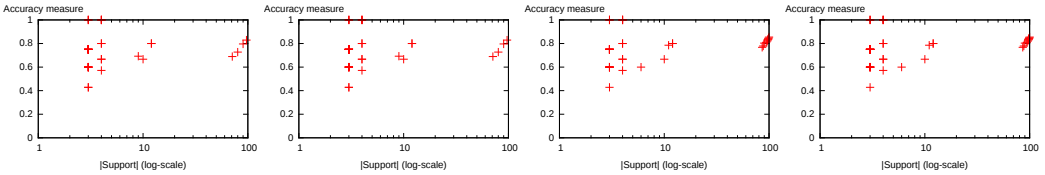


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

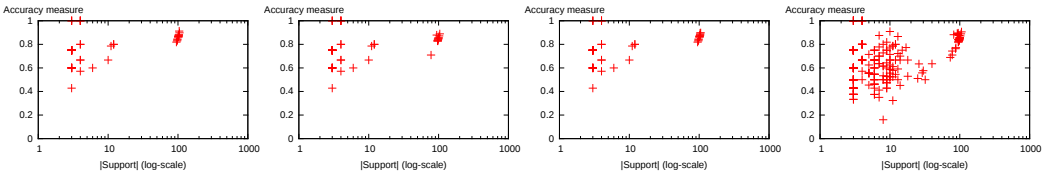


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 17: Accuracy for the third dataset (iii).

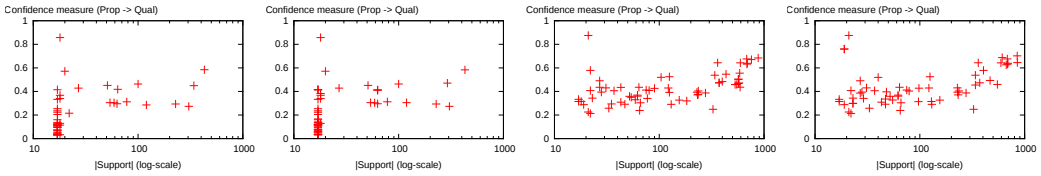


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

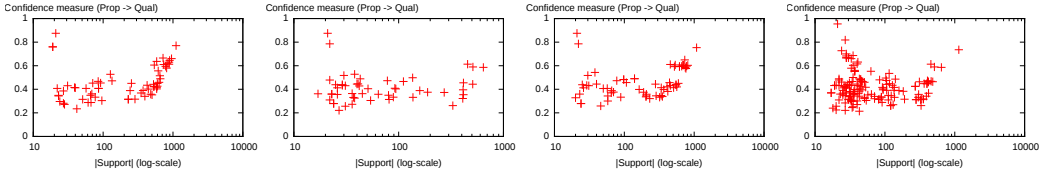


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 18: Accuracy for the fourth dataset (iv).

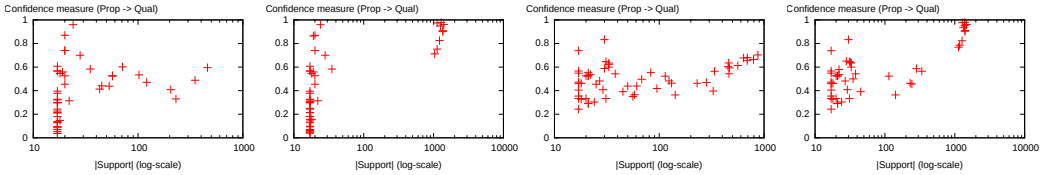


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

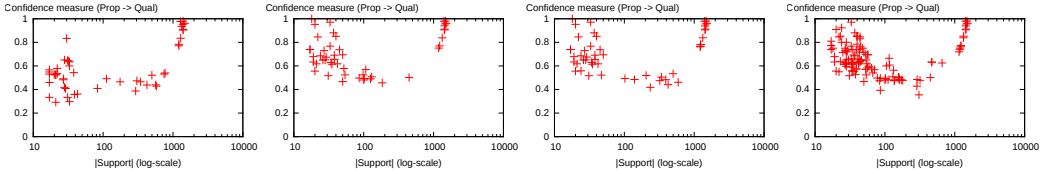


(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 19: Confidence rules Proprieties to Qualities for the first dataset (i).

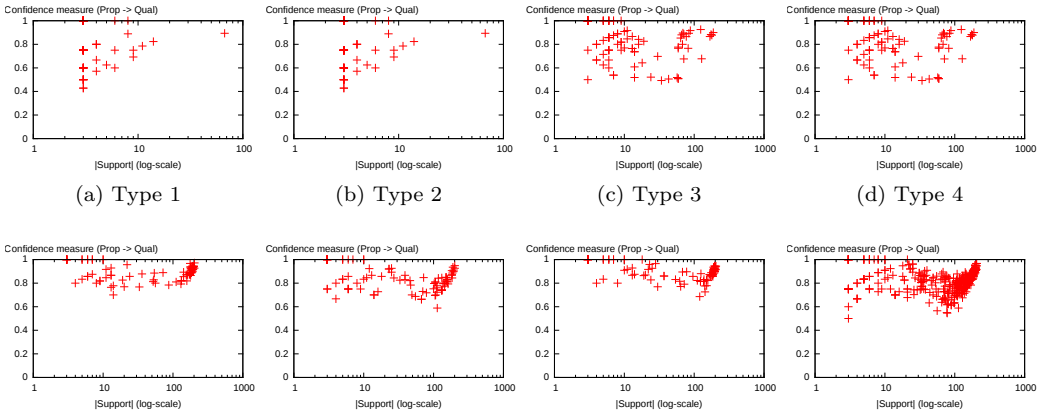


(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4



(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

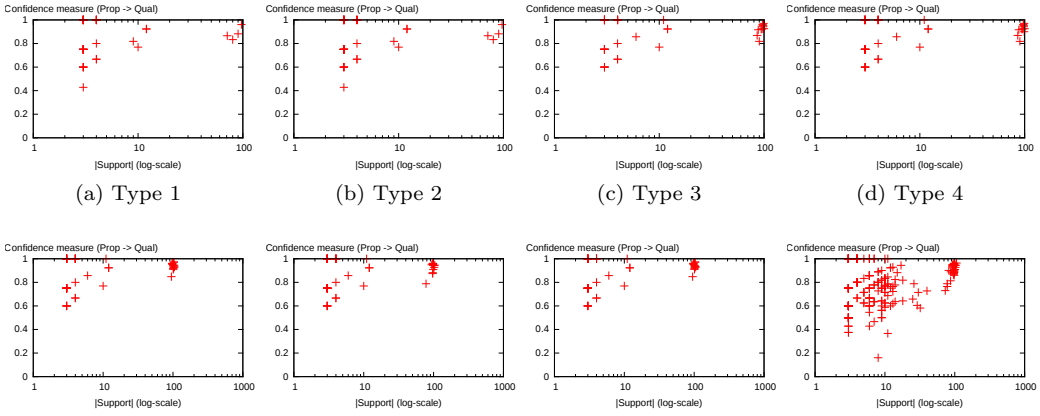
Figure 20: Confidence rules Proprieties to Qualities for the second dataset (ii).



(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

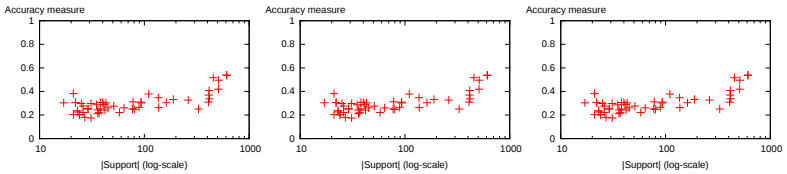
Figure 21: Confidence rules Proprieties to Qualities for the third dataset (iii).



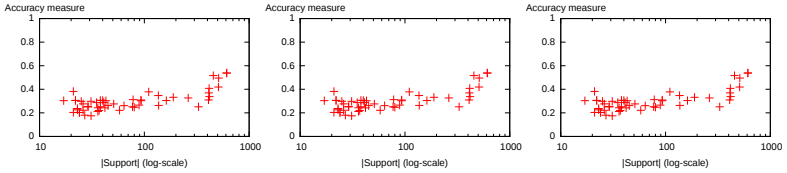
(a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

(e) Type 5 (f) Type 7 (g) Type 8 (h) Type 9

Figure 22: Confidence rules Proprieties to Qualities for the fourth dataset (iv).

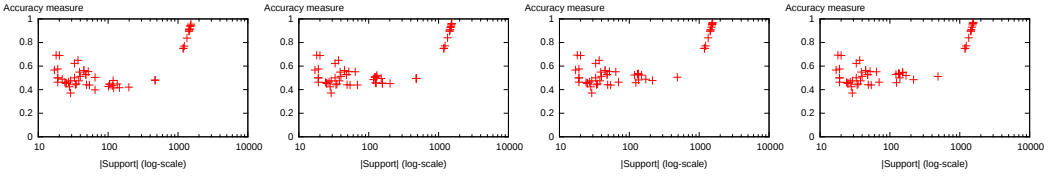


(a) Type 71 (b) Type 72 (c) Type 73

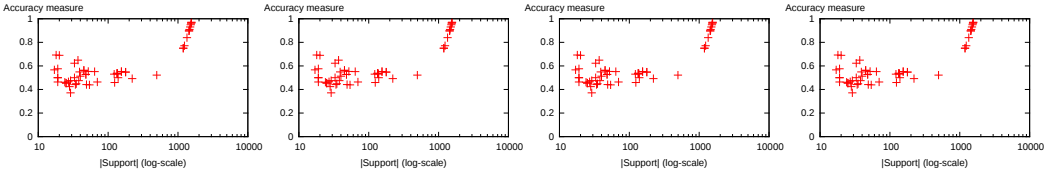


(d) Type 74 (e) Type 75 (f) Type 76

Figure 23: Accuracy for the first dataset (i).

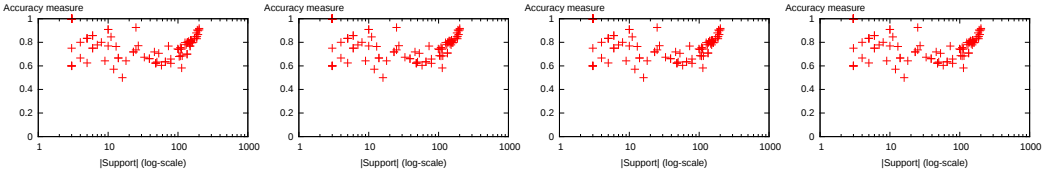


(a) Type 71 (b) Type 72 (c) Type 73 (d) Type 74

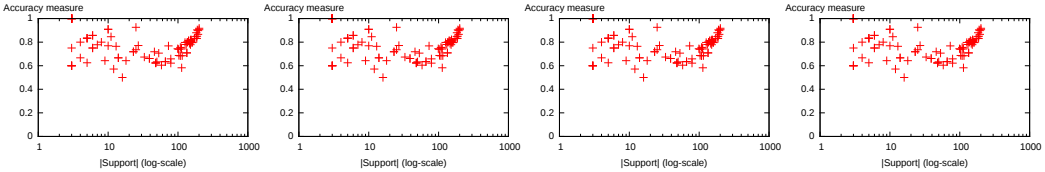


(e) Type 75 (f) Type 77 (g) Type 78 (h) Type 79

Figure 24: Accuracy for the second dataset (ii).

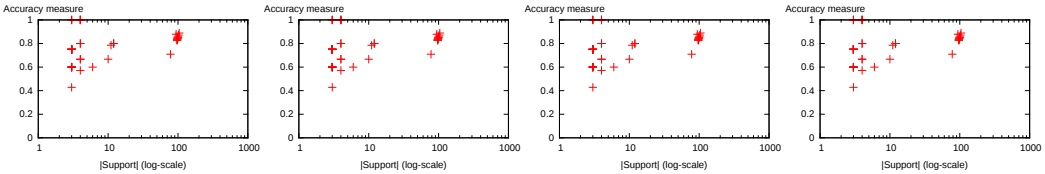


(a) Type 71 (b) Type 72 (c) Type 73 (d) Type 74

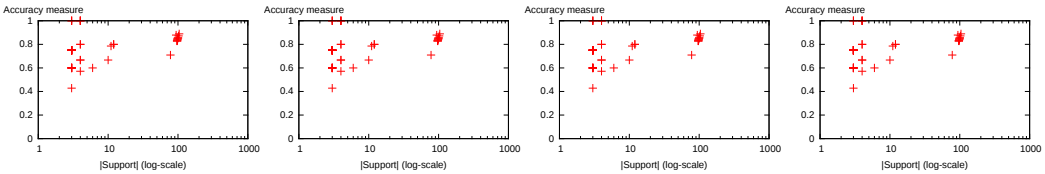


(e) Type 75 (f) Type 77 (g) Type 78 (h) Type 79

Figure 25: Accuracy for the third dataset (iii).

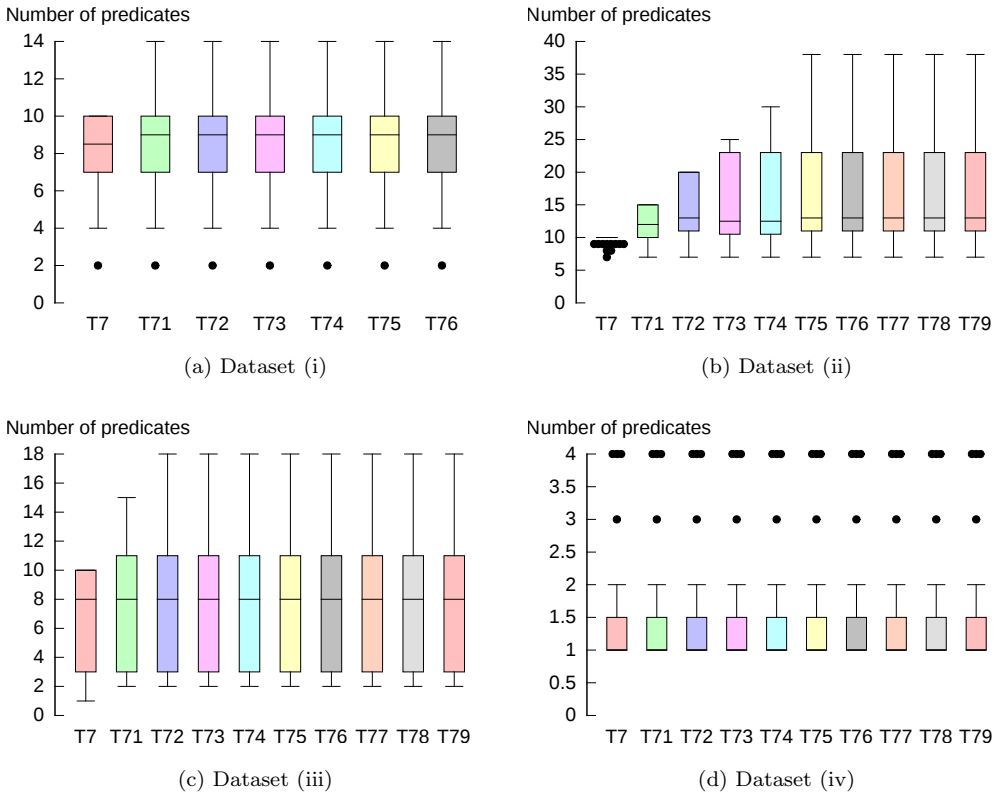


(a) Type 71 (b) Type 72 (c) Type 73 (d) Type 74



(e) Type 75 (f) Type 77 (g) Type 78 (h) Type 79

Figure 26: Accuracy for the fourth dataset (iv).



(a) Dataset (i)

(b) Dataset (ii)

(c) Dataset (iii)

(d) Dataset (iv)

Figure 27: Number of predicates of the redescriptions for the four datasets and for all Types defined for the new set of experiments.

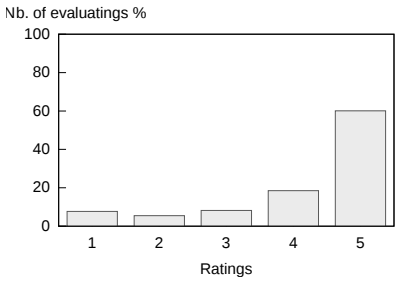


Figure 28: Ratings distribution.

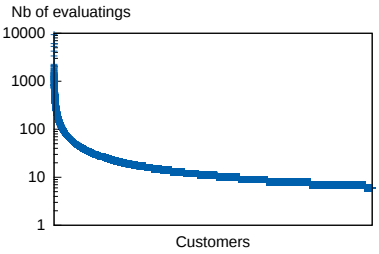


Figure 29: Number of evaluatings by customers.

Algorithm 1 Algorithm to handle hierarchy.

Input: The hierarchy file *hierarchy.xml*, the transaction file *transactions.data*, the thresholds *maxdist*, *mindist*, *maxsupp*, *minsupp*, *minsim*

Output: A set of redescription, R

$R \leftarrow \emptyset$
$I \leftarrow \emptyset$

Step 1 :
for all $firstNode$ in hierarchy.xml **do**
  Check constraints (4) and (5)
  if Constraints (4) and (5) hold for $firstNode$ **then**
    for all $secondNode$ in hierarchy.xml **do**
      Redescription $r = (q_L, q_R) = (firstNode, secondNode)$
      if All constraints hold for r **then**
        if I contains less than k elements **then**
          $I \leftarrow I \cup r$
        else
          if $sim(r) \geq sim(r')$ with $r' \in I$ **then**
            Remove the lower redescription from I
            $I \leftarrow I \cup r$
          end if
        end if
      end if
    end for
  end if
end for

Step 2 :
for all Redescription r in I **do**
  $T \leftarrow \{r\}$
  while $T \neq \emptyset$ **do**
    $r' = (q_L, q_R)$ received the better redescription from T
    $T \leftarrow \emptyset$
    for all $aNode$ in hierarchy.xml **do**
      $r_{left} = (q_L \cup aNode, q_R)$
      if All constraints hold for r_{left} and $sim(r_{left}) > sim(r)$ **then**
        $T \leftarrow T \cup r_{left}$
      end if
      $r_{right} = (q_L, q_R \cup aNode)$
      if All constraints hold for r_{right} and $sim(r_{right}) > sim(r)$ **then**
        $T \leftarrow T \cup r_{right}$
      end if
    end for
  end while
  $r' = (q_L, q_R)$ received the better redescription from T
  $R \leftarrow R \cup r'$
end for