

DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS A
REPORT A-2013-11

Methods for Redescription Mining

Esther Galbrun

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XV, University Main Building, on 4 December 2013, at twelve o'clock noon.

UNIVERSITY OF HELSINKI
FINLAND

Supervisors

Professor Hannu Toivonen, University of Helsinki, Finland

Assistant Professor Mikko Koivisto, University of Helsinki, Finland

Pre-examiners

Professor Bruno Crémilleux, University of Caen, France

Professor Naren Ramakrishnan, Virginia Tech, U.S.A.

Opponent

Professor Nada Lavrač, Jožef Stefan Institute, Slovenia

Custos

Professor Hannu Toivonen, University of Helsinki, Finland

Contact information

Department of Computer Science

P.O. Box 68 (Gustaf Hällströmin katu 2b)

FI-00014 University of Helsinki

Finland

Email address: postmaster@cs.helsinki.fi

URL: <http://www.cs.helsinki.fi/>

Telephone: +358 9 1911, telefax: +358 9 191 51120

Copyright © 2013 Esther Galbrun

ISSN 1238-8645

ISBN 978-952-10-9430-9 (paperback)

ISBN 978-952-10-9431-6 (PDF)

Computing Reviews (1998) Classification: G.2.2, G.2.3, H.2.8, I.2.6

Helsinki 2013

Unigraphia

Methods for Redescription Mining

Esther Galbrun

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
esther.galbrun@cs.helsinki.fi
<http://www.cs.helsinki.fi/people/galbrun/>

PhD Thesis, Series of Publications A, Report A-2013-11
Helsinki, November 2013, 72+77 pages
ISSN 1238-8645
ISBN 978-952-10-9430-9 (paperback)
ISBN 978-952-10-9431-6 (PDF)

Abstract

In scientific investigations data oftentimes have different nature. For instance, they might originate from distinct sources or be cast over separate terminologies. In order to gain insight into the phenomenon of interest, a natural task is to identify the correspondences that exist between these different aspects.

This is the motivating idea of *redescription mining*, the data analysis task studied in this thesis. Redescription mining aims to find distinct common characterizations of the same objects and, vice versa, to identify sets of objects that admit multiple shared descriptions.

A practical example in biology consists in finding geographical areas that admit two characterizations, one in terms of their climatic profile and one in terms of the occupying species. Discovering such redescriptions can contribute to better our understanding of the influence of climate over species distribution. Besides biology, applications of redescription mining can be envisaged in medicine or sociology, among other fields.

Previously, redescription mining was restricted to propositional queries over Boolean attributes. However, many conditions, like aforementioned climate, cannot be expressed naturally in this limited formalism. In this thesis, we consider more general query languages and propose algorithms

to find the corresponding redescription, making the task relevant to a broader range of domains and problems.

Specifically, we start by extending redescription mining to non-Boolean attributes. In other words, we propose an algorithm to handle nominal and real-valued attributes natively. We then extend redescription mining to the relational setting, where the aim is to find corresponding connection patterns that relate almost the same object tuples in a network.

We also study approaches for selecting high quality redescriptions to be output by the mining process. The first approach relies on an interface for mining and visualizing redescriptions interactively and allows the analyst to tailor the selection of results to meet his needs. The second approach, rooted in information theory, is a compression-based method for mining small sets of associations from two-view datasets.

In summary, we take redescription mining outside the Boolean world and show its potential as a powerful exploratory method relevant in a broad range of domains.

Computing Reviews (1998) Categories and Subject Descriptors:

G.2.2 Discrete Mathematics; Graph Theory; Graph Algorithms

G.2.3 Discrete Mathematics; Applications

H.2.8 Information Systems; Database Management; Database Applications; Data mining

I.2.6 Artificial Intelligence; Problem Solving, Control Methods, and Search; Heuristic Methods

General Terms:

Algorithms, Experimentation

Additional Key Words and Phrases:

Redescription Mining, Numerical Data, Relational Query Mining, Interactive Data Mining, Pattern Set Mining

Acknowledgements

Original Publications of the Thesis

This thesis is based on the following peer-reviewed publications, which are referred to as Articles I–IV in the text. They are reproduced at the end of the thesis. The articles do not constitute the basis of any other doctoral dissertation. The author’s contributions are described in Section 1.1.

- I. Esther Galbrun and Pauli Miettinen
**From Black and White to Full Color:
Extending Redescription Mining Outside the Boolean World**
In *Statistical Analysis and Data Mining*, 5(4):284–303, 2012.
DOI: <http://dx.doi.org/10.1002/sam.11145>
- II. Esther Galbrun and Pauli Miettinen
**A Case of Visual and Interactive Data Analysis:
Geospatial Redescription Mining¹**
In *Instant Interactive Data Mining Workshop*
at the *2012 European Conference on Machine Learning and Principles
and Practice of Knowledge Discovery in Databases, ECML/PKDD’12*
(Bristol, UK), 2012.
- III. Esther Galbrun and Angelika Kimmig
Finding Relational Redescriptions
In *Machine Learning*, Published Online First, 2013.
DOI: <http://dx.doi.org/10.1007/s10994-013-5402-3>
- IV. Matthijs van Leeuwen and Esther Galbrun
Compression-based Association Discovery in Two-View Data
Submitted for review.

¹Extended version of *Siren: An Interactive Tool for Mining and Visualizing Geospatial Redescriptions*, In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD’12* (Beijing, China), pages 1544–1547. ACM, 2012.

Contents

1	Introduction	1
1.1	Outline of the Contributions	2
2	Preliminaries	5
2.1	Problem Definition	6
2.2	Related Work	10
3	Query Languages	13
3.1	Propositional Queries	13
3.1.1	Predicates	15
3.1.2	Statements	16
3.2	Relational Queries	19
3.2.1	Predicates	20
3.2.2	Statements	23
4	Exploration Strategies	27
4.1	Query Mining and Pairing	28
4.2	Alternating Scheme	29
4.3	Greedy Atomic Updates	30
5	Pattern Selection	33
5.1	Individual Patterns	33
5.1.1	Quality Criteria	33
5.1.2	Constraint-based Mining	37
5.1.3	Interactive Data Mining	38
5.2	Sets of Patterns	39
5.2.1	Compression-based Model Selection	40
5.2.2	Subjective Interestingness	41

6	Illustrated Discussion	43
6.1	Overview of the Algorithms	43
6.2	<i>Computer Science Bibliography</i>	45
6.3	<i>Bioclimatic Niches</i>	48
6.4	<i>Political Candidates Profiles</i>	51
6.5	<i>Biomedical Ontology</i>	55
7	Conclusions	59
	References	61
	Articles	73

Chapter 1

Introduction

The present thesis is concerned with *redescription mining*. Roughly speaking, this data analysis task aims to find different ways of characterizing the same things and, vice versa, to find things that admit the same alternative characterizations.

As a practical example, consider the European regions of Scandinavia and Baltia. They share similar temperature and precipitation conditions and are both inhabited by the European Elk. Hence, this set of geographical areas admits two characterizations, one in terms of their climatic profile and one in terms of the occupying species.

The aim of data analysis in general is to gain useful knowledge from data, that is, to turn large amounts of data into actionable information. It is widely recognized that our understanding of a concept can be improved by considering it from different vantage points. To be more prosaic, several experiments might be carried out to study a phenomenon or, more generally, data might be available from different sources, cast in various terminologies or possess various semantics. This results in a group of datasets characterizing the same objects, known as a multi-view dataset. Then, it is of natural interest to relate and exploit these different aspects so as to better understand the concepts or phenomena at hand. This is the idea behind redescription mining.

Continuing with the example above, the data describes two different aspects of geospatial regions of Europe: their climate and their fauna. Characterizing the areas inhabited by a (set of) species in terms of the climate encountered, and the other way around, provides valuable information about the effects of climate on the species distribution. Finding such characterizations is actually an important problem in biology, known as bioclimatic niche finding [SN09, Gri17]. In this case, by providing an automated alternative to the tedious process of manually selecting species

and fitting a climatic model, redescription mining allows to explore many more combinations of conditions. Applications of redescription mining can be envisaged in a broad range of domains, including for instance social sciences and medicine.

This thesis consists of four original publications (Articles I–IV) and this introductory part. The purpose of the introduction is not to repeat the original publications. Rather, it aims to place the articles in their common context, articulate the issues addressed, and highlight the underlying transverse principles. In particular, the reader is referred to the original publications for careful review of related work, details regarding the algorithms and thorough experimental evaluation.

After providing an outline of the contributions of the original publications in the next section, we introduce the problem of mining redescriptions more thoroughly in Chapter 2. Then, in Chapters 3–5, we focus on three facets of the problem, respectively query languages, exploration strategies and pattern selection techniques. Based on these, we sketch the algorithms that constitute the main contributions of this thesis, as an opening to Chapter 6. We then proceed to illustrate the task with examples from different fields. We present results obtained with our proposed algorithms on various datasets, to illustrate the use of redescription mining in different domains and to serve as a basis for a critical discussion of the approach, before reaching conclusions in Chapter 7.

1.1 Outline of the Contributions

The contributions in this thesis are presented in the original Articles I–IV and can be summarized as follows.

- I. Previously, redescription mining could handle only Boolean data, making discretization a prerequisite to using the existing tools. We extended redescription mining to categorical and real-valued attributes with a greedy algorithm that determines on-the-fly the category or interval yielding the best accuracy.
- II. Building on the algorithm presented in Article I, we developed an interface for mining redescriptions from geospatial data, called SIREN. We discussed desirable features of visual and interactive data analysis tools, focusing on the case of geospatial redescription mining, exemplified by SIREN.
- III. We introduced relational redescription mining, lifting the problem to the first order level and thereby making the approach relevant

to a new range of applications. We mine for structurally different connection patterns that describe the same object pairs in a relational dataset.

- IV. Combining redescription mining with techniques from association rule mining and compression-based model selection, we present a novel method to find a small set of associations that explains how the two sides of Boolean two-view datasets are related.

The contribution of the author to all of the original publications was substantial.

Initially, Dr Pauli Miettinen suggested adapting his GREEDY redescription mining algorithm to handle real-valued variables. I implemented the algorithm, largely rewriting and extending the existing code, and performed the experiments. We participated equally in writing Article I.

Later, after repeated prompting from Dr Miettinen, I set out to implement a graphical user interface for our algorithm, the matter of Article II, which we co-wrote.

Of Article III, I am the main contributor. Dr Angelika Kimmig carried out the comparison experiments with the baseline tool and edited the paper, in particular, acting as an interpreter from the field of inductive logic programming.

The collaboration on Article IV was suggested by Dr Matthijs van Leeuwen, to combine ideas from redescription mining and from his field of expertise, exceptional model mining. The problem formalization, algorithm design and article writing was done jointly. My role was very minor in implementing the algorithm. I was responsible for running most experiments.

Chapter 2

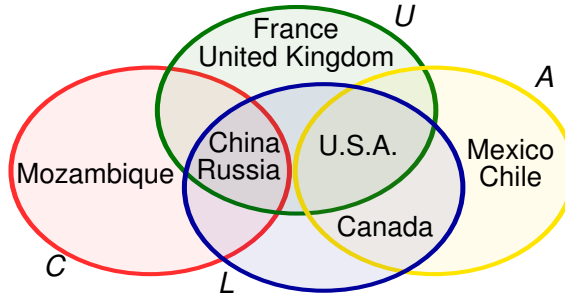
Preliminaries

In the field of data analysis, especially when considering vast amounts of data, in a process commonly called knowledge discovery in databases, *mining* usually refers to the task of extracting regularities, or patterns [HK00]. Specifically, novel, useful and understandable patterns are sought after [HMS01]. Faced with a potentially large set of factual data, obtained directly or derived from observations, for instance as the result of scientific experiments involving sensor measurements or censuses, the hope is that the analyst will be able to grasp the underlying reality by identifying recurrent patterns.

In particular, the purpose of redescription mining is to find alternative characterizations of almost the same objects. Such an approach allows to shed light on the concepts present in the dataset by identifying coherent sets of objects and related properties.

An instance of this task in the medical field could be, for example, to relate patients' background to their symptoms and to their diagnosis, so as to improve the understanding of illnesses. Revealing patterns that connect temperature and precipitation statistics to the fauna distribution constitutes another instance of the redescription mining task. Such discovery can contribute to our appreciation of the impact of climatic constraints on the habitat of these species. As mentioned previously, this pertains to the problem of ecological niche finding [SN09, Gri17].

In the relational setting, more specifically, redescription mining aims to identify correspondences between complex connection patterns, beyond the mapping of individual properties commonly considered in ontology matching and schema alignment [SE05, SAS11]. It is potentially useful for the exploration and maintenance of the massive amounts of structured information stored in knowledge bases [ABK⁺07, CBK⁺10, SKW07, RLT⁺12].



- | | | | |
|---|---|---|--|
| A | Located in the Americas | C | History of state communism |
| L | Land area above 8 Billion km ² | U | Permanent member
of the UN security council |

Figure 2.1: Example dataset. Geographic and geopolitical characteristics of countries represented as a Venn diagram. Adapted from [PR05].

Example 1. *Throughout this introduction, we use a running example to illustrate our discussion, with variations and refinements according to the successive points of focus. Adapting the prototypical example that appeared in the earliest redescription mining publications [RKM⁺04, PR05, ZR05], we consider a set of nine countries as our objects of interest, namely Canada, Chile, China, France, Mexico, Mozambique, Russia, the United Kingdom and the United States of America.*

Following Parida and Ramakrishnan [PR05], let us start with a simple toy dataset consisting of four properties characterizing these countries, represented as a Venn diagram in Figure 2.1. Consider the pair of statements below:

- *Country outside the Americas with land area above 8 billion square kilometers.*
- *Permanent member of the UN security council with a history of state communism.*

Both statements are satisfied by the same countries, namely China and Russia. They constitute alternative characterizations of the same subset of countries in terms of geographic and geopolitical properties, respectively. Hence, they form a redescription.

We now provide a formal albeit very general definition of redescrptions and the redescription mining task.

2.1 Problem Definition

Let \mathcal{O} be a set of elementary *objects* and \mathcal{A} a set of *attributes* characterizing properties of the objects or of relations between them. The attributes arise

from different sources, terminologies, etc., represented as a set of *views* V . A function v maps each attribute to the view to which it belongs, $v : \mathcal{A} \rightarrow V$. The dataset is fully specified by the triplet $(\mathcal{O}, \mathcal{A}, v)$.

A logical statement expressed over attributes in \mathcal{A} and evaluated against the dataset forms a *query*. A query language Q is a set of valid queries. To evaluate a statement against a dataset is to replace the variables in the statement by objects from the dataset and identify the substitutions for which the ground formula holds true. This subset of objects or of non-empty tuples of objects is called the *support* of query q and denoted as $\text{supp}(q)$. The set of *entities*, denoted as E , consists of all the possible substitutions for queries in Q . The set of attributes appearing in query q is denoted by $\text{att}(q)$ and we overload the function v to also denote the union of their views, $v(q) = \bigcup_{A \in \text{att}(q)} v(A)$. As a straightforward means to ensure that two queries provide different characterizations, their attribute sets are required to be disjoint. We consider a symmetric binary relation \sim over the power set of entities as a Boolean indicator of support similarity. Finally, we denote by \mathcal{C} a set of arbitrary constraints that can be used to specify a bias towards certain queries.

In this context, we define a redescription as follows.

Definition 1. *Given a dataset $(\mathcal{O}, \mathcal{A}, v)$, a query language Q over \mathcal{A} and a binary relation \sim , a **redescription** is a pair of queries $(q_A, q_B) \in Q \times Q$ such that $v(q_A) \cap v(q_B) = \emptyset$ and $\text{supp}(q_A) \sim \text{supp}(q_B)$.*

And redescription mining is simply the task of finding such pairs.

Problem 1 (Redescription Mining). *Given a dataset $(\mathcal{O}, \mathcal{A}, v)$, a query language Q over \mathcal{A} , a binary relation \sim , and a set \mathcal{C} of constraints, find all redescriptions that satisfy constraints in \mathcal{C} .*

Example 2. *Let us consider the example above in light of this terminology. The objects are nine countries, and the four attributes can be divided into two views, say \mathbf{G} and \mathbf{P} , corresponding to the domains of geography, i.e. attributes A and L , and geopolitics, i.e. attributes C and U , respectively. That is, we have that $\mathcal{A} = \{A, C, L, U\}$ and, for instance, $v(A) = \mathbf{G}$. The first statement forms a query over geographic attributes, which could be equivalently written as $q_{\mathbf{G}} = \neg A \wedge L$. When evaluated on the dataset, it is satisfied by two countries. Specifically, we have $\text{supp}(q_{\mathbf{G}}) = \{\text{China}, \text{Russia}\}$. The second query, over geopolitical attributes, $q_{\mathbf{P}} = U \wedge C$, has the same support. Thus, for any reasonable choice of similarity relation \sim we have that $\text{supp}(q_{\mathbf{G}}) \sim \text{supp}(q_{\mathbf{P}})$. Since in addition $v(q_{\mathbf{G}}) \cap v(q_{\mathbf{P}}) = \{\mathbf{G}\} \cap \{\mathbf{P}\} = \emptyset$, then $(q_{\mathbf{G}}, q_{\mathbf{P}})$ is a redescription.*

Here, we roughly consider data analysis methods with a focus on description to belong to data mining, while those methods with a focus on prediction are categorized into machine learning. In other words, the former area consists of techniques that aim at detecting regularities in the data, i.e. patterns, emphasizing the interpretability of the results, while the latter comprises techniques that aim to predict some properties or relations of unseen objects given a subset of observed ones. However, patterns resulting from data mining can constitute the building blocks of learning systems by providing higher level features, while machine learning tasks such as classification or regression can be found at the core of pattern discovery systems. The line between the two areas is easily blurred.

As its name suggests, redescription mining is a descriptive data analysis problem, a data mining task. Because the aim is not to learn a model to predict unseen data, but rather, to describe the data at hand as well as possible, the expressivity and interpretability of the results are particularly emphasized.

In our context, expressivity and interpretability are to be understood in the following acceptations: The variety of concepts that a language can represent determines its expressive power, or expressivity, while the interpretability of an element of the language relates to the ease with which the associated meaning can be apprehended. Interpretability is difficult to measure but is promoted by favoring concise, simple queries. As a consequence of this accent put on the descriptive aspect, certain query languages are more adequate for the task than others. In particular, throughout our work we adopt the following position with respect to query languages. Our preference goes to Boolean formulae specifying explicit constraints on the range of individual attributes. Linear functions defined over the attributes, on the other hand, are deemed to have limited interpretability and hence to be unsuitable for our purpose.

In any given instance of the redescription mining task, we consider a collection \mathcal{O} of elementary objects, sometimes also referred to as samples. The dataset consists of attributes in \mathcal{A} characterizing the properties of these objects and possibly of the relations linking them, as well. That is, we consider both propositional and relational datasets. These two settings are formalized and discussed in more depth together with the associated query languages in Chapter 3.

The set of views V represents the various sources, domains or terminologies from which the data originate. For instance, the attributes of our toy dataset above can be naturally split between geography (A and L) and geopolitics (C and U), while additional attributes could stem from the eco-

nomic, social or cultural domains. Climatic conditions on one hand and fauna on the other hand form two clearly distinct sets of attributes in biological niche finding, our example in the field of biology, while in the field of medicine, the objects of the study might be a set of patients with their personal background information, symptoms and elements of diagnosis, as three different views, for instance.

The purpose of redescription mining is to find *alternative characterizations* of almost the same objects, so as to relate concepts across different sources, domains or terminologies. Therefore, we require that the attributes over which both queries of a redescription are expressed come from disjoint sets of views, i.e. $v(q_A) \cap v(q_B) = \emptyset$, and say that such queries are *structurally different*. If two statements are logically equivalent, then the associated queries trivially have the same support. Uninteresting tautologies of this kind are ruled out by this requirement.

Conceptually, the number of views can be arbitrarily large. Yet, two settings are generally preferred. At one extreme, each individual attribute can be seen to form a separate view. This corresponds to the case where all attributes are gathered into a single dataset and the restriction that none of them appears in both queries simultaneously. At the other extreme, we might consider the attributes to be naturally split into two subsets that each constitutes a view.

From now on, we will focus primarily on the case where the attributes are split between two views, arbitrarily denoted as \mathbf{L} (for left-hand side) and \mathbf{R} (for right-hand side). That is, we have two subsets of attributes,

$$\mathcal{A}_{\mathbf{L}} = \{A \in \mathcal{A}, v(A) = \mathbf{L}\} \text{ and } \mathcal{A}_{\mathbf{R}} = \{A \in \mathcal{A}, v(A) = \mathbf{R}\},$$

such that $\mathcal{A}_{\mathbf{L}} \cup \mathcal{A}_{\mathbf{R}} = \mathcal{A}$ and, trivially, $\mathcal{A}_{\mathbf{L}} \cap \mathcal{A}_{\mathbf{R}} = \emptyset$. Then, a pair of structurally different queries simply consists of queries $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$ expressed over $\mathcal{A}_{\mathbf{L}}$ and $\mathcal{A}_{\mathbf{R}}$, respectively. Still, discussions on this setting, known as two-view or two-fold setting, extend naturally to other settings as well.

In the presence of multiple views, the correspondence between the elementary objects across the views might not be available. It might be that the sets of objects occurring in different views are not identical, that some objects are associated with redundant observations in one view, or that a single object from one view corresponds to multiple objects in another view, as is the case for instance of geospatial measurements with varying scales. Establishing the mapping between the views can be nontrivial and constitutes a problem of its own [TKOK11], ignored here. We assume that the input data consist of aligned views, i.e. that the bijection of objects across the views is known. Moreover, except for missing values which we

consider in Article I, dealing with measurement errors, sampling bias and similar problems affecting the data quality is outside the scope of our work.

The purpose of redescription mining is to find alternative characterizations of *almost the same objects*. Thus, the similarity of the supports of the queries is a defining quality of a redescription. We say that a pair of queries is *accurate* if their supports are similar. In general, the similarity relation \sim between support sets is specified as a set similarity function f together with a threshold σ , such that $E_a \sim E_b \iff f(E_a, E_b) \geq \sigma$. Specifically, the Jaccard coefficient is a common choice for f , as we mention in Section 5.1.1.

Nevertheless, structural difference and accuracy are typically not sufficient to guarantee the interestingness of the result. Other criteria also impact its quality. The set of constraints on the queries and their supports, \mathcal{C} , allows to specify a bias towards certain redescriptions, in particular to include background knowledge. Furthermore, we are generally looking for a set of redescriptions, the interestingness of which we want to evaluate not just individually but as a whole. Interestingness and the selection of high quality redescriptions constitute the focus of Chapter 5.

2.2 Related Work

Redescription mining is a multi-view data mining technique in that it exploits multiple views on the objects to identify interesting patterns. Similar ideas motivate multi-view learning approaches. Pioneered by Yarowsky [Yar95] in the context of word sense disambiguation, and by Blum and Mitchell [BM98] under the term co-training, the principle of exploiting distinct views to strengthen learning algorithms has attracted increasing interest. It has been applied to clustering [NG00, BS04, BS05], support vector machines [FHM⁺05], canonical correlation analysis [KK08] or factor analysis [VKKK12], among others. Arguably, redescription mining can be seen as the data mining pendant of multi-view techniques in machine learning.

Other data mining problems such as emerging patterns [NLW09], subgroup discovery [UZT⁺09] or exceptional model mining [LFK08] can also be seen as multi-view approaches, although they are rarely presented from that perspective. Indeed, the common aim of these techniques is to find queries over one view defining a set of objects with an uneven distribution in the other view when compared to the remaining objects. That is, these approaches consider one view as description and the other as target. In this

context, the symmetricity of its approach, in other words, the fact that the views are treated equally, is a distinguishing feature of redescription mining.

The relational setting is tightly related to inductive logic programming concepts in general [DR08, MDR94, QCJ93, Mug95, Sri07] and multi-relational query mining in particular [DT99, DRR04], from which terminology and notation are borrowed. The work presented here draws on techniques from various other areas of data analysis, such as, in particular, graph mining, constraint-based mining or model selection. References are provided in the relevant chapters and the original articles.

Redescription mining was introduced by Ramakrishnan et al. [RKM⁺04] in 2004. They proposed to find set theoretic expressions over indicator functions that define similar subsets of elements. There, indicator functions and set theoretic expressions are used as representation, equivalently to Boolean attributes and logical statements.

Several conceptually related problems are discussed in [PR05], including *story telling* [RKM⁺04], the task of finding a succession of approximate redescriptions such that the first and last are supported by two distinct given sets of entities. Another kindred task, finding *straddling-biclusters* [JMR08], combines mining redescriptions and biclusters. Finally, *query by output* is a closely related problem in the area of database systems [TCP09]. It aims at finding an instance-equivalent query for a given input query, or in the words of redescription mining, completing a query pair to form an accurate redescription. Considering SQL queries, it requires to determine not only good selection predicates but also relevant relations and projections.

The approaches proposed for redescription mining have been based on various ideas, including decision trees [RKM⁺04, Kum07], Karnaugh maps [ZR05], co-clusters [PR05], frequent itemsets [ZR05, GMM08] and greedy search [GMM08]. However, they all focused on Boolean propositional attributes, restricting the applicability of the method. Indeed, as a consequence, discretization, binarization or propositionalization were required preliminaries in order to mine redescriptions from datasets containing non-Boolean or relational attributes. Such preprocessing procedures typically require extensive domain knowledge, entail an information loss that can impact the subsequent analysis and greatly inflate the search space.

To address this issue, we extended the GREEDY algorithm [GMM08] with efficient on-the-fly discretization in its innermost loop, allowing to handle nominal and real-valued attributes, as reported in Article I. Next, we designed an algorithm to mine redescriptions from network data, presented

in Article III, using an alternating scheme similar to that presented by Ramakrishnan et al. [RKM⁺04].

Main strategies for mining redescription are discussed in Chapter 4.

To summarize, redescription mining automatically identifies both a pair of queries and an associated set of objects, neither of which needs to be specified in advance. The results of redescription mining offer a dual perspective. On one hand, the queries of a redescription contain attributes that are related to each other, as they can be used to characterize the same set of objects. On the other hand, the support of the redescription defines a particularly coherent set of objects, as it admits alternative characterizations.

So far we presented the problem of redescription mining from a conceptual point of view, deliberately keeping the discussion at a fairly high abstraction level. We will now focus on some more practical aspects, in other words, on choices that need to be made when implementing the general principle.

The three following chapters are organized around three primary facets of the problem that can be considered independently from each other, to some extent. First, we discuss query languages in Chapter 3, where possible choices for Q are defined formally. Second, we sketch strategies to explore the space of query pairs in search for redescrptions, in Chapter 4. Third, in Chapter 5, we look at what constraints \mathcal{C} can be used to characterize good redescrptions and study pattern selection techniques more generally.

Chapter 3

Query Languages

In this chapter, we explore variations on a theme and are concerned with choices of query languages. More precisely, we define the queries that are used in redescription mining, offering means to represent logical combinations of constraints on the range of individual attributes.

The formalism presented here corresponds to that previously used in Boolean propositional redescription mining [PR05, GMM08], which we adapted to support other types of attributes and later extended to the relational case, respectively in Article I and Article III.

Queries consist of logical statements evaluated against the dataset. The statements are obtained by combining atomic predicates built from individual attributes using Boolean operators. Replacing predicate variables by objects from the dataset and verifying whether the conditions of the predicates are satisfied returns a truth value. The objects or object tuples in substitutions satisfying the statement constitute the support of the query.

A query language is a set of acceptable queries, dependent on the supported types of attributes, the principles for building predicates and the syntactic rules for combining them into statements. We discuss, in turn, propositional and relational queries.

3.1 Propositional Queries

A *propositional dataset* consists of attributes characterizing properties of individual objects. We generally consider a homogeneous set of objects, in the sense that each attribute applies to all objects, regardless of possible missing values. In that context, the values taken by the attributes in \mathcal{A} for each of the objects are collected into a matrix D with $|\mathcal{O}|$ rows, one per

Table 3.1: Example dataset. Geographic attributes of world countries: localization in the southern hemisphere (1), existence of oceanic borders (2–4), continental location (5), land area (6) and maximum elevation (7).

Country	1) South Hemisphere	2) Atlantic Ocean	3) Indian Ocean	4) Pacific Ocean	5) Continent	6) Area (10 ⁹ km ²)	7) Elev. (m)
CA	false	true	false	true	{North America}	9.98	5959
CL	true	true	false	true	{South America}	0.76	6893
CN	false	false	false	true	{Asia}	9.71	8850
FR	false	true	false	false	{Europe}	0.64	4810
GB	false	true	false	false	{Europe}	0.24	1343
MX	false	true	false	true	{North America}	1.96	5636
MZ	true	false	true	false	{Africa}	0.79	2436
RU	false	false	false	true	{Asia, Europe}	17.10	5642
US	false	true	false	true	{North America}	9.63	6194

object, and $|\mathcal{A}|$ columns, one per attribute. In other words, $D(i, j) = A_j(o_i)$ is the value of attribute $A_j \in \mathcal{A}$ for object $o_i \in \mathcal{O}$.

As a special case, we say that the dataset is geospatial when the objects are associated to spatial coordinates, that is, when they can be located in a spatial reference system. Then, the support of the resulting geospatial queries and geospatial redescrptions over that dataset can be naturally represented on a map.

Example 3. *Refining Example 2, consider the dataset shown in Table 3.1. Again, the set of objects is a subset of world countries. Each of the seven attributes corresponds to a geographic property: localization in the southern hemisphere (1), existence of a border to the Atlantic Ocean (2), to the Indian Ocean (3), or to the Pacific Ocean (4), continental location (5), land area in billion squared kilometers (6) and maximum elevation in meters (7), referring to the mainland area only. This data can be represented as matrix G with ten rows and seven columns. Furthermore, we can identify attributes with the columns of this matrix. For instance, the first attribute, localization in the southern hemisphere, is denoted as the corresponding vector G_1 .*

This constitutes a simple example of a geospatial dataset, since the objects, i.e. the countries, can be associated to spatial coordinates such as latitude and longitude (omitted here).

Predicates are constructed over individual attributes and combined into statements to form the queries.

3.1.1 Predicates

The values that an attribute can take constitute its range. A predicate is constructed from an attribute by restricting the values to a selected subset of its range. Consider an attribute $A_j \in \mathcal{A}$ with range R . By fixing a subset $R_S \subseteq R$ we can turn the associated data column into a truth value assignment, that is, into a Boolean vector indicating which values are within the specified range. Using Iverson bracket, this is denoted as $[A_j \in R_S]$. In effect, this selects the subset of objects for which attribute A_j takes value in R_S , $s(A_j, R_S) = \{o_i \in \mathcal{O}, A_j(o_i) \in R_S\}$ and $[A_j \in R_S]$ is an indicator of membership in this subset.

Depending on their range, object attributes can be classified into types. In turn, we consider three types of attributes, Boolean, nominal and real-valued and the predicates constructed from them.

Boolean predicates. Boolean attributes take value either **false** or **true**. Equivalently, their range is $\{0, 1\}$. A Boolean attribute can be interpreted as a truth value assignment for the objects in a natural way and thus directly yields a predicate. For Boolean attributes we omit the bracket notation and simply denote the Boolean predicate $[A = \mathbf{true}]$ by A . Typically, we do not consider the complementary assignment $[A = \mathbf{false}]$ as it can be equivalently obtained with negation.

In our example, G_4 is a Boolean attribute that corresponds to a predicate with the following truth assignment on this dataset:

$$\langle 1, 1, 1, 0, 0, 1, 0, 1, 1 \rangle ,$$

selecting the six countries bordering the Pacific Ocean.

Nominal predicates. An attribute A whose range is an unordered set C , or its powerset, is called a nominal (or categorical) attribute, single-valued or multi-valued respectively. The elements of C are called the categories of attribute A . A truth value assignment is obtained by choosing a subset of the categories $C_S \subseteq C$ or a single category $c \in C$, to select objects that belong to these categories. The corresponding nominal predicates are denoted as $[A \in C_S]$ and $[A = c]$, respectively.

The first attribute from our example, the continental location, is a nominal attribute. Four countries located in the Americas satisfy the predicate

$$[G_5 \in \{\text{North America}, \text{South America}\}] ,$$

corresponding to the truth assignment $\langle 1, 1, 0, 0, 0, 1, 0, 0, 1 \rangle$. In this case, the attribute is multi-valued because objects can be associated to multiple categories: Russia spans over both Europe and Asia.

Practically, only single-valued nominal attributes and predicates with an individual category are considered. Multi-valued nominal attributes are equivalently represented with several Boolean attributes, one per category.

Real-valued predicates. An attribute A whose range is a subset of the real numbers $R \subseteq \mathbb{R}$ is a real-valued attribute. A truth value assignment for the objects can be obtained by choosing any subset of R . However, for interpretability reasons, it is typically constructed based on a single contiguous subset of R , i.e. an interval $[a, b] \subseteq R$, and denoted as $[a \leq A \leq b]$.

Notice that for a given real-valued attribute, there can be an infinity of intervals yielding the same truth-value assignment. Consider for example the sixth attribute, the land area, corresponding to the following vector

$$G_6 = \langle 9.98, 0.76, 9.71, 0.64, 0.24, 1.96, 0.79, 17.10, 9.63 \rangle .$$

Any pair (λ, ρ) where $\lambda \in]0.79, 1.96[$ and $\rho \in]9.98, 17.10[$ will yield the same truth value assignment

$$[\lambda \leq G_6 \leq \rho] = \langle 1, 0, 1, 0, 0, 1, 0, 0, 1 \rangle .$$

Hence, the definition of a consistent query language must include a criterion for selecting one among these equivalent intervals. Yet, it is disputable whether $[1 \leq G_6 \leq 10]$, because it has rounded bounds, is a better choice than tighter $[1.96 \leq G_6 \leq 9.98]$, and similarly, whether $[G_6 \leq 10]$ should be preferred over equivalent $[0.24 \leq G_6 \leq 10]$, for instance. Indeed, in both cases the two intervals correspond to the same truth assignment and favoring one over the other depends, in particular, on whether rounded bounds are considered more interpretable than tight intervals, or vice versa.

3.1.2 Statements

Predicates make up the building blocks of statements. Propositional predicates can be combined using Boolean operators, negation (\neg), conjunction (\wedge) and disjunction (\vee). The truth assignment for the associated query is obtained by combining the truth assignment of the individual predicates accordingly. Equivalently, the individual truth assignments define subsets of objects that can be combined by means of the corresponding set operators, complement (\setminus), intersection (\cap) and union (\cup). The resulting subset of objects is the support of the query.

For instance, in the context of Example 3, the query

$$q_1 = G_4 \wedge \neg G_1 \wedge [6000 \leq G_7]$$

describes countries bordering the Pacific outside the southern hemisphere where the highest peak reaches over 6000 meters. Two countries, China and the U.S.A., satisfy these conditions, that is, support this query.

In the propositional case, the possible substitutions for the queries, called the entities, are simply individual objects, $E = \mathcal{O}$.

Monotone conjunctions. Monotone conjunctions are the most restricted query language, where predicates are combined using only conjunction operators. For example, the following query is a monotone conjunction

$$q_2 = G_3 \wedge G_2 \wedge [1000 \leq G_7 \leq 2000] ,$$

while q_1 above does not belong to this language since it is not monotone.

Conjunctive monotone queries directly correspond to itemsets where each predicate is an item. Itemsets have been extensively studied in the literature, especially to design efficient frequency based mining algorithms [HCXY07, Goe03]. In particular, they are easily arranged in a partial order based on inclusion and verify the downward closedness property. That is, if query q_i is a subset of query q_j , then the support of q_i is a superset of the support of q_j . As a consequence, the search space of this query language can be explored efficiently in a level-wise fashion.

The language used in Article IV consists of monotone conjunctive queries over Boolean predicates, which, being the most restricted also affords efficient exhaustive search.

Monotone conjunctions are at the lower end of the scope of propositional queries, at the same time easy to interpret and to find. But excluding negations and disjunctions severely limits the expressivity.

Unrestricted queries. At the other end of the scope are unrestricted propositional queries, in which predicates can be combined using any of the three operators with no other limits than the usual rules of algebra. For example

$$\begin{aligned} q_3 &= (G_3 \vee G_2) \wedge \neg G_1 , \\ q_4 &= (\neg G_2 \wedge [G_5 = \text{Asia}]) \vee ([5000 \leq G_7] \wedge G_4) , \\ q_5 &= (G_1 \wedge \neg [4000 \leq G_7 \leq 6000]) \vee [3000 \leq G_7] , \text{ and} \\ q_6 &= (\neg (G_3 \vee ([G_5 = \text{Europe}] \wedge G_2))) \wedge [0.3 \leq G_6 \leq 0.9] \\ &\quad \wedge G_1) \vee ([G_7 \leq 6300] \wedge G_4) , \end{aligned}$$

$$\begin{aligned}
\langle \text{query} \rangle &\rightarrow (\langle \text{query} \rangle) \wedge \langle \text{literal} \rangle \\
\langle \text{query} \rangle &\rightarrow (\langle \text{query} \rangle) \vee \langle \text{literal} \rangle \\
\langle \text{query} \rangle &\rightarrow \langle \text{literal} \rangle \\
\langle \text{literal} \rangle &\rightarrow \langle \text{predicate} \rangle \\
\langle \text{literal} \rangle &\rightarrow \neg \langle \text{predicate} \rangle
\end{aligned}$$

Figure 3.1: Generative grammar of the linearly parsable propositional query language. The non terminal symbol $\langle \text{predicate} \rangle$ is a predicate as defined in Section 3.1.1.

as well as q_1 and q_2 above, all belong to this query language.

However, while permitting full expressivity of Boolean formulae, this unrestricted query language contains queries that are difficult to interpret, for instance because of deeply nested structures.

Consider, as a simple example, a query over numerous attributes, possibly with a complex nested structure, such as q_6 . Its support might match very well that of another query, resulting in a highly accurate redescription. However, because of the many entangled conditions, it will be difficult for the analyst to interpret it, that is, to understand the conveyed meaning, directly limiting its interestingness.

In addition, the resulting space of redescriptions lacks organizing structure and is therefore very hard to search.

Linearly parsable queries. As a compromise between these two extremes, we propose in Article I to use linearly parsable formulae as our propositional query language. This language comprises the queries generated by the simple formal grammar shown in Figure 3.1, where the non terminal symbol $\langle \text{predicate} \rangle$ is a predicate as defined in Section 3.1.1.

Simply put, these are queries which can be evaluated from left to right irrelevant of the binary operators precedence. Among the example queries q_1 – q_6 , all but q_4 and q_6 satisfy this criterion.

As an additional requirement of the language to ensure better interpretability, we restrict every attribute to appear only once. Query q_5 will be rejected since attribute G_7 appears twice. However, in this case, q_5 can be equivalently rewritten in the acceptable form $G_1 \vee [3000 \leq G_7]$.

Although in theory the choice of a query language is a building block of the problem definition, prior to the algorithm design, computability represents a strong practical constraint influencing the choice. For instance, linearly parsable queries naturally result from iterative atomic extensions, progressively appending literals, i.e. positive or negated predicates, to the

current query, as it happens in the GREEDY algorithm [GMM08]. Another example are the queries obtained with the CARTWHEELS algorithm [RKM⁺04], whose typical form directly reflects the decision trees used for mining them.

3.2 Relational Queries

When the dataset contains information about the relations between objects in addition to, or instead of, the properties of individual objects, it is called a *relational dataset*. An interaction between n objects is modelled as an n -ary relation, corresponding to a hyperedge of cardinality n in a hypergraph whose nodes represent the objects.

In the work presented here, we restrict ourselves to binary relations. That is, we consider only interactions involving two objects at once, as can be represented by usual graph edges. This restriction allow us to employ techniques from graph mining when processing the datasets. Indeed, this kind of relational dataset can be viewed as a multilabelled directed graph $(\mathcal{O}, \mathcal{R})$, where nodes correspond to the objects \mathcal{O} , and edges to relations \mathcal{R} between them. Two families of functions, \mathcal{N} and \mathcal{E} , label nodes and edges with their attributes, respectively. Relations of higher arity might be decomposed into binary relations, possibly by introducing intermediary objects.

Similarly to the propositional setting, predicates can be constructed from the attributes and combined into statements to form relational queries. We introduced relational queries for redescription mining in Article III.

Example 4. *Continuing with our example on world countries, we now look at a dataset representing their relations from a geopolitical point of view. This dataset involves other objects in addition to the nine countries: five international organizations, namely the Commonwealth of Nations, the European Union (EU), the North Atlantic Treaty Organization (NATO), the Organization of American States (OAS) and the United Nation Security Council, as well as five cities and seven languages.*

The relations existing between these objects can be represented as a directed and labelled network, as shown in Figure 3.2. Object attributes could be indicated as labels on the nodes. However, they are listed separately in Table 3.2 for better readability.

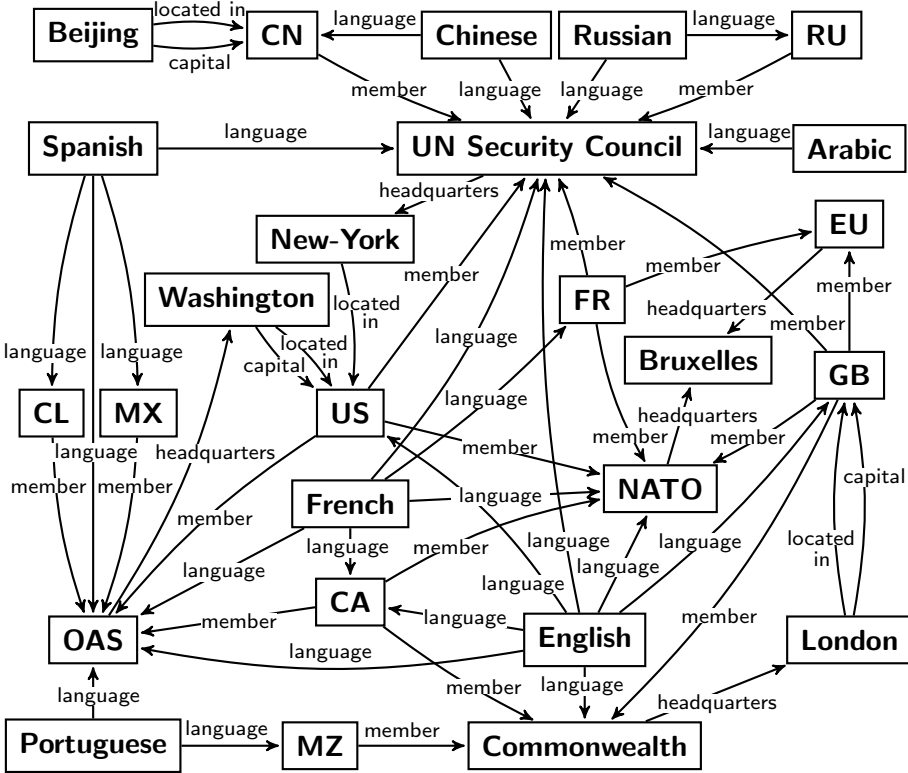


Figure 3.2: Example dataset. Geopolitical relations between world countries involving cities, international organizations and languages.

3.2.1 Predicates

A relational dataset might be heterogeneous, in the sense that not all attributes are defined for every object. The subset of objects that an attribute characterizes constitute its domain, such that $\text{dom}(N_i) \subseteq \mathcal{O}$ for node attributes and $\text{dom}(E_i) \subseteq \mathcal{O} \times \mathcal{O}$ for edge attributes.

For instance, in our example, population is recorded for both countries and cities and the year of foundation is defined for organizations only. All attributes are not gathered into a single matrix. Here, they are presented in distinct tables for the countries, cities and international organizations, in Tables 3.2 (a), (b) and (c), respectively.

Similarly to the propositional predicates seen previously, node and edge predicates can be built from object and relation attributes, respectively. In addition, we also consider comparison predicates.

Table 3.2: Example dataset. Geopolitical attributes.

(a) World countries.

Country	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	1 353.821
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) Cities.

City	6) Population (10 ⁶ hab.)
Beijing	16.801
Bruxelles	1.119
London	8.173
New-York	8.336
Washington D.C.	5.703

(c) International organizations.

Organisation	7) Year of Foundation
Commonwealth	1926
EU	1952
NATO	1949
OAS	1948
UN Security Council	1946

Node predicates. For a given node attribute N_i and subset R_S of its range, a node predicate $\nu_{N_i}^{R_S}(o)$ is true for an object o if and only if the node attribute N_i is defined and takes value in R_S for this object. Node predicates are the counterpart of propositional predicates, with the additional condition that the attribute needs to be defined, which is implicitly assumed in the propositional case. Using Iverson bracket notation, this is written as

$$\nu_{N_i}^{R_S}(o) = [o \in \text{dom}(N_i) \wedge N_i(o) \in R_S].$$

For example, objects for which population information is available and ranges from 10 to 30 millions support the predicate $\nu_{\text{population}}^{[10,30]}(o)$, that is, Chile, Mozambique and Beijing. Node predicates $\nu_{\text{independence}}^{[1800,1900]}(o)$ and $\nu_{\text{regime}}^{\text{Monarchy}}(o)$ respectively select countries that became independent during the XIXth century and monarchies, namely Canada, Chile and Mexico, on one hand, Canada and the United Kingdom on the other. In the latter case, we slightly abuse notation to denote, strictly speaking, the singleton set $\{\text{Monarchy}\}$.

Edge predicates. Likewise, for a given edge attribute E_i and subset R'_T of its range, an edge predicate $\epsilon_{E_i}^{R'_T}(o_1, o_2)$ is true for a pair of objects (o_1, o_2) if and only if the edge label E_i is defined for that pair and takes value in R'_T . This is equivalently expressed as

$$\epsilon_{E_i}^{R'_T}(o_1, o_2) = [(o_1, o_2) \in \text{dom}(E_i) \wedge E_i(o_1, o_2) \in R'_T].$$

When the range of the attribute is limited to a single value it simply indicates the existence of the relation without qualifying it. In our example, all edge attributes are of such existential type. In that case, we denote the corresponding predicate simply as $\epsilon_{E_i}(o_1, o_2)$.

In particular, the edge predicate $\epsilon_{\text{language}}(o_1, o_2)$ selects pairs of objects where the former is an official language of the latter. There are 21 such pairs in our example dataset, including the pairs of languages and organizations (Arabic, UN Security Council), (Spanish, OAS) and (French, NATO), as well as the pairs of languages and countries (Spanish, Mexico) and (Chinese, China). The membership predicate $\epsilon_{\text{member}}(o_1, o_2)$ is supported by 18 pairs of objects, a country and an organization, where the country is a member of the organization.

In comparison to these existential edge attributes we could consider a detailed variant, e.g. an attribute which does not simply indicate membership but more precisely qualifies the relation with values such as permanent, observing, or elected.

Comparison predicates. Finally, comparison predicates are built as follows. For a given object attribute we choose as a comparison function a binary relation \prec defined over its range. Then, a comparison predicate $\phi_{N_i}^{\prec}(o_1, o_2)$ is true for a pair of objects (o_1, o_2) if and only if both node labels $N_i(o_1)$ and $N_i(o_2)$ are defined and $N_i(o_1) \prec N_i(o_2)$. That is, a comparison predicate is defined as

$$\phi_{N_i}^{\prec}(o_1, o_2) = [o_1 \in \text{dom}(N_i) \wedge o_2 \in \text{dom}(N_i) \wedge N_i(o_1) \prec N_i(o_2)].$$

As an example, consider the less-than relation over the real-valued population attribute, such that the comparison predicate $\phi_{\text{population}}^<(o_1, o_2)$ holds true if and only if o_1 is less populated than o_2 . The pairs (Chile, Beijing), (London, New-York) and (Canada, Russia), among others, support this predicate.

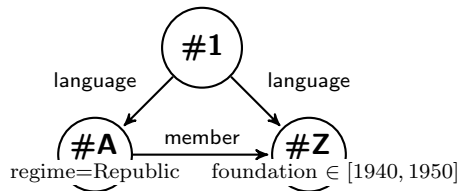


Figure 3.3: Graphical representation of a relational query $q_1(\#A, \#Z)$.

3.2.2 Statements

As in the propositional case, the predicates introduced in the previous section constitute building blocks, which are combined into statements to form queries.

In the relational setting, queries consist of monotone conjunctions of nodes, edges and comparison predicates with a subset of variables of interest selected as query variables. More precisely, borrowing terminology from inductive logic programming [MDR94] and using the *Prolog* notation [DEDC96], a relational query is a definite clause of the form

$$q(X_1, \dots, X_m) \text{ :- } b_1, \dots, b_n.$$

where the body elements b_i are node, edge or comparison predicates and q is a special predicate denoting the query. The query variables X_1, \dots, X_m in the head also occur in the body.

Such queries can be represented in graphical form. For this purpose, we adopt the following conventions. While data nodes are represented as squares, variables, i.e. query nodes, are represented as circles. Furthermore, we use the hash symbol together with a letter to denote the query variables and with a number to denote any other intermediate variable.

For instance, the graph in Figure 3.3 represents the following relational query

$$q_1(\#A, \#Z) \text{ :- } \nu_{\text{regime}}^{\text{Republic}}(\#A), \nu_{\text{foundation}}^{[1940,1950]}(\#Z), \epsilon_{\text{member}}(\#A, \#Z), \\ \epsilon_{\text{language}}(\#1, \#A), \epsilon_{\text{language}}(\#1, \#Z).$$

This query involves node predicates based on nominal and numerical attributes. However, we generally consider only Boolean and nominal object attributes to construct node predicates, while numerical attributes are used to construct comparison predicates.

To determine the support of the query, the statement is matched against the data: each variable in the query has to be matched to a node in the graph, respecting the predicates in the query body. We denote such

a match of variables Y_j to objects o_{i_j} by the corresponding substitution $\theta = \{Y_1/o_{i_1}, \dots, Y_l/o_{i_l}\}$; θ reduced to the query variables is called *answer substitution*. The set of all distinct answer substitutions of query q is its support, $\text{supp}(q)$. Hence, the support of query $q(X_1, \dots, X_m)$ is a set of m -tuples of objects.

The support of query q_1 above consists of pairs of objects, a republic and an organization founded in the 1940s, where the former is a member of the latter and they share an official language. Such is the case of Chile and the OAS or Canada and the NATO, among others.

As another example, by adding the intermediate variable $\#1$ to the head as a query variable renamed as, say, $\#B$, we obtain a query of arity three, $q_2(\#A, \#B, \#Z)$. Entities supporting this modified query are triplets consisting of a country, a language and an organization, including (Chile, Spanish, OAS) or (Canada, English, NATO) as well as (Canada, French, NATO).

Further examples are shown in Figure 3.4 respectively representing the following relational queries:

$$\begin{aligned}
 q_3(\#A) &:- \epsilon_{\text{capital}}(\#A, \#1), \epsilon_{\text{headquarters}}(\#2, \#A). \\
 q_4(\#A, \#Z) &:- \epsilon_{\text{capital}}(\#1, \#A), \epsilon_{\text{located in}}(\#2, \#A), \\
 &\quad \epsilon_{\text{headquarters}}(\#Z, \#2). \\
 q_5(\#A, \#Z) &:- \epsilon_{\text{capital}}(\#1, \#A), \epsilon_{\text{headquarters}}(\#Z, \#2), \\
 &\quad \phi_{\text{population}}^<(\#1, \#2). \\
 q_6(\#A, \#B, \#Z) &:- \epsilon_{\text{capital}}(\#1, \#A), \epsilon_{\text{language}}(\#3, \#A), \\
 &\quad \epsilon_{\text{located in}}(\#2, \#B), \epsilon_{\text{language}}(\#3, \#B), \\
 &\quad \epsilon_{\text{headquarters}}(\#Z, \#2), \phi_{\text{population}}^<(\#1, \#2).
 \end{aligned}$$

Interpretability of relational queries. Relational queries are a rather complex type of pattern. They are more easily understood in their graphical representation, allowing to visualize the different objects involved and their connections.

The limitation to monotone conjunctions aims to ensure the interpretability of the queries. First, queries involving both conjunctions and disjunctions would be still more complex and could not be represented as graphs, making interpretation very difficult. Second, because of the heterogeneity of the dataset, negation is equivocal. A predicate might not hold for an object or pair of objects for one of two reasons, either because the attribute is not defined or because it takes a different value. In most cases, the complementary predicate, selecting objects or object pairs for which

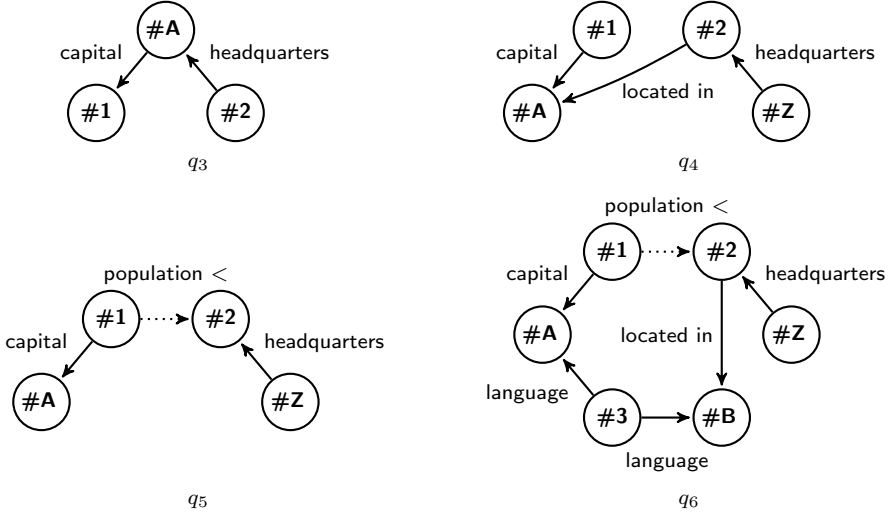


Figure 3.4: Four relational queries.

the attribute is defined but takes a different value or compares differently, can be obtained by replacing the value range by its complement or choosing a different comparison function. For example,

$$\begin{aligned} \nu_{\text{regime}}^{\text{Republic}}(\#A) & \text{ becomes } \nu_{\text{regime}}^{\text{Monarchy}}(\#A), \text{ and} \\ \phi_{\text{population}}^{\lt}(\#1, \#2) & \text{ becomes } \phi_{\text{population}}^{\geq}(\#1, \#2). \end{aligned}$$

In addition, we require clauses to be *linked*, meaning that the set of edge predicates connects any two query variables (X, Y) . Indeed, relational queries should characterize the connections between the objects of interest and unlinked queries are of little use for this purpose. In particular, query q_5 above does not satisfy this requirement, since the comparison predicate between variables $\#1$ and $\#2$ is not considered for linkage. All other queries are linked.

Furthermore, when mapping the statement onto the data, we require that each variable in the query be matched to a different node in the graph. As common in graph mining, we use subgraph isomorphism or, in terms of logic, θ_{OI} -subsumption [EMS⁺94], to match queries against the data graph. We consider that the resulting queries are more intuitive. They are also easier to search. For instance, variables $\#1$ and $\#2$ in query q_4 should be mapped to different objects. The substitution

$$\{\#A/\text{US}, \#1/\text{Washington D.C.}, \#2/\text{New-York}, \#Z/\text{UN Security Council}\}$$

complies with this requirement so the pair (US, UN Security Council) supports query q_4 . On the other hand, the pair (GB, Commonwealth) does not support this query because London is simultaneously the capital of the United-Kingdom and houses the headquarters of the Commonwealth.

Finally, the definitions above apply to queries of arbitrary arity but our work focuses on queries of arity two, those describing the relations between pairs of objects. While our proposed techniques are tuned towards this type of patterns, they might still be extended to queries of higher arity.

As a word of warning about the terminology, we point out that the term *variable* refers exclusively to a place holder for objects in this introduction and in Article III, while it is used to refer to a vector-valued attribute in the propositional setting described in Articles I and II.

The propositional setting presented in Section 3.1 readily corresponds to the relational case where attributes are restricted to node attributes over a homogeneous set of objects, i.e. such that $\mathcal{A} = \mathcal{N}$, $\mathcal{E} = \emptyset$, and $\text{dom}(A) = \mathcal{O}$, $\forall A \in \mathcal{A}$. In the absence of edge attributes, the connectivity requirement practically restricts the number of variables appearing in the body of the query to one. This variable also necessarily appears in the head as the only query variable. Hence, the support of such a query consists of tuples of size one. This directly maps to propositional monotone conjunctive queries whose support consists of a set of individual objects. In a propositional query, all predicates characterize the same object. For this reason, the variable that refers to that object always remained implicit. In fact, the notation $[A_j \in R_S]$ used in propositional queries is a short hand for

$$\nu_{A_j}^{R_S}(o) = [o \in \text{dom}(A_j) \wedge A_j(o) \in R_S].$$

This shows that the propositional setting is a restriction of the general relational setting.

Propositional redescription mining is directed towards the description of individual objects, as compared to the characterization of connection patterns between multiple objects in the relational setting. Propositionalization tools aim at turning relational datasets into propositional ones by constructing propositional attributes based on the local connections around individual nodes [KZ09, DEV12]. Redescriptions consisting of queries of arity one, such as q_3 , can be obtained by applying propositionalization coupled with a propositional redescription mining algorithm. More generally, as argued in Article III, replacing a fully relational method by propositionalization coupled with a propositional redescription mining algorithm does not allow to maintain the full connectivity information.

Chapter 4

Exploration Strategies

Given a query language, the space of possible queries needs to be explored in search of pairs that constitute good redescrptions. Combined with diverse constraints on the redescrptions, different query languages give rise to different search spaces. Beneficial properties of the language and constraints, such as anti-monotonicity, might allow for a particularly efficient exploration. However, this is not the case in general.

Considering propositional data, there are 2^{2^k} non-equivalent unrestricted Boolean expressions over a set of k predicates, the number of k -place truth functions. Hence, given a set of n predicates, there are

$$\kappa_n = \sum_{k=0}^n \binom{n}{k} 2^{2^k}$$

different expressions of arbitrary length. Furthermore, when looking at two datasets with $n_{\mathbf{L}}$ and $n_{\mathbf{R}}$ predicates respectively, there are potentially up to $(\kappa_{n_{\mathbf{L}}} - 1)(\kappa_{n_{\mathbf{R}}} - 1)$ pairs of non-empty queries to examine.

For reasons of interpretability, one would generally only consider queries involving at most a small fixed number of predicates and impose syntactic restrictions on the combination of predicates, significantly reducing the amount of candidate pairs. Still, it is generally too large to allow for an exhaustive enumeration. In the presence of non-Boolean attributes, the number of predicates that can be constructed might be extremely large. Furthermore, in the relational case, there can be infinitely many distinct valid queries. Thus, resorting to heuristics is a practical necessity.

The rest of this chapter outlines and compares three generic exploration strategies for mining redescrptions.

4.1 Query Mining and Pairing

The simplest exploration strategy consists of two steps. First, individual queries are mined from the dataset independently. Second, queries with similar supports are paired to form redescrptions.

On one hand, if the number of views is small, the most practical approach is to mine queries from each view separately, then to pair them across the views. On the other hand, if the number of views is large, for instance when each predicate is associated to a distinct view, one might mine queries over all predicates pooled together, then pair queries with similar supports that involve predicates from disjoint sets of views.

Example 5. *Continuing with our running example, we now consider the case of a propositional dataset divided into two views. Specifically, one view consists of the seven geographic attributes, G_1, \dots, G_7 , listed in Table 3.1, while the other consists of the four geopolitical attributes of Table 3.2 (a), henceforth denoted as P_1, \dots, P_4 .*

Faced with such data, the first mining strategy would be carried out by mining geographic and geopolitical queries independently, before pairing them based on support similarity.

The main advantage of such a mine-and-pair scheme is to allow the adaptation of frequent itemset mining algorithms in a very straightforward fashion. Over the last couple of decades, a great number of algorithms have been developed to mine monotone conjunctive queries over a fixed set of propositional predicates [AS94, PCY95, ZH02, CG02, HPYM04], to cite only a few among the most prominent examples. Typically, they exploit the anti-monotonicity of the support of queries to safely prune the search space, resulting in highly efficient complete enumeration procedures.

An alternative to mining and pairing is to replace the second step with a splitting procedure. That is, pool together all predicates for the initial mining step, then split the queries depending on views. However, the existence of a query does not imply that it can be split into two subqueries that both hold with the same supports. More generally, there is no guarantee that there will be a way to split the query found into two subqueries over disjoint views with sufficiently similar supports and even less so with relational queries, when the connectivity needs to be maintained.

When the data originate from two views, monotone conjunctive redescrptions can be mined exhaustively in a level-wise fashion similar to the *Apriori* algorithm [AS94, MTV94]. The support cardinality of both queries and of their intersection, as well as some associated measures, are

antimonotonic and can be used safely for pruning. However, support similarity functions are typically not antimonotonic, even in this simplest case. This strategy is adopted in Article IV, where the search for the best rule at each iteration is carried out exhaustively.

Mining and pairing is best suited for exhaustive search. We now turn to schemes that can be used for an exploration relying on heuristics.

4.2 Alternating Scheme

Another strategy for mining redescrptions is to use an alternating scheme. The general idea is to start with one query, find a good matching query to complete the pair, drop the first query and replace it with a better match, and continue to alternate in this way, constructing a fresh query on either side until no further improvement can be achieved.

For example we would start with an initial query $q_{\mathbf{L}}^{(0)}$ over geographic attributes and look for a good matching query over geopolitical attributes, $q_{\mathbf{R}}^{(1)}$. Next, we would look for another query on the geographic attributes, $q_{\mathbf{L}}^{(2)}$, that forms a better pair $(q_{\mathbf{L}}^{(2)}, q_{\mathbf{R}}^{(1)})$, and so on.

In fact, if one side of the redescription is fixed, finding an optimal query to complete the pair constitutes a binary classification task. The entities supporting the fixed side provide positive examples and the remaining entities might be considered as negative examples. Thus, any feature-based classification technique could potentially make up the basis for a redescription mining algorithm, with the associated query language consisting of the possible classification criteria.

However, to be consistent with our position on the interpretability of queries, we exclude for instance the direct use of linear classifiers. Indeed, the resulting weight vectors have reduced interpretability compared to explicit constraints on the range of attributes.

This alternating scheme was introduced by Ramakrishnan et al. [RKM⁺04]. Their CARTWHEELS algorithm is based on decision trees and the query language consists of the resulting rules. In Article III we propose an alternating scheme for mining relational redescrptions. It relies on a relational query miner in order to find matching queries to complement the current pair.

The question of finding good starting points arises naturally. One option is to randomly partition the entities into positive and negative examples, using one or several such partitions to initialize the search, instead of actual queries [RKM⁺04]. Queries that consist of a single predicate, i.e. the simplest possible queries, offer another choice for the initialization. This

option, adopted in Article III, is particularly appropriate for the relational setting, where the number of possible partitions is extremely large and a majority would not result in any query. In either case, accepting queries that do not match the fixed side very well during the first iterations can help increase the exploratory power of the algorithm.

For a fixed number of starting points and a limit on the number of alternations, the complexity of such a scheme depends primarily on the complexity of the chosen classification algorithm.

4.3 Greedy Atomic Updates

Finally, a third exploration strategy relies on iteratively finding the best atomic update to the current query pair. More precisely, given a pair of queries, one tries to apply atomic operations on either query to improve the candidate redescription, until no further improvement can be achieved. Conceptually, atomic operations at hand include the addition, deletion and edition of predicates. That is, one might add a fresh predicate to the query, remove a predicate from the query or alter some predicate already occurring in the query, in particular, by modifying the range of the truth value assignment.

For example, if our current candidate redescription is

$$(G_2 \wedge [1 \leq G_6 \leq 10] , [100 \leq P_4])$$

by adding, deleting and editing a predicate, we might modify it respectively to

$$\begin{aligned} & (G_2 \wedge [1 \leq G_6 \leq 10] , [100 \leq P_4] \vee [P_3 = \text{Monarchy}]) , \\ & ([1 \leq G_6 \leq 10] , [100 \leq P_4]) \text{ or} \\ & (G_2 \wedge [5 \leq G_6 \leq 10] , [100 \leq P_4]) . \end{aligned}$$

Memorization of the explored queries can be employed to prevent the algorithm from repeating itself. For the initialization, one might consider the pairs of best matching predicates constructed with any two attributes from different views.

This strategy, restricted to addition of predicates, i.e. extensions, was first introduced as the GREEDY algorithm by Gallo et al. [GMM08]. Building upon this work, we proposed the REREMi algorithm in Article I. The algorithm is strengthened with a beam search to keep the current top candidates at each step instead of focusing on the single best improvement.

Typically, such an algorithm would consider in turn each attribute to generate modified candidates, a subset of which will be selected and further

updated at the next step. The running time upper bound for this strategy is in the order of the product of the number of starting points, the maximal number of iterations and the beam width multiplied by the number of real-valued attributes times the squared number of objects plus the number of Boolean attributes and of categories of nominal attributes times the number of objects. For instance, in the example of Section 6.3, this product equals $100 \times 10 \times 4 \times (48 \times 2575^2 + 190 \times 2575) \approx 10^{12}$. In fact, not all objects can affect the support for a particular extension and determining the optimal extension attainable with a given real-valued attribute is quadratic in the number of cut points, which is at most the number of distinct values of the attribute and usually much smaller than the number of rows, as we argue in Article I. Thus, this strategy proved feasible in practice.

When global constraints on the query need to be enforced, like connectivity in the relational case, the alternating scheme presented previously is better suited compared to such atomic updates. On the other hand, atomic extensions might be favored when the construction of individual predicates is costly, as for instance when it involves finding the best interval for real-valued attributes. Indeed, in such cases, building a fresh query from scratch at each alternation can represent a waste of energy if the successive queries are close variations of their replacement and, in effect, the same predicates are generated over and over again.

Chapter 5

Pattern Selection

In this chapter we take a closer look into pattern selection. After having defined query languages and sketched methods for exploring the search space, we now discuss the evaluation of quality and the selection of redescrptions.

First, we consider the definition of quality criteria and their enforcement with respect to individual redescrptions. Such quality criteria, arising from background knowledge or particular domain requirements and modelled as a set of constraints \mathcal{C} , determine a bias towards individual redescrptions.

However, the aim of a data mining task generally lies in finding patterns that together describe the data well, instead of finding good patterns taken in isolation. That is, the analyst is interested in identifying a high quality set of patterns rather than a set of high quality patterns. Thus, we also consider the problem of mining sets of redescrptions.

5.1 Individual Patterns

In this section, we inventory criteria that affect the quality of redescrptions, before giving an outline of how they are enforced during the mining process.

5.1.1 Quality Criteria

Soon after the problem of association rule mining was defined [AIS93] and the first efficient solution, the now standard *Apriori* algorithm, was proposed [AS94, MTV94], it became clear that frequency and confidence are not sufficient to ensure the quality of the results [KMR⁺94, SVA97]. Similarly with redescrptions, while the structural difference in the queries and the similarity of their supports are defining features, they are not sufficient to guarantee the quality of the results. Other crucial aspects need to be taken into account.

The quality of a pattern is a rather abstract property. It results from a combination of characteristics that we try to evaluate with objective criteria. For instance, we might consider that a good redescription is a redescription with easily interpretable queries and statistically significant supports. That said, we still need to define precisely what is meant by interpretability and statistical significance, preferably in an operative way, by defining means to measure these characteristics.

Queries. Besides structural difference, in other words, the requirement that the attributes over which the queries forming a redescription are expressed must belong to distinct views, expressivity and interpretability are the main desirable characteristics for the queries of a redescription. For instance, long and nested formulae are generally hard to interpret, and are therefore of little interest for describing the data. Yet, too strong restrictions imposed on the syntactic complexity of queries might severely limit the expressive power of the language. Hence, a balance needs to be struck between these partly conflicting characteristics, which are moreover difficult to measure. The expressivity of the language and interpretability of individual elements are largely determined by the syntactic restrictions imposed on the construction of statements, discussed in Chapter 3. In addition, a simple means to control the complexity of a query is to limit its length as it is generated.

Support. The similarity between the supports of the queries of a redescription is a defining property of a redescription, also called its accuracy. As mentioned in Section 2.1, the similarity relation \sim is generally specified as a set similarity function together with a threshold. Various functions can be used for this purpose. For a pair of queries ($q_{\mathbf{L}}$, $q_{\mathbf{R}}$), we denote by $E_{1,1}$, $E_{1,0}$, $E_{0,1}$ and $E_{0,0}$ the subsets of entities that support both queries (i.e. $E_{1,1} = \text{supp}(q_{\mathbf{L}}) \cap \text{supp}(q_{\mathbf{R}})$), support only $q_{\mathbf{L}}$, support only $q_{\mathbf{R}}$ and do not support either queries, respectively. Then, examples of similarity functions include the following:

$$\begin{aligned} \text{matching number} &= |E_{1,1}| + |E_{0,0}| , \\ \text{matching ratio} &= \frac{|E_{1,1}| + |E_{0,0}|}{|E_{1,0}| + |E_{1,1}| + |E_{0,1}| + |E_{0,0}|} , \\ \text{Russel \& Rao coefficient} &= \frac{|E_{1,1}|}{|E_{1,0}| + |E_{1,1}| + |E_{0,1}| + |E_{0,0}|} , \\ \text{Jaccard coefficient} &= \frac{|E_{1,1}|}{|E_{1,0}| + |E_{1,1}| + |E_{0,1}|} , \end{aligned}$$

$$\begin{aligned} \text{Dice coefficient} &= \frac{2|E_{1,1}|}{|E_{1,0}| + 2|E_{1,1}| + |E_{0,1}|}, \text{ and} \\ \text{Rogers \& Tanimoto coefficient} &= \frac{|E_{1,1}| + |E_{0,0}|}{|E_{1,0}| + 2|E_{1,1}| + |E_{0,1}| + |E_{0,0}|}. \end{aligned}$$

The Jaccard coefficient is commonly used in redescription mining. This choice is motivated mainly by the simplicity of the measure and its agreement with the symmetric approach adopted in redescription mining. Indeed, the Jaccard coefficient weights the support of the two queries equally. In addition, it is scaled to the unit interval without involving the set of entities that support neither queries, $E_{0,0}$. This is an asset, particularly in the relational setting, when the dataset is heterogeneous or requires the use of the *open world assumption*, i.e. the assumption that a relation may exist despite not being recorded in the dataset, so that this set may not be easily and appropriately defined.

Besides accuracy, it can be desirable to fix lower or upper bounds on the support cardinality of the queries and possibly on that of the individual predicates involved as well. Also, in the relational case where entities consist of object tuples, more complex constraints can be imposed, for instance on the number of distinct objects appearing at any given position or on the number of distinct tuples up to reordering. These constitute secondary constraints on the redescrptions that might arise from the domain knowledge and help select redescrptions of interest.

Statistical significance. A crucial requirement for the redescrptions mined is that they be statistically significant. To provide new insight about the data at hand, a redescription should not be likely to arise at random from the underlying data distribution. In particular, the accuracy of a redescription should not be readily deducible from the support of its queries. For instance, if both queries cover almost all objects, the overlap of the supports is necessarily large, too, and a high accuracy is no surprise.

Hence, one way to measure the significance of a redescription is to estimate how likely such a pattern is to arise randomly. That is, the presence of the redescription is tested against the null-model where the two queries are assumed to be independent. Consider two statistically independent random queries whose marginal probabilities correspond to those of the queries under consideration. In other words, their marginal probabilities equal the fraction of covered entities $p_{\mathbf{L}} = |\text{supp}(q_{\mathbf{L}})| / |E|$ and $p_{\mathbf{R}} = |\text{supp}(q_{\mathbf{R}})| / |E|$, respectively. A p -value representing the probability that these independent queries have an overlap equal to or larger than the one observed can be

computed using the binomial distribution as follows

$$\text{pvalM}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \sum_{s=|E_{1,1}|}^{|E|} \binom{|E|}{s} (p_{\mathbf{L}}p_{\mathbf{R}})^s (1 - p_{\mathbf{L}}p_{\mathbf{R}})^{|E|-s}.$$

This is the probability of obtaining a set of same cardinality $|E_{1,1}|$ or larger if each element of a set of size $|E|$ has a probability equal to the product of marginals $p_{\mathbf{L}}$ and $p_{\mathbf{R}}$ to be selected, in accordance with the independence assumption.

Alternatively, a p -value can be computed as the probability that two sets of cardinalities $|\text{supp}(q_{\mathbf{R}})|$ and $|\text{supp}(q_{\mathbf{L}})|$, respectively, drawn independently at random from a set of size $|E|$ have an overlap of cardinality $|E_{1,1}|$ or larger. This is Fisher’s exact one-sided p -value [Fis38], evaluated using the hypergeometric distribution:

$$\text{pvalO}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \sum_{s=|E_{1,1}|}^{|E|} \frac{\binom{|\text{supp}(q_{\mathbf{L}})|}{s} \binom{|E| - |\text{supp}(q_{\mathbf{L}})|}{|\text{supp}(q_{\mathbf{R}})| - s}}{\binom{|E|}{|\text{supp}(q_{\mathbf{R}})|}}.$$

High p -values indicate that the independence assumption, i.e. the null hypothesis, cannot be rejected and the redescription is then considered less significant. The computation of such theoretical p -values relies on assumptions about the underlying data distribution. Both tests assume that all elements of the population can be sampled with equal probability, from a fixed distribution. The sampling distribution is calculated only on expectation in the former case, while the latter relies on the stronger assumption of fixed marginals. However, the real data might deviate from these simple assumptions, weakening the significance tests.

Theoretical p -values can be complemented by empirical statistical tests, carried out after randomizing the original data. Both approaches rely on statistical hypothesis testing. Developing a well-founded methodology based on this theory to assess the significance of redescriptions requires to consider a number of issues such as appropriate multiple testing with scaling and corrections, as well as property-preserving randomization and uniform sampling of datasets in the case of randomization tests.

These questions do not constitute the core of our contribution and we do not discuss them in depth here. Instead we refer the interested reader to the relevant literature [LR05, Edg95] for general considerations about statistical hypothesis testing and randomization tests or concerning their application to data mining, from the early study of statistical significance of association rules [BMS97, MS98] to recent developments [Oja11, Han12, Vuo12], among others [Web07, ZPT04].

In Article I, we assess the significance of propositional redescription using both approaches. Empirical p -values, in particular, were obtained following the approach of Gionis et al. [GMMT07]. Specifically, copies of the original data are generated and randomized so as to maintain, at least approximately, the row and columns marginals. Then, the mining process is run anew on each of the copies. Redescriptions from the original data that are more accurate than a chosen fraction of the redescription obtained from the randomized copies are deemed significant with respect to the preserved properties, others are discarded.

We did not study this aspect in the context of relational datasets. Evaluating the statistical significance of complex connection patterns such as our relational queries is a difficult but interesting question. It is open for future investigations, possibly building on works by Hanhijärvi et al. [HGP09] and by Günnemann et al. [GDJE12].

5.1.2 Constraint-based Mining

In the previous section, we discussed characteristics that impact the quality of a redescription and the associated evaluation criteria. Such criteria result in a set of constraints \mathcal{C} that limits the space of acceptable redescription. They can be enforced either during the exploration, by pruning the search space, or as a post-processing step, by filtering the output. Clearly, it is preferable to push the constraints as deeply as possible into the search algorithm, as this improves efficiency by preventing the generation of redescription only to discard them later on.

Significant effort has been directed towards the integration of various constraints into exhaustive search algorithms [SVA97, GR00], giving rise to *constraint-based data mining*. This integration relies on the classification of constraints according to properties such as anti-monotonicity and succinctness [NLHP98] that determine their behavior and consequently how they should be used to prune the search space safely but optimally.

However, these works focus on conjunctive query languages and exhaustive pattern enumeration, by extending the *A priori* algorithm. Allowing disjunctions makes these methods inapplicable because anti-monotonicity no longer holds. Then, heuristic approaches are preferred. Beam-search algorithms, in particular, depend on a ranking function to determine the top candidates that will be explored at the next step. Designing an appropriate score to ensure a satisfactory exploration of the search space with respect to a set of quality constraints is far from trivial.

Conceptually, constraint-based data mining is a step towards *inductive databases* [IM96], a framework for data mining where databases in addi-

tion to the usual data also contain patterns over this data [BKM99, DR02, BDRM05]. Inspired by the success of Codd’s model [Cod70] and the powerful closure property, this framework proposes to see data mining as the manipulation of patterns using a set of expressive operations, similarly to the way ordinary database records can be manipulated using relational algebra.

Consider the following abstract model of data mining introduced by Mannila and Toivonen [MT97]. Given a language of patterns \mathcal{L} , a dataset \mathcal{D} and a selection predicate \mathcal{S} , a data mining task aims at determining the theory of \mathcal{D} with respect to \mathcal{L} and \mathcal{S} ,

$$Th(\mathcal{D}, \mathcal{L}, \mathcal{S}) = \{\phi \in \mathcal{L}, \mathcal{S}(\phi, \mathcal{D})\}.$$

From the point of view of inductive databases, the computation of $Th(\mathcal{D}, \mathcal{L}, \mathcal{S})$ is a generic database operation of evaluating \mathcal{S} , known in this context as an inductive query. Redescription mining naturally integrates into this framework. The language of patterns consists of all query pairs, i.e. $\mathcal{L} = Q \times Q$, and the selection predicate takes the form of the accuracy and structural difference requirements together with the auxiliary constraints on the patterns.

Recently, constraint programming has been proposed as a declarative approach for constraint-based data mining [DRGN08, GNZDR11, KBC10]. It shares with inductive databases the aim of providing a generic language for specifying desirable characteristics of patterns, independently of the procedure used to identify the actual patterns. Constraint programming has been applied to redescription mining restricted to monotone conjunctions [GNDR13]. In general, this approach is currently unable to deal efficiently with certain classes of patterns and constraints, but promising for others.

5.1.3 Interactive Data Mining

Selecting patterns by explicitly specifying a set of desired characteristics and the associated means of evaluation offers much flexibility. However, such an ad-hoc approach also has its drawbacks. It might require extensive background knowledge and multiple rounds of trial and errors to familiarize with the tool and tune the parameters so as to obtain good results.

There, an interactive interface that allows the analyst to inspect patterns as they are generated and that readily provides feedback comes in handy. The SIREN interface presented in Article II is a first step in this direction. Potentially, by modifying the selection criteria, the analyst is

able to specify his interest dynamically, in response to the output produced hitherto by the mining algorithm.

However, such high flexibility and adaptability might actually enable the analyst to fine-tune the mining process to obtain only the results that confirm his expectations, putting the discovery of new knowledge in jeopardy.

5.2 Sets of Patterns

Even with strict quality requirements, the returned set of redescription might be large and contain near duplicates.

For instance, a collection of a dozen results which are minor variations of each other is an undesirable result, even if each redescription is highly accurate and of good quality when considered separately. The problem here lies in the redundancy of the redescrptions. When each of them conveys more or less the same information, communicating the whole set of patterns to the analyst represents a large cognitive overhead compared to returning only one, while the informative content remains almost unchanged.

One way of measuring the redundancy between two redescrptions is to compare the occurring attributes and covered entities, since they carry most of the information of a redescription. In particular, given a set of redescrptions, one can consider the similarity of their attribute sets and support intersection separately or compare the overlap of the area defined by the rows and columns involved in either propositional queries. Still, this is a rather crude way of measuring redundancy.

Overwhelming results are a major issue of data mining algorithms. In the domain of frequent itemset mining, it has been proposed to look for concise representations of the results, that is, to identify a small set of patterns from which the rest can be derived exactly or approximately. Such summaries also provide a condensed representation of the data [MT96]. This leads to the notions of closed itemsets [PBT99] and free-sets [BB00], among others (see [CRB04] for an overview).

More generally, given a dataset, a typical aim is to mine a small set of patterns that together describe the data well, rather than consider patterns taken in isolation. One approach to pattern set mining is to employ constraint-based techniques as discussed in Section 5.1.2, this time also taking into account constraints on the entire set of patterns such as support overlap or coverage [GNZDR11].

Methods rooted in Information Theory constitute more holistic approaches to selecting sets of patterns. In particular, alternatives based

on compression and on subjective interestingness, are presented in the following sections, respectively.

5.2.1 Compression-based Model Selection

Compression-based approaches for pattern set mining use compression as a selection criterion. They are motivated by the intuition that the data can be compressed more efficiently by exploiting its internal structure, so that uncovering more of the structure results in improved compression. Simultaneously, redundancies among the patterns result in increased compressed size and are therefore penalized.

Different techniques have been studied to select a model for a given dataset based on information theoretic principles such as the Minimum Description Length (MDL) [Grü07] or the Information Bottleneck (IB) [TPB00]. The MDL and IB approaches differ notably in the fact that the first requires a lossless compression scheme while the latter allows for lossy compression.

The central ingredient of the MDL recipe is the definition of an encoding scheme for the patterns. Then, patterns mined from the data can be stored in a table together with their associated code-words and used to encode the data. The aim is to find a set of patterns that yields the shortest encoding of the data while keeping the size of the code table minimal. A prime example of mining tool based on the MDL principle is the KRIMP itemsets miner [VvLS11].

Inspired by this approach, we propose a method for mining associations from two-view datasets, or, roughly speaking, for compressing the mapping between datasets using redescription, presented in Article IV.

Consider a pair of queries, (q_L, q_R) , with very similar supports, i.e. an accurate redescription. The fact that q_L holds implies that q_R is very likely to hold too, and vice versa. Therefore, such patterns provide information about the associations between the two views of the dataset and can be used to encode one view given the other. In other words, they allow to translate one view into the other, and we call them translation rules. Then, we look for a set of such rules that together capture the cross-view structure of the data well, as measured by their ability to compress it.

Pairs of propositional monotone conjunctive queries constitute our translation rules and we allow both unidirectional and bidirectional associations, as this provides more flexibility to capture the structure of the dataset and consequently increases the compression ability of our model. More precisely, we consider query pairs where the support of the former query is almost a subset of the support of the latter, i.e. such that the

presence of one query implies the presence of the other, but the converse need not be true.

To summarize, our proposed algorithm identifies pairs of monotone conjunctions which allow to encode one view of the data given the other, or vice versa. Practically, this results in a parameter-free method for mining pairs of queries.

5.2.2 Subjective Interestingness

All the approaches presented so far are concerned only with objective qualities of the patterns, in the sense that the quality depends only on the data and not on the beliefs or preconceived, possibly erroneous, understanding that the analyst possesses prior to the data mining task.

Early on, Silberschatz and Tuzhilin argued that the interestingness of patterns should be evaluated from the point of view of the user [ST95]. They proposed two subjective measures of interestingness. First, actionability depends on whether the analyst can react to the information provided [PSM94]. Second, unexpectedness depends on whether the information surprises the analyst, that is, whether the pattern contradicts the expectations of the analyst, formalized as a system of beliefs [PT98, PT00].

While such approaches arguably employ an extremely simplified representation of the analyst's expectations, they attempt to take these beliefs explicitly into account in the mining process. Therefore, they are called subjective, in contrast to other, objective, approaches.

Already a decade ago, Mannila [Man00] advocated the definition of a theoretical framework for data mining, arguing for the usefulness of such formalization and suggesting five possible candidates for the role, including inductive databases and data-mining as data compression. Pursuing this endeavor, De Bie [DB11a] recently proposed a subjective information theoretic framework for data mining. It is based on the idea that the data mining task can be considered as an exchange of information between the mining process and the analyst.

From this point of view, the data analyst has initial apriori beliefs about the data, modelled as a distribution over possible datasets. During the mining process, information about the data is communicated to the analyst in the form of patterns, allowing him to adjust his beliefs. Then, the amount of new information conveyed by a pattern, i.e. its subjective interestingness, is measured as the reduction of the uncertainty in the data miner's beliefs.

Significance testing approaches based on data randomization mentioned in Section 5.1.1 share some similarities with this line of work. The apriori knowledge of the analyst consists of the preserved properties, so that his

belief is modelled by sampling datasets that possess such properties. However, these empirical approaches are less scalable and suffer from limited resolution compared to the analytical alternative [DB11b]. In addition, they do not allow to model belief updates.

Relying on strong roots in information theory, this framework provides a principled way to define the subjective quality of patterns, as well as the cost of their transmission, i.e. their description length. Formalizing the mining process within this framework should allow to adapt existing algorithms and design new ones so as to maximize the transfer of information from the data to the analyst, for various families of patterns. In particular, integrating redescription mining into this framework is an attractive direction for future research.

Chapter 6

Illustrated Discussion

This chapter provides a practical illustration of the redescription mining task. We present examples of redescriptions mined with the different algorithms we developed, from datasets of diverse domains and using various query languages. These examples complement those of the original publications.

In a sense, the present chapter is a showcase for redescription mining. It is intended to exhibit the power of the method, its versatility, expressivity and interpretability, and not to constitute an experimental evaluation. Detailed documented assessments of the proposed algorithms can be found in the corresponding original publications.

Simultaneously, this exposition provides a basis for a critical discussion of redescription mining. Indeed, through these examples, we point out some weaknesses and drawbacks of the method, which could benefit from further investigations.

We start with a summary of the different algorithms proposed in this thesis with which the results illustrating this chapter were obtained. Each of the four proposed algorithms, indicated in bold in the text, combines aspects of redescription mining discussed in the previous chapters, as outlined below.

6.1 Overview of the Algorithms

As our main contribution, we extended redescription mining outside the world of propositional queries over Boolean attributes. We studied more general query languages and associated algorithms, making the task applicable to a broader range of domains and problems.

In Article I, we proposed the **REREM**I algorithm for propositional re-

description mining. Specifically, the query language considered consists of propositional linearly parsable queries (Section 3.1.2) over Boolean, nominal and real-valued predicates (Section 3.1.1). Our algorithm was built upon the GREEDY algorithm [GMM08], and similarly constructs queries by successive atomic extensions (Section 4.3). Compared to its predecessor, it can handle non-Boolean attributes and missing values and uses a beam search to maintain top candidates, improving exploration. The search is primarily driven by the Jaccard coefficient as the support similarity function (Section 5.1.1). Auxiliary constraints on the redescrptions are enforced ad-hoc by means of ranking and filtering (Section 5.1.2). In addition, we use randomization methods to assess the statistical significance of the obtained redescrptions (Section 5.1.1).

In Article III, we proposed an algorithm for relational redescription mining, called **ARRM**. Node and edge predicates are built over Boolean and nominal object and relation attributes, respectively, while comparison predicates are obtained from real-valued object attributes (Section 3.2.1). These three types of predicates are then combined together into relational queries (Section 3.2.2). To explore the space of queries, we resort to an alternating scheme (Section 4.2). We used the Jaccard coefficient as our measure of choice for accuracy, but it can easily be replaced with another set similarity function (Section 5.1.1). The generation of non-compliant candidates with respect to quality constraints is prevented whenever possible, and filtering is applied to the output to exclude remaining low-quality results (Section 5.1.2).

As we argue in Chapter 4, exploring the space of query pairs with iterative atomic updates is best suited to the propositional setting in the presence of real-valued attributes as it reduces the need for computationally intensive on-the-fly discretization. This approach naturally maps to linearly parsable queries. In the propositional setting, connectivity represents a global constraint on the queries that makes the alternating scheme the most appropriate exploration strategy.

The investigation of pattern selection methods, focusing on the example case of redescrptions, is our second major contribution.

First, we developed an interface for visualizing and mining propositional geospatial redescrptions (Section 3.1), presented in Article II. The proposed interface, called **SIREN**, relies on the REREMi algorithm as its core component. With this tool we take a first step towards interactive and instant redescription mining. Ultimately, such an endeavor could support an entirely interactive selection of redescrptions, by allowing to visualize

results as they are generated and, in response, adjust the parameters of the running algorithm (Section 5.1.3). By giving the analyst as much control as possible over the mining algorithm and filtering procedures, it provides a manually adjustable solution to the selection problem at hand.

Second, we proposed a compression-based method for mining small sets of directional associations from two-view datasets, which can be understood as a preliminary method for mining sets of redescrptions (Section 5.2.1). Specifically, the aim is to find a set of patterns that best describes one side of the data given the other side and vice versa. In other words, we seek to translate one side into the other and hence call such patterns *translation rules*. The algorithm to find them, dubbed **TRANSLATOR**, is presented in Article IV. This algorithm uses exhaustive search with pruning (Section 4.1) and is limited to propositional monotone conjunctions (Section 3.1.2).

Both approaches have advantages and drawbacks. The first approach is very flexible but lacks a theoretical basis, while the second approach constitutes a principled method rooted in information theory but is currently applicable only to a very restricted query language and does not allow to incorporate background knowledge.

The redescrptions presented in the rest of this chapter are sampled from larger sets of results. The indices appearing in the first column of the tables of examples stand for the position of the corresponding redescrptions in the entire result set ordered by decreasing Jaccard coefficient. In the text, we use the table reference together with this index to refer to a redescription, e.g. we refer to the second redescription in Table 6.2 as redescription 6.1(26). For each redescription, we indicate the right-hand side and left-hand side queries, denoted by q_L and q_R , respectively, as well as the Jaccard coefficient, J , and the cardinality of the support intersection, $|E_{1,1}|$.

6.2 Computer Science Bibliography

The first illustration concerns publication patterns in computer science research.

Dataset. Specifically, the dataset was obtained from the DBLP Computer Science Bibliography data base.¹ It consists of a pair of matrices with authors as the objects. The first matrix defines the venues in which each author has published, while the second defines other authors with whom they have published. DBLP_F is a dataset with 6455 authors and 304

¹Data retrieved from <http://dblp.uni-trier.de/db> in March 2010.

Table 6.1: Sample of redescrptions from DBLP_{FB} mined with REREMi.

q_L	q_R	J	$ E_{1,1} $
(1) ICDM \wedge CIKM \wedge APWEB \wedge SIGIR	Q. Yang \wedge W. Fan	0.714	10
(26) CCCG \wedge SODA \wedge GD	M. Yvinec \vee K. Kriegel \vee J. O'Rourke	0.409	27
(27) VLDB \wedge SDM \wedge SIGMOD \wedge KDD	J. Han \wedge P. S. Yu	0.407	11
(36) CCCG \wedge SODA \wedge SoCG	K. Kriegel \vee O. Devillers \vee K. L. Clarkson \vee D. M. Mount	0.383	49
(38) EUROCRYPT \wedge CRYPTO	S. Halevi \vee U. M. Maurer \vee Y. Desmedt \vee D. Naccache	0.382	58
(56) SDM \wedge SIGMOD \wedge ICDE \wedge KDD	(H. Mannila \vee P. S. Yu) \wedge J. Han	0.367	11
(59) COLT \wedge ICML	A. J. Smola \vee R. Khardon \vee S. P. Singh \vee Y. Singer	0.366	34
(60) STOC \wedge EUROCRYPT \wedge CRYPTO	R. Ostrovsky \vee P. Landrock	0.365	35
(71) PODC \wedge STOC \wedge EUROCRYPT	(K. Kurosawa \vee A. Sahai) \wedge R. Canetti	0.351	13
(72) SODA \wedge SoCG \wedge WADS	S. Bereg \vee F. P. Preparata \vee E. D. Demaine	0.351	54
(92) FOCS \wedge STOC \wedge EUROCRYPT \wedge CRYPTO	R. Ostrovsky	0.342	26

Table 6.2: Sample of redescrptions from DBLP_F mined with REREMi.

q_L	q_R	J	$ E_{1,1} $
(1) $[1 \leq \text{SEBD} \leq 8] \wedge [1 \leq \text{LPNMR}]$ $\wedge [1 \leq \text{SIGMOD} \leq 2]$	($[1 \leq \text{M. Lenzerini}]$ $\vee [1 \leq \text{F. Giannotti}]$) $\wedge [2 \leq \text{N. Leone}]$	0.909	10
(10) $[1 \leq \text{ICDM} \leq 12] \wedge [1 \leq \text{CIKM} \leq 5]$ $\wedge [1 \leq \text{APWEB} \leq 10] \wedge [1 \leq \text{SIGIR}]$	($[1 \leq \text{J. Xu}] \vee [1 \leq \text{B. Zhang}]$) $\wedge [1 \leq \text{W. Fan} \leq 9]$	0.667	10
(33) $[7 \leq \text{CCCG} \leq 22] \wedge [2 \leq \text{SoCG} \leq 9]$	$[2 \leq \text{M. H. Overmars}]$ $\wedge [4 \leq \text{E. D. Demaine}]$	0.524	11
(42) $[1 \leq \text{ICDM}] \wedge [1 \leq \text{DASFAA} \leq 10]$ $\wedge [1 \leq \text{WAIM}] \wedge [1 \leq \text{SIGMOD}]$	($[4 \leq \text{J. Pei}] \vee [1 \leq \text{L. Zhang}]$ $\vee [1 \leq \text{G. Yu}]$) $\wedge [1 \leq \text{J. Han}]$	0.500	10
(48) $[1 \leq \text{ESA} \leq 3] \wedge [7 \leq \text{GD}]$	$[1 \leq \text{F-J. Brandenburg}]$	0.476	10
(49) $[1 \leq \text{NIPS} \leq 20] \wedge [10 \leq \text{COLT}]$	$[1 \leq \text{N. Cesa-Bianchi} \leq 2]$	0.476	10
(57) $[5 \leq \text{FOCS}] \wedge [2 \leq \text{STOC}]$ $\wedge [4 \leq \text{CRYPTO} \leq 28]$	$[1 \leq \text{O. Goldreich}]$ $\wedge [1 \leq \text{S. Micali}]$	0.467	14
(70) $[1 \leq \text{PODC} \leq 9] \wedge [2 \leq \text{CRYPTO}]$ $\wedge [1 \leq \text{STOC}] \wedge [2 \leq \text{EUROCRYPT}]$	($[1 \leq \text{R. Venkatesan}]$ $\vee [1 \leq \text{R. Gennaro}]$) $\wedge [1 \leq \text{R. Ostrovsky}]$	0.455	15
(72) $[2 \leq \text{CRYPTO}] \wedge [2 \leq \text{STOC} \leq 23]$ $\wedge [2 \leq \text{EUROCRYPT}] \wedge [2 \leq \text{FOCS}]$	$[1 \leq \text{R. Ostrovsky}]$ $\wedge [1 \leq \text{R. Canetti}]$	0.452	14
(84) $[2 \leq \text{CRYPTO} \leq 10]$ $\wedge [2 \leq \text{EUROCRYPT} \leq 12]$	$[1 \leq \text{R. Gennaro}]$ $\vee [1 \leq \text{E. F. Brickell} \leq 2] \vee [1 \leq \text{V. Rijmen}]$	0.435	37
(98) $[4 \leq \text{SEBD} \leq 12]$	$[2 \leq \text{S. Paraboschi}]$ $\vee [1 \leq \text{F. Mandreoli}] \vee [1 \leq \text{G. Greco}]$	0.431	31

Table 6.3: Glossary of computer science venues.

Acronym	Venue
APWEB	Asia-Pacific Web Conference
CCCG	Canadian Conference on Computational Geometry
CIKM	International Conference on Information and Knowledge Management
COLT	Computational Learning Theory
CRYPTO	International Cryptology Conference
DASFAA	Database Systems for Advanced Applications
ESA	European Symposium on Algorithms
EUROCRYPT	Int. Conference on the Theory and Applications of Cryptographic Techniques
FOCS	IEEE Annual Symposium on Foundations of Computer Science
GD	Graph Drawing
ICDE	International Conference on Data Engineering
ICDM	IEEE International Conference on Data Mining
ICML	International Conference on Machine Learning
KDD	Knowledge Discovery and Data Mining
LPNMR	Logic Programming and Non-Monotonic Reasoning
NIPS	Neural Information Processing Systems
PODC	ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing
SDM	SIAM International Conference on Data Mining
SEBD	Sistemi Evoluti per Basi di Dati (Italian Symp. on Advanced DB Systems)
SIGIR	Int. Conference on Research and Development in Information Retrieval
SIGMOD	ACM SIGMOD Conference
SoCG	Symposium on Computational Geometry
SODA	Symposium on Discrete Algorithms
STOC	Symposium on the Theory of Computing
VLDB	Very Large Data Bases Conference
WADS	Workshop on Algorithms and Data Structures
WAIM	International Conference on Web-Age Information Management

conferences containing information on how many times each author has published in each venue and with each other author. We denote as DBLP_{FB} the Boolean version of this dataset, that is, the dataset obtained by turning every positive value to one.

Results. A sample of redescrptions mined with the REREMi algorithm from the Boolean and numerical versions of the DBLP dataset are shown in Tables 6.1 and 6.2 respectively. A glossary of the venues appearing in the redescrptions is provided in Table 6.3. In this context, a redescription consists of a pair of queries over venues and coauthors, respectively, obtained by combining non-negated predicates into linearly parsable queries using conjunctions and disjunctions. The two settings differ solely in the type of predicates used, Boolean in the first case, real-valued in the second.

The redescrptions obtained identify subareas of computer science such as machine learning or cryptology, characterized by prime conferences and key researchers of the respective fields. For instance, redescription 6.2(33) characterizes eleven researchers having multiple publications at SoCG and

CCCG, i.e. contributing to both these major computational geometry conferences, and having collaborated with either Prof. Mark Overmars or Prof. Erik D. Demaine, two key researchers in that area.

More generally, when analyzing bibliographic data, redescription mining can shed light on the communities of researchers that make up the field, arranged by specialty area, complementing other approaches to publication network and scientific collaboration analysis [New01a, New01b, MBKN13].

Discussion. The redescriptions found with either setting share strong similarities. The use of actual counts of publications rather than Boolean indicators allows for finer tuning of the queries. This achieves higher accuracies but also results in multiple redescriptions with support at the acceptability threshold. In this example, the minimum support threshold was set to 10. In the real-valued setting we obtained many redescriptions with support exactly 10, unlike in the Boolean setting. This points to a greater sensibility of the algorithm to such thresholds, due to its increased capacity to adjust the queries, which needs to be controlled to prevent the generation of spurious results.

Because an exhaustive exploration of the space of queries is not feasible, our algorithms rely on heuristics for finding the top redescriptions. In Article I, we present experiments with synthetic data showing that the REREMi algorithm is able to recover planted redescriptions. However, despite this empirical evidence, the approach is not guaranteed to find the strongest patterns in general.

6.3 *Bioclimatic Niches*

As a second illustration, we consider an application of redescription mining in the domain of biology, namely, to find bioclimatic envelopes. In biology, the bioclimatic constraints that must be met for a certain species to survive constitute that species' bioclimatic envelope or niche (here restricted only to environmental variables in the Grinnellian sense of the term [Gri17], not inter-species competition or such).

Dataset. We consider a dataset, denoted as *Bio*, characterizing the climate and fauna of Europe. Our objects consist of spatial areas of Europe, roughly squares of 50 km sides.² The data itself is composed from two publicly available data bases: the European mammal atlas [MJAB⁺99] and the Worldclim climate data [HCP⁺05]. The mammals data contains

²For details of the grid see www.luomus.fi/english/botany/afe/index.html.

Table 6.4: Sample of redescrptions from Bio mined with REREMi. t_X^{\min} , t_X^{\max} , and t_X^{avg} stand for minimum, maximum, and average temperature of month X in degrees Celsius, and p_X^{avg} stands for average precipitation of month X in millimeters.

q_L	q_R	J	$ E_{1,1} $
(3) Polar bear	$[-4.5 \leq t_{\text{Oct}}^{\max} \leq -1.0]$	0.973	36
(4) Polar bear	$[1.0 \leq t_{\text{Sep}}^{\max} \leq 3.5]$	0.973	36
(7) Wood mouse \vee Azores Noctule	$(([3.0 \leq t_{\text{Mar}}^{\max}] \wedge [9.8 \leq t_{\text{Oct}}^{\max}]) \vee [9.7 \leq t_{\text{Jul}}^{\max} \leq 14.0]) \wedge [0.4765 \leq t_{\text{Oct}}^{\text{avg}} \leq 19.5860]$	0.842	1703
(9) Bank Vole \vee Steppe Mouse \vee Northern Red-backed Vole \vee Harbor Seal	$[-9.2 \leq t_{\text{Dec}}^{\max} \leq 12.8000] \wedge [7.1556 \leq t_{\text{Aug}}^{\text{avg}} \leq 23.089] \wedge [34.714 \leq p_{\text{Jun}}^{\text{avg}}] \wedge [47.625 \leq p_{\text{Aug}}^{\text{avg}}]$	0.838	1696
(14) Wood mouse	$(([3.0 \leq t_{\text{Mar}}^{\max}] \wedge [4.2 \leq t_{\text{Nov}}^{\max}]) \vee [9.7 \leq t_{\text{Jul}}^{\max} \leq 13.2]) \wedge [-5.4944 \leq t_{\text{Dec}}^{\text{avg}} \leq 13.133]$	0.828	1685
(23) Cape Hare \vee European Hare \vee Algerian Mouse	$([15.208 \leq t_{\text{Jul}}^{\text{avg}} \leq 26.36] \wedge [-12.9 \leq t_{\text{Dec}}^{\min} \leq 8.9]) \vee [10.4 \leq t_{\text{Sep}}^{\text{avg}} \leq 12.187] \vee [112.75 \leq p_{\text{Apr}}^{\text{avg}}]$	0.808	1677
(30) Mountain Hare	$([t_{\text{Sep}}^{\text{avg}} \leq 12.992] \wedge [7.6 \leq t_{\text{Sep}}^{\max} \leq 17.2] \wedge [13.5 \leq t_{\text{Jul}}^{\max} \leq 22.5]) \vee [81.111 \leq p_{\text{Apr}}^{\text{avg}} \leq 81.222]$	0.782	688
(39) Balkan Snow Vole \vee Field Vole \vee Azores Noctule	$[11.5 \leq t_{\text{Jun}}^{\max} \leq 24.5] \wedge [12.2 \leq t_{\text{Jul}}^{\max} \leq 26.7] \wedge [34.714 \leq p_{\text{Jun}}^{\text{avg}} \leq 175.0] \wedge [42.0 \leq p_{\text{Sep}}^{\text{avg}} \leq 183.06]$	0.751	1343
(54) Harvest Mouse \wedge European Mole	$[-0.3 \leq t_{\text{Apr}}^{\min} \leq 8.8] \wedge [19.4 \leq t_{\text{Aug}}^{\max} \leq 27.2] \wedge [45.417 \leq p_{\text{Jun}}^{\text{avg}}] \wedge [48.75 \leq p_{\text{Aug}}^{\text{avg}} \leq 126.56]$	0.677	774
(56) (Daubenton's Bat \wedge Eurasian Pygmy Shrew) \vee Balkan Snow Vole	$([t_{\text{Nov}}^{\min} \leq 6.2] \wedge [14.0 \leq t_{\text{May}}^{\max} \leq 20.6] \wedge [48.75 \leq p_{\text{Aug}}^{\text{avg}} \leq 165.6500]) \vee [1.025 \leq t_{\text{Apr}}^{\text{avg}} \leq 1.0917]$	0.669	870

presence/absence information of mammal species in Europe, and the aggregated climate data contains minimum, average, and maximum monthly temperatures as well as average monthly precipitation.

Results. Table 6.4 presents redescrptions mined from this dataset with the REREMi algorithm. Since the objects considered in this task correspond to geographic locations, the redescrptions can be naturally plotted on maps. The maps generated with the SIREN interface for these sample results are shown in Figure 6.1.

These redescrptions accurately characterize areas, often contiguous, that share similar climatic conditions and constitute the habitat of particular species. For instance, an area spreading from the Pyrenees to the Baltic states is described in redescription 6.4(54) as the region where the harvest mouse and the European mole cohabit and where a conjunction of temperatures and precipitation conditions is encountered.

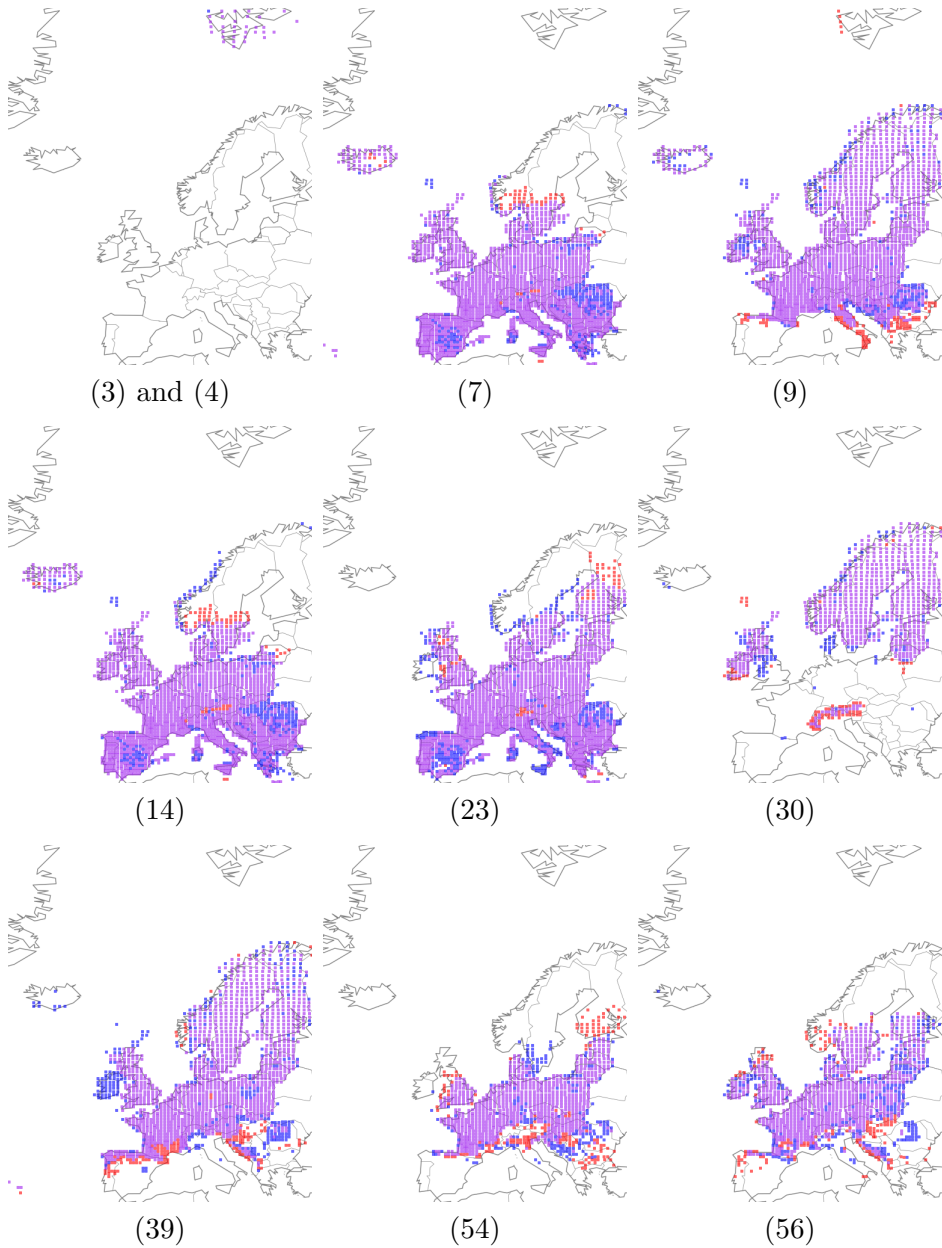


Figure 6.1: Support of the redescriptions on Bio shown in Table 6.4. For each redescription, purple, red and blue squares indicate areas where both queries hold ($E_{1,1}$), only the left query holds ($E_{1,0}$) and only the right query holds ($E_{0,1}$), respectively.

Redescriptions of this sort define the bioclimatic niche of species taken individually or in combination. Finding such niches is an important problem in biology that, for instance, can help predict the impact of global warming [PD03]. Redescription mining allows to study more complex combinations of species than would be otherwise possible with a laborious semi-automatic process requiring the manual selection of species.

Discussion. Notice that the mammals data is Boolean while the climate data is real-valued. Contrarily to the DBLP data, the range of the variables is not limited to small natural numbers and the amount of distinct occurring values can be as large as the number of objects. The algorithm determines the optimal discretization on-the-fly and the bounds are fixed to the shortest interval optimizing the accuracy. This can result in interval bounds with many decimals, up to the data precision, making the queries difficult to read. Therefore, taking into account a criterion that favors simple interval bounds could be considered, for the benefit of interpretability.

We note a drift towards redescriptions with largest allowed support cardinality, with a substantial part of the redescriptions covering large areas of the map. As with the bibliographic data, this is again partly an effect of the greater capacity to adjust the queries by tuning the interval bounds. Here, it manifests as fairly complex climatic queries, paired with disjunctions of possibly unrelated species. The resulting redescriptions, while fairly accurate, might be of little interest to biologists. This issue is mitigated by the use of p -values to check the significance of the results, but not adequately resolved yet.

In addition, if the discretization of a given variable at any step of the algorithm is not controlled, very similar candidates could be generated, affecting the diversity of the results.

To summarize, a more complex query language not only increases the search space but also calls for reinforced selection methods. Adding more parameters is not a viable solution. Thus, more holistic approaches, based for example on information theory, need to be explored.

6.4 *Political Candidates Profiles*

With the third illustration we turn to the field of politics.

Dataset. The dataset, dubbed **Elections**, consists of information about the candidates that participated in the 2011 Finnish parliamentary elections. The data was collected from www.vaalikone.fi, the “election en-

Table 6.5: Sample of redescriptions from `Elections` mined with `REREMI`.

(1)	q_L = party: National Coalition q_R = <i>Question</i> : Should authorization be granted for the replacement of the two nuclear reactors at the Loviisa power plant? <i>Answer</i> : Yes. \wedge <i>Question</i> : Which of the following statements best describes your views regarding Finland's financial support to other euro countries in the crisis? <i>Answer</i> : Supporting the euro is in the interest of Finland itself.	$J=0.444$ $ E_{1,1} =152$
(4)	q_L = party: Communist Party q_R = <i>Question</i> : What is your opinion on service outsourcing by local authorities to private companies? <i>Answer</i> : Outsourced services must be returned to the municipalities. \wedge <i>Question</i> : Should Finland apply for NATO membership? <i>Answer</i> : Never. \wedge <i>Question</i> : What do you think of the current Finnish immigration policy? <i>Answer</i> : Too tight.	$J=0.365$ $ E_{1,1} =57$
(5)	q_L = Municipal Representative q_R = <i>Question</i> : Recently, Russia banned property ownership by foreigners. On the other hand, Russians have bought thousands of properties in Finland. What should be done? <i>Answer</i> : Aquisition should be limited until there is reciprocity.	$J=0.352$ $ E_{1,1} =441$
(7)	q_L = county: Uusimaa q_R = <i>Question</i> : State tax revenue is equalized among municipalities, so that the money is transferred from the better-off to poorer municipalities. The largest contributors are Helsinki and Espoo, with approximately 500 million euros transfered to poorer municipalities this year. How should the system react? <i>Answer</i> : The metropolitan area should be able to keep a greater percentage of their income tax	$J=0.340$ $ E_{1,1} =200$
(9)	q_L = gender: female q_R = <i>Question</i> : In Finland, child benefit is paid for each child until the age of 17, regardless of parental income. Should the sytem be modified? <i>Importance</i> : High.	$J=0.321$ $ E_{1,1} =353$
(27)	q_L = county: Lapland q_R = <i>Question</i> : Which three countries should Finland befriend first if it were on Facebook? <i>Answer</i> : Sweden, Norway and Russia.	$J=0.086$ $ E_{1,1} =22$

gine” of the Finnish newspaper Helsingin Sanomat and made publicly available.³ One view contains candidate personal profile attributes, such as party, age, and education, while the answers provided to 30 multiple-choice questions and assigned importance form the other view. More precisely, for each of the thirty questions, the candidates were asked to choose the answer that best matched their opinion from a set of suggestions. In addition, they indicated what importance they attach to each question, that is, whether they consider the issue to be of high, medium or low importance. Each attribute-value of the profiles and each distinct question-answer and question-importance pair is represented by a Boolean attribute.

Results. Tables 6.5 and 6.6 present a sample of patterns mined from this dataset with the `REREMI` and `TRANSLATOR` algorithms respectively. In both cases, the queries consist of monotone conjunctions of Boolean predicates, and we indicate the accuracy and support for each example. In

³Data retrieved from <http://blogit.hs.fi/hsnext/hsn-vaalikone-on-nyt-avointa-tietoa> in May 2012.

Table 6.6: Sample of rules from Elections mined with TRANSLATOR.

(1)	$q_L = \text{party: National Coalition} \leftrightarrow$ $q_R = \text{Question: Taxes have increased quickly in Finland since the second half of the 90's. How should this be viewed? Importance: Medium. } \wedge$ <i>Question:</i> Should authorization be granted for the replacement of the two nuclear reactors at the Loviisa power plant? <i>Answer:</i> Yes. \wedge <i>Question:</i> Which of the following statements best describes your views regarding Finland's financial support to other euro countries in the crisis? <i>Answer:</i> Supporting the euro is in the interest of Finland itself. \wedge <i>Question:</i> Which of the following statements most closely matches your vision regarding the global Financial Transaction Tax (FTT) proposed by the EU? <i>Answer:</i> The EU should adopt an FTT, even if the rest of the world does not participate in the system. \wedge <i>Question:</i> Legislation regarding arms was tightened in the autumn of 2010, raising the age limit for handgun permits to 20 years. What should be done? <i>Answer:</i> The legislation is alright now. \wedge <i>Question:</i> What is your opinion on service outsourcing by local authorities to private companies? <i>Answer:</i> Outsourcing may be increased, municipalities should learn to improve the quality and prices of their services.	$J=0.211$ $ E_{1,1} =48$
(11)	$q_L = \text{county: Uusimaa} \leftarrow$ $q_R = \text{Question: State tax revenue is equalized among municipalities, so that the money is transferred from the better-off to poorer municipalities. The largest contributors are Helsinki and Espoo, with approximately 500 million euros transferred to poorer municipalities this year. How should the system react? Answer: The metropolitan area should be able to keep a greater percentage of their income tax}$	$J=0.340$ $ E_{1,1} =200$
(24)	$q_L = \text{party: Communist Party} \rightarrow$ $q_R = \text{Question: Should Finland apply for NATO membership? Answer: Never. } \wedge$ <i>Question:</i> What do you think of the current Finnish immigration policy? <i>Answer:</i> Too tight.	$J=0.249$ $ E_{1,1} =60$
(34)	$q_L = \text{party: Social Democratic Party} \wedge \text{Municipal Rep.} \rightarrow$ $q_R = \text{Question: Should Finland apply for NATO membership? Answer: Yes, but not at the beginning of the legislature. } \wedge$ <i>Question:</i> Recently, Russia banned property ownership by foreigners. On the other hand, Russians have bought thousands of properties in Finland. What should be done? <i>Answer:</i> Aquisition should be limited until there is reciprocity.	$J=0.111$ $ E_{1,1} =53$

addition, we indicate the direction of the rules found by the TRANSLATOR algorithm (\rightarrow , \leftarrow or \leftrightarrow), which are sorted in the order in which they were mined.

In general, the obtained patterns conform to the common understanding of the Finnish political landscape. Redescription 6.5(4), for instance, indicates that the Communist Party of Finland is opposed to the country entering NATO and to the outsourcing of municipal services, while it favors a more permissive immigration policy, opinions commonly attributed to that party.

In many countries, this kind of election recommendation engines, known as Voting Advice Application (VAA), are becoming a common feature at election times [CG10]. Simultaneously, election results, parliamentary activity, or government policies, for example, are made more widely accessible

under the action of open data movements.⁴ This offers potential for data analysis tools to promote political awareness among citizens and foster democratic participation. While earning increasing interest and recognition, these initiatives are still in their infancy and present a number of challenges [WNP09, EC13].

The principles underlying redescription mining are simple and interpreting the results requires neither expert training nor extensive domain knowledge. In contrast to more complex analysis methods, which might attract instinctive suspicions of opinion manipulation, this makes the approach suitable for applications targeted at the general public in this field.

Discussion. We observe that the results found by both methods are rather similar, a query pair found by one method often being a subpattern of one found by the other method or vice versa. For instance, redescription 6.5(1) is contained in rule 6.6(1) and vice versa with 6.5(4) and 6.6(24), while 6.5(7) and 6.6(11) have identical queries. However, TRANSLATOR finds directional rules. For instance, example 6.6(11) indicates that most candidates favorable to Helsinki and Espoo keeping more income tax come from the Uusimaa county, to which both municipalities belong, but that most candidates from that county do not share this opinion. This directional information is absent from REREMI's results.

The selection of patterns is a major difference between the two algorithms. REREMI focuses on finding redescriptions with high accuracy while TRANSLATOR emphasizes the quality of the entire collection of patterns with respect to compression ability. As a result, individual redescriptions found by the first method outmatch those found by the second method with respect to the Jaccard coefficient. The set of translation rules returned by the TRANSLATOR algorithm includes results that might have a low accuracy or a large p -value. For instance, the p -value of rule 6.6(34) equals 0.11 (see `pvalO`, in Section 5.1.1) and it would be rejected with most common significance levels. Still, this result set is purportedly more coherent as a whole than the one obtained with REREMI. In fact, the former allows to compress the data, although in this case the compression ratio is a modest 93%, while the latter actually inflates it with redundancies, with a compression ratio reaching 101%.

As a further advantage, the compression-based method does not require tuning any parameters other than, possibly, a minimum support threshold.

⁴See <http://openelectiondata.org>, <http://www.itsyourparliament.eu> or <http://opengovernmentdata.org>, for example.

However, it is not as scalable as the greedy search and remains to be adapted to more general query languages.

6.5 *Biomedical Ontology*

As the last piece of this exposition, we look at relational redescription mining in the biomedical domain.

Dataset. The UMLS dataset, obtained from the Alchemy repository,⁵ characterizes the relations between biomedical concepts in terms of the Unified Medical Language System ontology. It can be represented as a network of 135 nodes and 4181 edges. In contrast to the previous examples, this is a relational dataset, containing information about the links between different objects, here biomedical concepts. This particular dataset does not contain information about individual nodes, i.e. there are no node attributes. In this setting, our queries characterize pairs of objects in term of the relations that connect them. The redescrptions we are looking for are pairs of such queries, expressed over disjoint sets of edge attributes, that characterize roughly the same object pairs.

Results. Redescrptions from the UMLS dataset mined with the ARRM algorithm are shown in Table 6.7. Alternatively to the graphical representation used in that table, the queries can be written in full textual form, as a conjunction of relational predicates. For instance, redescription 6.7(6) can be written as the following pair of queries:

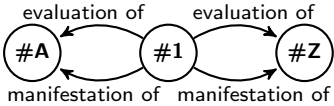
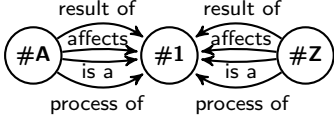
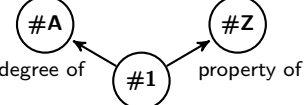
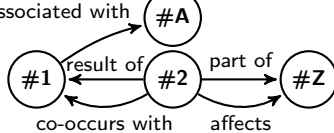
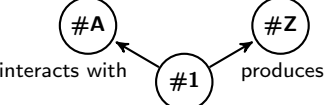
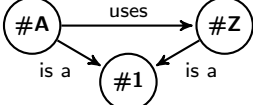
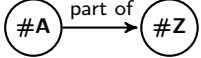
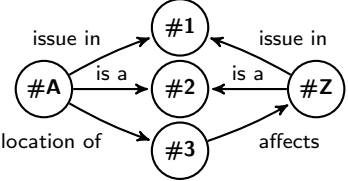
$$\begin{aligned}
 q_{\mathbf{L}}(\#A, \#Z) &:- \epsilon_{\text{degree of}}(\#1, \#A), \epsilon_{\text{property of}}(\#1, \#Z). \\
 q_{\mathbf{R}}(\#A, \#Z) &:- \epsilon_{\text{associated with}}(\#1, \#A), \epsilon_{\text{co-occurs with}}(\#2, \#1), \\
 &\quad \epsilon_{\text{result of}}(\#2, \#1), \epsilon_{\text{part of}}(\#2, \#Z), \epsilon_{\text{affects}}(\#2, \#Z).
 \end{aligned}$$

There are 34 pairs of data objects that map to nodes $\#A$ and $\#Z$ of query $q_{\mathbf{L}}$, and exactly these pairs also can be substituted for nodes $\#A$ and $\#Z$ of query $q_{\mathbf{R}}$. For instance,

$$\begin{aligned}
 &\{\#A/\text{Organism Attribute}, \#1/\text{Clinical Attribute}, \#Z/\text{Amphibian}\}, \text{ and} \\
 &\{\#A/\text{Organism Attribute}, \#1/\text{Anatomical Abnormality}, \\
 &\quad \#2/\text{Congenital Abnormality}, \#Z/\text{Amphibian}\}
 \end{aligned}$$

⁵Data retrieved from <http://alchemy.cs.washington.edu/data/umls> in Oct. 2012.

Table 6.7: Sample of redescrptions from UMLS mined with ARRM.

(1) $q_L(\#A, \#Z)$	$q_R(\#A, \#Z)$ $J = 1$ $ E_{1,1} = 182$
	
(6) $q_L(\#A, \#Z)$	$q_R(\#A, \#Z)$ $J = 1$ $ E_{1,1} = 34$
	
(12) $q_L(\#A, \#Z)$	$q_R(\#A, \#Z)$ $J = 0.833$ $ E_{1,1} = 40$
	
(15) $q_L(\#A, \#Z)$	$q_R(\#A, \#Z)$ $J = 0.649$ $ E_{1,1} = 170$
	

are substitutions for query $q_L(\#A, \#Z)$ and $q_R(\#A, \#Z)$, respectively, both corresponding to the object pair (Organism Attribute, Amphibian). Hence, this pair of queries forms a perfect relational redescription, i.e. a redescription of accuracy one, with a support of cardinality 34.

Ontologies are structured formal representations of the concepts within a domain and their relations. Because they define the semantics of the data, ontologies have an important role in the semantic web. In comparison, the schema of a database defines a practical representation of the data in order to allow efficient storage and retrieval, irrespective of meaning. A common problem in order to share information across sources, is to find correspondences between the occurring concepts.

Relational redescription mining provides expressive means to capture nearly equivalent connection patterns between objects in a heterogeneous network. This goes beyond current approaches in ontology alignment [SAS11] and schema matching [SE05] that typically aim to identify one-to-one mappings of concepts or relations.

More generally, relational redescription mining can help explore, understand and maintain complex relational datasets. For instance, it might be useful in large knowledge bases that store millions of objects and relations [ABK⁺07, CBK⁺10, SKW07], whose volume makes manual curation impossible.

Discussion. Compared to the propositional setting, while spurious redescrptions are less likely to arise in the relational setting, especially from sparse datasets, finding multiple nearly equivalent redescrptions is a more acute issue. For instance, an added relation or intermediate variable might respectively reduce or increase the number of satisfying substitutions for a query without affecting its support. More simply, a query and subqueries can be satisfied by the same substitutions. For instance, removing relations `is a` and `process of` from the right-hand side query of redescription 6.7(1) does not modify its support. Selecting and filtering such similar patterns, in other words identifying the best representative, is necessary to ensure the quality of the results and requires a tailored solution.

Furthermore, real-world datasets and especially real-world networks are often incomplete and might contain uncertain data, because of the data collection process or the nature of the information. In particular, some data can be inexact and the doubts about the actual values might be modelled as probabilities associated with the data. The problem of handling uncertainties has been considered in relational learning and in other data mining tasks [RKT07, PGdK09, Agg09]. Adapting such approaches to mine redescrptions in the presence of partial and of probabilistic information is thus an important direction for further investigations, in order to increase the practical applicability of the approach.

Chapter 7

Conclusions

The unifying theme of the present thesis is the data analysis task called redescription mining. It aims to find objects that admit multiple shared descriptions and, vice versa, to find distinct common characterizations for a set of objects.

Redescription mining is a task for exploratory data analysis. It provides insight into the data by means of pairs of expressive and interpretable queries, relating different views on the objects. It shares similarities with other data mining tasks like exceptional model mining and subgroup discovery, but is characterized by its symmetrical approach.

In this thesis, we extended redescription mining beyond propositional Boolean queries to real-valued attributes and relational queries. We designed the REREMi algorithm to mine redescriptions over nominal and real-valued attributes natively and introduced the ARRM relational redescription mining algorithm.

We also proposed two approaches for selecting high quality redescriptions. The SIREN interface for mining and visualizing redescriptions, on one hand, enables the user to interactively adjust the selection criteria. The TRANSLATOR algorithm, on the other hand, provides a principled solution to the selection problem. It is a parameter-free compression-based algorithm that encodes one side of the data using the other side, and vice versa, thereby capturing the associations across the two sides.

While its underlying principle is simple and intuitive, we showed that redescription mining constitutes a powerful tool for data exploration, potentially applicable in a large variety of domains.

Further developing the algorithms presented here and integrating them together should help alleviate current shortcomings such as spurious or redundant results and the absence of any analytical guarantee on finding the best redescrptions occurring in the data. The scalability of the algorithms and their generalization to varying numbers of views also demand investigation.

Specifically, devising methods with sound theoretic foundations and sufficient flexibility to select redescrptions, for instance drawing on recent advances in significance testing for data mining [Oja11, Han12, Vuo12] or modelling the information content of redescrptions in the subjective interestingness framework [DB11a], constitutes a major direction for future research.

Besides, uncertainties are inherent to most real-world scenarios. To promote its applicability in realistic situations, redescription mining should thus be enabled to account for uncertainties in the data, possibly by adapting techniques developed for other data analysis tasks [Agg09].

Finally, the actual value of our proposed methods can only be assessed by putting them to use, in collaboration with experts and practitioners of the respective fields.

References

- [ABK⁺07] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. G. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, ISWC/ASWC'07* (Busan, Korea), volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007.
- [Agg09] Aggarwal, C. C. *Managing and Mining Uncertain Data*, volume 35 of *Advances in Database Systems*. Springer, 2009.
- [AIS93] Agrawal, R., Imielinski, T., and Swami, A. N. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD'93* (Washington, D.C., U.S.A.), pages 207–216. ACM Press, 1993.
- [AS94] Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference on Very Large Data Bases, VLDB'94* (Santiago de Chile, Chile), pages 487–499. Morgan Kaufmann, 1994.
- [BB00] Boulicaut, J.-F., and Bykowski, A. Frequent closures as a concise representation for binary data mining. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications, PADKK'00* (Kyoto, Japan), volume 1805 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2000.
- [BDRM05] Boulicaut, J.-F., De Raedt, L., and Mannila, H., editors. *Revised Selected Papers of the European Workshop on Inductive Databases and Constraint Based Mining*, volume 3848 of *Lecture Notes in Computer Science*. Springer, 2005.

- [BKM99] Boulicaut, J.-F., Klemettinen, M., and Mannila, H. Modeling KDD processes within the inductive database framework. In *Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery, DaWaK'99* (Florence, Italy), volume 1676 of *Lecture Notes in Computer Science*, pages 293–302. Springer, 1999.
- [BM98] Blum, A., and Mitchell, T. M. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory, COLT'98* (Madison, Wisconsin, U.S.A.), pages 92–100. ACM, 1998.
- [BMS97] Brin, S., Motwani, R., and Silverstein, C. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD'97* (Tucson, Arizona, U.S.A.), pages 265–276. ACM Press, 1997.
- [BS04] Bickel, S., and Scheffer, T. Multi-view clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM'04* (Brighton, UK), pages 19–26. IEEE Computer Society, 2004.
- [BS05] Bickel, S., and Scheffer, T. Estimation of mixture models using co-EM. In *Proceedings of the 16th European Conference on Machine Learning, ECML'05* (Porto, Portugal), volume 3720 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 2005.
- [CBK⁺10] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, Jr., E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI'10* (Atlanta, Georgia, U.S.A.). AAAI Press, 2010.
- [CG02] Calders, T., and Goethals, B. Mining all non-derivable frequent itemsets. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD'02* (Helsinki, Finland), volume 2431 of *Lecture Notes in Computer Science*, pages 74–85. Springer, 2002.
- [CG10] Cedroni, L., and Garzia, D., editors. *Voting Advice Applications in Europe: The State of the Art*. ScriptaWeb, Naples, 2010.

- [Cod70] Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
- [CRB04] Calders, T., Rigotti, C., and Boulicaut, J.-F. A survey on condensed representations for frequent sets. In Boulicaut et al. [BDRM05], pages 64–80.
- [DB11a] De Bie, T. An information theoretic framework for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'11* (San Diego, California, U.S.A.), pages 564–572. ACM, 2011.
- [DB11b] De Bie, T. Maximum entropy models and subjective interestingness: An application to tiles in binary databases. *Data Mining and Knowledge Discovery*, 23(3):407–446, 2011.
- [DEDC96] Deransart, P., Ed-Dbali, A., and Cervoni, L. *Prolog: the Standard: Reference Manual*. Springer Verlag, 1996.
- [DEV12] Dinh, Q.-T., Exbrayat, M., and Vrain, C. A link-based method for propositionalization. In *Late Breaking Papers of the 22nd International Conference on Inductive Logic Programming, ILP'12* (Dubrovnik, Croatia), volume 975 of *CEUR Workshop Proceedings*, pages 10–25, 2012.
- [DR02] De Raedt, L. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2002.
- [DR08] De Raedt, L. *Logical and Relational Learning*. Springer, 2008.
- [DRGN08] De Raedt, L., Guns, T., and Nijssen, S. Constraint programming for itemset mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'08* (Las Vegas, Nevada, U.S.A.), pages 204–212. ACM, 2008.
- [DRR04] De Raedt, L., and Ramon, J. Condensed representations for inductive logic programming. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning, KR'04* (Whistler, British Columbia, Canada), pages 438–446. AAAI Press, 2004.

- [DT99] Dehaspe, L., and Toivonen, H. Discovery of frequent DATA-LOG patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.
- [EC13] Evans, A. M., and Campos, A. Open government initiatives: Challenges of citizen participation. *Journal of Policy Analysis and Management*, 32(1):172–185, 2013.
- [Edg95] Edgington, E. S. *Randomization Tests (3rd Edition)*, volume 147. CRC Press, 1995.
- [EMS⁺94] Esposito, F., Malerba, D., Semeraro, G., Brunk, C., and Pazzani, M. Traps and pitfalls when learning logical definitions from relations. In *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems, ISMIS'94* (Charlotte, North Carolina, U.S.A.), volume 869 of *LNCS*, pages 376–385. Springer, 1994.
- [FHM⁺05] Farquhar, J. D. R., Hardoon, D. R., Meng, H., Shawe-Taylor, J., and Szedmák, S. Two view learning: SVM-2K, theory and practice. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems, NIPS'05* (Vancouver, British Columbia, Canada), 2005.
- [Fis38] Fisher, R. *Statistical Methods for Research Workers*. Biological monographs and manuals. Oliver and Boyd, 1938.
- [GDJE12] Günnemann, S., Dao, P., Jamali, M., and Ester, M. Assessing the significance of data mining results on graphs with feature vectors. In *Proceedings of the 12th IEEE International Conference on Data Mining, ICDM'12* (Brussels, Belgium), pages 270–279. IEEE Computer Society, 2012.
- [GMM08] Gallo, A., Miettinen, P., and Mannila, H. Finding subgroups having several descriptions: Algorithms for re-description mining. In *Proceedings of the 8th SIAM International Conference on Data Mining, SDM'08* (Atlanta, Georgia, U.S.A.), pages 334–345. SIAM, 2008.
- [GMMT07] Gionis, A., Mannila, H., Mielikäinen, T., and Tsaparas, P. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data*, 1(3), 2007.

- [GNDR13] Guns, T., Nijssen, S., and De Raedt, L. k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):402–418, 2013.
- [GNZDR11] Guns, T., Nijssen, S., Zimmermann, A., and De Raedt, L. Declarative heuristic search for pattern set mining. In *Proceedings of the 2011 IEEE International Conference on Data Mining Workshops, ICDMW'11* (Vancouver, British Columbia, Canada), pages 1104–1111. IEEE Computer Society, 2011.
- [Goe03] Goethals, B. Survey on frequent pattern mining. Manuscript, 2003.
- [GR00] Garofalakis, M. N., and Rastogi, R. Scalable data mining with model constraints. *SIGKDD Explorations*, 2(2):39–48, 2000.
- [Gri17] Grinnell, J. The niche-relationships of the California Thrasher. *The Auk*, 34(4):427–433, 1917.
- [Grü07] Grünwald, P. D. *The Minimum Description Length Principle*. Adaptive computation and machine learning series. MIT Press, 2007.
- [Han12] Hanhijärvi, S. *Multiple Hypothesis Testing in Data Mining*. PhD thesis, Aalto University School of Science, Department of Information and Computer Science, Finland, 2012.
- [HCP⁺05] Hijmans, R. J., Cameron, S., Parra, L., Jones, P., and Jarvis, A. Very high resolution interpolated climate surfaces for global land areas. *International Journal of Climatology*, 25:1965–1978, 2005. www.worldclim.org.
- [HCXY07] Han, J., Cheng, H., Xin, D., and Yan, X. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.
- [HGP09] Hanhijärvi, S., Garriga, G. C., and Puolamäki, K. Randomization techniques for graphs. In *Proceedings of the 2009 SIAM International Conference on Data Mining, SDM'09* (Sparks, Nevada, U.S.A.), pages 780–791. SIAM, 2009.
- [HK00] Han, J., and Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

- [HMS01] Hand, D., Mannila, H., and Smyth, P. *Principles of Data Mining*. Adaptive computation and machine learning series. MIT Press, 2001.
- [HPYM04] Han, J., Pei, J., Yin, Y., and Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.
- [IM96] Imielinski, T., and Mannila, H. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
- [JMR08] Jin, Y., Murali, T. M., and Ramakrishnan, N. Compositional mining of multirelational biological datasets. *ACM Transactions on Knowledge Discovery from Data*, 2(1), 2008.
- [KBC10] Khiari, M., Boizumault, P., and Crémilleux, B. Constraint programming for mining n-ary patterns. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming, CP'10* (St. Andrews, Scotland, UK), volume 6308 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2010.
- [KK08] Klami, A., and Kaski, S. Probabilistic approach to detecting dependencies between data sets. *Neurocomputing*, 72(1-3):39–46, 2008.
- [KMR⁺94] Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. I. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the 1994 ACM International Conference on Information and Knowledge Management, CIKM'94* (Gaithersburg, Maryland, U.S.A.), pages 401–407. ACM, 1994.
- [Kum07] Kumar, D. *Redescription Mining: Algorithms and Applications in Bioinformatics*. PhD thesis, Department of Computer Science, Virginia Tech, U.S.A., 2007.
- [KZ09] Kuzelka, O., and Zelezný, F. Block-wise construction of acyclic relational features with monotone irreducibility and relevancy properties. In *Proceedings of the 26th Annual International*

- Conference on Machine Learning, ICML'09* (Montreal, Quebec, Canada), volume 382 of *ACM International Conference Proceeding Series*, pages 569–576. ACM, 2009.
- [LFK08] Leman, D., Feelders, A., and Knobbe, A. J. Exceptional model mining. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD'08, Part II* (Antwerp, Belgium), volume 5212 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2008.
- [LR05] Lehmann, E. E. L., and Romano, J. P. *Testing Statistical Hypotheses*. Springer Science+ Business Media, 2005.
- [Man00] Mannila, H. Theoretical frameworks for data mining. *SIGKDD Explorations*, 1(2):30–32, 2000.
- [MBKN13] Martin, T., Ball, B., Karrer, B., and Newman, M. E. J. Coauthorship and citation in scientific publishing. *arXiv preprint arXiv:1304.0473*, 2013.
- [MDR94] Muggleton, S., and De Raedt, L. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
- [MJAB⁺99] Mitchell-Jones, A. J., Amori, G., Bogdanowicz, W., Krystufek, B., Reijnders, P., Spitzenberger, F., Stubbe, M., Thissen, J., Vohralik, V., and Zima, J. *The Atlas of European Mammals*. Academic Press, London, 1999. www.european-mammals.org.
- [MS98] Megiddo, N., and Srikant, R. Discovering predictive association rules. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD'98* (New York City, New York, U.S.A.), pages 274–278. AAAI Press, 1998.
- [MT96] Mannila, H., and Toivonen, H. Multiple uses of frequent sets and condensed representations (extended abstract). In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96* (Portland, Oregon, U.S.A.), pages 189–194. AAAI Press, 1996.
- [MT97] Mannila, H., and Toivonen, H. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

- [MTV94] Mannila, H., Toivonen, H., and Verkamo, A. I. Efficient algorithms for discovering association rules. In *Proceedings of the 1994 AAAI Workshop on Knowledge Discovery in Databases, KDD'94, Technical Report WS-94-03* (Seattle, Washington, U.S.A.), pages 181–192. AAAI Press, 1994.
- [Mug95] Muggleton, S. Inverse entailment and progol. *New Generation Computing*, 13(3&4):245–286, 1995.
- [New01a] Newman, M. E. J. Scientific collaboration networks. i. network construction and fundamental results. *Physical Review E*, 64:016131, 2001.
- [New01b] Newman, M. E. J. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, 64:016132, 2001.
- [NG00] Nigam, K., and Ghani, R. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 2000 ACM International Conference on Information and Knowledge Management, CIKM'00* (McLean, Virginia, U.S.A.), pages 86–93. ACM, 2000.
- [NLHP98] Ng, R. T., Lakshmanan, L. V. S., Han, J., and Pang, A. Exploratory mining and pruning optimizations of constrained association rules. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, SIGMOD'98* (Seattle, Washington, U.S.A.), pages 13–24. ACM Press, 1998.
- [NLW09] Novak, P. K., Lavrac, N., and Webb, G. I. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.
- [Oja11] Ojala, M. *Randomization Algorithms for Assessing the Significance of Data Mining Results*. PhD thesis, Aalto University School of Science, Department of Information and Computer Science, Finland, 2011.
- [PBTL99] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory, ICDT'99* (Jerusalem, Israel), volume 1540 of *Lecture Notes in Computer Science*, pages 398–416. Springer, 1999.

- [PCY95] Park, J. S., Chen, M.-S., and Yu, P. S. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, SIGMOD'95* (San Jose, California, U.S.A.), pages 175–186. ACM Press, 1995.
- [PD03] Pearson, R. G., and Dawson, T. P. Predicting the impacts of climate change on the distribution of species: Are bioclimate envelope models useful? *Global Ecology and Biogeography*, 12:361–371, 2003.
- [PGdK09] Pei, J., Getoor, L., and de Keijzer, A., editors. *Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data, U'09*. ACM, 2009.
- [PR05] Parida, L., and Ramakrishnan, N. Redescription mining: Structure theory and algorithms. In *Proceedings of the 20th National Conference on Artificial Intelligence and the 7th Innovative Applications of Artificial Intelligence Conference, AAAI'05* (Pittsburgh, Pennsylvania, U.S.A.), pages 837–844. AAAI Press / The MIT Press, 2005.
- [PSM94] Piatetsky-Shapiro, G., and Matheus, C. J. The interestingness of deviations. In *Proceedings of the 1994 AAAI Workshop on Knowledge Discovery in Databases, KDD'94, Technical Report WS-94-03* (Seattle, Washington, U.S.A.), pages 25–36. AAAI Press, 1994.
- [PT98] Padmanabhan, B., and Tuzhilin, A. A belief-driven method for discovering unexpected patterns. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, KDD'98* (New York City, New York, U.S.A.), pages 94–100. AAAI Press, 1998.
- [PT00] Padmanabhan, B., and Tuzhilin, A. Small is beautiful: Discovering the minimal set of unexpected patterns. In *Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'00* (Boston, Massachusetts, U.S.A.), pages 54–63. ACM, 2000.
- [QCJ93] Quinlan, J. R., and Cameron-Jones, R. M. FOIL: A midterm report. In *Proceedings of the 4th European Conference on Machine Learning, ECML'93* (Vienna, Austria), volume 667

- of *Lecture Notes in Computer Science*, pages 3–20. Springer, 1993.
- [RKM⁺04] Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., and Helm, R. F. Turning CARTwheels: An alternating algorithm for mining redescrptions. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'04* (Seattle, Washington, U.S.A.), pages 266–275. ACM, 2004.
- [RKT07] Raedt, L. D., Kimmig, A., and Toivonen, H. ProbLog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI'07* (Hyderabad, India), pages 2462–2467, 2007.
- [RLT⁺12] Rettinger, A., Lösch, U., Tresp, V., d'Amato, C., and Fanizzi, N. Mining the semantic web — statistical learning for next generation knowledge bases. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.
- [SAS11] Suchanek, F. M., Abiteboul, S., and Senellart, P. PARIS: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment (PVLDB)*, 5(3):157–168, 2011.
- [SE05] Shvaiko, P., and Euzenat, J. A survey of schema-based matching approaches. *Journal on Data Semantics IV*, pages 146–171, 2005.
- [SKW07] Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW'07* (Banff, Alberta, Canada), pages 697–706. ACM, 2007.
- [SN09] Soberón, J., and Nakamura, M. Niches and distributional areas: Concepts, methods, and assumptions. *Proceedings of the National Academy of Sciences of the United States, PNAS*, 106(Supplement 2):19644–19650, 2009.
- [Sri07] Srinivasan, A. *The Aleph Manual*. University of Oxford, 2007.
- [ST95] Silberschatz, A., and Tuzhilin, A. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the*

- First International Conference on Knowledge Discovery and Data Mining, KDD'95* (Montreal, Quebec, Canada), pages 275–281. AAAI Press, 1995.
- [SVA97] Srikant, R., Vu, Q., and Agrawal, R. Mining association rules with item constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, KDD'97* (Newport Beach, California, U.S.A.), pages 67–73. AAAI Press, 1997.
- [TCP09] Tran, Q. T., Chan, C.-Y., and Parthasarathy, S. Query by output. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, SIGMOD'09* (Providence, Rhode Island, U.S.A.), pages 535–548. ACM Press, 2009.
- [TKOK11] Tripathi, A., Klami, A., Oresic, M., and Kaski, S. Matching samples of multiple views. *Data Mining and Knowledge Discovery*, 23(2):300–321, 2011.
- [TPB00] Tishby, N., Pereira, F., and Bialek, W. The information bottleneck method. *arXiv preprint arXiv:physics/0004057*, 2000.
- [UZT⁺09] Umek, L., Zupan, B., Toplak, M., Morin, A., Chauchat, J.-H., Makovec, G., and Smrke, D. Subgroup discovery in data sets with multi-dimensional responses: A method and a case study in traumatology. In *Proceedings of the 12th Conference on Artificial Intelligence in Medicine, AIME'09* (Verona, Italy), volume 5651 of *Lecture Notes in Computer Science*, pages 265–274, 2009.
- [VKKK12] Virtanen, S., Klami, A., Khan, S. A., and Kaski, S. Bayesian group factor analysis. *Journal of Machine Learning Research - Proceedings Track*, 22:1269–1277, 2012.
- [Vuo12] Vuokko, N. *Testing the Significance of Patterns with Complex Null Hypotheses*. PhD thesis, Aalto University School of Science, Department of Information and Computer Science, Finland, 2012.
- [VvLS11] Vreeken, J., van Leeuwen, M., and Siebes, A. Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.

- [Web07] Webb, G. I. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.
- [WNP09] Walgrave, S., Nuytemans, M., and Pepermans, K. Voting aid applications and the effect of statement selection. *West European Politics*, 32(6):1161–1180, 2009.
- [Yar95] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, ACL'95* (MIT, Cambridge, Massachusetts, U.S.A.), pages 189–196. Morgan Kaufmann Publishers / ACL, 1995.
- [ZH02] Zaki, M. J., and Hsiao, C.-J. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining, SDM'02* (Arlington, Virginia, U.S.A.). SIAM, 2002.
- [ZPT04] Zhang, H., Padmanabhan, B., and Tuzhilin, A. On the discovery of significant statistical quantitative rules. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'04* (Seattle, Washington, U.S.A.), pages 374–383. ACM, 2004.
- [ZR05] Zaki, M. J., and Ramakrishnan, N. Reasoning about sets using redescription mining. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'05* (Chicago, Illinois, U.S.A.), pages 364–373. ACM, 2005.

Articles

Article I

Esther Galbrun and Pauli Miettinen

**From Black and White to Full Color:
Extending Redescription Mining Outside the Boolean World**

In *Statistical Analysis and Data Mining*, 5(4):284–303, 2012.

DOI: <http://dx.doi.org/10.1002/sam.11145>

Copyright © Wiley, 2012. Reprinted with permission.

From Black and White to Full Color: Extending Redescription Mining Outside the Boolean World[†]

Esther Galbrun^{1*} and Pauli Miettinen^{2‡}

¹*Helsinki Institute for Information Technology (HIIT), Department of Computer Science, University of Helsinki, Finland*

²*Max-Planck Institute for Informatics, Saarbrücken, Germany*

Received 23 June 2011; revised 17 February 2012; accepted 8 March 2012

DOI:10.1002/sam.11145

Published online 10 April 2012 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: Redescription mining is a powerful data analysis tool that is used to find multiple descriptions of the same entities. Consider geographical regions as an example. They can be characterized by the fauna that inhabits them on one hand and by their meteorological conditions on the other hand. Finding such redescrptors, a task known as niche-finding, is of much importance in biology. Current redescription mining methods cannot handle other than Boolean data. This restricts the range of possible applications or makes discretization a pre-requisite, entailing a possibly harmful loss of information. In niche-finding, while the fauna can be naturally represented using a Boolean presence/absence data, the weather cannot. In this paper, we extend redescription mining to categorical and real-valued data with possibly missing values using a surprisingly simple and efficient approach. We provide extensive experimental evaluation to study the behavior of the proposed algorithm. Furthermore, we show the statistical significance of our results using recent innovations on randomization methods. © 2012 Wiley Periodicals, Inc. *Statistical Analysis and Data Mining* 5: 284–303, 2012

Keywords: redescription mining; bioclimatic niche finding; numerical data; missing data; data mining

1. INTRODUCTION

Finding multiple ways to characterize the same entities is a problem that appears in many areas of science. In medical sciences, for example, one typically wants to find a subset of patients sharing similar symptoms and similar genes. In biology, the bioclimatic constraints that must be met for a certain species to survive constitute that species' bioclimatic envelope (or niche¹), and finding such envelopes can help, e.g. to predict the results of global warming [2].

But this process is only semi-automatic. For instance, to find the bioclimatic envelopes, an expert first selects a species and then uses some method to find the envelope

for that particular species. More complex combinations of species, or even any combinations at all, are rarely studied, as manually iterating over all possible combinations would be far too laborious.

It is here where *redescription mining* comes to help. In redescription mining the input contains entities with two sets of characterizing variables. The task is to find a pair of queries, one query for both sets of variables, such that both queries describe (almost) the same set of entities. In niche-finding, the entities would be spatial locations, one set of variables would be the fauna and the other set would contain the bioclimatic variables. A very simple example of a redescription in this setting could say that the area where polar bears live is the area where March's mean temperature is between -16 and -11°C and May's mean temperature is between -3 and -7°C .

Until now, redescription mining algorithms (see ref. [3–6]) have not been able to handle other than Boolean data. Hence they have not been able to help in the aforementioned cases, not at least without some pre-processing.

Correspondence to: Esther Galbrun (esther.galbrun@cs.helsinki.fi)

[†] A preliminary version of this paper appeared in SDM 2011.

[‡] Part of this work was done when the author was with HIIT.

¹ The term *niche* is in this paper used in Grinnellian sense [1], considering only environmental variables, not inter-species competition or such.

The rest of this paper is organized as follows. The next two sections, Sections 2 and 3, present notation and definitions, and related work, respectively. We explain the basic structure of our algorithm in Section 4. In Section 5 we present various extensions to the basic algorithm, including methods for trading some accuracy for speed and handling missing values. The experimental evaluation spans Sections 6–8, with focus on studying the properties of the algorithm and its extensions, comparing it to other algorithms, and a real-world example of niche finding, respectively. Section 9 concludes the paper.

Contributions. In this paper, we extend redescription mining to categorical and real-valued data with an algorithm that efficiently computes the optimal discretization on-the-fly. The algorithm can handle missing data. We present experimental studies with synthetic and real-world data to verify that our algorithm scales and returns good results. We also assess the significance of our results by testing them against different null models. Our primary application for real-valued redescription mining is niche-finding, to which we present interesting and intuitive results. The proposed method is also applicable to other domains, e.g. medicine.

2. NOTATION AND DEFINITIONS

This paper considers redescrptors over two sets of variables, V_L and V_R . The set of entities is denoted by E . We will represent the data using two matrices, D_L and D_R . Both matrices have $|E|$ rows and D_i has $|V_i|$ columns. The value of $D_L(i, j)$ is the value of $v_j \in V_L$ for $e_i \in E$. If I is a set of row indices (or a characterizing vector thereof), $D(I, j)$ is the column j of D restricted to the rows in I . The data are 5-tuple $\mathcal{D} = (V_L, V_R, E, D_L, D_R)$. We identify variables in V_L and V_R with the corresponding columns in D_L and D_R when there is no risk of ambiguity.

We consider three types of variables: Boolean, categorical, and numerical (real-valued). If $v \in V$ is Boolean, we interpret the column corresponding to it as a truth value assignment for $e \in E$ in a natural way. If $v \in V$ is real-valued, we consider an interval $[a, b]$, and the truth value assignment induced by the relation $v \in [a, b]$. A special case of this is when v is categorical. Then we consider the relation $v = c$, where c is some category. We will denote these truth value assignments using Iverson notation: $[a \leq v \leq b]$ is the Boolean (column) vector that has 1 in the rows where $v \in [a, b]$, and 0 elsewhere; $[v = c]$ is defined analogously.

These truth assignments and their negations constitute the set of *literals* for variables in V . Notice that there are infinitely many intervals yielding the same truth

value assignment for some real-valued $v \in V$. To avoid ambiguity, we consider only the *shortest* interval yielding some truth value assignment. An exception to this is when leaving one side of the interval unbounded is equivalent. We then consider half-lines $(-\infty, b]$ or $[a, +\infty)$, respectively, but for the sake of brevity they are also called intervals. Notice that we can always reconstruct the interval given the data and the truth value assignment corresponding to the interval.

Literals can be combined with Boolean operators \wedge (and) and \vee (or). A *Boolean formula* is made by combining literals with Boolean operators. A *query over V* is a Boolean formula with literals of V . A *redescription R of $\mathcal{D} = (V_L, V_R, E, D_L, D_R)$* is a pair of queries (q_L, q_R) over V_L and V_R , respectively. For a redescription $R = (q_L, q_R)$, we use $V_L(R)$ to denote the variables of q_L ; $V_R(R)$ is defined analogously.

The support of a query q on \mathcal{D} , $\text{supp}_{\mathcal{D}}(q)$, is a set $\{e \in E : q \text{ is true for } e\}$. The support of a redescription $R = (q_L, q_R)$, $\text{supp}_{\mathcal{D}}(q_L, q_R)$, is the intersection of supports of q_L and q_R , $\text{supp}(q_L, q_R) = \text{supp}(q_L) \cap \text{supp}(q_R)$. We will omit the subscripts when they are clear from the context.

A redescription $R = (q_L, q_R)$ is *exact* if and only if $\text{supp}(q_L) = \text{supp}(q_R)$. If a redescription is not exact, it is approximate. The *accuracy* of a redescription $R = (q_L, q_R)$ is measured using the *Jaccard coefficient*

$$J(R) = J(q_L, q_R) = \frac{|\text{supp}(q_L, q_R)|}{|\text{supp}(q_L) \cup \text{supp}(q_R)|}.$$

Formally, redescription mining is defined as follows:

Problem 1 (Redescription Mining) *Given data $\mathcal{D} = (V_L, V_R, E, D_L, D_R)$ and a set of constraints C , find all redescrptions R_1, R_2, \dots of \mathcal{D} that satisfy constraints in C .*

We leave open the exact constraints in C for a while and will turn back to it in Subsection 5.1.

The formulation of redescription mining above assumes that the describing variables are partitioned into two sets, V_L and V_R , and looks for pairs of queries over these two sets, respectively. Formulations of redescription mining exist that do not include this requirement. Typically, they consider a single set of describing variables and search for pairs of queries, with the constraint that the two subsets of variables appearing in the queries of any pair be disjoint. The methods presented in this paper can be naturally adapted to that alternative formulation.

Our proposed methods could also be adapted to handle multiple data sets, i.e. settings with more than two sets of variables: V_A, V_B, \dots, V_N , where one looks for tuples of queries (q_A, q_B, \dots, q_N) over the different sets of variables, respectively.

3. RELATED WORK

3.1. Rule Discovery

A characterizing property of redescription mining is its ‘many views’ approach, i.e. it deals with entities that can be explained using different sets of variables. This approach, however, is not unique to redescription mining.

One of the most traditional ‘many views’ approaches is classification, though it is not typically considered as such. There the data gives one characterization of the entities and the class another. Mining a single query can be considered as a classification task. Fixing one query at a time gives us binary class labels and we try to find a good classifier to it.

Logical Analysis of Data [7] is a particular example among classification approaches in the presence of Boolean attributes and target. It aims at finding a perfect classifier of fixed form, e.g. a horn clause, a DNF, a CNF, or linear or quadratic Boolean formula.

Closer to the idea of redescription mining is Multi-label Classification [8], where the goal is to learn classifiers for conjunctions of labels. Perhaps the main difference to redescription mining is this restriction to conjunctions of classes. There is also a big difference in the goals: redescription mining is descriptive while multi-label classification is predictive.

The common aim of mining Emerging Patterns, Contrast Set Mining, and Subgroup Discovery is to find queries whose support is distributed very unevenly with respect to the target attribute.

Emerging Patterns [9] is targeted at Boolean data and uses monotone conjunctive queries, i.e. itemsets. The purpose is to find itemsets whose presence is statistically dependent on the positive or negative labeling of the objects. In the extreme case, the itemset would be present only in the positive example and would form a perfect classifier for the data at hand. However, this is not generally the case.

Contrast Set Mining [9] can be used with a nominal target attribute to identify a monotone conjunctive query that best discriminates between the objects from one class and the rest of the objects.

Subgroup Discovery [10] aims in a more general sense at finding a query such that the objects in the defined subgroup have atypical values for a target attribute, possibly ordinal or numerical, compared to other objects. This is extended to several target attributes in Exceptional Model mining. In that framework, defined by Leman et al. [11], and in its recent instance [12], one considers a model defined over the target attributes and tries to identify a subgroup of objects where the fitted model differs significantly from the model fitted to the rest of the data.

A related approach is presented by Garriga, Heikinheimo, and Seppänen [13]. It uses frequent itemsets on the binary

attributes to form a partition of the original entities such that for each subset it is possible to construct a specific model that fits well on the numerical attributes. In other words, this approach tries to partition the original data into subgroups.

Redescription mining differs from the above techniques in that it aims at simultaneously finding multiple descriptions of a subset of entities which is not previously specified, selecting the few relevant among a potentially large set of variables. In contrast to these methods, it does not have a set of describing features and target attributes, but rather several sets of describing variables. Yet, when there are two sets of describing attributes and a chosen query language, we can define a one-directional redescription problem. Queries can be built over one set of attributes, defining subgroups whose quality is measured in terms of how exactly and concisely they can be described by queries over the other set of attributes. In a sense, this can be loosely understood as a case of Exceptional Model Mining where the model is the chosen query language and fitting the model to a subgroup corresponds to finding as concise and exact a query for it over the target attributes as possible. The aim of redescription mining is then to solve this problem in both directions simultaneously.

Redescription mining was introduced by Ramakrishnan et al. [5], and has since attained continuous research interest (e.g. see refs. [3,4,6,14]). The approaches proposed for redescription mining have been based on various ideas, including decision trees [5,14], Karnaughmaps [6], co-clusters [4], and frequent itemsets [3,6].

The CARTWHEELS algorithm [5] is an alternating method that uses decision trees. One side of the redescription is fixed, giving binary labels for the entities and a decision tree over the other variables, i.e. a good classifier with respect to those labels, is constructed. At the next step, the labeling given by this decision tree is considered as the target and a new decision tree is built on the other set of variables. The branches of the two trees that correspond to positive labels form a pair of queries that can be considered as a redescription, as well as the pair of queries associated to the negative branches. This construction based on decision trees gives the redescriptions an atypical form. Consider the example of a tree of depth two, with the first level branching variable A and the second level branching variables B and C . Then, $(A \wedge B) \vee (\neg A \wedge C)$ is a good example of query obtained by joining branches of such a tree. The fact that the same variable occurs multiple times with and without negation can make the query difficult to interpret.

Gallo et al. [3] propose two approaches to redescription mining. The first one is based on mining frequent itemsets from both data sets separately and combining them together. The second one is a greedy method that forms the basis of our work. This will be discussed with further details.

A natural extension of redescription mining is *storytelling* [4], where the aim is to find consecutive redescriptors. That is, given data $\mathcal{D} = (V_L, V_R, E, D_L, D_R)$, the goal is to find queries $q_L^1, q_R^1, q_L^2, q_R^2, \dots$ such that consecutive pairs of queries $(q_L^1, q_R^1), (q_R^1, q_L^2), (q_L^2, q_R^2), \dots$ all form a valid non-exact redescription. We will not cover storytelling in this paper.

3.2. Data Discretization

Generalizing algorithms based on Boolean attributes to real-valued data has been a recurrent problem in data mining. Most solutions are based on some sort of pre-processing: typically categorical data is represented using one variable per category, and quantitative data is turned into categorical data using some type of bucketing.

When labels are available on the original data, as is the case for Subgroup Discovery with a single output feature, a supervised discretization method can be devised for the problem at hand. In the method proposed by Grosskreutz and Rüping [15], the discretization happens within the algorithm and relies on a property of the function measuring subgroup quality to merge basic intervals in a bottom-up fashion. Yet, the end points for the basic intervals are determined as a pre-processing step in a way that is not necessarily optimal with respect to their later use.

In most settings, though, no labeling of the data is available and one has to resort to unsupervised discretization. This approach raises several questions, from the choice of the number of buckets to the size of the resulting data. A more elegant approach was provided by Srikant and Agrawal [16], who presented a machinery that solves most of the problems automatically. Their method is still based on *a priori* bucketing, and moreover, it is very specific to association rule mining, making it hard (or impossible) to apply to redescription mining.

The problem of on-the-fly discretization during a classification task was studied by Fayyad and Irani [17]. Although the task is different, the results we obtained for dynamically choosing the most accurate extension shares similarities with their result.

Using redescription mining algorithms with non-Boolean data is not a new idea. Already in ref. [5], the CART WHEELS algorithm was used to extract bioinformatics data that was non-Boolean. As the algorithm requires Boolean input, the data had to be bucketed as a pre-processing step. But pre-processing typically requires considerable domain knowledge and might still be impossible or yield exponential growth in the number of variables. This is in contrast to our algorithm, where the optimal discretization is determined at each iteration within the algorithm, requiring no pre-processing. Nothing, of course, prevents users to pre-process their data, should that be needed.

3.3. Niche Finding

In biology, the problem of finding species' bioclimatic envelope is a rather new one (see, e.g. ref. [18] and references therein), but the idea of ecological niches dates back to the early 20th century [1]. There is also some level of ambiguity in what exactly is meant by the term niche [18]. In this paper, we consider a bioclimatic envelope of a (group of) species to be a set of limits in climate variables (such as monthly mean temperature) that defines the region occupied by the species.²

Despite the vague definition, the past 10 years have seen a number of methods to model the bioclimatic envelopes. The methods are based, for example, on regression, neural networks, and genetic algorithms (see [19]). But to the best of the authors' knowledge, none of these methods allows automatically finding both the set of species and their envelope.

Other niche finding tasks have been formulated, for example, in a linguistic context [20]. In this paper, we consider only the biological problem.

4. THE BASIC ALGORITHM

In this section, we present the core of our algorithm. Various extensions to it, such as handling missing values, are presented in the next section.

4.1. Motivation and Background

The redescription mining problem is defined for general Boolean queries, yet none of the proposed algorithms explores the full search space (using instead decision trees of fixed depth [5], monotone CNF and DNF formulae [4], or only (possibly negated) conjunctions [6]). Such restrictions are easy to understand, given the huge search space formed by all Boolean formulae (2^{2^n} distinct formulae can be defined over n variables). With non-Boolean data the search space is even more overwhelming. It is therefore evident that when devising an algorithm usable with real-world data, the space of all Boolean functions cannot be considered in its entirety.

4.1.1. Type of Boolean queries mined

How to restrict the search space? This question can be considered from at least three different perspectives: the expressive power of the resulting queries, the ease of finding them, and their interpretability. For example,

²That is, we consider realized niches using correlative methods (see ref. [2]).

monotone conjunctive queries (i.e. frequent item sets) are easy to interpret and relatively easy to find, given the monotonicity of the search space, but they lack expressive power. On the other hand, general Boolean queries have a high expressivity, but are hard to find. Furthermore, deeply nested structures and variables appearing multiple times can lead to difficulties in interpreting them.

Our aim is to restrict the search to queries that provide a good compromise between the three aforementioned properties. For this purpose, we follow the approach taken by Gallo, Miettinen, and Mannila [3]. First, we evaluate the queries from left to right irrelevant of the operator precedence. In other words, we only consider queries that can be parsed in linear order, without trees. For example, $(a \vee b) \wedge \neg c$ is such a query, but $(a \wedge b) \vee (c \wedge d)$ is not. Second, we allow every variable to appear only once. Queries of this type are strict generalizations of purely conjunctive or disjunctive queries, save the tautological cases $a \wedge \neg a$ and $a \vee \neg a$. We consider such queries to be relatively easy to interpret while still having a satisfying expressive power. While becoming smaller, the search space still remains exponential. Therefore, we also employ a heuristic pruning, as will be explained later.

4.1.2. *On-the-fly bucketing vs. pre-processing*

Binning the variables into buckets is a standard pre-processing technique to make non-Boolean data Boolean (see Section 3.2). But it has its drawbacks. For example, the resulting data have a special structure, with all variables corresponding to different buckets of a given non-Boolean variable being mutually disjoint. The algorithms are typically not adjusted to this property.

Moreover, the bucketing must be made in a pre-processing step, and cannot be modified by the algorithm later on. If the quality of the bucketing was poor, so will be the results. But the user typically does not know whether a certain bucketing yields good results before running the algorithm, so repeated trials and errors are needed to achieve satisfactory results.

Our approach of doing the bucketing on-the-fly avoids these problems. The algorithm will select the optimal bucket for each case when necessary. This removes the need of pre-processing and repeated trials. Furthermore, our algorithm can use different buckets for the same variable in different redescrptions, should that yield better results.

4.2. **Outline of the Algorithm**

We use a strategy similar to beam-search to explore the solution space. The basic idea is to construct queries bottom-up, starting from singleton redescrptions (i.e. both queries contain only one literal) and progressively

extending them by appending operators and literals. For example, we could start with a pair $(a, \neg b)$, and try to extend it to $(a \wedge c, \neg b)$, $(a \vee c, \neg b)$, $(a \wedge \neg c, \neg b)$, etc. After evaluating all possible one-step extensions, we select the best candidates and extend them in turn. This process requires a book-keeping procedure to avoid repeatedly generating the same queries, as we will explain below. When no new redescription can be generated, we move to the next initial pair. The outline of the algorithm, called RE-REMI, is given in Fig. 1.

Our algorithm shares similarities with the GREEDY algorithm presented by Gallo et al. [3]. But unlike Gallo et al., and following the idea of beam-search, we allow several extensions to be generated from a given redescription in each step. In this way we can explore the search space more extensively. Notice also our algorithm’s resemblance to bottom-up frequent itemset mining algorithms. Indeed, finding redescrptions can be seen as a generalization of association rule mining [6], although without the monotonicity property.

4.3. **Efficient Computation of the Accuracy for Boolean Variables**

Given two queries, q_A and q_B , to decide which of the possible extensions of q_A yields the best redescription, we need to compute the accuracy (i.e. Jaccard coefficient) for four different types of extensions for each Boolean variable v : $J(q_A \wedge v, q_B)$, $J(q_A \wedge \neg v, q_B)$, $J(q_A \vee v, q_B)$ and $J(q_A \vee \neg v, q_B)$. Doing this in a straightforward way, determining a single Jaccard coefficient requires the computation of three distinct supports over the data. But this is not necessary. To compute $J(q_A \wedge v, q_B)$, we need to consider only the rows in $\text{supp}(q_A)$ —others will never be in $\text{supp}(q_A \wedge v)$. On the other hand, rows in $\text{supp}(q_A)$ will be in $\text{supp}(q_A \vee v)$ in any case and can be omitted when computing $J(q_A \vee v, q_B)$. Let us formalize this intuition.

Let $E_{1,0}$ be the set of entities for which only the first query holds (i.e. $E_{1,0} = \text{supp}(q_A) - \text{supp}(q_B)$), $E_{0,1}$ those for which only the second query holds, $E_{1,1}$ those for which both queries hold, and $E_{0,0}$ those for which neither of the queries hold. Finally, these sets restricted to $\text{supp}(v)$ are denoted as $E_{x,y}(v)$ (e.g. $E_{1,0}(v) = E_{1,0} \cap \text{supp}(v)$). The same notation is also used with real-valued variables and Iverson notation, as in $E_{1,0}([\lambda \leq v \leq \rho])$.

It is well-known that the Jaccard coefficient $J(q_A, q_B)$ can be expressed as

$$J(q_A, q_B) = \frac{|E_{1,1}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}|}. \tag{1}$$

Similarly, we can write $J(q_A \wedge v, q_B) = |E_{1,1}(v)| / (|E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|)$. Analogous formulae can be derived for all different extensions (see Fig. 2).

Input: Data $\mathcal{D} = (V_L, V_R, E, D_L, D_R)$, nonnegative integers k_p and k_i , constraints C .
Output: A set of redescription, \mathcal{R} .

- 1: $\mathcal{R} \leftarrow \emptyset$
- 2: $\mathcal{I} \leftarrow \{k_p \text{ best initial singleton redescription}\}$
- 3: **for** $S \in \mathcal{I}$ **do**
- 4: $\mathcal{K} \leftarrow \{S\}$
- 5: $F_L(S), F_R(S) \leftarrow$ free variables for S
- 6: **if** $F_L(S) \neq \emptyset$ or $F_R(S) \neq \emptyset$ **then**
- 7: $\mathcal{E} \leftarrow \{S\}$
- 8: **while** $\mathcal{E} \neq \emptyset$ **do**
- 9: **for each** $R \in \mathcal{E}$ **do**
- 10: **for side** $s \in \{\mathbf{L}, \mathbf{R}\}$ and operator $\circ \in \{\vee, \wedge\}$ **do**
- 11: **if** R can be extended on side s with operator \circ and literal $l \in F_s(R)$ **then**
- 12: $\mathcal{K} \leftarrow \mathcal{K} \cup \{\text{best such extension of } R \text{ admitting constraints } C\}$
- 13: $\mathcal{K} \leftarrow \{k_i \text{ best redescription from } \mathcal{K}, \text{ with updated free variables.}\}$
- 14: $\mathcal{E} \leftarrow \{R \in \mathcal{K} : F_L(R) \neq \emptyset \text{ or } F_R(R) \neq \emptyset\}$
- 15: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{K}$
- 16: Filter \mathcal{R} with constraints C
- 17: **return** \mathcal{R}

Fig. 1 REREM: Mine data sets for redescription.

$$J(q_A \wedge v, q_B) = \frac{|E_{1,1}(v)|}{|E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|}$$

$$J(q_A \wedge \neg v, q_B) = \frac{|E_{1,1}| - |E_{1,1}(v)|}{|E_{1,0}| - |E_{1,0}(v)| + |E_{0,1}| + |E_{1,1}|}$$

$$J(q_A \vee v, q_B) = \frac{|E_{1,1}| + |E_{0,1}(v)|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}(v)|}$$

$$J(q_A \vee \neg v, q_B) = \frac{|E_{1,1}| + |E_{0,1}| - |E_{0,1}(v)|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}| - |E_{0,0}(v)|}$$

Fig. 2 Formulae for computing the Jaccard coefficient for different extensions.

Notice that $E_{1,0}$, $E_{0,1}$, and $E_{1,1}$ can be computed once for a given redescription. Then, for each candidate variable, it is enough to perform three intersection operations to obtain $E_{1,0}(v)$, $E_{0,1}(v)$, and $E_{1,1}(v)$. Furthermore, $E_{0,0}$ and $E_{0,0}(v)$ can be deduced from $\text{supp}(v)$ and E , and we do not have to consider the rows in which neither q_A nor q_B hold. This observation can significantly speed up the algorithm.

4.4. Adding Categorical Variables

Handling the categorical variables is rather straightforward. We consider only the relation $v = c$, where c is the label. The above computation of Jaccard applies, as we can write, for example,

$$J(q_A \wedge [v = c], q_B) = \frac{|E_{1,1}([v = c])|}{|E_{1,0}([v = c])| + |E_{0,1}| + |E_{1,1}|}$$

What is different to the Boolean case is that we must select the class label c . But this we can do easily by trying all class labels and selecting the one that improves the

Jaccard most. Naturally, we can use the aforementioned speedup techniques for Jaccard when selecting the label, as each label just defines different Boolean vector.

4.5. Extension to Real-Valued Variables

With the real-valued data, our approach is to do bucketing on-the-fly, finding the optimal bucket to add in every step. Assume that our algorithm tries to extend, say, query q_A of redescription (q_A, q_B) with a real-valued variable v . The algorithm considers the extended query $q_A \wedge [\lambda \leq v \leq \rho]$ for different thresholds λ and ρ and selects those that maximize the accuracy of the extension. Naturally, the optimal λ and ρ are different for different extensions. The two thresholds are set simultaneously since setting one bound first and possibly the other later would prevent the greedy search from finding some of the most specific intervals.

How can we find λ and ρ efficiently? To tackle this question, we adapt our approach from the previous section, using a result similar to that of Fayyad and Irani [17].

We consider only the *shortest* interval yielding any truth value assignment: only values in $D(E, v)$, i.e. values taken by the variable v , can be interval bounds. We could try all possibilities, but if the data contains n entities, this can require n^2 time, quickly becoming infeasible since we have to compute the accuracies for each *candidate* extension. However, as for the Boolean case, only two subsets of entities for each type of extension can impact the Jaccard coefficient: those in $E_{1,1}$ and $E_{1,0}$ for conjunctions and those in $E_{0,1}$ and $E_{0,0}$ for disjunctions. Furthermore, only values separating entities from the two sets need to be considered.

DEFINITION 1: When extending a redescription with a numerical variable v , we say that value u is a *lower cut point* if for any fixed value w larger than u , $[u, w]$ is the shortest interval yielding a locally optimal support for the extension. Equivalently, we say that w is an *upper cut point* if for any fixed value u smaller than w , $[u, w]$ is the shortest interval yielding a locally optimal support for the extension. In addition, $-\infty$ and $+\infty$ can be lower and upper cut points, respectively.

EXAMPLE: Consider the example of extending a redescription (q_A, q_B) by appending a variable v with a non-negated conjunction. The bins in Fig. 3 represent the values taken by the variable, sorted in increasing order. Black circles indicate entities belonging to $E_{1,1}$, white circles indicate entities from $E_{1,0}$. Our aim is to find bounds λ and ρ that maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}([\lambda \leq v \leq \rho])|}{|E_{1,0}([\lambda \leq v \leq \rho])| + |E_{0,1}| + |E_{1,1}|}. \quad (2)$$

In this example, there is one entity in $E_{1,1}$ with value v_4 , but none in $E_{1,0}$ with value v_3 . Therefore, v_4 cannot be an optimal choice for λ since choosing v_3 instead would always increase the accuracy. For the same reason, $-\infty$, v_2 , v_3 , v_7 , v_{10} , and v_{11} are lower cut points, as only these values can be an optimal choice for λ here. Similarly, v_1 , v_2 , v_5 , and v_7 are upper cut points.

To improve the speed of computing optimal extensions, we need to identify lower and upper cut points efficiently. To that end, we present succinct characterizations of the cut points for different types of extensions. We use (v_1, v_2, \dots, v_k) to denote the values taken by the variable v for entities in $E_{1,1}$ or $E_{1,0}$, that is, values in $D(E_{1,0} \cup E_{1,1}, v)$, sorted in increasing order. Similarly, the values in $D(E_{0,0} \cup E_{0,1}, v)$ ordered increasingly are denoted as $(v'_1, v'_2, \dots, v'_l)$.

PROPOSITION 1: For a *non-negated conjunction*, a lower cut point is a value v_i such that $v_i \in D(E_{1,1}, v)$ and $v_{i-1} \in D(E_{1,0}, v)$, or $-\infty$ if $i = 1$ and $v_i \in D(E_{1,1}, v)$. An upper cut point is a value v_j such that $v_j \in$

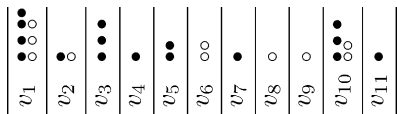


Fig. 3 Example of repartition of the entities for one variable. Each bin represents a value taken by the variable. Black circles indicate entities belonging to $E_{1,1}$, white circles indicate entities from $E_{1,0}$.

$D(E_{1,1}, v)$ and $v_{j+1} \in D(E_{1,0}, v)$, or $+\infty$ if $j = k$ and $v_j \in D(E_{1,1}, v)$.

For a *negated conjunction*, a lower cut point is a value v_i such that $v_i \in D(E_{1,0}, v)$ and $v_{i-1} \in D(E_{1,1}, v)$, or $-\infty$ if $i = 1$ and $v_i \in D(E_{1,0}, v)$. An upper cut point is a value v_j such that $v_j \in D(E_{1,0}, v)$ and $v_{j+1} \in D(E_{1,1}, v)$, or $+\infty$ if $j = k$ and $v_j \in D(E_{1,0}, v)$.

For a *non-negated disjunction*, a lower cut point is a value v'_i such that $v'_i \in D(E_{0,1}, v)$ and $v'_{i-1} \in D(E_{0,0}, v)$, or $-\infty$ if $i = 1$ and $v'_i \in D(E_{0,1}, v)$. An upper cut point is a value v'_j such that $v'_j \in D(E_{0,1}, v)$ and $v'_{j+1} \in D(E_{0,0}, v)$, or $+\infty$ if $j = l$ and $v'_j \in D(E_{0,1}, v)$.

For a *negated disjunction*, a lower cut point is a value v'_i such that $v'_i \in D(E_{0,0}, v)$ and $v'_{i-1} \in D(E_{0,1}, v)$, or $-\infty$ if $i = 1$ and $v'_i \in D(E_{0,0}, v)$. An upper cut point is a value v'_j such that $v'_j \in D(E_{0,0}, v)$ and $v'_{j+1} \in D(E_{0,1}, v)$, or $+\infty$ if $j = l$ and $v'_j \in D(E_{0,0}, v)$.

Proof: We concentrate on the case of non-negated conjunction; other cases are similar. The aim is to maximize $j(\lambda, \rho)$ (cf. Eq. 2). We start with the lower bound λ . First, suppose $v_i \notin D(E_{1,1}, v)$. Then v_i must occur in $E_{1,0}$ and we have $|E_{1,0}([v_i \leq v \leq \rho])| > |E_{1,0}([v_{i+1} \leq v \leq \rho])|$ while $|E_{1,1}([v_i \leq v \leq \rho])| = |E_{1,1}([v_{i+1} \leq v \leq \rho])|$. Hence, $j(v_i, \rho) < j(v_{i+1}, \rho)$. So v_i is not an optimal value for λ . Second, if $v_{i-1} \notin D(E_{1,0}, v)$, following a similar reasoning, we notice that $j(v_{i-1}, \rho) > j(v_i, \rho)$ and v_i is not an optimal value for λ . Finally, in case $v_1 \in D(E_{1,1}, v)$, setting $\lambda = v_1$ can be optimal and we simply leave the lower bound undefined, i.e. we use the half-line $(-\infty, \rho]$ that yields the same support.

The case of the upper bound ρ is analogous. If $v_i \notin D(E_{1,1}, v)$, then $v_i \in D(E_{1,0}, v)$ and we have $|E_{1,0}([\lambda \leq v \leq v_{i-1}])| < |E_{1,0}([\lambda \leq v \leq v_i])|$ while $|E_{1,1}([\lambda \leq v \leq v_{i-1}])| = |E_{1,1}([\lambda \leq v \leq v_i])|$. Hence, $j(\lambda, v_{i-1}) > j(\lambda, v_i)$ and v_i is not an optimal value for ρ . On the other hand, if $v_{i+1} \notin D(E_{1,0}, v)$ then $j(\lambda, v_{i-1}) > j(\lambda, v_i)$ and v_i is not an optimal value for ρ . Finally, when v_k occurs in $E_{1,1}$ for the last index k , setting $\rho = v_k$ can be optimal and we use the half-line $[\lambda, +\infty)$ which is equivalent with respect to the support.

The case of negated conjunctions is the reverse of this. Again, we only need to consider entities in $E_{1,0}$ or in $E_{1,1}$, but this time we will try to find an interval $[\lambda \leq v \leq \rho]$ such that the set $|E_{1,0}([\lambda \leq v \leq \rho])|$ is large while $|E_{1,1}([\lambda \leq v \leq \rho])|$ is small, so as to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| - |E_{1,1}([\lambda \leq v \leq \rho])|}{|E_{1,0}| - |E_{1,0}([\lambda \leq v \leq \rho])| + |E_{0,1}| + |E_{1,1}|}. \quad (3)$$

The case of disjunctions is very similar to conjunctions, but focusing on entities in $E_{0,1}$ and $E_{0,0}$ instead of $E_{1,1}$ and $E_{1,0}$, respectively.

For non-negated disjunctions the aim is to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| + |E_{0,1}(\{\lambda \leq v \leq \rho\})|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,0}(\{\lambda \leq v \leq \rho\})|}, \quad (4)$$

that is, we try to maximize the number of entities in $E_{0,1}$ while minimizing that in $E_{0,0}$.

The aim for negated disjunctions is to maximize

$$j(\lambda, \rho) = \frac{|E_{1,1}| + |E_{0,1}| - |E_{0,1}(\{\lambda \leq v \leq \rho\})|}{|E| - |E_{0,0}(\{\lambda \leq v \leq \rho\})|}, \quad (5)$$

the reverse of the non-negated case, similarly to conjunctions. ■

To search for an optimal bucket for variable v , we only need to consider upper and lower cut points. Denoting by n_λ the number of lower cut points and by n_ρ the number of upper cut points, the size of the search space is $(n_\lambda + 1)(n_\rho + 1)$. In many cases, this is considerably smaller than the naïve n^2 (but see Section 5.2 for a method to deal with large $(n_\lambda + 1)(n_\rho + 1)$).

4.6. Putting it All Together: The REREMi Algorithm

As we mentioned previously, the algorithm starts by evaluating all possible pairs of singleton redescription (i.e. literals) and keeps only the k_p best pairs (line 2). Alternatively, it is possible to extend all pairs with accuracy higher than some threshold or exhaust all the pairs in order to discover redescription with low first level accuracy. But after some number of initial pairs, a drop in the accuracy of the generated redescription can typically be observed. Limiting k_p is therefore reasonable.

Generating the initial pairs from real-valued data requires some extra work. There are two options. First, if one of the matrices (say D_L) is Boolean while the other is real-valued, we create the initial pairs by considering redescription $R = (v_L, \emptyset)$ for each $v_L \in V_L$, and extending their right-hand side using the standard on-the-fly bucketing approach. Second, if both sides are real-valued, an exhaustive search of all possible intervals needs to be performed. This might be computationally very expensive, and in Section 5.3 we present a method to find the initial pairs faster with possible loss in accuracy.

Each of the initial pairs is extended in turn (lines 3–15), selecting at each step the k_i most promising candidates (line 13). A value of 4 for k_i , for example, enables to keep the candidates for both operators and both sides on the first step. Two sets of variables, $F_L(R) \subset V_L$ and $F_R(R) \subset V_R$, are associated to each redescription R . They contain the variables that can be used to expand that redescription, which we call the free variables of R . The free variables are

determined so as to avoid generating several times the same redescription. The variables leading from R to some already generated one-step extension, i.e. redescription obtained by appending one literal to it, are not free for R ; this includes the variables that appear in R .

For the purpose of determining free variables, the algorithm maintains a list of the redescription generated so far. We assign a set of keys to every redescription, one for each literal involved. The redescription can be indexed using these keys in space bounded by the number of explored redescription multiplied by a factor quadratic in the maximal allowed query length. Then, given any redescription containing l literals, we can retrieve the sets of redescription associated to each of its keys in l accesses to the index. Its one-step extensions are the redescription of length $l + 1$ in the intersection of these sets.

In addition, when the query on either side of the redescription has reached the maximum number of variables, all remaining free variables for that side are removed. Among the selected candidates, those that have some free variables are put into the set \mathcal{E} of redescription to be extended during the next iteration (line 14). The loop ends when \mathcal{E} is empty, that is, when there is no extendible redescription left.

5. EXTENSIONS TO THE ALGORITHM

With the basic algorithm presented in the previous section, we now turn to study some extensions to it. We first start by studying the possible constraints one can apply for the redescription, then present two methods that can be used to speed up the algorithm (possibly trading off some accuracy) with real-valued data. Finally, we explain how to extend the algorithm to handle missing values.

5.1. Constraints on the Redescription

In this section, we discuss the different constraints one can apply to redescription. The accuracy is a simple constraint: leave out all redescription with accuracy lower than some threshold. But in addition to being accurate, we would like the redescription to be statistically significant. That is, the support of a redescription (q_L, q_R) should carry some new information, given the support of the queries. To measure this, we test against the null-model representing the case in which the two queries would be independent. We compute a p -value that represents the probability that two random queries with marginal probabilities (i.e. the fraction of entities supporting them) equal to those of q_L and q_R have an intersection equal to or larger than $|\text{supp}(q_L, q_R)|$. This probability uses the binomial distribution and is

given by

$$\text{pvalM}(q_L, q_R) = \sum_{s=|\text{supp}(q_L, q_R)|}^{|E|} \binom{|E|}{s} (p_R)^s (1 - p_R)^{|E|-s},$$

where $p_R = |\text{supp}(q_L)| |\text{supp}(q_R)| / |E|^2$. The higher the p -value, the more likely it is to observe such a support for independent queries, and the less significant the query.

Similarly, when appending a literal l to a redescription, we would like it to be as informative as possible. On one hand, if q_A and l have very similar supports, $q_A \vee l$ will not carry much more information than the original q_A , so we want q_A and l to be as uncorrelated as possible and intersect less than would occur randomly. On the other hand, when extending q_A to $q_A \wedge l$ we expect q_A and l to be correlated and intersect more than would occur randomly. Hence we define $\text{pvalE}(q_s, \wedge l) = \text{pvalM}(q_s, l)$ and $\text{pvalE}(q_s, \vee l) = 1 - \text{pvalM}(q_s, l)$.

Also important is the size of the support of the redescrptions and the number of entities by which each variable contributes to it, since redescrptions characterizing too few entities or almost all of them are of no interest.

These constraints on p -value and support can be applied more or less strictly during the beam search, using thresholds to penalize or simply disqualify candidate redescrptions that do not comply with them. The redescription p -value and support size can also be used to filter out uninteresting results *a posteriori*. Of course, the stricter the constraints applied within the candidate selection, the faster the search, but the more likely it becomes to miss candidates that would expand to acceptable redescrptions.

The type of query can also be selected, for example to disallow negations or use only disjunctions. Most of the constraints need not be tuned for good results, but they can be used to incorporate domain knowledge or to guide the algorithm to search for special redescrptions.

5.2. Interval Approximation

In cases where the search space of possible intervals is still too large (i.e. $(n_\lambda + 1)(n_\rho + 1)$ is too big, cf. Section 4.5), we use a faster search to find an interval whose accuracy is a good lower bound to the optimal one.

Let $(t_0, t_1, \dots, t_{n_\lambda+n_\rho+1})$ be the ordered list of cut points (as per Proposition 1), with the special cases $t_0 = -\infty$ and $t_{n_\lambda+n_\rho+1} = +\infty$. Let $l, i,$ and u be any indices such that $S_1 = [t_l, t_i]$, $S_2 = [t_i, t_{i+1}]$, and $S_3 = [t_{i+1}, t_u]$ are valid intervals. Note that we can have $t_i = t_{i+1} = w$, when the value w is both a *lower* and an *upper cut point*.

On one hand, if the accuracy obtained by merging intervals S_1 and S_2 is lower than that of S_2 alone, then merging $S_1, S_2,$ and S_3 yields lower accuracy than S_2 alone

or merging S_2 and S_3 , for any interval S_3 . That is, if $j(t_l, t_{i+1}) < j(t_i, t_{i+1})$, then

$$j(t_l, t_u) < \max(j(t_i, t_{i+1}), j(t_i, t_u)).$$

We use this property to find the best interval by upward aggregation. Starting with the first interval, we construct at each iteration an interval of the form $I_i = [\lambda_i, t_i]$. We go through the possible optimal values in ascending order, while keeping track of the best accuracy encountered: $\lambda_{i+1} = t_i$ if $j(\lambda_i, t_{i+1}) < j(t_i, t_{i+1})$, and $\lambda_{i+1} = \lambda_i$ otherwise.

On the other hand, if the accuracy obtained by merging intervals S_1 and S_2 is greater than that of S_2 alone, there might still be an interval S_3 such that merging S_2 and S_3 yields a higher accuracy than S_1, S_2 and S_3 together. That is, even if $j(t_i, t_{i+1}) < j(t_l, t_{i+1})$, it does not necessarily follow that $j(t_i, t_u) < j(t_l, t_u)$. Therefore, we also compute the best interval using downward aggregation, starting with the last interval and iterating over the possible optimal values in reverse order. Then we combine the two best intervals to eliminate possible undesirable values on either ends. Let I_u and I_d denote the best intervals found using upward and downward aggregations, respectively. The final interval returned is either $I_u, I_d,$ or $I_u \cap I_d$, depending on which maximizes $j()$. Using this method, we can compute an interval that approximates the optimal accuracy in $O(n_\lambda + n_\rho)$. This is especially useful in cases where the rows in $E_{1,0}$ and $E_{1,1}$ (for conjunctions) or $E_{0,1}$ and $E_{0,0}$ (for disjunctions) are not clearly separated, saving heavy computations when encountering variables that are intuitively poor extensions.

5.3. Approximating the Initial Pairs for Real-Valued Data

The approximate search presented in Section 5.2 might not be sufficient to find the initial pairs in reasonable time. This is especially the case when the data contain dense variables with many different values on both sides. We then resort to unsupervised bucketing to reduce the number of intervals tested and make the computation tractable.

During the initial pair generation, when the number of intervals to test for a given pair of variables exceeds a predefined threshold, different values of the variables are aggregated together to construct non-overlapping contiguous buckets. Only the intervals corresponding to the bounds of the buckets are tested. The aggressiveness of the method can be adapted by tuning the threshold on the original number of intervals and the number of entities that can be grouped.

Unlike with usual pre-bucketing approaches, the static buckets are only used for this special cases of initial

pair generation; during the later extension steps, the algorithm determines the intervals dynamically as described previously.

5.4. Handling Missing Values

Real-world data often contains missing values, i.e. cases where not all values are known for all entities. In order to handle such data, we consider the extended truth value set {True, False, Missing}. Any truth value assignment for a variable with missing values will lead to Missing as the value for those entities. When evaluating Boolean queries, the following new cases emerge: the negation of Missing is Missing; the value of $X \wedge \text{Missing}$ is False if X is False and Missing otherwise; the value of $X \vee \text{Missing}$ is True if X is True and Missing otherwise.

With missing values, there are five new groups of entities given a redescription (q_A, q_B) : either q_A or q_B can be Missing while the other is True or False, or they are both Missing. Following the notation of already-defined sets $E_{1,0}$, $E_{0,1}$, $E_{1,1}$, and $E_{0,0}$, we denote these sets by $E_{1,?}$, $E_{0,?}$, $E_{?,1}$, $E_{?,0}$, and $E_{?,?}$.

The presence of missing values requires us to re-define how we compute the quality of the redescription, i.e. the Jaccard coefficient. A common approach is to ignore those entities that have missing values. With relatively few missing values that are evenly distributed, this should give a reasonably good estimate of the true accuracy. We call this method *rejective Jaccard*, denoted J_R , to distinguish from the cases with no missing values.

The rejective Jaccard has its problems, though. If the missing values are not evenly distributed, we may have to reject too much data to be able to obtain reasonable estimates. Therefore, we also consider two other estimates: the *optimistic Jaccard* (J_O) and the *pessimistic Jaccard* (J_P), defined as follows:

$$J_O(q_A, q_B) = \frac{|E_{1,1}| + |E_{1,?}| + |E_{?,1}| + |E_{?,?}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{1,?}| + |E_{?,1}| + |E_{?,?}|} \quad (6)$$

$$J_P(q_A, q_B) = \frac{|E_{1,1}|}{|E_{1,0}| + |E_{0,1}| + |E_{1,1}| + |E_{0,?}| + |E_{?,0}| + |E_{?,?}|} \quad (7)$$

The optimistic Jaccard gives the upper bound and the pessimistic the lower bound of the Jaccard coefficient when truth values are assigned to the missing values. In other words, the optimistic Jaccard corresponds to the accuracy obtained if one could assign truth values to the missing values in the most favorable way while the pessimistic Jaccard is the accuracy that would result from an adversarial assignment of the missing values.

These three accuracy measures change the way our algorithm behaves by changing the objective function.

Optimizing J_O , for example, means that we try to find a maximum upper bound, while optimizing J_P means we try to find a maximum lower bound. We study these effects in Section 6.4.

Other than the different accuracy measures, the missing values are rather straightforward to implement. We need to keep track of the new sets, $E_{1,?}$, $E_{0,?}$, $E_{?,1}$, $E_{?,0}$, and $E_{?,?}$, for efficient computation of the different versions of Jaccard (cf. Section 4.3), but this does not alter the main principle. Similarly with real-valued data, having missing values only adds notational complexity—the concepts remain the same.

6. EXPERIMENTAL EVALUATION OF ALGORITHM'S PROPERTIES

We now turn to the experimental evaluation of our algorithm. We divide this in three parts. The first part (this section) studies the various properties of our algorithm with both synthetic and real-world data. The next two sections compare our algorithm to other methods and present a real-world application of our algorithm, namely biological niche finding.

But before starting with the experiments, we explain a method for assessing the significance of the results based on randomizations and explain the data sets used.

An implementation of the REREMi algorithm and the synthetic data generator are available online.³

6.1. Assessing the Significance with Randomization Methods

When mining the redescrptions, we compute various p -values in order to prune uninteresting rules. But these p -values are based on assumptions about the distribution of 0s and 1s in the (bucketed) data. Given the generality of our algorithm, we cannot assume the distributions to model exactly the underlying distribution of values. Hence, we also use property-preserving randomization methods. Such methods sample random matrices that share some property with the original matrix. The algorithm is then rerun using a random matrix as an input, and this process is repeated multiple times. If the results with random matrices contain multiple redescrptions that have same or higher accuracy than some redescription found from the original data, that redescription is deemed insignificant; otherwise it is significant (with respect to the property preserved by the randomization method).

For these experiments, we used two randomization methods. The first method permutes the matrix, preserving

³ <http://www.cs.helsinki.fi/u/galbrun/redescriptors/>.

the values of the matrix. On symmetric matrices, we used a variation that preserves their symmetry: only the upper-right triangle was permuted, and the lower-left triangle was copied from there. The property of a matrix having the same values is not a very strong one. Hence, we also used a method to preserve the distribution of values in columns and rows of the matrix [21]. This method is called swap-randomization. While swap-randomization is in many ways stronger than permutation, the latter is used for two reasons. First, its use was suggested by Ojala et al. [21]. Second, unlike swap-randomization, permutation can preserve symmetrical matrices.

6.2. The Real-World Data Sets

For the real-world data, we used two basic data sets: DBLP and Bio. The former is obtained from the DBLP database,⁴ and its entities are authors. The first matrix defines the conferences in which each of them has published, while the second defines other authors with whom each of them has published. The entities of the latter data set are spatial areas, that is, approximately 50 km² over Europe.⁵ The data itself is composed from two publicly available databases: European mammal atlas [22] and Worldclim climate data [23]. The mammals data contain presence/absence information of mammal species in Europe, and the climate data contain minimum, average, and maximum monthly temperatures as well as average monthly precipitation. Notice that the mammals data are Boolean while the climate data are continuous.

We created variations of these two basic data sets for specific purposes. From the DBLP data we created three variations: DBLP_F, DBLP_N, and DBLP_B. The first, DBLP_F, is a big data set with 6455 authors and 304 conferences containing information on how many times each author has published in each conference and with each other author. The second, DBLP_N, also contains numerical information but is restricted to 19 hand-picked conferences⁶ and 2345 authors. The third, DBLP_B, is like DBLP_N, but Boolean: every positive value of DBLP_N is replaced with 1. These data are identical to the one used by Gallo et al. [3].

For the Bio data, we constructed two variations. The basic Bio data contain the whole data while smaller Bio_{small} concentrates only on Northern Europe (specifically, areas between 50 and 71°North). Also, it does not include monthly maximum or minimum temperatures,

leaving only monthly average temperature and average precipitation.

The statistics of the real-world data sets used in the experiments are presented in Table 1.

6.3. Finding Planted Redescriptions

To study the behavior of our algorithm we first apply it to synthetic data. The idea is to generate data with planted redescrptions and check whether our algorithm is able to recover the redescrptions. Generating matrices such that, for example, no subset of the query forms an exact redescrption, is not trivial.

To generate a pair of synthetic Boolean 500 × 10 matrices, we plant on both matrices one Boolean formula, either conjunction or disjunction over 3 variables with 50 supporting rows, such that the resulting redescrption is exact. Then we add random noise of density between 0.01 and 0.1. The noise can either be conservative or destructive, leaving the redescrption exact or not. A synthetic real-valued data matrix is then obtained replacing 1s and 0s by values uniformly sampled from the intervals [0.75, 1] and [0, 0.25], respectively.

Applied to some 200 synthetic Boolean matrix pairs with conservative noise the algorithm managed to find all planted queries. In the case of destructive noise and fully Boolean data, the algorithm managed to find the planted queries in only about 40% of the data sets, but always found queries with higher accuracy than that of the planted redescrption. The algorithm cannot be considered faulty in these cases: It behaved as assumed, finding the redescrptions with the highest accuracies.

Applied to synthetic data sets where one matrix is Boolean and the other real-valued, both with conservative noise, the algorithm managed to find the planted redescrptions or equivalent in 76 cases out of 80. The planted redescrptions that were not found had contributions below the acceptance threshold. Thus, the algorithm again worked as assumed.

6.4. Experiments with Missing Values

As for the full data, our first experiment with missing values is on synthetic data. We removed values from synthetic Boolean matrices with planted redescrptions (cf. Section 6.3) uniformly at random. We removed 1%, 5%, and 10% of the values (0s or 1s) and replaced them with markers for missing values. Then we mined them with our algorithm and checked how well the planted redescrptions were discovered.

For the planted redescrptions with conservative noise, the planted redescrption was found in 97% of the cases. With increasing ratio of missing values the algorithm more

⁴ <http://www.informatik.uni-trier.de/~ley/db>.

⁵ Details of the grid can be found in www.fmnh.helsinki.fi/english/botany/afe/index.html.

⁶ Namely www, SIGMOD, VLDB, ICDE, KDD, SDM, PKDD, ICDM, EDBT, PODS, SODA, FOCS, STOC, STACS, ICML, ECML, COLT, UAI, and ICDT.

Table 1. Statistics of the real-world data sets used in the experiments.

Data set	Description	Dimensions	Type	Density
DBLP _F	Authors × Conferences	6455 × 304	Integer values	0.033
	Authors × Authors	6455 × 6455	Integer values	0.002
DBLP _N	Authors × Conferences	2345 × 19	Integer values	0.194
	Authors × Authors	2345 × 2345	Integer values	0.005
DBLP _B	Authors × Conferences	2345 × 19	Boolean Indicators	0.194
	Authors × Authors	2345 × 2345	Boolean Indicators	0.005
Bio	Locations × Mammals	2575 × 194	Boolean Indicators	0.166
	Locations × Climate	2575 × 48	Real values	—
Bio _{small}	Locations × Mammals	1271 × 119	Boolean indicators	0.259
	Locations × Climate	1271 × 24	Real values	—

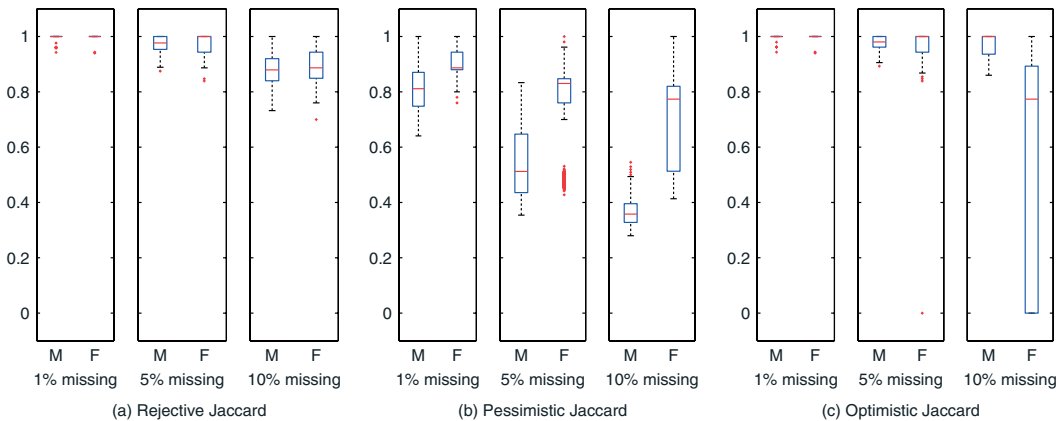


Fig. 4 Distribution of redescription accuracies with missing values, conservative noise, and different accuracy measures. (a) Rejective Jaccard. (b) Pessimistic Jaccard. (c) Optimistic Jaccard. The three panels of subplots correspond to, from left to right, 1%, 5%, and 10% of missing values. In each panel, the left-hand box is the distribution of accuracies with missing data and using the corresponding accuracy measure (M) and the right-hand box is the distribution of accuracies of the same redescriptions computed over full data and using normal Jaccard (F). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

often did not find the complete planted redescription, but rather one or several fragments of high accuracies.

We report the distributions of the accuracies in Figs 4 (conservative noise) and 5 (destructive noise). The results compare the different variations of the Jaccard (rejective, pessimistic, and optimistic) as well as their estimates to the true Jaccard computed over the full data (i.e. with no missing values).

In both figures, it is clear that the rejective Jaccard is the best. With conservative noise (Fig. 4(a)) missing values have hardly any effect, and while the destructive noise reduces the quality (Fig. 5(a)), different levels of missing values have very small effect and the estimated Jaccards are close to the true Jaccards. Pessimistic Jaccard is the second-best option. While its estimates look bad (Figs 4(b) and 5(b)), the true Jaccards are almost as good as those with rejective Jaccard. One has to remember that the estimate of the pessimistic Jaccard is only the lower bound, and therefore having very low estimate is not bad

in itself. The last option, optimistic Jaccard (Figs 4(c) and 5(c)) has the worst behavior with higher level of missing values (although with 1% and 5% of missing values the true Jaccards are slightly better than those of pessimistic Jaccard).

As the missing values were evenly scattered over the data, the good behavior of rejective Jaccard was no surprise.

Next, we experimented with the Bio_{small} data set, generating copies with 1% or 5% of randomly chosen values removed. For each setting, we generated 10 copies and mined them. We compared the queries found in the data with missing values to those found in the full data. We only used rejective Jaccard due to its good performance with synthetic data.

The results are reported in Fig. 6. With 1% of missing values the results are almost as good as the results with full data, but with 5% of missing values the true Jaccards are somewhat worse, even if the estimates are still high.

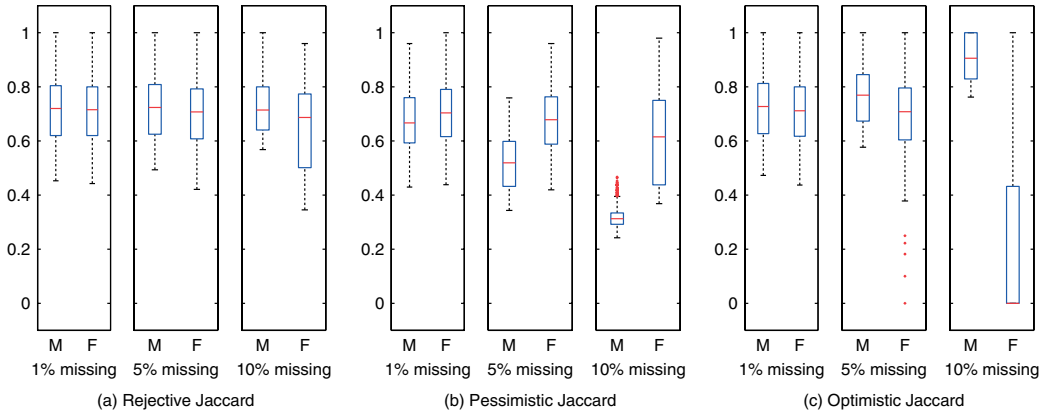


Fig. 5 Distribution of redescription accuracies with missing values, destructive noise, and different accuracy measures. (a) Rejective Jaccard. (b) Pessimistic Jaccard. (c) Optimistic Jaccard. The three panels of subplots correspond to, from left to right, 1%, 5%, and 10% of missing values. In each panel, the left-hand box is the distribution of accuracies of the same redescrptions computed over full data and using normal Jaccard (F). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

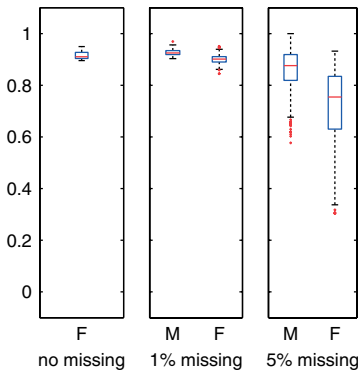


Fig. 6 Results with Bio_{small} data set having missing values. Left: distribution of similarities with all values known. Middle: distribution of similarities with 1% of values missing. Right: distribution of similarities with 5% of values missing. In middle and right panels, left-hand box is rejective Jaccard with missing values (M), and the right-hand box is the normal Jaccard of the same redescrptions computed over full data (F). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

6.5. From Boolean to Numerical: The DBLP Data

The purpose of this experiment is to study the behavior of our algorithm with two versions of the same data, one Boolean and one numerical. For this we used the two variants of the DBLP data: $DBLP_B$ and $DBLP_N$.

Some results with $DBLP_N$ are presented in Table 2, while some example results with $DBLP_B$ are in Table 3. In all tables, J stands for the Jaccard, supp is the support

of the redescription, and the p -value is computed as in Section 5.1.

For these experiments, we disabled negations to allow later comparison (see Section 7.2). Comparing the two tables, it is obvious that $DBLP_N$ contains more information, allowing REREMi to find more accurate redescrptions (the best Jaccard in Table 2 is 0.625). But the rules are also more specific than those in Table 3, with small support and often requiring multiple publications in the same conference, making them possibly harder to interpret than those of Table 3. Yet, in both cases the algorithm correctly identifies subfields of computer science and well-known authors from those fields.

6.6. Approximating the Initial Pairs

To study the effects of approximating the initial pairs, we took the climate data of the Bio_{small} data set and divided it into two: the monthly average temperatures and monthly average rainfall. This gave us two real-valued data sets with 12 variables each. We call this data *Climate*.

This seemingly small data set is already hard for exhaustive initial pairs selection due to the nature of the data: almost each data point contains different values. Consequently, the exhaustive search was not finished in 2 days.

Instead of exhaustive search we used the binning-based initial-pairs searching method explained in Section 5.3. We tried different bin sizes, ranging from 10 to 100. We compared this with fully pre-bucketing approach, bin size again ranging from 10 to 100.

Table 2. Example results of REREMi with $DBLP_N$.

q_L	q_R	J	supp	p -value
(1) $[1.0 \leq \text{STOC} \leq 6.0] \wedge [8.0 \leq \text{COLT}]$	$[1.0 \leq \text{D.P. Helmbold}] \vee [1.0 \leq \text{M. Frazier}]$ $\vee [2.0 \leq \text{N. Cesa-Bianchi} \leq 2.0]$	0.625	15	0.000
(2) $[1.0 \leq \text{VLDB} \leq 18.0] \wedge [2.0 \leq \text{ICDM}]$ $\wedge [1.0 \leq \text{SDM} \leq 5.0]$ $\wedge [3.0 \leq \text{ICDE}]$	$(([5.0 \leq \text{W. Wang}] \vee [1.0 \leq \text{J. Pei}])$ $\wedge [2.0 \leq \text{P.S. Yu}]) \vee [6.0 \leq \text{G. Das} \leq 6.0]$	0.600	12	0.000
(3) $[5.0 \leq \text{COLT}]$	$[2.0 \leq \text{P.L. Bartlett}] \vee [1.0 \leq \text{M.K. Warmuth}]$ $\vee [1.0 \leq \text{E.B. Kinber}] \vee [1.0 \leq \text{S.A. Goldman}]$	0.472	42	0.000

Table 3. Example results of REREMi with $DBLP_B$.

q_L	q_R	J	supp	p -value
(1) $\text{STOC} \wedge \text{COLT} \wedge \text{ICML}$	$\text{Y. Freund} \vee \text{N. Littlestone} \vee \text{P.M. Long} \vee \text{S. Kwek}$	0.500	21	0.000
(2) $\text{VLDB} \wedge \text{ICDM} \wedge \text{SDM} \wedge \text{SIGMOD}$	$(\text{J. Han} \wedge \text{P.S. Yu}) \vee \text{C.-R. Lin} \vee \text{S. Lonardi}$	0.444	16	0.000
(3) $\text{ICDM} \wedge \text{SDM} \wedge \text{KDD}$	$\text{J. Lin} \vee \text{I.S. Dhillon} \vee \text{P.S. Yu} \vee \text{V. Kumar}$	0.338	44	0.000
(4) $\text{FOCS} \wedge \text{SODA} \wedge \text{STOC}$	$\text{B. Awerbuch} \vee \text{S. Khanna} \vee \text{R.E. Tarjan} \vee \text{N. Alon}$	0.324	158	0.000

The results were clear. There was very little variation between different bin sizes for initial pairs in terms of accuracy: the smaller bins gave overall slightly better results, but all sizes had top 20 redescrptions with accuracy clearly above 0.8 with best above 0.9. Pre-bucketing approaches, on the other hand, were clearly inferior, with the best results (obtained with bin size 10) being below 0.7.

In terms of speed, the pre-bucketed approaches were the fastest, with even the slowest (bin size 10) taking less than 4 min. The fastest method with initial bins (bin size 100) took roughly 10 times longer, 38 min. The slowest method (initial bin size 10) took 1 day and 24 min. In all the methods with initial bins, the time differences are explained by the time spent on constructing the initial pairs; after that, they all took roughly the same time.

The results were as expected. Using binning only for initial pairs (as opposed to working with pre-bucketed data) gave much better results, but also took more time. It is, however, noteworthy that increasing the bin size used for selecting the initial pairs did not have considerable effect on the quality. Apparently, the algorithm was able to overcome the possibly weaker initial pairs in later phases.

6.7. Running Times

Does our algorithm scale? The data sets we used were not extremely large (although $DBLP_F$ is already of considerable size), but we feel confident to say that our algorithm is scaling reasonably well. All experiments were conducted in a single core of an 8 core Intel Xeon 2.8 GHz processor and with 32 GB of memory.

The statistics of the running times on the different data sets are presented in Table 4. There we can see that original

data take much more time to run than randomized data. This is because with randomized data, there are less potential redescrptions, and the algorithm can prune the search space much faster. The longest time required, 1 h with $DBLP_F$, is still very good, especially when one remembers that the algorithm needs to be run only once for the original data.

To evaluate more precisely the evolution of our algorithm's running time with respect to the size of the data, we generated synthetic matrices of increasing sizes. As we just mentioned, the running times on random data is not representative. Therefore, we need to insure that some redescrptions are present in the synthetic data. Thus, we constructed diagonal block matrices, where the blocks on the diagonal are matrices obtained as described in Section 6.3 while off-diagonal blocks are filled with random noise. We used original matrices of size 500×10 for the fully Boolean setting and of size 250×10 for the setting where one side is Boolean and the other real-valued. We let our algorithm run on 10 such synthetic data sets for each studied size. The algorithm consists of two major steps: first, computing singleton redescrptions by trying out all pairs of variables and second extending the best pairs in turn. In the case of fully Boolean data, the computation of initial pairs clearly dominates as the data grows larger, since it is quadratic in the number of columns while finding extension is linear. The running times remain reasonable even for very large data sets.

When real-valued data are involved, the running times are much greater, as expected. Therefore, we had to restrict the experiments to smaller data sets. The running times for that experiment are displayed in Fig. 7. The increase in running times is due to the fact that extending a redescrption is no longer linear and the search for initial pairs also has increased complexity.

Table 4. Running times for the different data sets. For mean time, ‘—’ denotes that there was only one data set.

Data set	mean	max
DBLP _F	—	60 min
DBLP _F permuted	5 min 16 s	6 min 32 s
DBLP _F swap	16 min	18 min
DBLP _N	—	7 min
DBLP _B	—	1 min
Bio	—	10 min
Bio permuted and swap	6 min 48 s	10 min 30 s

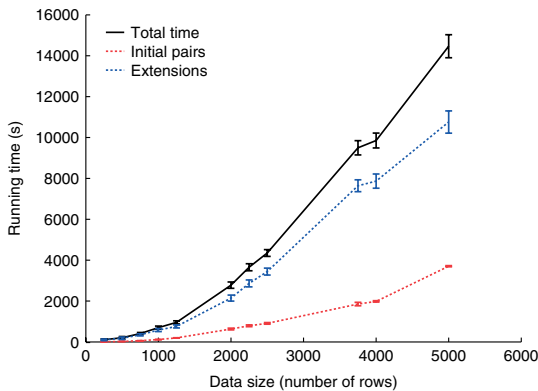


Fig. 7 Running times for mining redescrptions on synthetic data of increasing sizes with one side Boolean and the other real-valued. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

7. EXPERIMENTAL COMPARISON AGAINST OTHER METHODS

With the previous sections results showing that our algorithm has solid performance, we can now turn to comparing our algorithm to other methods.

7.1. Comparison to Association Rule Mining

The first experiment with real-world data mirrors the experiments with synthetic data: the task is to study how well our algorithm finds redescrptions from the data. But as we cannot know all redescrptions present in the real-world data, we narrow our scope to monotone conjunctive redescrptions from Boolean data. These redescrptions are simply bi-directional association rules, and hence can be found by mining all frequent itemsets from both data matrices and using the itemset pairs as redescrptions. This gives us the ground truth, against which we can compare our algorithm.

Statistical Analysis and Data Mining DOI:10.1002/sam

For these experiments we used the DBLP_B data (see Section 6.2). We used the ECLAT frequent itemset miner [24]. The redescrptions were generated as follows: first, all closed frequent itemsets with support greater than five were mined for both data sets. The itemsets were then combined into redescrptions. Only those with accuracy greater than 0.1, support above 10 but below 100 (inclusive), and *p*-value higher than 0.01 were retained. The same parameters were set to REREMi, and it was only allowed to find monotone conjunctive redescrptions.

The best four redescrptions found using ECLAT had a Jaccard similarity between 0.366 and 0.333. REREMi found exactly the same redescrptions. After these four redescrptions, the ECLAT approach found several redundant redescrptions: they were minor variations of the first four redescrptions. REREMi did not report these redundant redescrptions, which we consider a positive feature—the user should not be overwhelmed by the quantity of results.

The next non-redundant redescrption found by ECLAT approach was also found by REREMi. The same trend continued throughout the results: ECLAT approach found several thousands of redescrptions, but most of them were redundant.

Applying ECLAT on swapped and permuted randomized copies of the original data (500 pairs of matrices for each method) following the same approach did not return any redescrption. Therefore, all original redescrptions are considered significant with respect to these null hypotheses.

Although these experiments cannot guarantee that REREMi will always find the best redescrptions, they suggest that it is able to find most of the important ones.

7.2. Comparison to the Work of Gallo et al.

As we used the same data set as Gallo et al. [3], DBLP_B, we compared the results obtained with REREMi to theirs. Following Gallo et al., we did not allow negations. The example results are presented in Table 3.

The results obtained by REREMi had higher Jaccard similarity than those obtained by Gallo et al.: the highest accuracy they report is 0.35, while REREMi returns a redescrption with accuracy 0.5, and 9 redescrptions have accuracy above 0.35.

An effect of the beam search is illustrated by the following for example. The results presented by Gallo et al. contain the following redescrption of accuracy 0.30:

$$SDM \wedge ICDE \quad P. S. Yu \vee J. Lin \vee M. Schubert \vee Y. Ma.$$

Our results do not contain this redescrption, but

$$SDM \wedge ICDM \wedge KDD \quad P. S. Yu \vee J. Lin \vee I. S. Dhillon \vee V. Kumar,$$

instead, with accuracy 0.34. Indeed, when starting from the initial pair (SDM, P. S. Yu), keeping at each step only the best candidate does not allow to find this redescription as it is obtained from a candidate ranked lower during the extension process.

While allowing for a better exploration of the search space, our algorithm required only a third of GREEDY's running time (3 min) with the same data. This shows that the proposed algorithm is faster yet more exhaustive on Boolean redescription mining than the GREEDY algorithm.

Otherwise, the results are similar; both algorithms identify sets of conferences from different fields of computer science together with well-known authors from those fields. Both algorithms are also able to identify interdisciplinary researchers, say, theoretical machine learners who publish in both machine learning and theoretical conferences (row 1 of Table 3).

7.3. Comparison to CARTWHEELS and Pre-Bucketing Numerical Data

We now turn to another algorithm for mining redescrptions, CARTWHEELS [5]. We used the implementation of CARTWHEELS provided by the authors. Because of this, we do not have any control over the results reported by CARTWHEELS (e.g. minimum support, type of queries, p -values).

Boolean Data: First, we tried CARTWHEELS with DBLP_B. The algorithm returned in total 35 redescrptions before running out of the available 32 GB of memory. Of these, only 5 were retained after removing rules with p -value higher than 0.05. All of the remaining redescrptions had p -values between 0.0111 and 0.02, making them insignificant on the highest significance level (99%). The rules also covered almost the whole data, having at least 1567 (of 2345) rows in the support. This high support is a consequence of using mostly negated variables. Results are reported in Table 5.

As can be seen from Table 5, the results have high accuracy, which is not surprising, given their high support.

Results also have many negations, making them less interesting. We let REREMi find results with negations, too, and while we were not able to find results with as high accuracy (results omitted), they all had p -values essentially 0. We conclude that while CARTWHEELS finds more accurate redescrptions, they are somewhat insignificant, and less interesting than the ones found by REREMi.

Pre-Bucketing Numerical Data: In this section, we compare our on-the-fly bucketing approach to CARTWHEELS with bucketing as a pre-processing step.

To bucket the data, one has to select the bucketing method. For a fair comparison, we used three different methods with a number of buckets per variable varying between 10 and 150, ran CARTWHEELS for all of these configurations, and selected the best results. The three methods were (i) buckets of equal width, where the range of the values in a column was divided into equally long buckets; (ii) buckets of equal height, where each bucket contained approximately equally many entities; and (iii) segmentation, where the entities were separated into segments (i.e. buckets) minimizing the sum-of-square distances to the segment's mean (the segmentation was obtained using Bellman's algorithm [25]).

The CARTWHEELS algorithm was unable to handle the full Bio data, so we used Bio_{small} instead.

The results are reported in Table 6. The aim of this experiment is to study how good accuracies can be obtained with pre-processed buckets compared to REREMi.

The first results are the five best (with respect to the accuracy) from CARTWHEELS using 10 buckets of approximately equal number of entities (this method produced the best overall results, although 10 segments gave very similar results). The redescrptions have again very high accuracy, and, again, they cover almost all of the studied area. Furthermore, two of them are clearly insignificant and one is not very significant, according to the p -values. Results of this type are rarely of any interest for users, as they do not convey any interesting information.

We pruned out results that had too high support (leaving less than 250 entities uncovered) or too high p -value. CARTWHEELS can still return rather accurate

Table 5. Example results of CARTWHEELS with DBLP_B.

q_L	q_R	J	supp	p -value
(1) (STOC \wedge \neg FOCS) \vee \neg STOC	B. Dageville \vee (\neg B. Dageville \wedge \neg A. Wigderson)	0.736	1673	0.011
(2) (STOC \wedge \neg FOCS) \vee \neg STOC	T. Grust \vee (\neg T. Grust \wedge \neg A. Wigderson)	0.736	1673	0.011
(3) EDBT \vee (\neg EDBT \wedge \neg STOC)	(P. Datta \wedge P. Langley) \vee (\neg P. Datta \wedge \neg A. Wigderson)	0.693	1577	0.021
(4) ICDM \vee (\neg ICDM \wedge \neg STOC)	(C. Olston \wedge \neg C. Chekuri) \vee (\neg C. Olston \wedge \neg A. Wigderson)	0.691	1570	0.017
(5) PKDD \vee (\neg PKDD \wedge \neg STOC)	T. Grust \vee (\neg T. Grust \wedge \neg A. Wigderson)	0.689	1567	0.019

Table 6. Example redescrptions from BiO_{small} data: t_X^{\min} , t_X^{\max} , and t_X^{avg} stand for minimum, maximum, and average temperature of month X in degrees Celsius, and p_X^{avg} stands for average precipitation of month X in millimeters.

q_L	q_R	J	supp	p -value
CARTWHEELS				
(1) (European Pine Vole ∧ European Pine Marten) ∨(¬European Pine Vole)	$([57.5 \leq p_{Nov}^{\text{avg}} \leq 62.706] \wedge \neg[75.03 \leq p_{Jun}^{\text{avg}} \leq 82.6]) \vee (\neg[57.5 \leq p_{Nov}^{\text{avg}} \leq 62.706])$	0.980	1244	0.007
(2) (Argali ∧ Hazel Dormouse) ∨(¬Argali)	$([78.291 \leq p_{Aug}^{\text{avg}} \leq 82.282] \wedge \neg[0.8245 \leq t_{Nov}^{\text{avg}} \leq 2.2357]) \vee (\neg[78.291 \leq p_{Aug}^{\text{avg}} \leq 82.282])$	0.974	1237	0.158
(3) (Brown Bear ∧ ¬Arctic Fox) ∨(¬Brown Bear)	$([43.163 \leq p_{Dec}^{\text{avg}} \leq 46.92] \wedge \neg[t_{Jul}^{\text{avg}} \leq 10.427]) \vee (\neg[43.163 \leq p_{Dec}^{\text{avg}} \leq 46.92])$	0.974	1237	0.074
(4) (Nathusius' Pipistrelle ∧ ¬Mediterranean Water Shrew) ∨(¬Nathusius' Pipistrelle)	$([40.896 \leq p_{Nov}^{\text{avg}} \leq 46.743] \wedge \neg[16.32 \leq t_{Jul}^{\text{avg}} \leq 16.751]) \vee (\neg[40.896 \leq p_{Nov}^{\text{avg}} \leq 46.743])$	0.974	1235	0.000
(5) (Brown Bear ∧ ¬Arctic Fox) ∨(¬Brown Bear)	$([60.13 \leq p_{Apr}^{\text{avg}} \leq 74.305] \wedge \neg[82.6 \leq p_{Jun}^{\text{avg}}] \vee (\neg[60.13 \leq p_{Apr}^{\text{avg}} \leq 74.305]))$	0.964	1224	0.337
CARTWHEELS pruned				
(1) (Eurasian Least Shrew) ∨ (¬Eurasian Least Shrew ∧ ¬House mouse)	$([75.154 \leq p_{Aug}^{\text{avg}} \leq 78.291] \wedge \neg[1.622 \leq t_{Feb}^{\text{avg}} \leq 3.44]) \vee (\neg[75.154 \leq p_{Aug}^{\text{avg}} \leq 78.291] \wedge \neg[5.488 \leq t_{Mar}^{\text{avg}}])$	0.832	944	0.000
(2) (Wisent) ∨(¬Wisent ∧ ¬House mouse)	$([34.467 \leq p_{Feb}^{\text{avg}} \leq 41.402] \wedge \neg[5.488 \leq t_{Mar}^{\text{avg}}]) \vee (\neg[34.467 \leq p_{Feb}^{\text{avg}} \leq 41.402] \wedge \neg[3.15 \leq t_{Jan}^{\text{avg}}])$	0.824	940	0.000
(3) (Red-necked Wallaby) ∨ (¬Red-necked Wallaby ∧ House mouse)	$([6.12 \leq t_{Nov}^{\text{avg}}] \wedge \neg[14.028 \leq t_{Sep}^{\text{avg}}]) \vee (\neg[6.12 \leq t_{Nov}^{\text{avg}}] \wedge [1.622 \leq t_{Feb}^{\text{avg}} \leq 3.44])$	0.522	175	0.000
(4) (European Mole ∧ ¬House mouse) ∨ (¬European Mole ∧ House mouse)	$([5.0439 \leq t_{Nov}^{\text{avg}} \leq 6.12] \wedge \neg[1.691 \leq t_{Mar}^{\text{avg}} \leq 2.9013]) \vee (\neg[5.0439 \leq t_{Nov}^{\text{avg}} \leq 6.12] \wedge [6.12 \leq t_{Nov}^{\text{avg}}])$	0.674	225	0.000
(5) (European Mole ∧ House mouse) ∨ (¬European Mole ∧ ¬House mouse)	$([5.0439 \leq t_{Nov}^{\text{avg}} \leq 6.12] \wedge [1.691 \leq t_{Mar}^{\text{avg}} \leq 2.9013]) \vee (\neg[5.0439 \leq t_{Nov}^{\text{avg}} \leq 6.12] \wedge \neg[6.12 \leq t_{Nov}^{\text{avg}}])$	0.896	937	0.000
REREMI				
(1) ¬ House mouse	$(\neg[3.4133 \leq t_{Mar}^{\text{avg}}] \vee [1.790 \leq t_{Aug}^{\text{avg}}] \vee [6.5917 \leq p_{May}^{\text{avg}} \leq 6.6083]) \wedge \neg[2.3667 \leq t_{Jan}^{\text{avg}} \leq 3.0667]$	0.948065	931	0.000
(2) ¬ House mouse	$([t_{Feb}^{\text{avg}} \leq 2.20] \wedge [t_{Mar}^{\text{avg}} \leq 3.41]) \vee [1.790 \leq t_{Aug}^{\text{avg}}] \vee [6.5917 \leq p_{May}^{\text{avg}} \leq 6.6083]$	0.947101	931	0.000
(3) ¬ House mouse ∧ ¬ Eastern gray squirrel	$[1.8375 \leq t_{Jan}^{\text{avg}} \leq 1.90] \vee \neg[0.25 \leq t_{Feb}^{\text{avg}}]$	0.934359	911	0.000
(4) ¬ House mouse ∧ ¬ Raccoon ∧ ¬ Eastern gray squirrel	$([5.37 \leq t_{Apr}^{\text{avg}} \leq 6.1357] \vee [5.04 \leq t_{Nov}^{\text{avg}} \leq 5.10] \vee \neg[1.25 \leq t_{Dec}^{\text{avg}}]) \wedge [3.775 \leq t_{Mar}^{\text{avg}}]$	0.925965	888	0.000
(5) ¬ Grey Red-Backed Vole ∧ ¬ Wolverine ∧ ¬ Brown Bear ∧ ¬ Siberian Flying Squirrel	$(\neg[t_{Jan}^{\text{avg}} \leq -6.4615] \wedge [14.994 \leq t_{Jul}^{\text{avg}}] \vee [t_{Jul}^{\text{avg}} \leq 4.6267] \vee [-4.9133 \leq p_{Dec}^{\text{avg}} \leq -4.871])$	0.923681	823	0.000
REREMI bucketed				
(1) (Striped Field Mouse ∨ House mouse) ∧ Wood mouse	$[5.925 \leq t_{Apr}^{\text{avg}} \leq 7.0] \vee [7.0 \leq t_{Apr}^{\text{avg}} \leq 7.9077] \vee [7.9077 \leq t_{Apr}^{\text{avg}} \leq 8.46] \vee [8.46 \leq t_{Apr}^{\text{avg}}]$	0.807	442	0.000
(2) House mouse	$[-0.21534 \leq t_{Feb}^{\text{avg}} \leq 1.622] \vee [1.622 \leq t_{Feb}^{\text{avg}} \leq 3.44] \vee [3.44 \leq t_{Feb}^{\text{avg}}]$	0.778	308	0.000
(3) European Rabbit	$[1.691 \leq t_{Mar}^{\text{avg}} \leq 2.9013] \vee [2.9013 \leq t_{Mar}^{\text{avg}} \leq 3.9685] \vee [3.9685 \leq t_{Mar}^{\text{avg}} \leq 5.488] \vee [5.488 \leq t_{Mar}^{\text{avg}}]$	0.774	441	0.000
(4) House mouse	$[2.9013 \leq t_{Mar}^{\text{avg}} \leq 3.9685] \vee [3.9685 \leq t_{Mar}^{\text{avg}} \leq 5.488] \vee [5.488 \leq t_{Mar}^{\text{avg}}]$	0.773	307	0.000
(5) North American Beaver ∧ Reindeer	$[t_{Jan}^{\text{avg}} \leq -11.9] \wedge [14.994 \leq t_{Jul}^{\text{avg}} \leq 15.785] \wedge [7.1073 \leq t_{Sep}^{\text{avg}} \leq 8.5667] \wedge [70.333 \leq p_{Jul}^{\text{avg}} \leq 73.385]$	0.700	7	0.000

redescriptions, but the quality drops quickly after the first ones.

The last results in this experiment are from REREMi. As CARTWHEELS obtained almost all of its results using negations, we allowed also REREMi to use negations. As can be seen, REREMi positions itself between non-pruned and pruned CARTWHEELS. But unlike the non-pruned CARTWHEELS, REREMi does not return insignificant results.

We also experimented with bucketed data and Boolean REREMi. The results were similar to those with pruned CARTWHEELS, but without the quick drop in accuracy. Also, they were considerably worse than the results with REREMi using on-the-fly bucketing.

We conclude that with similar constraints, the on-the-fly bucketing of REREMi gives the best results, although in this application, not allowing negations could be considered more reasonable an option.

8. REAL-WORLD APPLICATION: FINDING BIOCLIMATIC ENVELOPES

Our final experiment is a real-world application of finding bioclimatic envelopes.

REREMi was run on full Biodata for a maximum of 100 initial pairs, minimum score for the initial pairs 0.2, minimum support 15, minimum number of uncovered entities 500, and minimum contribution 3, disallowing negated variables. This was done because the negated variables can lead to counterintuitive redescriptions in this type of application. The algorithm found 69 redescriptions within these constraints. Again, we regard it positively that our algorithm returns only a reasonable amount of results.

The data were randomized using swap-randomization and permutation. With both methods, 500 random matrices were generated, and in both cases the best redescription had lower accuracy than the lowest original accuracy. Hence,

Table 7. Example redescriptions from Bio data: t_X^{\min} , t_X^{\max} , and t_X^{avg} stand for minimum, maximum, and average temperature of month X in degrees Celsius, and p_X^{avg} stands for average precipitation of month X in millimeters.

QL	QR	J	supp	p-value
(1) Polar Bear	$[-7.0727 \leq t_{\text{May}}^{\text{avg}} \leq -3.375]$	0.973	36	0.000
(2) Polar Bear	$[-16.694 \leq t_{\text{Mar}}^{\text{avg}} \leq -11.462]$	0.973	36	0.000
(3) Bank Vole ∨ Northern Red-backed Vole ∨ Steppe Mouse ∨ Harbor Seal	$(([11.20 \leq t_{\text{Jul}}^{\text{max}} \leq 15.40] \vee [13.10 \leq t_{\text{Aug}}^{\text{max}} \leq 27.40]) \wedge [42.5 \leq p_{\text{Jul}}^{\text{avg}}]) \vee [17.10 \leq t_{\text{Apr}}^{\text{max}} \leq 17.50]$	0.818	1679	0.000
(4) European Elk	$([-9.80 \leq t_{\text{Feb}}^{\text{max}} \leq 0.40] \wedge [12.20 \leq t_{\text{Jul}}^{\text{max}} \leq 24.60] \wedge [56.852 \leq p_{\text{Aug}}^{\text{avg}} \leq 136.46]) \vee [183.27 \leq p_{\text{Sep}}^{\text{avg}} \leq 238.78]$	0.814	582	0.000
(5) Arctic Fox ∨ Stoat	$(([2.60 \leq t_{\text{Jun}}^{\text{max}} \leq 8.50] \vee [7.20 \leq t_{\text{Sep}}^{\text{max}} \leq 22.20]) \wedge [36.667 \leq p_{\text{Aug}}^{\text{avg}}]) \vee [21.133 \leq t_{\text{Jul}}^{\text{avg}} \leq 21.20]$	0.813	1477	0.000
(6) Greater White-toothed Shrew ∧ Egyptian Mongoose	$([15.60 \leq t_{\text{Aug}}^{\text{min}} \leq 19.00] \wedge [1.625 \leq p_{\text{Aug}}^{\text{avg}} \leq 7.4444] \wedge [66.222 \leq p_{\text{Dec}}^{\text{avg}} \leq 137.27]) \vee [19.083 \leq t_{\text{Oct}}^{\text{avg}} \leq 19.10]$	0.790	49	0.000
(7) House Mouse ∨ Caucasian Squirrel ∨ Marbled Polecat	$(([3.50 \leq t_{\text{Jan}}^{\text{max}}] \wedge [4.40 \leq t_{\text{Feb}}^{\text{max}}] \vee [3.5071 \leq t_{\text{Mar}}^{\text{avg}} \leq 4.1727]) \wedge [3.30 \leq t_{\text{Dec}}^{\text{max}}])$	0.765	1034	0.000
(8) Southwestern Water Vole ∨ Azores Noctule ∨ Common Noctule ∨ Blind Mole	$([17.10 \leq t_{\text{Mar}}^{\text{max}}] \vee [19.30 \leq t_{\text{Aug}}^{\text{max}} \leq 26.90] \vee [12.40 \leq t_{\text{Nov}}^{\text{max}} \leq 14.50]) \wedge [14.60 \leq t_{\text{Sep}}^{\text{max}}]$	0.697	1072	0.000
(9) Brown Long-eared Bat	$([13.70 \leq t_{\text{Sep}}^{\text{max}} \leq 22.70] \vee [8.4111 \leq t_{\text{Nov}}^{\text{avg}} \leq 8.6444]) \wedge [17.30 \leq t_{\text{Jul}}^{\text{max}} \leq 28.40] \wedge [-8.15 \leq t_{\text{Jan}}^{\text{avg}} \leq 6.0083]$	0.693	963	0.000
(10) Harvest Mouse ∧ European Mole	$[-0.30 \leq t_{\text{Apr}}^{\text{min}} \leq 8.70] \wedge [19.40 \leq t_{\text{Aug}}^{\text{max}} \leq 27.20] \wedge [45.417 \leq p_{\text{Jun}}^{\text{avg}}] \wedge [48.75 \leq p_{\text{Aug}}^{\text{avg}} \leq 126.33]$	0.677	774	0.000
(11) (Serotine Bat ∨ Lesser Mole Rat) ∧ European Mole	$[19.70 \leq t_{\text{Jul}}^{\text{max}}] \wedge [16.90 \leq t_{\text{Sep}}^{\text{max}} \leq 23.70] \wedge [43.111 \leq p_{\text{Jul}}^{\text{avg}} \leq 149.5] \wedge [31.875 \leq p_{\text{Oct}}^{\text{avg}} \leq 119.5]$	0.634	664	0.000
(12) Wood Mouse ∧ Natterer's Bat ∧ Eurasian Pygmy Shrew	$([3.20 \leq t_{\text{Mar}}^{\text{max}} \leq 14.50] \wedge [17.30 \leq t_{\text{Aug}}^{\text{max}} \leq 25.20] \wedge [14.90 \leq t_{\text{Sep}}^{\text{max}} \leq 22.80]) \vee [19.60 \leq t_{\text{Jul}}^{\text{avg}} \leq 19.956]$	0.623	681	0.000

all redescrptions were considered significant with respect to these null hypotheses. The algorithm processed the full data in about 13 min.

Some of the results are displayed in Table 7. The redescrptions have varying support sizes: some cover only a small part of the data, while others cover almost the whole data. Yet, they all have very high accuracy. The first two redescrptions cover exactly the same area. They represent the Svalbard archipelago (see Fig. 8(a)). The climate in Svalbard is so different from other areas that it allows multiple ways to define it, causing multiple redescrptions. The fourth redescrption has only European Elk on the left-hand side, but the right-hand side is more complex, characterizing very accurately the environment in Scandinavia and Baltia (Fig. 8(b)), the area occupied by the European Elk. The remaining redescrptions are more complex. The fifth redescrption (Fig. 8(c)) covers Northern and Central Europe, while the last covers only Central Europe (Fig. 8(d)).

We point out that while the results of Garriga et al. [13] are superficially similar, the differences in the methods used and the goals pursued make the results incomparable.

9. CONCLUSIONS

We have presented a new algorithm to mine redescrptions from real-valued data. Unlike previous algorithms, ours does not require any pre-processing when used with non-Boolean data. It is based on a beam-search type of method. We have shown with both synthetic and real-world data sets that our algorithm performs better than its peers. In particular, the experiments show the benefits of on-the-fly bucketing against pre-processing.

The non-Boolean redescrption mining has many applications in various fields of science, of which niche-finding is the one we have studied here. One of the most prominent future works would be to collaborate with biologists and ecologists and explore the true value of redescrption mining in finding the bioclimate envelopes. Also other fields should be considered. Medical data describing patients where the matrices would contain genetic or physiological characteristics and symptoms, respectively, is one example.

While our algorithm seems to be working fine, it is by no means the final word. Building better algorithms is, as always, an important future direction. For a concrete

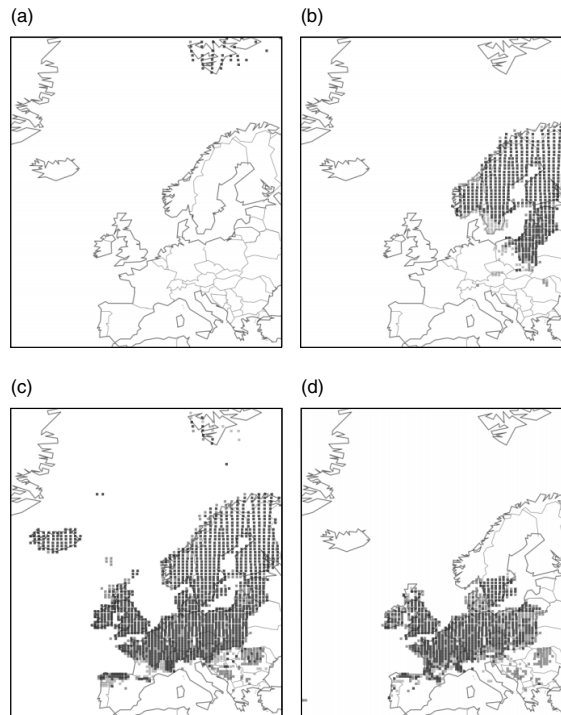


Fig. 8 Support of redescrptions when mining Bio data. (a) Row 1, (b) row 4, (c) row 5, and (d) row 12 in Table 7. Light gray, medium gray and dark gray respectively indicate areas where only the left query holds, only the right query holds and both queries hold.

example, the selection of initial pairs seems to have space for improvements.

Finally, having proofs of the behavior of the algorithms is important. Is there, for example, an algorithm for real-valued redescription mining for which one can prove that it finds a redescription, provided that the redescription is sufficiently strong?

ACKNOWLEDGEMENTS

The authors are grateful to Dr. Jussi Eronen for helpful comments and suggestions.

REFERENCES

- [1] J. Grinnell, The niche-relationships of the California Thrasher, *Auk* 34(4) (1917), 427–433.
- [2] R. G. Pearson and T. P. Dawson, Predicting the impacts of climate change on the distribution of species: Are bioclimate envelope models useful?, *Glob Ecol Biogeogr* 12 (2003), 361–371.
- [3] A. Gallo, P. Miettinen, and H. Mannila, Finding subgroups having several descriptions: Algorithms for redescription mining, In *Proceedings of SDM*, 2008, 334–345.
- [4] L. Parida and N. Ramakrishnan, Redescription mining: Structure theory and algorithms, In *Proceedings of AAAI*, 2005, 837–844.
- [5] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm, Turning CARTwheels: An alternating algorithm for mining redescrptions, In *Proceedings of KDD*, 2004, 266–275.
- [6] M. J. Zaki and N. Ramakrishnan, Reasoning about sets using redescription mining, In *Proceedings of KDD*, 2005, 364–373.
- [7] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. B. Muchnik, An implementation of logical analysis of data, *IEEE Trans Knowl Data Eng* 12(2) (2000), 292–306.
- [8] G. Tsoumakas, I. Katakis, and I. Vlahavas, Mining multi-label data, In *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, eds. Springer, 2010, 667–685.
- [9] P. K. Novak, N. Lavrac, and G. I. Webb, Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining, *J Mach Learn Res* 10 (2009), 377–403.
- [10] L. Umek, B. Zupan, M. Toplak, A. Morin, J.-H. Chauchat, G. Makovec, and D. Smrke, Subgroup discovery in data sets with multi-dimensional responses: A method and a case study in traumatology, In *Proceedings of AIME*, 2009, 265–274.
- [11] D. Leman, A. Feelders, and A. J. Knobbe, Exceptional model mining. In *Proceedings of ECML/PKDD*, 2008, 1–16.
- [12] M. van Leeuwen, Maximal exceptions with minimal descriptions, *Data Min Knowl Disc* 21(2) (2010), 1–18.
- [13] G. C. Garriga, H. Heikinheimo, and J. K. Seppänen, Cross-mining binary and numerical attributes, In *Proceedings of ICDM*, 2007, 481–486.
- [14] D. Kumar, Redescription Mining: Algorithms and Applications in Bioinformatics. Ph.D. Thesis, Department of Computer Science, Virginia Tech, 2007.
- [15] H. Grosskreutz and S. Rüping, On subgroup discovery in numerical domains, *Data Min Knowl Disc* 19(2) (2009), 210–226.
- [16] R. Srikant and R. Agrawal, Mining quantitative association rules in large relational tables, In *Proceedings of SIGMOD*, 1996, 1–12.
- [17] U. Fayyad and K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, In *Machine Learning*, 1993, 1022–1027.
- [18] J. Soberón and M. Nakamura, Niches and distributional areas: Concepts, methods, and assumptions, *PNAS* 106 (Suppl 2) (2009), 19644.
- [19] J. Soberón and A. T. Peterson, Interpretation of models of fundamental ecological niches and species' distributional areas, *Biodiv Inform* 2 (2005), 1–10.
- [20] V. Pericliev and R. E. Vladès-Pérez, Differentiating 451 languages in terms of their segment inventories, *Studia Linguistica* 56(1) (2002), 1–27.
- [21] M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila, Randomization methods for assessing data analysis results on real-valued matrices, *Stat Anal Data Min* 2(4) (2009), 209–230.
- [22] A. J. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Krystufek, P. Reijnders, F. Spitzenberger, M. Stubbe, J. Thissen, V. Vohralik, and J. Zima, *The Atlas of European Mammals*, London, Academic Press, (1999). www.european-mammals.org.
- [23] R. J. Hijmans, S. Cameron, L. Parra, P. Jones, and A. Jarvis, Very high resolution interpolated climate surfaces for global land areas, *Int J Climatol* 25 (2005), 1965–1978. www.worldclim.org.
- [24] M. J. Zaki, Scalable algorithms for association mining, *IEEE Trans Knowl Data Eng* 12(3) (2000), 372–390.
- [25] R. Bellman, On the approximation of curves by line segments using dynamic programming, *Comm ACM* 4(6) (1961), 284.

Article II

II

Esther Galbrun and Pauli Miettinen

A Case of Visual and Interactive Data Analysis: Geospatial Redescription Mining

*In Instant Interactive Data Mining Workshop
at the 2012 European Conference on Machine Learning and Principles and
Practice of Knowledge Discovery in Databases, ECML/PKDD'12
(Bristol, UK), 2012.*

Copyright © The Authors, 2012.

A Case of Visual and Interactive Data Analysis: Geospatial Redescription Mining

Esther Galbrun¹ and Pauli Miettinen²

¹ Helsinki Institute for Information Technology,
Department of Computer Science,
University of Helsinki, Finland,
galbrun@cs.helsinki.fi

² Max Planck Institute for Informatics,
Saarbrücken, Germany,
pmiettin@mpi-inf.mpg.de

Abstract. We present a method for visual and interactive geospatial redescription mining. The goal of geospatial redescription mining is to characterize geospatial areas using two different descriptions, such as their bioclimatic features and fauna. Indeed, one application of geospatial redescription mining is finding bioclimatic niches, i.e. explaining the distribution of species using their bioclimatic envelope. Allowing users to find the geospatial redescriptions in an interactive way, and to see the results in clear visualizations, is fundamental for the applicability of the method. We present several goals we think a good interactive and visual redescription mining method should fulfil, and we explain how our proposed method achieves (most of) them. Finally, we also discuss some open problems in interactive redescription mining.

1 Introduction

Finding multiple ways to characterize the same entities is a problem that appears in many areas of science. Describing geographical regions in terms of both their bioclimatic conditions and the fauna that inhabits them is one instance in the field of biology. A simple example of a redescription in this setting could state that areas where Moose live are areas where February's maximum temperature is between -10 and 0 degrees Celsius and July's maximum temperature between 12 and 25 degrees Celsius. This is actually the redescription shown in the foreground panel of Figure 1.

The results of redescription mining, the redescriptions, can be approached from two points of view. On one hand, by considering the variables and conditions appearing in the queries, which provide valuable information in themselves; on the other hand, by studying the support set of the redescriptions, i.e. the subset of entities where both queries of a redescription hold.

To analyse the redescriptions, the ability to visualize the support sets is very helpful. When building a tool for mining redescriptions from geospatial data, plotting the support on a map, as in the foreground panel of Figure 1, is a natural

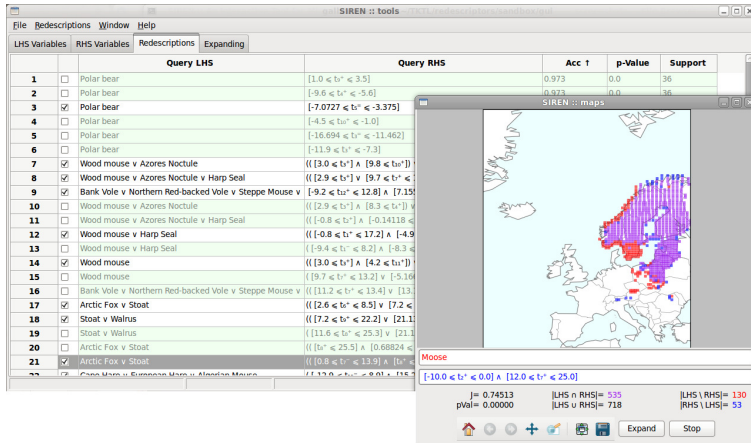


Fig. 1. The SIREN interactive mining and visualization tool. The panel in the background contains a list of redescrptions while the foreground panel displays the map of a selected redescription. In this example, left hand side queries are over fauna while right hand side queries are over monthly bioclimatic conditions, that is, temperatures and precipitation.

visualization. But a static display of the results is not enough: the user must also be able to interact with the program. This interaction can be conceptually divided into two sub-phases: interacting with the data mining algorithm and interacting with the result visualization. The analysis is an alternation of these two phases, with the user moving back-and-forth between issuing commands to find new results and examining obtained ones. We argue that a good interactive data mining tool should support both types of interaction and facilitate the alternation between different phases.

In this paper we give a systematic outline of contributive features to fulfill that aim, considering the example of mining geospatial redescrptions. We then present a pair of algorithms, ReREMI and SIREN, and explain how they implement interactivity and visualization for that task. Lastly, we discuss some possible pitfalls associated with interactive, visual mining. But first, we formally define the redescription mining problem.

2 Redescription Mining

Redescription mining aims at simultaneously finding multiple descriptions of a subset of entities which is not previously specified. This is in contrast with other methods like Emerging Patterns Mining (EPM), Contrast Set Mining (CSM)

and Subgroup Discovery (SD) (see [8] for a unifying survey) or general classification methods, where target subsets of entities are specified via labels. Currently, redescription mining is a purely descriptive approach, its predictive power remains to be explored. Since its introduction in [12] various algorithms have been proposed for Boolean redescription mining, based on approaches including decision trees [12, 6], co-clusters [9], and frequent itemsets [2]. In [1], we extended redescription mining to categorical and numerical variables.

More formally, we consider data that contains entities E with two sets of characterizing variables, e.g. the fauna and the bioclimatic conditions. Boolean variables can be interpreted as a truth value assignment in a natural way. For categorical and real-valued variables, truth value assignments are induced by relations denoted using Iverson bracket $[v = c]$ and $[a \leq v \leq b]$, respectively, where c is some category and $[a, b]$ an interval. For more details about how the actual optimal categories and intervals are chosen, please refer to [1]. These truth assignments and their negations constitute *literals* which can be combined using the Boolean operators \wedge (and) and \vee (or) to form *queries*. The support of a query q is the subset of entities for which the query holds true, that is $\text{supp}(q) = \{e \in E : q \text{ is true for } e\}$. We refer to the two sets of variables informally as left and right hand side data, and the queries over them as left and right hand side queries, denoted as $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$, respectively. Then, a redescription is simply a pair of queries over variables from the two sets, $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$. Its *accuracy* is measured using the *Jaccard coefficient*

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

We compute a p -value that represents the probability that two random queries with marginal probabilities (i.e. the fraction of entities supporting them) equal to those of $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$ have an intersection equal to or larger than $|\text{supp}(q_{\mathbf{L}}, q_{\mathbf{R}})|$. This probability uses the binomial distribution and is given by

$$\text{pvalM}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \sum_{s=|\text{supp}(q_{\mathbf{L}}) \cap \text{supp}(q_{\mathbf{R}})|}^{|E|} \binom{|E|}{s} (p_{\mathbf{L}})^s (1 - p_{\mathbf{L}})^{|E|-s},$$

where $p_{\mathbf{L}} = |\text{supp}(q_{\mathbf{L}})| / |E|$. The higher the p -value, the more likely it is to observe such a support for independent queries, and the less significant the query.

The task consists in finding significant accurate redescrptions, in other words, pairs of queries, one query for both sets of variables, such that both queries describe almost the same set of entities.

When the data is geospatial, that is, the entities are connected to geographical locations, the task is called *geospatial redescription mining*. A meaningful geospatial redescription should define coherent areas using expressive queries.

Niche-finding is a particular instance of geospatial redescription mining — and a task of great importance for biologists. The bioclimatic constraints that must be met for a certain species to survive constitute that species' bioclimatic

envelope, or niche [3]. Finding such envelopes can help, e.g. to predict the results of global warming [10]. A number of methods, involving regression, neural networks, and genetic algorithms (see [13]) have been developed over the past ten years to model the bioclimatic envelope, MAXENT [11] and BIOMOD [14], being good examples of modelling tools used in this domain. The former provides a graphical user interface while the latter is a text-based tool. But to the best of our knowledge, none of these methods allows automatically finding both the set of species and their envelope.

3 Goals for Interactive and Visual Redescription Mining

In this section we discuss our goals for an interactive and visual redescription mining tool. Some of these goals are general to any interactive and visual data mining tool (and we spend less time on discussing why they are desirable), some are specific to redescription mining. We divide the discussion between interaction and visualization, though we emphasize that these goals are not independent.

3.1 Visualization of Results

As a basis for our discussion, we use the taxonomy of interactions for visual analytics proposed by Heer and Shneiderman [4]. The bold-face terms correspond to their taxonomy.

The most fundamental goal when designing a tool for visual data analysis is, of course, to have a good **visualization**. With geospatial redescrptions, a map is the most natural option. Thus our tool should be able to plot the redescrptions on a map. But in order to effectively select the content of the visualizations, the user needs means to **filter** and **sort** the results mined. In the case of redescription mining, the user should be able to sort the returned redescrptions based on different criteria, such as accuracy, support size, statistical significance, or query length (i.e. number of literals). To some extent, filtering can be regarded as sorting with a cut-off value. Hence, sorting should naturally use the same criteria and similar results display as sorting. Additional criteria might affect sorting, including the described geographical area and redundancy.

During the analysis, the user should be allowed to **derive** new data. That is, new variables obtained by aggregating existing variables might better capture the studied phenomenon. Hence, their introduction during the mining process would support the analysis. While modifying the way the information is represented, deriving new variables is also a means to interact with the mining process.

In order to manipulate the views, the user needs to be able to **select** the data he wants to visualize. In the present case, he can primarily choose a redescription to plot. Then, he can edit the queries, modifying literals and altering the bounds of real-valued variables. The user might need to **navigate** inside the view, typically looking first at the redescription over the whole area, before zooming and panning to see more details. On a high level, the user might only be able to see whether either query hold on a region. Focusing on particular

area, he might obtain more detailed information about the actual state of the variables and what makes a query hold or not in a particular location, for instance by clicking or hovering over a dot in the map. Several views and the data might need to be **coordinated**. Modifications made to a redescription should be reflected immediately on the map(s). In addition, it could be useful to allow the user to bind maps together, so that panning and zooming are applied to all maps simultaneously. In that way, detailed comparison of the support of different redescriptions would be facilitated. Maps can be opened in detachable tabs, to be inspected side by side or sequentially and be **organized** using the system's or a dedicated windows tiling.

For any interactive tool, undo and redo are minimal functionalities to allow reverting actions, making interaction safe and comfortable. The user should be able to save the current status of the analysis process, i.e. all current redescriptions, opened lists and maps to punctuate the process. **Recording** the interaction history and turning it into editable and parameterizable macros provides a mean to repeat a sequence of actions and automate repetitive tasks. The tool should support **annotation** in order to keep track of the thought path during the analysis. For example, this could be achieved by generating annotable screen shots of the current window of interest, and by adding comments to the interaction history and macros. Organizing the history and macros into blocks would help to further clarify the logical structure of the analysis. Furthermore, with the ability to link to objects in the current environment, such as redescriptions, groups of entities or literals, these could be explicitly related to each other. Data analysis is often a collaborative effort, involving several users. Then, **sharing** information becomes crucial. Easy export and import of redescriptions lists, maps and macros, possibly with comments and annotations is a very important feature towards that aim. Finally, giving clear names to the actions and providing feedback on their application helps **guiding** users along the analysis process. Example macros with detailed explanations, to be replayed step-by-step, represent a good means to introduce new users to the tool. These latter goals pertain automating interactions, attaching a meaning to sequences of interactions, allowing segmented interactivity, e.g. when different users collaborate, using the tool in turn. Hence they are also closely tied to the interaction with the mining process, to which we now turn.

3.2 Interaction with the Mining Process

A desirable behavior for an interactive program is the production of meaningful results at any time. In other words, if the mining process is stopped, it can nevertheless return results which are valid, albeit possibly partial. This is related to the possibility to obtain preliminary results while the mining is still underway. Such a feature contributes to the ability of the program to respond quickly to instructions from the users. It is also possible to first run the algorithm allowing only short queries, say, at most a couple of literals on either side, and let the user choose the ones that seem promising and should be further extended. Low latency or even instantaneity is a core quality of an interactive tool and is important

to catch and keep the user’s attention. At least, the tool should provide instant feedback about what is happening.

The automation level of the whole mining process could be adaptative. From fully manual, where the users writes down redescrptions and the tools simply evaluates them, to fully automated where the program mines the list of best redescrptions using static predefined constraints, it could also be partially automated, with the tool suggesting best extensions at each step and asking for approval from the user.

Consider extending an existing redescription with a real-valued literal. Instead of a map plot based on their geographical location, a figure where the areas are represented as colored dots plotted along the x -axis depending on the value taken by the chosen variable would be useful for determining the optimal interval for that variable. Indeed, the user could observe which values occur in locations that belong to different parts of the current redescription. Then, he could fix the bounds in consequence, for example using sliders. The tool could even indicate which are the best bounds but also update the best corresponding upper bound when the user moves the lower bound, and vice versa. This is a prime example of instant interaction with the mining process through apt visualization.

Proper interactivity with the program also requires allowing the user to specify constraints for the search. Possible constraints include specifying variables or geographical areas that should be excluded from the redescrptions or modifying the minimum acceptable accuracy. For such constraints that constitute filtering criteria, there can be three different degrees of integration with the algorithm, that is, depending on how far they are pushed into the mining process instead of applied a posteriori. The degree zero of integration happens when the user manually filters the raw output. Instead, the program can automatically filter its results before reporting. The highest degree of integration implies incorporating such filtering criteria during the search to avoid generating the unwanted results in the first place. Still, a compromise needs to be found between supporting deep integration and accepting a broad range of constraints, e.g. through a flexible specification framework. Indeed, these are typically two conflicting goals.

More generally, the user should be able to specify interest and lack of it. Selecting a redescription to be edited and extended is a way of expressing curiosity towards the involved conditions or area. Similarly, he should be able to prevent the algorithm to search further in directions he deems uninteresting. One way of doing so is to merely pick out variables or locations that should be ignored. Another way is to select a redescription and specify that results of this kind are of no interest.

4 Our Proposed Tool

In this section, we present our proposed tool, which consists of a pair of algorithms, REREMi and SIREN. First, we explain how it implement interactivity and visualization for redescription mining. Then, we give a concrete illustration

of its usage by means of a use case. The current version of the tool is only able to achieve part of the goals stated in the previous section. In this section we focus on presently available features—the others are *châteaux en Espagne*.

4.1 The Algorithms

SIREN is an interactive tool for mining and visualizing geospatial redescrptions.³ At its core is the REREMI redescription mining algorithm [1].

This greedy algorithm uses an efficient on-the-fly discretization technique to extend redescription mining to categorical and numerical variables. It considers queries over such variables that can be parsed in linear order, without trees, with every variable allowed to appear only once. They constitute a subset of Boolean formulae that provides a good compromise between expressive power, difficulty of the search, and interpretability.

Yet, the search space remains exponential and we still resort to heuristic pruning. We use a strategy similar to beam-search to explore the solution space. The basic idea is to construct queries bottom-up, starting from singleton redescrptions (i.e. both queries contain only one literal) and progressively extending them by appending operators and literals. After evaluating all possible one-step extensions, we select the best candidates and extend them in turn. This process stops when no new redescription can be generated.

Redescrptions with too high p -value can be filtered out during the search. We exploit some simple observations to make the computation of accuracy more efficient. This allows to evaluate candidates faster, which is particularly important for an interactive tool.

Owing to his beam-search-like behaviour, REREMI is an any-time algorithm. The intermediate redescrptions explored during the search are returned at each step. This way, the user is able to see the candidates present in the beam and might stop the extension process if he so wishes. The possibility to remove a candidate from the beam, cutting off a less promising branch from the search, remains to be implemented.

In SIREN, threading is employed to delegate mining tasks to REREMI in the background. This preserves the tool's responsiveness while the communication is maintained to provide feedback about the ongoing mining, to return results as they are obtained and to allow the user to directly interact with the process.

Finally, SIREN allows automatic filtering of redundant redescrptions. That is, redescrptions that cover approximately the same area even if they have (some-what) different sets of variables. The user can select a redescription and ask SIREN either to filter out all redescrptions that are redundant with respect to the selected one, or to go through the whole list of redescrptions filtering out all redescrptions that are redundant with respect to some earlier-encountered (i.e.

³ More details about SIREN's features, additional screenshots and a demonstration video are available online at <http://www.cs.helsinki.fi/u/galbrun/redescriptors/siren/>.

better) redescription. Naturally, the decisions made by SIREN can be reverted whenever the user wishes to.

SIREN and REREMI are implemented in Python. The interface is built with the `wxPython` Open Source GUI toolkit, ensuring cross-platform compatibility. The `matplotlib` library enables to generate high quality figures, seamlessly integrated in the interface. SIREN allows for simple editing of the redescrptions thanks to flexible parsing of different representations. It can handle any data provided in a compatible format.

4.2 Use Case

We exemplify the usage of SIREN by going through a generic work-flow of mining geospatial redescrptions, detailing typical steps in the process. This specific example concerns the application of SIREN on the task of bioclimatic niche finding using data that describes spatial areas of Europe, squares of side roughly 50 kilometers. The left hand side data contain information about the mammals that live in these areas, while the right hand side consists of bioclimatic variables⁴. Nonetheless, SIREN is a flexible tool that can be used with different datasets from various application domains.

Initial redescription mining. A natural starting point for the analysis of any given data is to use a redescription mining algorithm to find an initial set of redescrptions. This can be done within SIREN by running the extension mechanism on an empty redescription. Following the principle of first providing an overview of the results then focusing on specific items, the redescrptions found are presented as a list from which the user can select a redescription of his choice to examine more closely and plot on the map. Figure 1 shows two panels, containing an overview of the current results as a list, in the background, and a single redescription plotted on a map, in the foreground. The list supports sorting and filtering on various criteria.

Extending a redescription. Sometimes the user wants to focus only on one of the queries, on some particular variable of interest or on a part of an existing redescription. SIREN allows the user to automatically extend a given redescription, i.e. let the algorithm add new literals to the queries to make the redescription as accurate as possible.

In the climatic niche-finding task, for instance, we might select a species, say, the Southwestern Water Vole and look for best extensions starting from that single variable. Here, the best found extension has accuracy 0.665 (per Jaccard coefficient):

Southwestern Water Vole \vee Gray Dwarf Hamster \vee Savi’s Pine Vole
 \vee Mediterranean Monk Seal

$$[11.2 \leq t_3^+] \wedge [0.51 \leq t_1^- \leq 11.333] \wedge [42.75 \leq p_{10}^- \leq 131.81] \\ \wedge [50.556 \leq p_{11}^- \leq 176.75],$$

⁴ The data comes from two publicly available datasets: European mammal atlas [7] and Worldclim climate data [5].

This redescription indicates that areas where any of the four species lives correspond to areas where the maximum temperature in March is above 11.2 degrees Celsius, the average temperature in January between 0.51 and 11.333 degrees Celsius and the average precipitations in October and November range from 42.75 to 131.81 millimeters and from 50.556 to 176.75 millimeters, respectively.

Returned extensions can be plotted on maps opened inside several windows, so as to be visualized side by side and compared as shown in Figure 2.

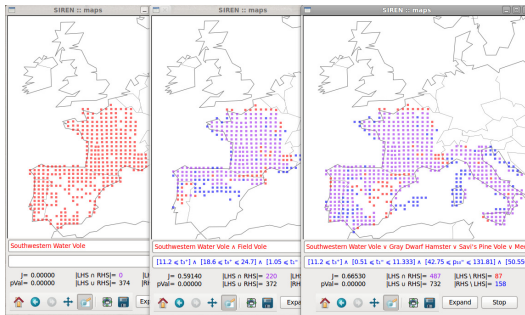


Fig. 2. Several map panels. Comparing intermediate extensions automatically generated for a chosen starting variable. Red, blue and purple represents areas where the left hand side query holds, only the right hand side query holds and where both queries hold, respectively.

Editing a redescription. It is typical that the user wants to edit some of the obtained redescriptions. For example, some results might be overly complex, or have exceedingly precise boundaries for numerical variables. The user can easily select a redescription to modify, open it in a map panel and edit it. Boundaries can be altered, literals added or removed. SIREN instantly updates the map and important statistics (accuracy, p -value, etc.) of the redescription, allowing the user to see the effects of the modifications immediately and verify, e.g. whether the new redescription would still be acceptably accurate.

Continuing with our example above, we might want to reduce the precision of the climatic constraints to integers. We could edit the query as follows:

$$[11 \leq t_3^+] \wedge [0 \leq t_1^- \leq 12] \wedge [42 \leq p_{10}^- \leq 132] \wedge [50 \leq p_{11}^- \leq 177],$$

and obtain a redescription of slightly decreased accuracy.

Using subsets of variables. SIREN allows the user to specify variables to temporarily avoid when extending or mining redescriptions. In our running example, we might want to force the algorithm to search alternative redescriptions that do not involve any precipitation. For that purpose, we simply unselect

all such variables before running the extension anew. We will obtain the best extensions containing only temperatures in the bioclimatic query, such as the following redescription of accuracy 0.653:

Southwestern Water Vole \vee Cape Hare \vee Savi’s Pine Vole
 \vee Mediterranean Monk Seal

$$([11.2 \leq t_3^+] \wedge [20.1 \leq t_7^+ \leq 32.9] \wedge [0.51 \leq t_1^- \leq 11.333]) \vee [34.0 \leq t_8^+].$$

Note that this redescription was not returned previously since the beam search focused on better ones involving precipitation variables. In addition to basic parameter tuning, this feature allows the user to specify additional constraints, thereby interacting with the mining process to adjust it according to his interest and what appears most promising during the analysis.

Filtering redundant redescrptions. The results returned during the extension mentioned previously may contain many redundant redescrptions found at different steps. We can easily sort them, e.g. by accuracy, select one of interest and filter all the following results redundant with respect to it.

Sharing the results. Finally, SIREN facilitates distributing the results: redescrptions can be exported in easy-to-read format and the maps associated to redescrptions can be readily converted to publication-ready graphics.

5 Discussion

This paper presents a tool for interactive and visual redescription mining. While we believe that the goals—and the methods we present to achieve them—are easy to accept as reasonable, we want to point out that there are still many open problems, both conceptual and technical, that need to be solved.

In the heart of interactive data mining is the user’s ability to tell the algorithm that he wants more or less certain type of results. In principle, this is not a problem in SIREN: the user simply selects a redescription he wants to remove from the beam search or extend more. The problem, however, is that there can be (and usually are) other, similar redescrptions that the user might also want to remove or extend. He can do that manually, of course, but with larger number of redescrptions, the process becomes unbearably tedious very soon.

A solution to this problem would be to remove (or extend) all similar redescrptions. But how to define the similarity? To give an example, consider a case when the user finds a redescription saying that the area where the Polar Bear lives is the area with January’s mean temperature below -20 degrees Celsius, in other words, Polar Bear lives in cold. This is hardly a surprising result, and the user might want to remove it (and other similar results) from the search. But we can characterize the cold areas using other variables than just January’s mean temperature, so it is not enough to just stop extending any redescription with Polar Bear and January’s mean temperature in it. Also, we cannot just remove all the redescrptions with Polar Bear—that could remove some very interesting redescrptions, too. Finally, we could consider the area in which the redescription

holds. But even that leaves a lot to be hoped for: if we remove all redescrptions that contain that area, we probably remove too many redescrptions, but if we instead remove redescrptions contained in the area, we probably miss most of the redescrptions we should remove.

The problem of removing and extending similar redescrptions is closely related to that of redundancy reduction. There are often multiple redescrptions that represent the same phenomenon (think of the Polar Bears living in the cold areas), and ideally, we would like to present only one of them to the user. In other words, we do not want to present to the user any redescrptions that do not add any (or add only marginally) new information over the redescrptions he has already seen. But as with deciding which redescription is similar to a selected one, also quantifying the redundancy between redescrptions is a difficult problem.

When interpreting a redescription, one should always bear in mind the assumptions attached to it. For example, whether some variables were disabled or whether the focus was put on some area when it was generated. Hence, keeping track of the constraints used when mining a redescription is essential. However, if the user is allowed to stop the extension process, modify the constraints and resume the search, this might be fairly intricate and interpretation of the results become impossible.

The goal of data mining is to find new and interesting information from the data. In interactive data mining in general, and with the tools discussed in this paper in particular, the user can guide the data mining method towards the results he prefers. This raises new problems. First, we have to control that the data supports the results the user finds and second, we must be careful that the user actually finds new information, not just the information he already knew.

The first problem, making sure that the obtained results are supported by the data, is ages old in sciences. In short, it is the question of testing the significance of a hypothesis, and there is a vast body of statistical literature about it. Our proposed algorithms mitigate the problem by computing a p -value, but as it is based on a fixed null hypothesis, it is not adequate in every case.

The second problem is more conceptual: taken to an extreme, the interactivity removes the data mining from the interactive data mining. If the user more or less directly tells the algorithm the redescription he wants to see, the SIREN program turns into a mere plotting interface. Even on the less extreme case, the user can easily (an unwittingly) guide the algorithm towards the kind of results he wanted to see. Together with the fact that we can only check against a fixed null hypothesis, this causes a considerable risk of false findings. The onus is on the user to make sure he does not misuse the algorithm.

References

1. Galbrun, E., Miettinen, P.: From Black and White to Full Colour: Extending Redescription Mining Outside the Boolean World. *Statistical Analysis and Data Mining* (2012), in press

2. Gallo, A., Miettinen, P., Mannila, H.: Finding subgroups having several descriptions: Algorithms for redescription mining. In: SDM. pp. 334–345 (2008)
3. Grinnell, J.: The niche-relationships of the California Thrasher. *The Auk* 34(4), 427–433 (1917)
4. Heer, J., Shneiderman, B.: Interactive dynamics for visual analysis. *Commun. ACM* 55(4), 45–54 (2012)
5. Hijmans, R.J., Cameron, S., Parra, L., Jones, P., Jarvis, A.: Very high resolution interpolated climate surfaces for global land areas. *Int. J. Climatol.* 25, 1965–1978 (2005), www.worldclim.org
6. Kumar, D.: Redescription mining: Algorithms and applications in bioinformatics. Ph.D. thesis, Department of Computer Science, Virginia Tech (2007)
7. Mitchell-Jones, A.J., Amori, G., Bogdanowicz, W., Krystufek, B., Reijnders, P., Spitzenberger, F., Stubbe, M., Thissen, J., Vohralik, V., Zima, J.: *The atlas of European mammals*. Academic Press, London (1999), www.european-mammals.org
8. Novak, P.K., Lavrac, N., Webb, G.I.: Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.* 10, 377–403 (2009)
9. Parida, L., Ramakrishnan, N.: Redescription mining: Structure theory and algorithms. In: AAAI. pp. 837–844 (2005)
10. Pearson, R.G., Dawson, T.P.: Predicting the impacts of climate change on the distribution of species: Are bioclimate envelope models useful? *Global Ecol. Biogeogr.* 12, 361–371 (2003)
11. Phillips, S., Anderson, R., Schapire, R.: Maximum entropy modeling of species geographic distributions. *Ecological modelling* 190(3), 231–259 (2006)
12. Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., Helm, R.F.: Turning CARTwheels: An alternating algorithm for mining redescrptions. In: KDD. pp. 266–275 (2004)
13. Soberón, J., Peterson, A.T.: Interpretation of models of fundamental ecological niches and species’ distributional areas. *Biodiv. Inform.* 2(0) (2005)
14. Thuiller, W., Lafourcade, B., Engler, R., Araújo, M.B.: Biomod – a platform for ensemble forecasting of species distributions. *Ecography* 32(3), 369–373 (2009), <http://dx.doi.org/10.1111/j.1600-0587.2008.05742.x>

Article III

Esther Galbrun and Angelika Kimmig

Finding Relational Redescriptions

In *Machine Learning*, Published Online First, 2013.

DOI: <http://dx.doi.org/10.1007/s10994-013-5402-3>

Copyright © 2013, The Authors.

III

Finding relational redescription

Esther Galbrun · Angelika Kimmig

Received: 13 December 2012 / Accepted: 25 July 2013
© The Author(s) 2013

Abstract We introduce relational redescription mining, that is, the task of finding two structurally different patterns that describe nearly the same set of object pairs in a relational dataset. By extending redescription mining beyond propositional and real-valued attributes, it provides a powerful tool to match different relational descriptions of the same concept.

We propose an alternating scheme for solving this problem. Its core consists of a novel relational query miner that efficiently identifies discriminative connection patterns between pairs of objects. Compared to a baseline Inductive Logic Programming (ILP) approach, our query miner is able to mine more complex queries, much faster. We performed extensive experiments on three real world relational datasets, and present examples of redescriptions found, exhibiting the power of the method to expressively capture relations present in these networks.

Keywords Redescription mining · Relational query mining · Inductive Logic Programming · Graph mining · Relational data mining

1 Introduction

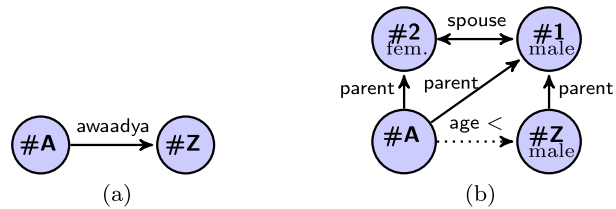
With the increasing amount of data available from heterogenous sources nowadays, establishing links between different perspectives on the same concept becomes ever more important, as recognized, for instance, in schema matching and ontology alignment for the

Editors: Fabrizio Riguzzi and Filip Železný.

E. Galbrun (✉)
Department of Computer Science and Helsinki Institute for Information Technology HIIT, University of Helsinki, P.O. Box 68, 00014 Helsinki, Finland
e-mail: esther.galbrun@cs.helsinki.fi

A. Kimmig
Departement Computerwetenschappen, KU Leuven, Celestijnenlaan 200A, bus 2402, 3001 Heverlee, Belgium
e-mail: angelika.kimmig@cs.kuleuven.be

Fig. 1 Example of redescription from the `Kinship` dataset: (a) kinship relation *Awaadya* between ($\#A$, $\#Z$) and (b) corresponding genealogical connection. The description formalism is introduced in Sect. 2



semantic web (Shvaiko and Euzenat 2005) and can contribute to the discovery of patterns in knowledge bases (Galárraga et al. 2013). One way of creating such links is to find sets of objects together with their descriptions in different terminologies, as done in *redescription mining* (Ramakrishnan et al. 2004; Galbrun and Miettinen 2012). However, this technique has so far only considered propositional or real valued attributes. In this paper, we extend redescription mining to the relational or network-based setting. In other words, we consider the task of finding two structurally different patterns that describe nearly the same set of object pairs in a relational dataset. We focus on a restricted language (binary relations with object identity) amenable to efficient graph mining techniques, resulting in an exploratory method that relies primarily on occurrence information and requires no extensive background knowledge such as a declarative bias.

Consider the following example from the `Kinship` dataset, which provides information about kinship terminology and family relationships within an Australian indigenous community (cf. Sect. 6). In Fig. 1, graph (a) represents the kinship relation *Awaadya* between the speaker $\#A$ and another person $\#Z$, corresponding to the relation between a child and his older brother, as given by graph (b). These two graphs are alternative ways to describe the same pairs of individuals ($\#A$, $\#Z$) and hence form a redescription.

Given a dataset and a query language, the underlying principle of redescription mining is to find pairs of queries, i.e. descriptions, that are structurally different yet describe (nearly) the same entities. Propositional redescription mining considers as its entities single objects characterized by their individual properties. Instead, in this novel relational setting, entities consist of pairs of objects characterized by both relations linking them and individual properties.

To find pairs of descriptions, redescription mining can adopt an alternating approach: one description is fixed, the other one is updated, and roles are swapped in the next iteration (Ramakrishnan et al. 2004). Following this approach, we present ARRМ, an algorithm for Alternating Relational Redescription Mining.

Our alternating scheme relies on an approach to finding relational descriptions, such as relational query mining (Dehaspe and Toivonen 1999; De Raedt and Ramon 2004). However, the generate-and-test approach of query mining systems requires large numbers of expensive coverage tests based on subgraph isomorphism. More importantly, they typically do not ensure that patterns connect the nodes of interest, thus producing many patterns that do not correspond to redescrptions. Hence, we propose an efficient algorithm, which we call FPQM, that finds descriptions for a given set of example pairs. It first mines for path patterns that connect many example pairs, then combines those into more expressive graph patterns. This reduces the number of coverage tests needed by constructing queries based on the data.

A comparison of FPQM to a baseline ILP tool on real world data shows that our approach can identify complex descriptions matching known ones, and is much faster. Examples of results obtained with ARRМ illustrate the power of the method to capture the relations present in a network. Compared to individual relational queries, the added expressivity brought by

redescriptions permits to better elucidate different uses of the same predicate and find more accurate patterns.

This paper extends upon our previous work (Galbrun and Kimmig 2012). As our main contributions, we define relational redescription mining, present novel algorithms to solve this task and provide an experimental evaluation of the proposed approach. Compared to the earlier version, we improved the combination and selection procedures of the query mining algorithm and realized an alternating scheme to address the full relational redescription mining problem. Also, we considered two additional datasets in our experiments and performed an extensive evaluation of the alternating scheme, including comparisons to several baselines.

We proceed as follows. Section 2 introduces relational redescription mining, Section 3 discusses related work, Sections 4 and 5 present the proposed path-based relational query miner and alternating scheme for relational redescription mining, respectively. Both are then experimentally evaluated in Sect. 6, before we conclude in Sect. 7.

2 Definitions and notations

This paper introduces *relational redescription mining*, the task of finding two structurally different queries that describe nearly the same set of object pairs in a relational dataset. Informally, we view descriptions as connected graphs expressed in terms of attributes of the data. For instance, graph (b) in Fig. 1 is an example of such descriptions for the objects of interest #A and #Z in terms of relations and attributes `spouse`, `parent`, `male`, `female` and `age`[<]. We now introduce the concepts required for a more formal definition of the problem. We focus on binary relations, as these can be represented in the form of graphs, allowing us to base our algorithms on graph concepts.

We view *relational data* as a directed graph $(\mathcal{O}, \mathcal{R})$, where nodes correspond to the objects \mathcal{O} , and edges to relations \mathcal{R} between them. Two families of functions, \mathcal{N} and \mathcal{E} , label nodes and edges with their *attributes*, respectively.

For instance, in the kinship domain, \mathcal{O} is the set of individuals from the community. Node attributes are $\mathcal{N} = \{\text{sex}, \text{age}\}$ and edge attributes $\mathcal{E} = \{\text{kin}, \text{gen}\}$, where `kin` maps into the set of kinship terms and the values of `gen` are the genealogical relations `parent` and `spouse`.

From node and edge attributes, we obtain three types of Boolean functions, or predicates, that serve as basic building blocks of queries. The first type, a *node predicate* $v_{N_i}^v(o)$, is true for an object o if and only if the node label $N_i(o)$ is defined and takes value v . The second type, an *edge predicate* $u_{E_i}^u(o_1, o_2)$, is true for a pair of objects (o_1, o_2) if and only if the edge label $E_i(o_1, o_2)$ is defined and takes value u . The third type, a *comparison predicate* $\phi_{N_i}^{rel}(o_1, o_2)$ for a binary relation *rel* over the range of node labeling function N_i is true for a pair of objects (o_1, o_2) if and only if both node labels $N_i(o_1)$ and $N_i(o_2)$ are defined and $rel(N_i(o_1), N_i(o_2))$ holds.

As an example, graph (b) in Fig. 1 uses, among others, node predicates $v_{\text{sex}}^{\text{male}}(\#1)$ and $v_{\text{sex}}^{\text{female}}(\#2)$, edge predicates $\epsilon_{\text{gen}}^{\text{parent}}(\#A, \#1)$ and $\epsilon_{\text{gen}}^{\text{spouse}}(\#1, \#2)$, and comparison predicate $\phi_{\text{age}}^<(\#A, \#Z)$.

For an object o , the set $F_N(o)$ of its *node features* contains the node predicates that hold true for that object. For a pair of objects (o_1, o_2) , the sets $F_E(o_1, o_2)$ and $F_C(o_1, o_2)$ of *edge* and *comparison features* contain the edge and comparison predicates that hold true for that pair, respectively. Note that the data, or network, is fully specified by the features of all objects, which implicitly provide all relevant information about the objects and their relations and attributes.

A *graph query* is a definite clause of the form $q(X, Y) :- b_1, \dots, b_n$, where the body elements b_i are node, edge or comparison predicates, q is a special predicate denoting the pattern and the *query variables* X and Y in the head also occur in the body. Instantiations of query variables are the object pairs of interest. We require graph queries to be *linked*, meaning that the set of edge predicates in the body connects the query variables. That is, a query is linked if there exists a sequence of variables Z_0, \dots, Z_k with $Z_0 = X, Z_k = Y$, such that for all $i = 1, \dots, k$, there is an index j such that $b_j \in F_E(Z_{i-1}, Z_i) \cup F_E(Z_i, Z_{i-1})$. A *path query* is a graph query whose query variables are connected by an acyclic path consisting of all edge predicates in the body. We denote the set of attributes for which the body of query q contains predicates by $\text{att}(q)$.

In the remainder of this paper, we use $\#A$ and $\#Z$ to denote the query variables and $\#1, \#2, \text{et cetera}$ to denote any other intermediate variables. Node attributes are indicated inside the node under the identifier, using lowercase, as for the edge attributes also. For instance, graph (b) in Fig. 1 corresponds to the graph query

$$q_b(\#A, \#Z) :- \epsilon_{\text{gen}}^{\text{parent}}(\#A, \#1), \quad \epsilon_{\text{gen}}^{\text{parent}}(\#A, \#2), \quad \nu_{\text{sex}}^{\text{male}}(\#1), \\ \epsilon_{\text{gen}}^{\text{spouse}}(\#1, \#2), \quad \epsilon_{\text{gen}}^{\text{spouse}}(\#2, \#1), \quad \nu_{\text{sex}}^{\text{female}}(\#2), \\ \epsilon_{\text{gen}}^{\text{parent}}(\#Z, \#1), \quad \phi_{\text{age}}^<(\#A, \#Z), \quad \nu_{\text{sex}}^{\text{male}}(\#Z).$$

This query has attribute set $\text{att}(q_b) = \{\text{sex}, \text{gen}, \text{age}\}$ and is linked due to the spouse and parent edges. Note that the $\text{age}^<$ edge in the graphical representation corresponds to a comparison predicate and is thus not considered for linkage and represented with a dotted line.

As common in graph mining, we use subgraph isomorphism or, in terms of logic, θ_{O_I} -subsumption (Esposito et al. 1994), to match queries against the data graph. That is, each variable in the query has to be matched to a different node in the graph, respecting the predicates in the query body. This choice is motivated by the intuitiveness and interpretability of the resulting queries and the ease of search. We denote such a match of variables V_j to objects o_i by the corresponding substitution $\theta = \{V_1/o_{i_1}, \dots, V_n/o_{i_n}\}$; θ reduced to query variables is called *answer substitution*. The set of all (distinct) answer substitutions of query q on the given network is its support, $\text{supp}(q)$. For instance, for query q_b above, the support contains all pairs of nodes (n_a, n_z) such that when matching $\#A$ to n_a and $\#Z$ to n_z , there is at least one match of $\#1$ and $\#2$ to other nodes that satisfies the query body.

For simplicity, we use the closed world assumption throughout this work. In other words, given a set of positive example pairs, all remaining pairs are considered to be negative examples. Altering the functions for scoring and filtering the queries presented in Sect. 4.3 allows to modify this assumption.

For a given set of example pairs E^+ and query q , we denote the sets of true positives (example pairs covered), false positives (other pairs covered) and false negatives (example pairs not covered) by $E_{1,1} = \text{supp}(q) \cap E^+$, $E_{0,1} = \text{supp}(q) \setminus E^+$, and $E_{1,0} = E^+ \setminus \text{supp}(q)$, respectively.

A redescription is a pair of queries $R = (q_L, q_R)$. Extending the previous notation, let $E_{1,0}$ be the set of entities (i.e. object pairs) which support only the first query (i.e. $E_{1,0} = \text{supp}(q_L) \setminus \text{supp}(q_R)$), $E_{0,1}$ those which support only the second query and $E_{1,1}$ those which support both queries. The *accuracy* of a redescription is commonly measured using the Jaccard coefficient, that is,

$$J(q_L, q_R) = \frac{|\text{supp}(q_L) \cap \text{supp}(q_R)|}{|\text{supp}(q_L) \cup \text{supp}(q_R)|} = \frac{|E_{1,1}|}{|E_{1,0} + E_{0,1} + E_{1,1}|}.$$

This measure takes values in the unit interval without using the number of all existing pairs of nodes for scaling, which would result in practically undistinguishable values. Furthermore, $E_{1,0}$ and $E_{0,1}$ are weighted equally, in agreement with the symmetric view of re-description mining.

In order to avoid trivial re-descriptions in the form of queries that are small variants of each other, we impose a syntactic requirement on the queries, which need to have disjoint attribute sets. While it would also be possible to require the use of attributes from two disjoint vocabularies, such a fixed split may not always be easily specified upfront. If available, such a split can be incorporated into the disjointness condition.

Given this background, we define *relational re-description mining* as follows:

Problem 1 (Relational re-description mining) *Given a relational dataset in the form of node, edge and comparison features $\{F_N, F_E, F_C\}$ and an accuracy threshold j , find re-descriptions (q_L, q_R) such that $\text{att}(q_L) \cap \text{att}(q_R) = \emptyset$ and $J(q_L, q_R) \geq j$.*

For simplicity of exposition, we restricted our discussion to queries of arity two. However, the definitions extend naturally to queries of higher arity, that is, queries whose body consists of binary predicates but with more than two variables occurring in their head.

3 Related work

Redescription mining emphasizes the insights obtained from expressive, interpretable patterns and their instances in the given data rather than prediction over unseen data. Relational pattern languages are thus a natural candidate for re-descriptions, but existing approaches have focused on propositional features (Ramakrishnan et al. 2004) and real-valued attributes (Galbrun and Miettinen 2012). They operate on matrices with a row for every object in the data and a column for every attribute or feature. Our ARRM algorithm follows an alternating scheme similarly to the former method.

These approaches have been shown to find simple re-descriptions of single nodes in a bibliographic network, describing a researcher either in terms of the conferences he publishes at, or in terms of his co-authors, that is, using only attributes in the form of the labels of neighboring nodes (Gallo et al. 2008; Galbrun and Miettinen 2012). However, as we illustrate in Sect. 6, considering pairs of objects and features based on their connections in terms of complex longer distance relations would inflate the size of the matrix and thus the search space of the algorithm. In contrast, our relational approach dynamically adapts the feature space to the subtask at hand, allowing for more focused exploration of connection patterns.

Learning relational queries is a key goal in Inductive Logic Programming (ILP) and multi-relational data-mining, and a central component of our relational re-description mining scheme. Multi-relational query miners often use a refinement operator to extend frequent queries found at the previous level, typically by adding a literal with at least one already used variable to the end of the clause body (Dehaspe and Toivonen 1999; De Raedt and Ramon 2004). While this principle results in connected clauses for unary patterns, patterns of higher arity are likely to ignore some of the query variables, or to contain disconnected components around individual query variables, and thus fail to provide insight into the relations between them. This connectivity problem has been addressed by relational pathfinding (Richards and Mooney 1992; Ong et al. 2005) and function learning (Santos et al. 2009). Pathfinding refines clauses by adding a sequence of literals if no single lit-

eral is able to connect query variables, where candidate sequences are generated based on connections of a single example's query variables in the data rather than by enumerating abstract paths. Function learning avoids evaluating unconnected queries by generating candidate queries from individual examples. The queries in our method are similarly anchored in the data, but are directly selected based on their frequency across all examples, in an approach inspired by graph mining techniques (Yan and Han 2002). Furthermore, to reduce the search-space, logic-based methods typically make heavy use of declarative bias, which needs to be provided by the user. This is not the case with our method, where the search space is pruned using solely occurrence information computed on the dataset.

Relational patterns play an important role in various techniques that explore and analyze relational datasets. For instance, the aim of subgroup discovery (Wrobel 1997; Lavrac et al. 2002) is to identify groups of objects that differ from the overall population in an interesting way. There, individual objects are described in terms of the relations they participate in. In contrast, relational redescription mining is interested in finding pairs of queries that describe different connection patterns between two objects. In the context of making predictions based on relational patterns, query mining has also been extended to learn association rules with conjunctive heads (Goethals and Van den Bussche 2002), which can be seen as associations between conjunctive redescriptions, and to flexible numbers of query variables (Goethals et al. 2005). Yet, the type of rules mined is fairly restrictive compared to the descriptions considered in our approach. These complex association rules are similar to the tuple-generating dependencies used in schema mapping and data exchange. The CLIO system learns such mappings by exploiting relational dependencies in the two schemata, where the user indicates some correspondences between attributes, as input (Miller et al. 2000). Our work uses shared objects to determine attribute correspondences, and does not rely on explicit information on how relations can be combined. Aligning or mapping objects can also be considered part of the overall discovery process, as for instance in the PARIS approach to ontology alignment (Suchanek et al. 2011), which discovers correspondences on the level of both instances and schemas. But as most other approaches to schema matching (Shvaiko and Euzenat 2005), PARIS focuses on one-to-one mappings of relations and does not consider more complex queries. A recent exception is the work of Zhang et al. (2012), who nevertheless only consider paths up to length four and rely on approximation schemes for Markov Logic inference to keep the approach feasible.

Recently, there is increasing interest in large knowledge bases like DBpedia (Auer et al. 2007), NELL (Carlson et al. 2010) or YAGO (Suchanek et al. 2007), which store binary relations between millions of objects. Such massive amounts of incomplete and often noisy information are a challenge for most existing relational learning approaches, and call for adapted representations and learning methods (Rettinger et al. 2012), such as the graph based techniques used in our redescription miner. Nebot and Llavori (2012) propose to extract relational association rules from a medical ontology by applying frequent itemset mining. Their method, however, requires extensive expert knowledge. In the context of NELL, weighted combinations of path patterns, learnt based on random walks, have been used for retrieval tasks (Lao et al. 2011; Lao and Cohen 2010). In contrast to our approach, paths are not combined into graphs, but are instead weighted by their importance. Galárraga et al. (2013) introduce a fast association rule mining approach that, as our work, is inspired by ILP techniques, but tailored towards binary relations, in this case using a database with aggressive indexing to speed up querying.

4 The frequent-paths based relational query miner

Before introducing the alternating relational redescription mining algorithm in Sect. 5, we now discuss its core component, FPQM, a novel relational query miner based on frequent paths. Given a network, FPQM aims to find the best graph queries with respect to quality criteria Γ that discriminate a set of positive object pairs E^+ . The specific criteria in Γ are discussed in Sect. 4.3.

An outline of the FPQM algorithm is presented in Fig. 2. The three-phase approach considers only linked queries and limits the number of costly subgraph isomorphism-based coverage tests. First, the path queries that cover at least a given number of positive examples are mined (line 2). Second, such path queries are combined to construct graph queries (lines 3–9). Finally, the best describing among these graph queries are selected and returned (lines 10–13). We now discuss each of these steps in turn.

4.1 Mining frequent path queries

The first phase of FPQM (line 2) finds the set of linked path queries that are frequent among the example pairs, as any frequent graph query connecting pairs of interest has to be a combination of such paths. It is similar in spirit to relational pathfinding, but using frequency, that is, the number of positive examples covered, as a selection criterion. The key idea behind this phase of the algorithm is to transform the problem into a sequence of constrained frequent itemset mining tasks, which can be solved efficiently using an off-the-shelf tool. More specifically, these subtasks are defined as follows:

Given a set of transactions \mathcal{T} , each representing all features of a path of length k connecting a positive example pair, and a frequency threshold γ ,
find all itemsets that (a) cover transactions for at least γ different example pairs, and (b) correspond to a path query, that is, contain an edge feature for every pair of neighboring nodes on the path.

Fig. 2 FPQM: Frequent-paths based relational query miner. Details on FREQUENTPQ are provided in Figure 3 and discussed in Sect. 4.1; TO TRANSACTION is discussed in Sect. 4.1, ALIGNEDBOTTOMCLAUSE, FIMGRAPHS and COMPUTEQUERY in Sect. 4.2

Input: A network with a set of positive examples E^+ , a frequency threshold γ , an extension threshold κ , a contribution threshold δ and a set of quality criteria Γ .
Output: A set of relational queries Q .

```

1:  $Q \leftarrow \emptyset$ ;  $D \leftarrow \emptyset$ ;  $S \leftarrow \emptyset$ ;  $\mathcal{T} \leftarrow \emptyset$ 
2:  $C \leftarrow \text{FREQUENTPQ}(E^+, \gamma, \kappa)$ 
3: for  $e \in E^+$  do
4:    $a \leftarrow \text{ALIGNEDBOTTOMCLAUSE}(e, C)$ 
5:    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(e, \text{TO TRANSACTION}(a))\}$ 
6: for  $f \in \text{FIMGRAPHS}(\mathcal{T}, \gamma)$  do
7:    $q \leftarrow \text{COMPUTEQUERY}(f)$ 
8:   if  $q$  is acceptable according to  $\Gamma$  then
9:      $D \leftarrow D \cup \{q\}$ 
10: for  $q \in D$  ordered according to  $\Gamma$  do
11:   if  $|E_{1,1}(q) \setminus S| \geq \delta$  then
12:      $Q \leftarrow Q \cup \{q\}$ 
13:      $S \leftarrow S \cup E_{1,1}(q)$ 
14: return  $Q$ 

```

Input: A network with a set of positive example object pairs E^+ , a frequency threshold γ , and an extension threshold κ .

Output: A set of frequent paths queries C .

```

1:  $k \leftarrow 0$ 
2:  $\mathcal{P}_k \leftarrow$  paths of length 0, i.e. starting nodes in  $E^+$ 
3: while  $\mathcal{P}_k \neq \emptyset$  do
4:    $k \leftarrow k + 1$ ;  $\mathcal{P}_k \leftarrow \emptyset$ ;  $\mathcal{T} \leftarrow \emptyset$ 
5:   for each  $P' \in \mathcal{P}_{k-1}$  do  $\triangleright P[i]$  is node at position  $i$  in path  $P$ 
6:     for each  $n \in \text{neighbors}(P'[k-1])$  do
7:        $P \leftarrow P'$ 
8:       if  $n \notin P$  then
9:          $P[k] \leftarrow n$ 
10:         $\mathcal{P}_k \leftarrow \mathcal{P}_k \cup \{P\}$ 
11:        if  $(P[0], P[k]) \in E^+$  then  $\triangleright$  example pair connected
12:           $\mathcal{T} \leftarrow \mathcal{T} \cup \{(P[0], P[k]), \text{TOTRANSACTION}(P)\}$ 
13:     $F \leftarrow \text{FIMPATHS}(\mathcal{T}, \gamma)$ 
14:     $C \leftarrow C \cup F$ 
15:     $E_k \leftarrow E_{k-1} \cup \bigcup_{f \in F} E_{1,1}(f)$ 
16:    if  $k > \kappa$  and  $E_{k-\kappa} = E_k$  then  $\triangleright$  no new example pair covered for  $\kappa$  steps
17:       $\mathcal{P}_k \leftarrow \emptyset$ 
18: return  $C$ 

```

Fig. 3 FREQUENTPQ: Mining frequent path queries from a network. Details on TOTRANSACTION and FIMPATHS are provided as part of the discussion in Sect. 4.1

Algorithm FREQUENTPQ, detailed in Fig. 3, extracts linked paths of increasing length that connect example pairs in the network, for each length creating and solving the corresponding frequent itemset mining task. We use the set of all starting nodes in the examples (line 2) as seed paths for the main loop that processes paths of increasing length. The algorithm terminates if no example pair has been covered for the first time in the last κ iterations. In the k th iteration, the nested loop in lines 5–12 extends paths in \mathcal{P}_{k-1} to paths of length k , discards cyclic paths, and stores the resulting paths in \mathcal{P}_k . For each path connecting an example pair, TOTRANSACTION produces the corresponding transaction, and adds it along with the connected example pair to \mathcal{T} (line 12). Next, frequent path queries for a given frequency threshold γ are mined from \mathcal{T} (line 13) and added to the set C of queries to be returned. We keep track of the set E_k of covered examples for the termination criterion.

For a given path, TOTRANSACTION creates a transaction based on the following encoding. An item is a tuple (i, f) , where i is either a single node or a pair of nodes, and f a feature. We refer to pairs of nodes adjacent on a path as *backbone edges*, and to all other pairs of nodes appearing in the path as *crossing edges*. Given a path of length k , its first and last nodes receive identifiers $\#A$ and $\#Z$, respectively, and intermediate nodes along the path $\#1$ to $\#(k-2)$. The corresponding transaction created by TOTRANSACTION contains an item (n, a) for each node feature a of a node n on the path, an item $((n, m), e)$ for each edge feature e of a backbone edge (n, m) , and an item $((n, m), c)$ for each comparison feature c of a backbone or crossing edge (n, m) . For example, the path of length three ($\#A, \#1, \#Z$) from Fig. 1(b) is represented by the following itemset:

$$\{((\#A, \#1), \text{parent}), (\#1, \text{male}), ((\#Z, \#1), \text{parent}), ((\#A, \#Z), \text{age}^<), (\#Z, \text{male})\}.$$

Note that this encoding is reversible, that is, given any transaction, we can extract the corresponding path query, replacing each identifier by a unique variable.

The transactions for all paths of the current length form the input for the corresponding frequent itemset mining task (line 13). Any algorithm that solves this mining task could be used here. Our solution exploits a declarative approach to mining patterns under constraints (Guns et al. 2011), as the corresponding system allows for user-defined constraints on frequent itemset mining tasks. This is important for our approach, as the definition of support employed when mining for frequent path queries in \mathcal{T} differs from the usual: the support of an itemset is not the number of transactions that contain it, but instead the number of distinct corresponding example pairs. In other words, we count the number of answer substitutions rather than the number of instantiations and are interested in finding path queries that connect many different example pairs rather than path queries having many instances for a given pair. Furthermore, in order to maintain connectivity, we require that some item corresponding to an edge constraint must be present for each backbone edge. More specifically, the approach of Guns et al. (2011) combines a declarative task specification language with a generic constraint programming solver. The user provides the transactions to mine itemsets from and specifies the desired constraints on itemsets in this language. The system interprets the task as a constraint program, to which it finds all solutions by calling the constraint solver. Algorithm FIMPATHS thus simply combines the transactions \mathcal{T} with a specification of our support and connectivity constraints, passes them to the system of Guns et al. (2011) to obtain the result, and transforms each itemset in the result into the corresponding path query as discussed above.

4.2 Combining path queries into graph queries

The second phase of FPQM obtains more expressive graph queries by combining frequent path queries. Again, we use a reduction to frequent itemset mining to obtain frequent queries. For each positive example pair, we combine the frequent paths covering it into a graph and represent it as a transaction (Fig. 2, lines 3–5). Frequent graph queries are then mined from these transactions and further filtered based on user-defined acceptance criteria (lines 6–9).

As an illustration of graph queries, Fig. 4 depicts two path queries p1 and p2 as well as three example graph queries that are obtained by merging query variables, and potentially other nodes as well, of one or more copies of these paths. Clearly, allowing multiple copies of a path permits an infinite number of combinations. However, merging intermediate nodes that assign conflicting values to attributes results in invalid queries, and only finitely many among the valid queries are supported by the data. Therefore, we merge paths based on their

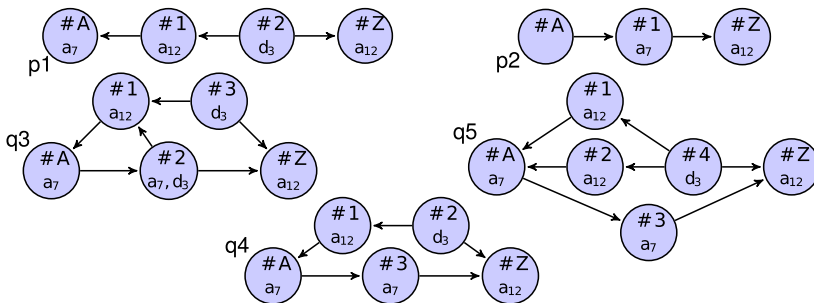


Fig. 4 Example of three graph queries (q3-q5) combining path queries p1 and p2

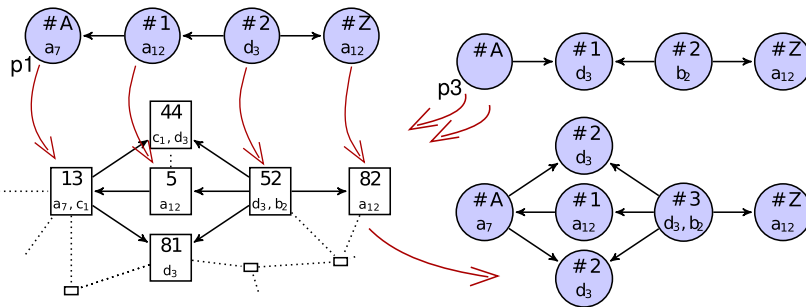


Fig. 5 Example of construction of an aligned bottom clause; see Sect. 4.2 for details

instantiations in the data rather than based on their query representation. This ensures that we only construct valid queries with non-empty support.

More specifically, given a set C of path queries with query variables ($\#A, \#Z$) and a positive example (o_1, o_2) , we call *bottom clause* the union of all possible instantiations of bodies of queries in C that map $(\#A, \#Z)$ to (o_1, o_2) . This is similar to the bottom clause obtained in some ILP approaches, but consists only of features from frequent path queries.

In order to represent such a bottom clause as a transaction, we need to assign identifiers to its intermediate nodes. This is done by `ALIGNEDBOTTOMCLAUSE` (line 4), which iteratively labels nodes based on the identifiers of nodes they instantiate in the paths queries as well as their neighboring nodes. This identifies groups of nodes occurring in the same context. Figure 5 illustrates the alignment of the bottom clause for positive example $(13, 82)$ and two path queries $p1$ and $p3$. The graph with rectangular nodes and numerical identifiers represents the relevant part of the data network. In this example, node 44 and node 81 both are instantiated by identifier $\#1$ in $p3$ only, with identical neighbors, and therefore receive the same new identifier $\#2$. While such *duplicate variables* are interesting from an expressivity point of view (as under θ_{OI} -subsumption they implement counting), they also result in multiple query instantiations for the same pair of answer nodes, which can be undesirable from an efficiency point of view. In this paper, we do not exploit the extra expressivity. Such duplicate variables receive the same label and will not be distinguished later on. Note that multiple copies do not necessarily generate duplicate nodes. For instance, in Fig. 4, nodes $\#3$ and $\#2$ of query $q3$ both correspond to node $\#2$ of $p1$ in two copies, yet are no duplicates because one is combined with node $\#1$ of $p2$ and the other is not.

Once the bottom clauses have been aligned via the new identifiers, all their features are collected into transactions using `TOTRANSACTION` (line 5) as in Sect. 4.1. Given these transactions \mathcal{T} and the frequency threshold γ , the next step consists in finding all itemsets that are frequent among \mathcal{T} . As each example has at most one associated transaction, the support of an itemset here is simply the number of distinct transactions containing it, that is, a frequent itemset covers at least γ transactions. Similarly to `FIMPATHS`, `FIMGRAPHS` (line 6) combines the transactions \mathcal{T} with a specification of this frequency constraint and passes them to the system of Guns et al. (2011) to obtain the result. We do not include connectivity constraints, as those cannot be enforced at the time of mining here. As its last step, `FIMGRAPHS` therefore filters out unconnected queries. For each frequent itemset returned by `FIMGRAPHS`, `COMPUTEQUERY` recovers the corresponding query as discussed for path queries above, maps it onto the data to determine all instances and computes the support (line 7). Note that only graphs that are frequent among positive examples are generated as a result of this procedure. Queries satisfying the user provided criteria Γ (discussed next) are collected in D .

4.3 Selecting a subset of queries

The final phase of FPQM (lines 10–13) greedily selects a subset of best queries with limited overlap among the ones stored in D . The queries are processed in turn, from best to worst according to the ranking criterion in Γ . If the part of the support of the current query that has not yet been covered, called its *contribution*, is sufficiently large (larger than δ), it is accepted and supported positive examples are marked as covered. Otherwise it is rejected. All accepted queries are returned as the output of FPQM.

More specifically, Γ offers a number of parameters that can (but need not) be used to orient the search toward queries with preferred characteristics and to rank queries. First, thresholds can be imposed on the coverage of queries, including the following:

$$\begin{aligned} \text{cover diff.} &= |E_{1,1}| - |E_{0,1}| & \text{support} &= |E_{1,1}| + |E_{0,1}| & \text{precision} &= \frac{|E_{1,1}|}{|E_{1,1}| + |E_{0,1}|} \\ \text{cover ratio} &= \frac{|E_{1,1}|}{|E_{0,1}|} & \text{accuracy} &= \frac{|E_{1,1}|}{|E_{1,0}| + |E_{1,1}| + |E_{0,1}|} & \text{recall} &= \frac{|E_{1,1}|}{|E_{1,0}| + |E_{1,1}|}. \end{aligned}$$

Second, additional constraints can be imposed on the form of queries. For instance, one can restrict the number of nodes or edges to limit the complexity of the patterns and facilitate interpretation, or require intermediate nodes to have a minimum number of instantiations in order to exclude queries that only apply in the neighborhood of a specific node.

The frequency threshold γ used in the two frequent itemset mining steps can be automatically determined from the minimum support as well as from the accuracy or the recall, given the number of positive examples.

4.4 An efficient relational query miner

To summarize, FPQM mines frequent path queries, combines them into graph queries, and finally selects a good subset of queries.

As discussed above, we filter out duplicate variables when constructing graph clauses to avoid a combinatorial explosion of the number of instances. To reduce the amount of filtering necessary, the frequent path mining step drops paths with high multiplicity, that is, with a number of instances a factor greater than the number of supporting pairs, as these will likely create duplicate variables. Also, when mapping the queries onto the data, we fix a limit on the number of instances that can be generated. For each query edge, the number of instances that will be obtained after mapping it is estimated from the current number of instances and the number of matching data edges. Query edges with lowest estimates are processed first and if the current estimate crosses the chosen threshold, remaining edges are dropped from the query.

To further improve efficiency, the computationally costly operation of finding graph instances is shared between queries. Indeed, if an itemset I is a subset of another itemset J , the query represented by J will be a refinement of that corresponding to I . Hence, the instances of I 's query are a superset of those of J 's query and can be used to initialize its mapping. Only the extra predicates will have to be added to the mapping.

5 An alternating scheme for relational redescription mining

We now turn to the alternating scheme for finding relational redescriptions, ARRM, sketched in Fig. 6. Given an initial set of queries, the algorithm first grows a forest of queries by repeatedly running FPQM to find the best descriptions of each query, which it adds as the query's children. The second phase then extracts redescriptions from this forest.

Input: A network, a set of initial queries I and quality criteria Γ .
Output: A set of relational redescription \mathcal{R} .

- 1: $\mathcal{S} \leftarrow \emptyset; \mathcal{M} \leftarrow \emptyset; \mathcal{K} \leftarrow \emptyset;$
- 2: **for** $i \in I$ **do**
- 3: $K.query \leftarrow i; K.parent \leftarrow \emptyset; K.generation \leftarrow 0$
- 4: $\mathcal{K} \leftarrow \mathcal{K} \cup \{K\}; \mathcal{M} \leftarrow \mathcal{M} \cup \{K\}$
- 5: **while** $\exists K \in \mathcal{K}$ **do**
- 6: $\mathcal{K} \leftarrow \mathcal{K} \setminus \{K\}$
- 7: **for** $q \in \text{FPQM}(\text{supp}(K.query), \Gamma)$ **do**
- 8: $L.query \leftarrow q; L.parent \leftarrow K; L.generation \leftarrow K.generation + 1$
- 9: $\mathcal{E} \leftarrow \{S \in \mathcal{M}, \text{supp}(S) = \text{supp}(L) \wedge \text{att}(S) = \text{att}(L)\}$
- 10: $\mathcal{M} \leftarrow \mathcal{M} \cup \{L\}$
- 11: **if** $L.generation \leq \tau \wedge \mathcal{E} = \emptyset$ **then**
- 12: $\mathcal{K} \leftarrow \mathcal{K} \cup \{L\}$
- 13: $\mathcal{U} \leftarrow \{(M, N) \in \mathcal{M}^2, (M.parent = N) \vee (\text{supp}(M) = \text{supp}(N) \wedge \text{att}(M) \cap \text{att}(N) = \emptyset)\}$
- 14: **for** $R \in \mathcal{U}$ ordered according to Γ **do**
- 15: **if** $|E_{1,1}(R) \setminus \mathcal{S}| \geq \delta$ **then**
- 16: $\mathcal{R} \leftarrow \mathcal{R} \cup \{R\}$
- 17: $\mathcal{S} \leftarrow \mathcal{S} \cup E_{1,1}(R)$
- 18: **return** \mathcal{R}

Fig. 6 ARRM: Alternating Relational Redescription Mining

5.1 Initialization

The algorithm expects an input set I of queries, based on which its two key data structures, the candidate list \mathcal{K} and the set of explored descriptions \mathcal{M} are initialized (lines 2–4). The simplest means to generate this input set is to consider the queries obtained from each edge predicate taken individually, that is, for each value v of each edge predicate a

$$c(\#A, \#Z) :- \epsilon_a^v(\#A, \#Z).$$

In addition, one might consider simple combinations such as

$$c(\#A, \#Z) :- \epsilon_a^v(\#1, \#A), \quad \epsilon_a^v(\#1, \#Z). \quad \text{or}$$

$$c(\#A, \#Z) :- \epsilon_a^v(\#A, \#1), \quad \epsilon_a^v(\#Z, \#1).$$

In particular, this might be useful in cases where the nodes that appear in the first (respectively second) position of edges supporting ϵ_a^v do not appear in any other predicate. The user may also specify a set of initial queries manually.

5.2 Growing a forest of queries

Given the current candidate list \mathcal{K} , each query K in \mathcal{K} is processed in turn, using the FPQM algorithm presented above to find new queries describing the same set of examples, but using different attributes (lines 5–12).

That is, the supporting pairs of the current candidate K constitute the positive examples, and the network for that round is obtained from the original network by removing all predicates based on attributes used in K . Given this input, FPQM is used to find the best

describing children queries (line 7), which are added to the set of explored descriptions \mathcal{M} , linked to the candidate in the query forest, and, if they meet the criteria discussed below, appended to the list of candidates \mathcal{K} . This process is repeated until \mathcal{K} is empty.

The algorithm thus grows a forest of queries breadth first, branching from the initial candidates. A branch might be interrupted for one of three reasons. First, if no child is returned by the query mining procedure the expansion stops naturally. Second, if a child has a support (modulo symmetry) and attributes set identical to some query found previously (non empty \mathcal{E} , line 11), it is not added to the candidates for expansion. Indeed, since the algorithm is entirely deterministic this would not generate new queries but practically introduce a loop in the exploration. Third, a maximum exploration depth τ can be fixed as part of Γ and branches that reach this length will not be expanded further (line 11).

The shape of the forest is affected primarily by the quality criteria Γ (cf. Sect. 4.3). Strict criteria limit the *fertility* of queries, i.e. the number of results returned by FPQM, and thus the branching factor of the forest. With high fertility, only shallow exploration will be manageable. On the other hand, stricter selection of the children, provided that some degree of diversity is maintained, allows one to explore more generations. To direct the exploration towards more accurate queries, the accuracy of the current query can be applied as a filtering criterion for its children queries.

5.3 Retrieving redescriptions

Once the candidate list \mathcal{K} has been exhausted, the next step consists in retrieving good redescriptions from the forest of queries. We gather into \mathcal{U} all pairs of adjacent queries along the forest. By construction, such pairs have overlapping supports and disjoint attribute sets. Also, we scan the forest for pairs of queries that are not adjacent but have identical support and disjoint attribute sets. These form extra candidate redescriptions that are added to \mathcal{U} (line 13).

Finally, the same greedy procedure as in FPQM, but considering redescriptions instead of individual queries, is applied to \mathcal{U} to select the final set of redescriptions (lines 14–17).

6 Experiments

We now turn to the experimental evaluation of our approach. In the first experiment, we focus on FPQM alone. We explore the influence of its parameters and compare the approach to an existing relational query miner, which we use as a baseline for both pattern quality and running time. The second set of experiments concerns the full alternating scheme. It again explores algorithm parameters, compares to a propositional approach, and also provides a qualitative assessment of redescriptions discovered.

Our algorithm was implemented in Python, using FIM CP (Guns et al. 2011) to mine path and graph queries (FIMPATHS and FIMGRAPHS). In all experiments, it was run on a single core of an 8 core Intel Xeon 2.8 GHz processor and with 32 GB of memory. In all runs, we limited the number of trials for mining paths to $\kappa = 2$, the number of instances when mapping queries to 20000, the mining time per FIM CP call to 1 min and the absolute minimum support of any query to 3.

The implementation of our algorithms and the prepared datasets are available online.¹

¹<http://www.cs.helsinki.fi/u/galbrun/redescriptors/>.

Table 1 Datasets statistics: number of nodes, edges, node predicates, edge predicates and comparison predicates

Dataset	#nodes	#edges	#node pred.	#edge pred.	#comp. pred.
Kinship	381	24053	3	31	1
UMLS	135	4181	–	46	–
UWCSE	1042	1674	6	7	5

6.1 Datasets

Three relational datasets are used throughout the experiments. The first dataset, *Kinship*, was extracted from the *Alyawarra Ethnographic Database*.² The other two datasets, *UWCSE* and *UMLS*, were obtained from the *Alchemy repository*.³ Table 1 summarizes the characteristics of the datasets.

Kinship provides personal and genealogical information about individual members of an indigenous community of Australia, the *Alyawarra*, as well as the kinship terms they use for their relationships to other persons. A glossary of kinship terms is available, to which we can compare our findings. To simplify the notation, we use the indices of kinship terms from the glossary, rather than the terms themselves. For instance, *awaadya* is later denoted as *kin10*. *Kinship* information is not available for all living individuals; we mark as *relevant* those for whom it is complete.

UMLS characterizes the relations between biomedical concepts in terms of the *Unified Medical Language System ontology*.

UWCSE contains information about relationships between persons and courses within the computer science department of the university of Washington. It includes two predicates of arity three, namely *taught by* (*Course*, *Person*, *Time*) and *ta* (*Course*, *Person*, *Time*), which our algorithm cannot handle natively. We therefore split those into binary predicates linked together by a newly introduced course identifier.

6.2 Mining relational queries

The first series of experiments focuses on our proposed path-based relational query mining algorithm, *FPQM*. After reporting general observations about the behaviour of the algorithm, it is compared to a baseline relational query miner.

Algorithm behaviour As expected, the minimum support γ strongly affects the number of path queries found in the first step of *FPQM* (for instance, from 2 when $\gamma = 0.30$ up to 368 when $\gamma = 0.05$, on average for *Kinship*). Raising γ can actually result in increased running times because in the absence of frequent short queries, paths will be extended to greater lengths, possibly at high computational expenses and without success. On the other hand, the algorithm will be overwhelmed by the quantity of patterns when γ is set too low. Clearly, it is advisable to set γ as low as possible according to the size and density of the dataset.

²<http://habc.eu/csac/wiki/knsrc/KinSources/AU01Alyawarra1971>.

³<http://alchemy.cs.washington.edu/>.

Raising the extension threshold κ can lead to the generation of numerous variations of shorter connecting paths. This potential for enriching the queries typically comes at a high computational cost and κ should thus be kept low.

We also studied the effect of the greedy selection of queries on the cover of positive examples. The aim of this phase is mainly to reduce the high redundancy of the results. Indeed, we observed that before greedy selection, we easily obtained hundreds of queries, but a very limited number of queries (1–3) is generally sufficient to obtain almost the same cover. Hence, this pruning phase is crucial for limiting the fertility of clauses while maintaining the coverage quality.

Relational query miner comparison Next, we compare our proposed path-based algorithm FPQM to a baseline relational query miner on the three datasets. For each edge predicate in turn, we take the supporting pairs of nodes as positive examples and mine queries over the remaining attributes. We do not consider cases with less than four positive examples. Furthermore, with `kinship` we only consider pairs of nodes that are both relevant.

As a baseline, we use a modified version of C-ARMR (De Raedt and Ramon 2004) (implemented in Prolog) that mines top- k queries with respect to the difference in support on positive and negative examples. Given a set of positive examples and considering all remaining pairs of nodes as negative examples, we allowed C-ARMR to mine for top-5 queries with positive score. As discussed in Sect. 3, the implementation does not ensure that query variables are linked. To address this problem, we refine unlinked queries if they cover at least one positive example, but never include them in the result. This is similar in spirit to generating candidates based on the data as common in relational pathfinding and function learning (Richards and Mooney 1992; Ong et al. 2005; Santos et al. 2009), but avoids the need to adapt the canonical refinement operator used in our implementation. Experiments with C-ARMR have been performed on a single core of a C2Q machine (2.4 GHz 4 GB for `kinship`, 2.83 GHz 8 GB for `UMLS` and `UWCSE`).

In order to obtain results comparable to C-ARMR's, we also use the cover difference to select queries with FPQM (cf. Sect. 4.3). Similarly, only positively scoring queries are output. In addition, we require the accuracy of the queries to exceed $j = 0.05$, so as to obtain a minimum frequency threshold for mining paths and graphs.

Tables 2–4 present quantitative results and running times for the three datasets respectively. For C-ARMR, which only scores individual patterns, we use the disjunction of top-5 patterns with positive scores. As C-ARMR returns all equally scoring patterns in case of ties, these disjunctions can have more than five elements. For each case, we report the total number of positive examples $|E^+|$, as well as the number of queries returned $|Q|$, the number of true and false positives $|E_{1,1}|$ and $|E_{0,1}|$, the aggregated accuracy J (i.e. the Jaccard coefficient between the set of example pairs and the union of the supports of output queries) and the running times T for both C-ARMR and FPQM.

On `kinship`, cf. Table 2, we restrict the number of body literals in C-ARMR's queries to at most five, as running times become prohibitive for longer queries due to large numbers of unlinked or non-discriminative queries. Under this restriction, we observe comparable or better accuracies for patterns found by FPQM, which moreover is at least one order of magnitude faster. On average, FPQM returns fewer queries than C-ARMR because it includes a selection procedure to remove redundant queries, which is not the case with C-ARMR.

Furthermore, as a direct consequence of this restriction, no pattern with positive score was found for six of the `kinship` terms, whereas FPQM is able to identify more complex patterns for these cases. For four predicates, C-ARMR found queries where FPQM did not return any. In three of these cases, the positive support of individual queries is lower than 3 or

Table 2 Comparison of C-ARMR and FPQM on Kinship: number of positive examples $|E^+|$, number of true and false positives $|E_{1,1}|$ and $|E_{0,1}|$, aggregated accuracy J, number of queries $|Q|$, and running time (T) in seconds

Predicate	C-ARMR						FPQM				
	$ E^+ $	$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)	$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)
kin1	228	16	50	0.06	12	531	12	7	0.05	1	5
kin2	489	38	29	0.07	5	532	154	74	0.27	4	40
kin3	231	36	102	0.11	14	423	78	40	0.29	5	41
kin4	379	24	31	0.06	179	663	0	0	0.00	0	7
kin5	493	11	9	0.02	12	564	0	0	0.00	0	2
kin6	508	87	2	0.17	20	433	148	33	0.27	4	19
kin7	453	50	7	0.11	6	462	209	93	0.38	8	20
kin8	817	56	1	0.07	21	502	92	1	0.11	2	2
kin9	805	64	203	0.06	6	513	166	33	0.20	3	19
kin10	462	41	3	0.09	7	413	49	41	0.10	2	1
kin11	505	38	4	0.07	6	442	42	34	0.08	2	0
kin12	739	75	11	0.10	23	598	81	61	0.10	2	1
kin13	299	0	0	0.00	0	396	159	123	0.38	14	46
kin14	447	0	0	0.00	0	449	87	17	0.19	2	14
kin15	43	0	0	0.00	0	445	20	6	0.41	3	10
kin16	943	130	148	0.12	15	551	203	53	0.20	3	85
kin17	1256	157	127	0.11	11	582	0	0	0.00	0	93
kin18	392	61	3	0.15	9	466	61	3	0.15	1	3
kin19	569	36	24	0.06	16	507	0	0	0.00	0	88
kin20	13	0	0	0.00	0	338	7	0	0.54	2	5
kin21	272	43	13	0.15	6	437	109	38	0.35	4	48
kin22	142	20	132	0.07	9	453	49	32	0.28	6	8
kin23	193	0	0	0.00	0	343	53	29	0.24	5	15
kin26	6	0	0	0.00	0	219	0	0	0.00	0	3

their accuracy lower than 0.05. These values are below the thresholds used in FPQM hence they did not qualify as good quality queries. For the remaining predicate, kin17, FPQM was overwhelmed by the quantity of frequent queries, an issue easily solved by raising the accuracy threshold.

On inspection, the obtained queries correspond to definitions provided in the glossary. Some deviations are observed, such as an intermediate genealogical level or a difference in gender of some individual.

On UMLS, the difference in running time gets even more pronounced. Here, C-ARMR could not identify the top-5 patterns up to four body literals within two hours for any term, and failed to do so for up to three literals for twelve predicates, taking between 9 and 95 minutes for the remaining ones. Table 3 therefore reports results with up to two body literals. FPQM, on the other hand, takes seconds or at most up to a few minutes per predicate, and often finds more accurate queries, as it does not suffer from the restricted expressiveness of short queries.

On UWCSE, no queries could be found for most predicates. Contrarily to C-ARMR, FPQM did not return any query for taught by due to individual queries having insufficient accu-

Table 3 Comparison of C-ARMR and FPQM on UMLS. Legend as in Table 2

Predicate	C-ARMR						FPQM				
	$ E^+ $	$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)	$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)
adjacent to	7	5	4	0.45	11	38	3	1	0.38	1	2
affects	1022	437	0	0.43	7	129	635	124	0.55	5	65
analyzes	52	50	15	0.75	7	38	52	14	0.79	2	1
ass. eff. of	65	50	2	0.75	29	55	65	15	0.81	1	9
associated with	239	110	46	0.39	5	102	187	122	0.52	11	107
carries out	38	36	36	0.49	5	46	36	0	0.95	1	2
causes	360	280	116	0.59	5	72	354	82	0.80	5	164
complicates	263	189	95	0.53	5	71	263	29	0.90	4	171
concep. part of	18	0	0	0.00	0	283	3	1	0.16	1	2
connected to	4	4	8	0.33	82	45	0	0	0.00	0	2
consists of	9	9	8	0.53	18	38	8	0	0.89	1	6
contains	11	6	2	0.46	5	50	5	0	0.45	1	1
co-occurs with	67	42	24	0.46	16	66	61	34	0.60	5	190
degree of	34	30	6	0.75	43	52	30	0	0.88	1	90
dev. form of	4	4	2	0.67	15	58	3	0	0.75	1	5
diagnoses	48	20	4	0.38	9	66	48	0	1.00	4	61
disrupts	154	112	68	0.50	5	74	125	77	0.54	4	24
evaluation of	63	28	8	0.39	7	52	46	6	0.67	2	33
exhibits	45	18	0	0.40	6	34	45	24	0.65	1	1
indicates	27	18	4	0.58	5	189	18	0	0.67	1	10
ingredient of	28	0	0	0.00	0	203	0	0	0.00	0	2
interacts with	451	155	137	0.26	5	73	451	451	0.50	6	9
isa	500	7	1	0.01	10	188	0	0	0.00	0	22
issue in	268	68	0	0.25	6	94	256	0	0.96	4	3
location of	319	159	117	0.36	5	94	269	97	0.65	9	21
manages	6	0	0	0.00	0	152	4	0	0.67	1	3
manif. of	194	138	82	0.50	5	69	193	92	0.67	9	154
measurement of	64	31	1	0.48	5	80	62	9	0.85	5	95
measures	180	65	0	0.36	18	56	164	86	0.62	3	203
method of	25	18	16	0.44	7	39	22	15	0.55	2	0
occurs in	90	60	18	0.56	5	79	87	8	0.89	4	36
part of	200	176	122	0.55	5	63	102	0	0.51	3	4
performs	90	18	0	0.20	7	47	90	0	1.00	5	83
precedes	73	72	13	0.84	62	60	72	0	0.99	3	268
prevents	32	30	0	0.94	13	59	30	0	0.94	1	9
process of	437	437	165	0.73	5	64	0	0	0.00	0	97
produces	276	140	65	0.41	5	86	254	47	0.79	6	61
property of	44	34	0	0.77	1	156	34	0	0.77	1	18
result of	586	306	63	0.47	5	95	361	137	0.50	10	63
surrounds	8	8	9	0.47	84	51	4	0	0.50	1	0
treats	56	50	0	0.89	21	68	50	10	0.76	3	25
uses	65	48	0	0.74	6	81	50	6	0.70	2	2

Table 4 Comparison of C-ARMR and FPQM on UWCSE. Legend as in Table 2

Predicate	$ E^+ $	C-ARMR					FPQM				
		$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)	$ E_{1,1} $	$ E_{0,1} $	J	$ Q $	T (s)
advised by	113	11	14	0.09	8	646	9	4	0.08	1	0
ta	195	0	0	0	0	257	0	0	0	0	0
taught by	286	21	30	0.07	8	446	0	0	0	0	4
tempAdvised by	37	0	0	0	0	392	0	0	0	0	0

racy. We observe that in the absence of good connecting queries, C-ARMR still takes several minutes to complete the search, while FPQM returns within a few seconds.

These experiments thus indicate that the FPQM algorithm can find more complex queries much faster, compared to a standard query mining approach.

6.3 Finding relational redescription

We now turn to the full relational redescription mining algorithm, ARRM. First, we study its behaviour, particularly the impact of queries selection. Next, we compare it to three alternative approaches, namely using C-ARMR instead of FPQM within the alternating scheme, using propositionalization followed by a propositional redescription miner, and restricting ARRM to path queries rather than graph queries. Finally, we provide concrete illustrative examples of redescrptions found by ARRM.

Algorithm behaviour The criterion for ranking queries is the major lever in the selection procedure, directly impacting the exploration. We investigate this aspect by letting ARRM mine redescrptions from all three datasets with varying parameters. More specifically, we considered positive support (s), cover ratio (r) and accuracy (j) as options for descending ranking of the queries. We used various maximum exploration depths (τ_x), from a single round up to the number of alternations after which the algorithm stopped with no candidates left, or at most 15. In addition, as an alternative to graph queries (G), we limited the algorithm to paths queries (P) by shortcutting the combining phase.

Looser selection methods that yield more outputs per candidate query (upwards of 5 on average) were also studied. They result in very broad query trees whose dimensions become unmanageable already after the first round. They showed no improvements in the quality of redescrptions after uncomplete runs lasting over a day and hence were abandoned.

The remaining criteria in Γ control the complexity of the queries, avoiding the generation of patterns that are unduly specific or contain overly many constraints. Such patterns have a very limited explanatory power and are considered worthless. These criteria were fixed as follows for each dataset in all runs. We required any node to be instantiated by at least 3 distinct data nodes. For Kinship, we limited the number of nodes and of edges in a query to $n = 7$ and $e = 10$, respectively. The minimum contribution of a clause was set to $d = 0.66$ of its support and the minimum accuracy for the first turn was set to $j = 0.05$. For subsequent turns the Jaccard coefficient of the parent query was used as a minimum threshold. For UMLS, we set these thresholds to $n = 5$, $e = 10$, $d = 0.25$ and $j = 0.33$. For UWCSE, we set $j = 0.33$ and the absolute minimum contribution to 3 but did not limit the number of nodes and edges.

Table 5 presents statistics of the results obtained for each run as well as running times. We observe important variations in the support and accuracy across datasets, reflecting the

Table 5 Quantitative results and running times (T) for mining redescrptions under different parameterizations (cf. Sect. 6.3 for details). $|\mathcal{R}|$ and $|\mathcal{M}|$ are the number of redescrptions returned and of queries explored, respectively. Fert. is the average clause fertility

Dataset Parameters	$ \mathcal{R} $	$ E_{1,1} $			J			$ \mathcal{M} $	Fert.	Tot. T	T/clause	
		min	max	avg	min	max	avg				max	avg
Kinship												
Gj τ_1	39	4	784	141	0.05	0.18	0.10	31	1.90	32 min	4 min	62 s
Gj τ_5	62	5	784	91	0.05	0.50	0.20	228	1.13	3 h 14 min	10 min	51 s
Gj τ_8	64	5	784	88	0.05	0.53	0.22	263	1.05	3 h 58 min	10 min	54 s
Gr τ_1	59	4	283	84	0.05	0.11	0.07	31	2.77	33 min	4 min	64 s
Gr τ_5	177	4	753	31	0.05	0.61	0.25	744	1.36	8 h 14 min	10 min	39 s
Gr τ_{10}	205	4	753	27	0.06	0.66	0.27	1190	1.08	14 h 37 min	10 min	44 s
Gr τ_{12}	203	4	753	27	0.06	0.66	0.27	1203	1.08	14 h 45 min	10 min	44 s
Gs τ_1	37	4	784	153	0.05	0.17	0.08	31	1.77	32 min	4 min	62 s
Gs τ_5	57	4	784	110	0.05	0.28	0.11	202	1.46	2 h 41 min	8 min	48 s
Gs τ_{10}	65	8	784	101	0.05	0.28	0.12	320	1.32	4 h 47 min	9 min	53 s
Gs τ_{15}	69	8	784	98	0.05	0.28	0.13	350	1.29	5 h 20 min	10 min	55 s
Pj τ_1	34	4	784	155	0.05	0.18	0.10	31	1.55	23 min	2 min	44 s
Pj τ_5	57	4	784	102	0.05	0.43	0.15	178	1.17	1 h 11 min	2 min	24 s
Pj τ_7	56	4	784	102	0.05	0.43	0.15	207	1.06	1 h 28 min	2 min	25 s
Pr τ_1	43	4	466	108	0.05	0.16	0.08	31	2.06	23 min	2 min	46 s
Pr τ_5	117	4	732	43	0.05	0.43	0.16	374	1.36	2 h 10 min	2 min	20 s
Pr τ_{10}	139	4	732	36	0.05	0.66	0.19	601	1.13	3 h 47 min	2 min	22 s
Ps τ_1	34	4	784	163	0.05	0.17	0.09	31	1.52	23 min	2 min	44 s
Ps τ_5	49	4	784	123	0.05	0.28	0.12	169	1.33	1 h 6 min	2 min	23 s
Ps τ_{11}	57	4	784	111	0.05	0.35	0.13	222	1.23	1 h 21 min	2 min	21 s
UMLS												
Gj τ_1	19	5	404	100	0.35	0.94	0.65	42	0.71	9 min 31 s	76 s	13 s
Gj τ_3	20	6	552	118	0.35	1.00	0.79	87	0.68	13 min 59 s	76 s	9 s
Gr τ_1	19	5	374	99	0.35	0.94	0.64	42	0.83	9 min 30 s	77 s	13 s
Gr τ_4	20	6	374	106	0.35	1.00	0.80	111	0.80	14 min 51 s	77 s	8 s
Gs τ_1	18	5	437	107	0.33	0.75	0.51	42	0.67	9 min 24 s	76 s	13 s
Gs τ_4	19	6	552	130	0.35	1.00	0.64	86	0.65	16 min 51 s	76 s	11 s
Pj τ_1	16	5	404	103	0.36	0.94	0.63	42	0.62	8 min 2 s	77 s	11 s
Pj τ_4	17	6	552	124	0.38	1.00	0.78	80	0.62	11 min 21 s	77 s	8 s
Pr τ_1	16	5	404	104	0.36	0.94	0.62	42	0.76	8 min 1 s	77 s	11 s
Pr τ_4	18	6	404	107	0.38	1.00	0.79	101	0.78	11 min 26 s	77 s	6 s
Ps τ_1	15	5	437	110	0.33	0.75	0.48	42	0.57	8 min 4 s	77 s	11 s
Ps τ_4	16	6	552	138	0.35	1.00	0.63	78	0.62	13 min 49 s	77 s	10 s
UWCSE												
Gj τ_1	15	5	278	53	0.02	0.24	0.07	8	2.50	5 s	2 s	<1 s
Gj τ_3	19	5	278	49	0.02	0.28	0.10	35	0.94	24 s	10 s	<1 s
Gr τ_1	16	5	278	53	0.02	0.24	0.07	8	3.75	5 s	2 s	<1 s
Gr τ_4	21	5	278	47	0.02	0.28	0.10	50	1.16	31 s	9 s	<1 s
Gs τ_1	14	5	278	42	0.02	0.24	0.06	8	2.00	6 s	2 s	<1 s
Gs τ_3	17	5	278	41	0.02	0.24	0.06	28	0.89	23 s	10 s	<1 s
Pj τ_1	15	5	278	53	0.02	0.24	0.07	8	2.50	4 s	2 s	<1 s
Pj τ_3	19	5	278	49	0.02	0.28	0.10	35	0.94	21 s	9 s	<1 s
Pr τ_1	16	5	278	53	0.02	0.24	0.07	8	3.50	5 s	2 s	<1 s
Pr τ_4	21	5	278	47	0.02	0.28	0.10	48	1.17	26 s	8 s	<1 s
Ps τ_1	14	5	278	42	0.02	0.24	0.06	8	2.00	3 s	1 s	<1 s
Ps τ_3	17	5	278	41	0.02	0.24	0.06	28	0.89	18 s	9 s	<1 s

Table 6 Quantitative results and running times (T) for mining redescrptions with C-ARRM as the query miner (cf. Sect. 6.3 for details). Legend as in Table 5

Dataset Parameters	$ \mathcal{R} $	$ E_{1,1} $			J			$ \mathcal{M} $	Fert.	Tot. T	T/clause	
		min	max	avg	min	max	avg				max	avg
Kinship τ_3	8	4	75	34	0.07	0.15	0.11	53	2.04	19 h 59 min	2 h	22 min
UMLS τ_8	49	4	437	76	0.08	1.00	0.72	178	1.35	3 h	3 min	61 s
UWCSE τ_2	3	40	61	52	0.07	0.15	0.10	32	2.67	4 min 16 s	51 s	8 s

variety of the redescrptions found. First, alternating for a few turns does allow one to find more accurate patterns than simply matching isolated predicates (τ_1). Note that contrarily to Tables 2–4, accuracies here are not aggregated over multiple patterns but obtained from individual redescrptions. Running times per query vary from less than a second up to several minutes. Of course, the total running time needed for the full exploration depends heavily on the number of queries explored, $|\mathcal{M}|$. As mentioned previously, an important factor impacting the running times is the presence of symmetries which results in duplicate variables, leading to numerous instances and hence more costly mapping.

The effects of different parameterizations are limited with UMLS and UWCSE, where the algorithm consistently mined a small number of relatively simple queries. Parameterization is more critical with Kinship, where more complex queries are needed to capture the various meanings of kinship terms. In particular, with the former two datasets, the algorithm stops after a few alternations upon finding only redundant queries. Therefore, further raising τ does not affect the outcome. In all datasets, ranking queries by cover ratio allowed to find the best redescrptions while ranking by support appears suboptimal.

Alternative query miner As the alternating scheme is independent of the query mining algorithm used (line 7 in Fig. 6), we created a modified version of ARRM, where we replace FPQM by C-ARRM. We use the same parameterizations of C-ARRM as in the experiments above, with the exception of the number of body literals for Kinship, which we had to restrict to at most three (rather than five) here to keep running times feasible. This is due to the much larger search space over kinship terms compared to that over genealogical terms. We rank queries by cover ratio, as this was the best setting in the previous experiment.

Table 6 presents quantitative results and running times for mining redescrptions from the three datasets under this replacement. In all three cases, the algorithm stopped after a few alternations (3 for Kinship, 8 for UMLS and 2 for UWCSE) without candidates for further expansion. In the case of UMLS and UWCSE the obtained redescrptions are almost on par with those found using FPQM, albeit somewhat less accurate, while the redescrptions found for Kinship have clearly lower accuracy. This can be explained as an effect of the length restriction required to keep C-ARRM running times feasible. As UMLS and UWCSE contain simple redescrptions, the length limit of two only moderately affects the quality of the outcome. With Kinship, however, restricting query length to at most three prevents the algorithm from finding the more complex redescrptions of higher accuracy the original ARRM identifies. Together with the significantly higher running times of the C-ARRM-based variant, these results show that we can find more accurate redescrptions more quickly by using our new FPQM method rather than C-ARRM within the alternating scheme.

A propositional approach We next compare to an existing propositional redescrption mining algorithm, following ideas from propositionalization (Kuzelka and Zelezny 2009;

Table 7 Quantitative results and running times for mining redescrptions by applying the REREMi algorithm on the propositionalized dataset (cf. Sect. 6.3 for details). $|D|$, $|F|$, T prop. and T mining are the number of rows and columns in the propositionalized dataset and the running time for propositionalization and for mining, respectively

Dataset Parameters	$ \mathcal{R} $	$ E_{1,1} $			J			$ D $	$ F $	T prop.	T mining
		min	max	avg	min	max	avg				
Kinship	66	3	1489	178	0.00	0.17	0.07	49074	13169	3 min 42 s	34 min 19 s
UMLS	100	6	182	36	0.04	1.00	0.74	18088	4998	23 s	13 min 13 s
UWCSE	58	3	41	19	0.01	0.24	0.09	55882	2048	15 s	1 min 45 s

Dinh et al. 2012) to suitably transform the data. Specifically, we extracted features from the dataset by enumerating paths up to a given length joining each pair of objects and ran the REREMi propositional redescription mining algorithm (Galbrun and Miettinen 2012) on the resulting propositional dataset. We considered paths of length one and two for all datasets. In addition we also extracted paths over genealogical attributes of length up to five for the Kinship dataset.

As can be seen from Table 7, this method is able to find comparable redescrptions in competitive running times when applied to UMLS and UWCSE, where paths of length at most two suffice to capture the relations in the network. However, it fails with Kinship, where longer paths and more complex combinations are needed.

Especially in the latter case, our relational approach to redescription mining has clear advantages over the propositional approach. First, it can easily explore longer paths, whose inclusion results in feature matrices too large to handle with the propositional miner. Second, it bases the selection of paths considered on the current set of positive examples, rather than on all possible pairs as necessary when performing propositionalization once before mining. Third, it explicitly combines paths into more expressive graph queries.

The second point could be addressed by repeated propositionalization during mining, which would require to modify the propositional approach. To simulate this setting, we performed experiments where we restricted our algorithm to path queries. Results are reported in Table 5. Once more, we observe that results are often comparable for the simpler cases of UMLS and UWCSE, but our fully relational method finds substantially more accurate redescrptions on Kinship, with an average accuracy of 0.27 when using graphs (Gr) compared to 0.19 when using paths only (Pr).

Examples of redescrptions We now provide a few example redescrptions found by ARRM on the three datasets. Figure 7 shows an example of redescription found from the Kinship dataset. Graph (1b) represents the genealogical link between a female individual and the daughter of her paternal aunt, which is one of the meanings of kin14 found in the glossary. Kinship terms often have several meanings and may also be used in broader senses. In such cases, a configuration of several terms, as in graph (1a), better captures one of the senses than a term taken in isolation. For instance, this redescription has an accuracy of 0.59, a significant improvement over the best match found for edge predicate kin14 alone, of accuracy 0.06 or the best pair of path queries similarly involving this predicate, of accuracy 0.17.

Figure 8 shows two examples of redescrptions found for UMLS: a perfect redescription, i.e. with accuracy 1, and a pair of symmetrical queries.

Finally, three redescrptions from UWCSE are displayed in Fig. 9. Graphs (4a) and (6a) both represent the advisee–advisor relationship, with different matching queries. One states

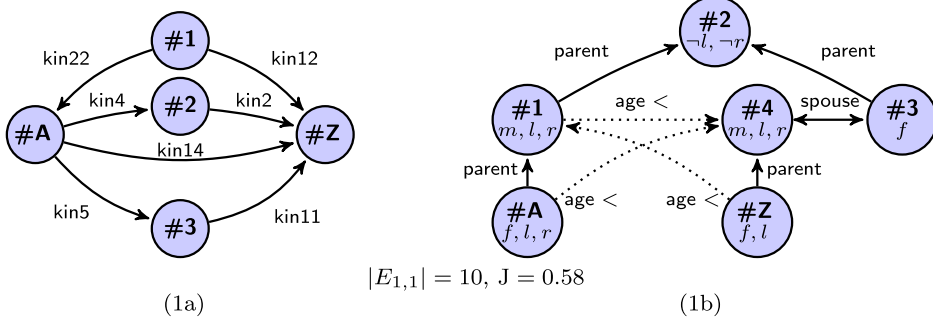


Fig. 7 Example redescription found from Kinship. Node predicates are male (m), female (f), living (l) and relevant (r). Negated Boolean attributes are denoted with \neg

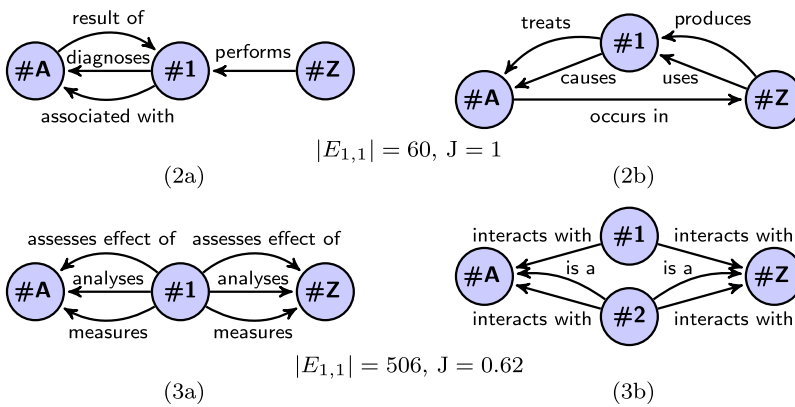


Fig. 8 Example redescriptions found from UMLS

that they share a common publication (4b), the other that the advisee is a teaching assistant for the course taught by his supervisor (6b), forming two redescriptions of accuracy 0.24 and 0.10, respectively. The other redescription involves a coauthorship relation again, this time between faculty members (5a) and the matching query indicates that the persons of interest have taught different sessions of a same course (5b).

To summarize, we observe that our alternating scheme is able to find redescriptions that capture the relations existing in a network. It can exploit the added expressivity compared to individual queries to better elucidate the different uses of the same predicate and find more accurate patterns.

7 Conclusions

We have introduced the problem of relational redescription mining. As a solution, we proposed an alternating scheme with a novel efficient relational query miner based on frequent paths as its core.

We demonstrated that our query miner can find more complex queries than a baseline ILP approach, much faster. The proposed alternating scheme is able to capture the relations existing in a network with expressive redescriptions, as shown in experiments with three relational datasets.

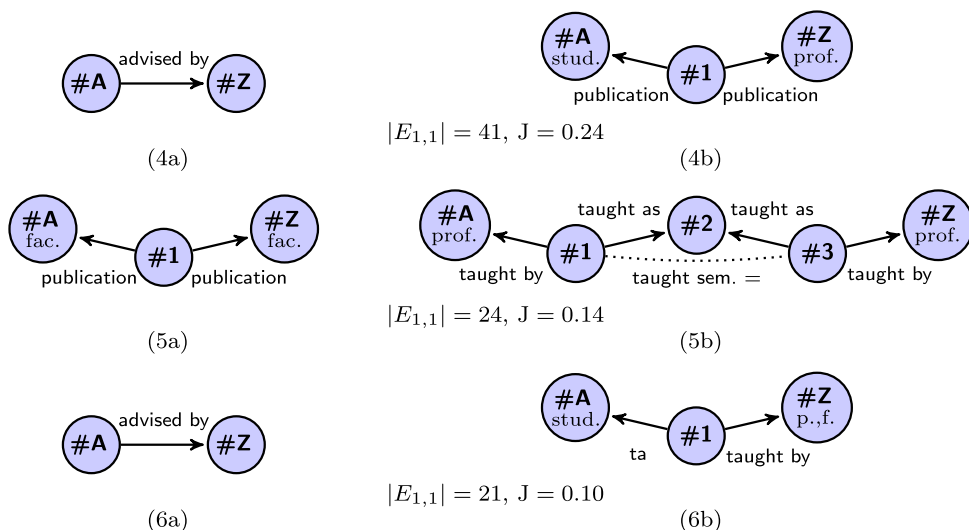


Fig. 9 Example redescrptions found from UWCE. Node predicates are professor, student and faculty

The power of relational redescription mining should be investigated further on other datasets and application domains. On the algorithmic side, extending the approach to probabilistic networks and considering queries of higher arity provide essential directions for future work.

Acknowledgements A. Kimmig is a postdoctoral fellow of the Research Foundation Flanders (FWO Vlaanderen). Part of the work was done while EG was visiting KU Leuven.

References

- Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. G. (2007). DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th international semantic web conference (ISWC/ASWC 2007)* (pp. 722–735).
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R. Jr., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In M. Fox & D. Poole (Eds.), *AAAI*. Menlo Park: AAAI Press.
- De Raedt, L., & Ramon, J. (2004). Condensed representations for inductive logic programming. In *9th international conference on principles of knowledge representation and reasoning* (pp. 438–446). Menlo Park: AAAI Press.
- Dehaspe, L., & Toivonen, H. (1999). Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3(1), 7–36.
- Dinh, Q. T., Vrain, C., & Exbrayat, M. (2012). A link-based method for propositionalization. In F. Riguzzi & F. Železný (Eds.), *Late breaking papers of the 22nd international conference on inductive logic programming (ILP'12), CEUR-WS* (Vol. 975, p. 10–25).
- Esposito, F., Malerba, D., Semeraro, G., Brunk, C., & Pazzani, M. (1994). Traps and pitfalls when learning logical definitions from relations. In Z. W. Ras & M. Zemankova (Eds.), *LNCS* (Vol. 869, pp. 376–385). Berlin: Springer.
- Galárraga, L. A., Teflioudi, C., Hose, K., & Suchanek, F. M. (2013). AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In D. Schwabe et al. (Eds.), *Proceedings of the 22nd international world wide web conference (WWW'13), IW3C2/ACM* (pp. 413–422).
- Galbrun, E., & Kimmig, A. (2012). Towards finding relational redescrptions. In J. G. Ganascia, P. Lenca, & J. M. Petit (Eds.), *Lecture notes in computer science: Vol. 7569. Discovery science* (pp. 52–66). Berlin: Springer.

- Galbrun, E., & Miettinen, P. (2012). From black and white to full colour: extending redescription mining outside the Boolean world. *Statistical Analysis and Data Mining*, 5(4), 284–303.
- Gallo, A., Miettinen, P., & Mannila, H. (2008). Finding subgroups having several descriptions: algorithms for redescription mining. In *Proceedings of the SIAM international conference on data mining, SDM 2008* (pp. 334–345).
- Goethals, B., & Van den Bussche, J. (2002). Relational association rules: Getting WARMeR. In D. J. Hand, N. M. Adams, & R. J. Bolton (Eds.), *LNCS: Vol. 2447. Pattern detection and discovery* (pp. 125–139). Berlin: Springer.
- Goethals, B., Hoekx, E., & Van den Bussche, J. (2005). Mining tree queries in a graph. In *11th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 61–69). New York: ACM.
- Guns, T., Nijssen, S., & De Raedt, L. (2011). Itemset mining: a constraint programming perspective. *Artificial Intelligence*, 175(12–13), 1951–1983.
- Kuzelka, O., & Zelezný, F. (2009). Block-wise construction of acyclic relational features with monotone irreducibility and relevancy properties. In A. P. Danyluk, L. Bottou, & M. L. Littman (Eds.), *ACM international conference proceeding series: Vol. 382. ICML* (p. p 72). New York: ACM.
- Lao, N., & Cohen, W. W. (2010). Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1), 53–67.
- Lao, N., Mitchell, T. M., & Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing (EMNLP 2011)* (pp. 529–539).
- Lavrac, N., Zelezný, F., & Flach, P. A. (2002). Rsd: relational subgroup discovery through first-order feature construction. In S. Matwin & C. Sammut (Eds.), *Lecture notes in computer science* (Vol. 2583, pp. 149–165). Berlin: Springer.
- Miller, R. J., Haas, L. M., & Hernandez, M. A. (2000). Schema mapping as query discovery. In *International conference on very large data bases*.
- Nebot, V., & Llavori, R. B. (2012). Finding association rules in semantic web data. *Knowledge-Based Systems*, 25(1), 51–62.
- Ong, I. M., de Castro Dutra, I., Page, D., & Santos Costa, V. (2005). Mode directed path finding. In J. Gama, R. Camacho, P. Brazdil, A. Jorge, & L. Torgo (Eds.), *LNCS: Vol. 3720. ECML* (pp. 673–681). Berlin: Springer.
- Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., & Helm, R. F. (2004). Turning CARTwheels: an alternating algorithm for mining redescrptions. In *10th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 266–275). New York: ACM.
- Rettinger, A., Löscher, U., Tresp, V., d'Amato, C., & Fanizzi, N. (2012). Mining the semantic web—statistical learning for next generation knowledge bases. *Data Mining and Knowledge Discovery*, 24(3), 613–662.
- Richards, B. L., & Mooney, R. J. (1992). Learning relations by pathfinding. In *10th national conference on artificial intelligence* (pp. 50–55). Menlo Park/Cambridge: AAAI Press/MIT Press.
- Santos, J. C. A., Tamaddoni-Nezhad, A., & Muggleton, S. (2009). An ILP system for learning head output connected predicates. In L. S. Lopes, N. Lau, P. Mariano, & L. M. Rocha (Eds.), *LNCS: Vol. 5816. EPIA* (pp. 150–159). Berlin: Springer.
- Shvaiko, P., & Euzenat, J. (2005). A survey of schema-based matching approaches. In S. Spaccapietra (Ed.), *LNCS: Vol. 3730. Journal on data semantics IV* (pp. 146–171). Berlin: Springer.
- Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). YAGO: a core of semantic knowledge. In *Proceedings of the 16th international conference on world wide web (WWW 2007)* (pp. 697–706).
- Suchanek, F. M., Abiteboul, S., & Senellart, P. (2011). PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proceedings of the VLDB Endowment*, 5(3), 157–168.
- Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In H. J. Komorowski & J. M. Zytkow (Eds.), *Lecture notes in computer science* (Vol. 1263, pp. 78–87). Berlin: Springer.
- Yan, X., & Han, J. (2002). gSpan: graph-based substructure pattern mining. In *2nd IEEE international conference on data mining* (pp. 721–724). Los Alamitos: IEEE Comput. Soc.
- Zhang, C., Hoffmann, R., & Weld, D. S. (2012). Ontological smoothing for relation extraction with minimal supervision. In *AAAI conference on artificial intelligence*.

Article IV

Matthijs van Leeuwen and Esther Galbrun

Compression-based Association Discovery in Two-View Data

Submitted for review.

Copyright © The Authors.

Compression-based Association Discovery in Two-View Data

Matthijs van Leeuwen*

Esther Galbrun†

Abstract

Two-view datasets are datasets whose attributes are naturally split into two sets, each providing a different view on the same set of objects. We introduce the task of finding small and non-redundant sets of associations that provide insight in how the two views are related. To achieve this, we propose a novel approach based on the Minimum Description Length principle, in which sets of rules are used to *translate* one view to the other and vice versa. These sets of rules form our models and are dubbed *translation tables*.

We introduce three TRANSLATOR algorithms to find models that compress the data well if there is structure across the two views on the data. The first algorithm is parameter-free and iteratively adds the rule that improves compression most, the other two algorithms use heuristics to achieve better trade-offs between runtime and compression. The empirical evaluation on real-world data shows that only modest numbers of associations are needed to characterize two-view structure that is present in the datasets, while the obtained translation rules are easily interpretable and provide valuable insight in the data.

1 Introduction

Two-view datasets are datasets whose attributes are split into two sets, providing two alternative views on the same set of objects. Two-view data is a form of multi-view data, in which an arbitrary number of views can occur. In practice, a data analyst is often given different sets of information over the same set of objects, and asked to analyze associations across these views.

In the medical domain, for example, persons could be the objects of interest, and one could have both demographic and medical data. In this case, the two views represent clearly different *types* of information. Alternatively, one could have data for the same persons over two different periods in *time*, e.g. before and after treatment. Retailers could also have sets of data collected in different time periods, or else they might be interested in analyzing sales data concerning the same set of products from different *locations*.

In all these examples, there are two views that convey different information concerning the same objects. An apparent question to a data analyst would be: *what*

associations are present across the two views? In general, it is of interest to investigate how much information one view provides about the other, and vice versa.

In particular, we are interested in associations that span both views, e.g. a medical condition that frequently occurs before treatment might imply certain symptoms after treatment with high probability. The association might even hold in both directions, in which case both observations occur mostly together. As another example, for music tracks we could have both machine-extracted audio features and manually curated information on evoked emotions. A question could then be which emotions are evoked by which type of audio features; how are audio features associated to emotions?

Redescription Mining [10] provides a partial answer to these questions by finding pairs of queries, one for each view, that are satisfied by almost the same set of objects. Such query pairs are called *redescriptions*, and quality is usually measured with the Jaccard coefficient of the queried object sets. A substantial limitation is that associations are required to hold in both directions, i.e. a redescription can be interpreted as a bidirectional high confidence association rule. Furthermore, redescriptions are judged individually and the complete set of redescriptions is therefore likely to be redundant.

The obvious alternative would be to mine association rules [1] across the two views. Unfortunately, this approach only considers unidirectional rules and has other disadvantages [13], the most important being the so-called *pattern explosion*: humongous amounts of highly similar rules are found and, consequently, support and confidence thresholds are hard to tune.

A recent trend that addresses these problems is the development of pattern-based *models* using the Minimum Description Length (MDL) principle [6]. Examples include methods for Boolean [11] and uncertain data [2]. Advantages of this approach are 1) that small pattern-based models are interpretable and hence provide insight in the data, and 2) that compression allows the models to be used for other tasks [4].

Approach and contributions. We introduce a compression-based framework for the induction of pattern-based models that *describe* the associations across two views of a dataset, providing *insight* in how the two sides are related. To achieve this goal, we use

*Machine Learning, Department of Computer Science, KU Leuven, Leuven, Belgium
matthijs.vanleeuwen@cs.kuleuven.be.

†Helsinki Institute for Information Technology (HIIT), Department of Computer Science, University of Helsinki, Finland
galbrun@cs.helsinki.fi.

both bidirectional and unidirectional rules as *translation patterns*. This allows us to construct succinct models that allow for easy interpretation.

Our models, dubbed *translation tables*, consist of sets of rules that span the two views. Without loss of generality, we will refer to the two views as left-hand side and right-hand side of the data and denote these by $\mathcal{D}_{\mathbf{L}}$ and $\mathcal{D}_{\mathbf{R}}$ respectively. A *translation rule* $X \rightarrow Y$ implies that if X occurs in a transaction in $\mathcal{D}_{\mathbf{L}}$, then Y is likely to occur in the corresponding transaction of $\mathcal{D}_{\mathbf{R}}$. Vice versa for $X \leftarrow Y$, and $X \leftrightarrow Y$ implies the combination of both. When strong associations across the two views are present in the data, a ‘good’ model should accurately *map*, or *translate*, $\mathcal{D}_{\mathbf{R}}$ to $\mathcal{D}_{\mathbf{L}}$ and vice versa. Hence, a good set of rules establishes a descriptive mapping between the two views and effectively *translates* one view into the other and vice versa.

The MDL principle is *the* induction principle for descriptions and is therefore a good choice for quantifying how good a certain set of rules (i.e. a translation table) is. It essentially states that given a set of models, the best model is the one resulting in the smallest total compressed size. Here, total compressed size includes both the size of the data and the model itself, so that data and model complexity are appropriately balanced. Given this, the task at hand becomes that of finding the translation table that best compresses the mapping between the two sides of a given two-view dataset.

This task is substantially different from existing ones and requires a novel approach: instead of directly compressing the data, which would not give the desired results, we instead compress the mapping across the two views. Thus, we obtain a succinct summary of the relevant associations that span the two views, in the form of a set of both uni- and bidirectional rules. After discussing related work in Section 2, we elaborate on this problem and formalize it in Section 3.

Exhaustive search for the globally optimal translation table is practically unfeasible, but it is possible to find the single rule that gives the largest gain in compression given a dataset and current translation table. Hence, in Section 4 we propose the anytime and parameter-free TRANSLATOR algorithm for finding good translation tables on datasets with a moderate number of attributes. It starts with an empty model and iteratively adds the best rule until no rule that improves compression can be found. Then, we also introduce two variants that select rules from a fixed candidate set, making the approach applicable on larger datasets.

The proposed model and algorithms are empirically evaluated in Section 5. The obtained compression ratios indicate how much two-view structure is present in the datasets. Comparisons to existing methods

unambiguously show that those do not provide adequate solutions to the problem that we consider. Finally, we show by means of examples that the found translation rules are expressive and intelligible.

2 Related Work

Both Exceptional Model Mining (EMM) [7] and Redescription Mining (RM) [10, 5] are concerned with finding patterns in two-view data. EMM aims at finding subsets of the data that stand out with respect to a designated ‘target’ view. As such, EMM is highly asymmetric, with one side used for descriptions and the other as target, as is the case with multilabel classification [12]. RM treats both sides equally but discovers individual high-confidence bidirectional rules, whereas we aim for a non-redundant set of both unidirectional and bidirectional rules.

Association rules have been widely studied since their introduction [1]. Acknowledging the problem of large result sets, methods have been proposed to select subsets of the rules, for example via statistical testing [13]. Supervised pattern set mining methods [3] approach the problem mostly from a classification perspective, that assumes the existence of a single property of interest, i.e. the class label or target. Our proposed method is different as it aims at inducing *compact* models that *describe* structure *across two-view data* and also allows for bidirectional rules.

As mentioned previously, the MDL principle [6] has been used for the induction of pattern-based models before. Consequently, our approach is related to KRIMP [11] and its counterparts, for instance for uncertain data [2]. However, it differs from existing methods for several reasons. First, we consider two-view datasets; concatenating the two sides of the data and applying KRIMP yields very different results. KRIMP uses itemsets, whereas we consider rules. Second, our framework compresses the mapping across two views rather than the data directly, to ensure that we (only) find associations *across* the two sides of the data.

3 Compression for Two-View Associations

We consider Boolean data over a set of objects denoted by \mathcal{O} . Each object is characterized by a transaction over two sets of items, $\mathcal{I}_{\mathbf{L}}$ and $\mathcal{I}_{\mathbf{R}}$ (\mathbf{L} for left, \mathbf{R} for right). That is, each transaction t can be regarded as a pair of itemsets $t = (t_{\mathbf{L}}, t_{\mathbf{R}})$ concerning the same object $o \in \mathcal{O}$, such that $t_{\mathbf{L}} \subseteq \mathcal{I}_{\mathbf{L}}$ and $t_{\mathbf{R}} \subseteq \mathcal{I}_{\mathbf{R}}$. A *two-view dataset* \mathcal{D} is a bag of transactions. Let $|\mathcal{D}|$ denote its size, i.e. $|\{t \in \mathcal{D}\}|$. We use $\mathcal{D}_{\mathbf{L}}$ (resp. $\mathcal{D}_{\mathbf{R}}$) to denote dataset \mathcal{D} projected onto $\mathcal{I}_{\mathbf{L}}$ (resp. $\mathcal{I}_{\mathbf{R}}$). An itemset X is said to *occur* in a transaction t iff $X \subseteq t_{\mathbf{L}} \cup t_{\mathbf{R}}$. The *support* of an itemset X in dataset \mathcal{D} is the bag of transactions in

which X occurs, i.e. $\text{supp}_{\mathcal{D}}(X) = \{t \in \mathcal{D} \mid X \subseteq t_{\mathbf{L}} \cup t_{\mathbf{R}}\}$. We typically omit the index when \mathcal{D} is unambiguous from the context. The *confidence* of a rule $X \rightarrow Y$ is defined as

$$\text{conf}(X \rightarrow Y) = \frac{|\text{supp}(X \cup Y)|}{|\text{supp}(X)|}.$$

To be able to compare rule sets consisting of both unidirectional and bidirectional rules to redescrptions, we propose the following. For all rules and redescrptions, confidence can be expected to be high in *at least one direction*. We therefore use the maximum confidence in either direction, and define conf^+ as

$$\text{conf}^+(X \diamond Y) = \max\{\text{conf}(X \rightarrow Y), \text{conf}(X \leftarrow Y)\}.$$

This slightly resembles all-confidence [9], which also combines confidences for different ‘rule instantiations’.

3.1 MDL for translations. The Minimum Description Length principle [6] states that given a set of models \mathcal{M} and a dataset \mathcal{D} , the best model is that model $M \in \mathcal{M}$ that minimizes

$$L(\mathcal{D} \mid M) + L(M),$$

where $L(\mathcal{D} \mid M)$ is the length, in bits, of the data encoded with M and $L(M)$ is the length, in bits, of the model. Simply put, the best model is the one that gives the best compression of data and model combined.

Thus, to formally define our problem we need to specify 1) the space of possible models \mathcal{M} and how they compress the data, and 2) how to compute encoded lengths of both model and data. In the standard situation, such as with KRIMP [11], encoding the data is straightforward: each transaction is encoded by the model. However, the current problem is different and we are not interested in encoding the data *directly*. Instead, we are interested in encoding the *translation* that connects the two sides of a two-view dataset, to capture any cross-view associations. By using the MDL principle we automatically balance the complexity of our model with the complexity of the translation.

Making this more precise, a *translation* is an exact mapping from one view of a multi-view dataset to another view. In two-view data, we have two such mappings: one from left to right and one from right to left, which we denote by $\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}}$ and $\mathcal{D}_{\mathbf{L} \leftarrow \mathbf{R}}$ respectively. In other words, $\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}}$ can be regarded as a function that translates $t_{\mathbf{L}}$ to $t_{\mathbf{R}}$ for each $t \in \mathcal{D}$.

It is these translations that we want to summarize by compression. Combining the left-to-right and right-to-left translations to make the problem symmetric, the

total encoded length of a bidirectional translation given a model, denoted by $L(\mathcal{D}_{\mathbf{L} \leftrightarrow \mathbf{R}} \mid M)$, is defined as

$$L(\mathcal{D}_{\mathbf{L} \leftrightarrow \mathbf{R}} \mid M) = L(\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}} \mid M) + L(\mathcal{D}_{\mathbf{L} \leftarrow \mathbf{R}} \mid M).$$

Since translation is fully symmetric, for ease of presentation we introduce all definitions and methods for $\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}}$ only, but assume the reverse direction to be defined analogously. Our model class \mathcal{M} is defined as the set of possible *translation tables* for a given dataset.

DEFINITION 1. (TRANSLATION RULE) Let $\mathcal{I}_{\mathbf{L}}$ and $\mathcal{I}_{\mathbf{R}}$ be two sets of items. A translation rule, denoted $X \diamond Y$, consists of a left-hand side itemset $X \subseteq \mathcal{I}_{\mathbf{L}}$, a direction $\diamond \in \{\rightarrow, \leftarrow, \leftrightarrow\}$, and a right-hand side itemset $Y \subseteq \mathcal{I}_{\mathbf{R}}$.

DEFINITION 2. (TRANSLATION TABLE) A translation table T is a three-column table where each row contains a translation rule $X \diamond Y$; one column for each of X , \diamond , and Y .

Given this, we need to specify how to translate the data with a translation table. Because transactions in \mathcal{D} are assumed to be independent from one another, translation can be done individually for each of them. To translate $t_{\mathbf{L}}$ into $t_{\mathbf{R}}$, we initialize $t'_{\mathbf{R}} = \emptyset$ and consider each $X \diamond Y \in T$ in turn. For each $X \rightarrow Y$ and $X \leftrightarrow Y$, we check whether the antecedent occurs in the left-hand side, i.e. whether $X \subseteq t_{\mathbf{L}}$. If this is the case, we add Y to $t'_{\mathbf{R}}$. This translation scheme is presented as Algorithm 1; it takes $t_{\mathbf{L}}$ and T as input and returns a translated transaction $t'_{\mathbf{R}}$, i.e. $t'_{\mathbf{R}} = \text{TRANSLATE}_{\mathbf{L} \rightarrow \mathbf{R}}(t_{\mathbf{L}}, T)$. Note that with this scheme, the order of the rules in T does not influence translation.

3.2 Encoding translations. Ideally, we would have $t'_{\mathbf{R}} = t_{\mathbf{R}}$ for each transaction. Unfortunately, for any realistic dataset \mathcal{D} it will be impossible to find a translation table T that achieves this. Therefore, to ensure lossless encoding of $\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}}$, as required by the MDL principle, we introduce a *correction table* $C_{\mathbf{R}}$. For each transaction t , $c_{\mathbf{R}}^t \in C_{\mathbf{R}}$ is the *difference* between $t_{\mathbf{R}}$ and the translated itemset $t'_{\mathbf{R}}$, i.e. $c_{\mathbf{R}}^t = t_{\mathbf{R}} \oplus t'_{\mathbf{R}}$, where \oplus denotes exclusive or.

Algorithm 1 The $\text{TRANSLATE}_{\mathbf{L} \rightarrow \mathbf{R}}$ algorithm

Input: Transaction $t_{\mathbf{L}}$, translation table T

Output: Translated transaction $t'_{\mathbf{R}}$

- 1: $t'_{\mathbf{R}} \leftarrow \emptyset$
 - 2: **for all** $X \diamond Y \in T$ **do**
 - 3: **if** $\diamond \in \{\rightarrow, \leftrightarrow\} \wedge X \subseteq t_{\mathbf{L}}$ **then**
 - 4: $t'_{\mathbf{R}} \leftarrow t'_{\mathbf{R}} \cup Y$
 - 5: **return** $t'_{\mathbf{R}}$
-

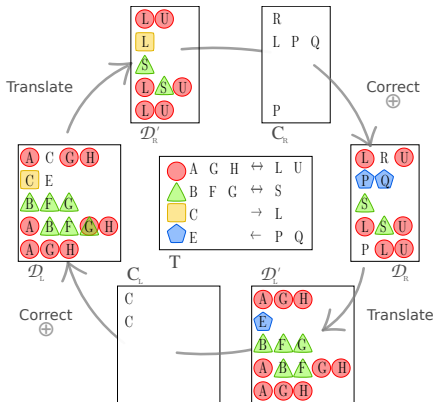


Figure 1: Translation example on a toy dataset, including the two views of the dataset \mathcal{D}_L and \mathcal{D}_R , the translated datasets \mathcal{D}'_L and \mathcal{D}'_R , the correction tables C_L and C_R , and the translation table T in the centre.

We can perfectly translate \mathcal{D}_L into \mathcal{D}_R using T and correction table C_R : for each $t_R \in \mathcal{D}_R$ we have $t_R = \text{TRANSLATE}_{L \rightarrow R}(t_L, T) \oplus C_R$. Hence, the encoded length of the left-to-right translation given T becomes

$$L(\mathcal{D}_{L \rightarrow R} | T) = L(C_R | T),$$

and what remains is to determine how to compute the encoded lengths of the translation and correction tables.

To illustrate the translation scheme, Figure 1 shows example translations in both directions on a toy dataset. Starting from \mathcal{D}_L , the first rule in T is matched and applied in the first, fourth and fifth rows (indicated with red circles). This results in items L and U in the corresponding transactions in \mathcal{D}'_R . After all rules have been applied using the TRANSLATE algorithm, corrections in C_R are applied using exclusive or. This both adds and removes items, and results in \mathcal{D}_R .

3.3 Computing encoded lengths. To encode a translation table, we need to specify how to encode the itemsets it contains. The simplest solution is to encode each item independently, assigning a code with length based on its empirical probability of occurring in the data. For each $I \in \mathcal{I}_L$ this probability is given by

$$P(I | \mathcal{D}_L) = \frac{|\{t \in \mathcal{D}_L | I \in t\}|}{|\mathcal{D}|}.$$

From information theory, we have that the optimal code length corresponding to probability distribution P is $L(I | \mathcal{D}_L) = -\log_2 P(I | \mathcal{D}_L)$. The encoded length

of an itemset X is now given by

$$L(X | \mathcal{D}_L) = \sum_{I \in X} L(I | \mathcal{D}_L) = - \sum_{I \in X} \log_2 P(I | \mathcal{D}_L).$$

We use this encoding for the itemsets in the first column of a translation table, and similarly for itemsets over \mathcal{I}_R in the third column. For the directions, i.e. the second column of the table, a first bit indicates whether a rule is unidirectional or bidirectional, and a second bit represents the direction in case of a unidirectional rule. The length of a direction \diamond is thus

$$L(\diamond) = \begin{cases} 1 & \text{if } \diamond = \leftrightarrow \\ 2 & \text{otherwise} \end{cases}$$

Summing up, the encoded length of a translation table T is given by

$$\begin{aligned} L(T) &= \sum_{X \diamond Y \in T} L(X \diamond Y), \text{ with} \\ L(X \diamond Y) &= L(X | \mathcal{D}_L) + L(\diamond) + L(Y | \mathcal{D}_R). \end{aligned}$$

For the encoding of the correction tables, note that we are only interested in the discovery of *cross-view* associations. This implies that we should not exploit any structure *within* one of the two views for compression, because that would prevent us from finding all cross-view structure. That is, we assume that we can capture all *relevant* structure in the translation table, and the contents of the correction table should be regarded as residue. Under this assumption, we can use the same ‘independent’ encoding for the itemsets in the correction tables as for the translation table, giving

$$L(C_R | T) = \sum_{c \in C_R} L(c | \mathcal{D}_R).$$

One could argue that using the empirical item distributions of the individual views for encoding the correction tables results in suboptimal codes, and that the actual code usage distributions should be used instead. By so doing, however, we avoid introducing additional codes and hence additional tables that map items to these codes. Furthermore, this choice makes it possible to devise an exact algorithm for finding the best rule given a current translation table, which would otherwise be practically unfeasible.

This completes the encoding, and the problem of finding a good set of associations can now be formally defined as finding the translation table that minimizes the total encoded length:

PROBLEM 1. *Given a two-view dataset $\mathcal{D} = (\mathcal{D}_L, \mathcal{D}_R)$ with corresponding translation $\mathcal{D}_{L \leftrightarrow R}$, find*

$$\arg \min_{T \in \mathcal{T}} L(\mathcal{D}_{L \leftrightarrow R}, T) = L(T) + L(C_L | T) + L(C_R | T),$$

where \mathcal{T} is the set of possible translation tables for \mathcal{D} , and $C_{\mathbf{R}}$ and $C_{\mathbf{L}}$ are the correction tables for $\mathcal{D}_{\mathbf{L} \rightarrow \mathbf{R}}$ given T and $\mathcal{D}_{\mathbf{R} \rightarrow \mathbf{L}}$ given T , respectively.

4 The Translator algorithms

We now introduce three different TRANSLATOR algorithms, which take a two-view dataset \mathcal{D} as input and induce a good translation table T by starting from an empty table and iteratively adding rules.

Given a dataset \mathcal{D} , there are $2^{|\mathcal{I}_{\mathbf{L}}|}$ (resp. $2^{|\mathcal{I}_{\mathbf{R}}|}$) possible itemsets for the left-hand side (resp. right-hand side). Since each pair of itemsets can form three different rules ($\rightarrow, \leftarrow, \leftrightarrow$), there are $|\mathcal{R}| = 3 \times 2^{|\mathcal{I}_{\mathbf{L}}|} \times 2^{|\mathcal{I}_{\mathbf{R}}|}$ possible rules. Without further assumptions on the number of rules in a translation table, each possible subset of \mathcal{R} needs to be considered. Since there is no structure that can be used to prune the search space, we resort to a greedy method, as is usual with MDL [11].

Before presenting our algorithms, we investigate how to efficiently compute the gain in compression that can be attained by adding a rule to a translation table.

Each item in a correction table C occurs for one of two reasons: either an item is missing after translation and needs to be added, or it is introduced erroneously and needs to be removed. Hence, we can split C into two separate tables U and E , as follows. Let $U_{\mathbf{R}}$, for *Uncovered*, be a table such that $U_{\mathbf{R}}^t = t_{\mathbf{R}} \setminus t'_{\mathbf{R}}$ for each $t \in \mathcal{D}$, where $t'_{\mathbf{R}} = \text{TRANSLATE}(t_{\mathbf{L}}, T)$ as before. Similarly, let $E_{\mathbf{R}}$, for *Errors*, be a table such that $E_{\mathbf{R}}^t = t'_{\mathbf{R}} \setminus t_{\mathbf{R}}$ for each $t \in \mathcal{D}$. From this it follows that $U \cap E = \emptyset$ and $C = U \cup E$.

In practice, U initially equals \mathcal{D} ; T is empty, and all items are uncovered. By adding rules to T , more items become covered, U becomes smaller, and thus the encoded length of C decreases. On the other hand, E is empty when we start and can only become larger (but to a lesser extent than the decrease of C , or rules would not be added). Once an error is inserted into E it cannot be removed by adding rules.

Now, let $\Delta_{\mathcal{D}, T}(X \diamond Y)$ denote the decrease in total compressed size obtained by adding a rule $r = X \diamond Y$ to a translation table T , i.e. $\Delta_{\mathcal{D}, T}(X \diamond Y) = L(\mathcal{D}_{\mathbf{L} \leftrightarrow \mathbf{R}}, T) - L(\mathcal{D}_{\mathbf{L} \leftrightarrow \mathbf{R}}, T \cup \{r\})$. Given the previous, this can be defined as the reduction in length of the correction table minus the length of the rule itself, as follows:

$$(4.1) \quad \Delta_{\mathcal{D}, T}(X \diamond Y) = \Delta_{\mathcal{D}|T}(X \diamond Y) - L(X \diamond Y),$$

$$(4.2) \quad \Delta_{\mathcal{D}|T}(X \rightarrow Y) = \sum_{t \in \mathcal{D} \wedge X \subseteq t_{\mathbf{L}}} L(Y \cap U_{\mathbf{R}}^t \mid \mathcal{D}_{\mathbf{R}}) - L(Y \setminus (t_{\mathbf{R}} \cup E_{\mathbf{R}}^t) \mid \mathcal{D}_{\mathbf{R}}).$$

These equations follow directly from the definitions given so far. $\Delta_{\mathcal{D}|T}(X \leftarrow Y)$ is defined analogously with

Algorithm 2 The TRANSLATOR-EXACT algorithm

Input: Two-view dataset \mathcal{D}

```

1:  $T \leftarrow \emptyset$ 
2: repeat
3:    $r^* \leftarrow \arg \max_{r \in \mathcal{R}} \Delta_{\mathcal{D}, T}(r)$ 
4:   if  $L(\mathcal{D}, T \cup \{r^*\}) < L(\mathcal{D}, T)$  then
5:      $T \leftarrow T \cup \{r^*\}$ 
6: until no rule added to  $T$ 
7: return  $T$ 

```

\mathbf{L} and \mathbf{R} reversed, and $\Delta_{\mathcal{D}|T}(X \leftrightarrow Y)$ is simply the sum of the two unidirectional variants. Given this, the best candidate rule is the one that maximizes $\Delta_{\mathcal{D}, T}(X \diamond Y)$.

4.1 Iteratively finding the best rule. The idea of the TRANSLATOR-EXACT algorithm, presented in Algorithm 2, is to iteratively add the optimal rule to the current translation table. The greedy scheme starts from an empty translation table, and iteratively adds the rule that improves compression most, until no further improvement can be achieved. Note that the order of the rules in the table does not matter, and that provisional results can be inspected at any time.

To find the optimal rule r^* that maximizes the gain in compression, we use a search based on the ECLAT algorithm [15], traversing the pattern space depth-first while maintaining transaction sets for both X and Y and pruning where possible. Without additional pruning, all non-empty itemset pairs X and Y that occur in the data would be enumerated. For each such pair, all three possible rules are evaluated, i.e. one for each possible direction. To find r^* we only need to keep track of the best solution found so far.

To make search efficient, it is essential to find good solutions as early as possible, and to prune the search space based on the best solution so far. Unfortunately, $\Delta_{\mathcal{D}, T}(X \diamond Y)$ is not (anti)monotonic. However, each XY should occur in the data and therefore all XY that do not occur in \mathcal{D} are pruned (we do not consider rules for which either $X = \emptyset$ or $Y = \emptyset$, as these are not cross-view associations). Furthermore, from the definition of the gain of a rule in Equation 4.2, we observe that any positive gain must come from covering items that are currently uncovered. We can exploit this with a pruning technique similar to those used in high-utility itemset mining [14]. We trivially have that $L(Y \cap U_{\mathbf{R}}^t \mid \mathcal{D}_{\mathbf{R}}) \leq L(U_{\mathbf{R}}^t \mid \mathcal{D}_{\mathbf{R}})$ for any Y and $U_{\mathbf{R}}^t$, and will use it to derive an upper-bound.

That is, for each $t_{\mathbf{R}} \in \mathcal{D}$ the gain for that transaction is upper-bounded by the encoded size of its *uncovered items*. Let $tub(t_{\mathbf{R}})$ denote this transaction-based upper-bound, defined as $tub(t_{\mathbf{R}}) = L(U_{\mathbf{R}}^t \mid \mathcal{D}_{\mathbf{R}})$. Since

for any transaction $tub(t_{\mathbf{R}})$ is constant during search for a single rule, these values are computed once prior to search. We can now check in which rows of the database a rule would be applied and sum the transaction-based bounds. For any rule $X \rightarrow Y$, this gives the following:

$$\Delta_{\mathcal{D},T}(X \rightarrow Y) \leq \sum_{t \in \mathcal{D} \text{ s.t. } t_{\mathbf{L}} \supseteq X} tub(t_{\mathbf{R}}).$$

For a given $X \diamond Y$, the bidirectional instantiation always has the highest potential gain, meaning that we should sum the bounds for the two directions. Finally, we should take the size of the rule into account: extensions of the current rule will be at least as large as the current rule. We thus define the rule-based upper-bound, denoted rub , as

$$rub(X \diamond Y) = \sum_{t \in \mathcal{D} \text{ s.t. } t_{\mathbf{L}} \supseteq X} tub(t_{\mathbf{R}}) + \sum_{t \in \mathcal{D} \text{ s.t. } t_{\mathbf{R}} \supseteq Y} tub(t_{\mathbf{L}}) - L(X \leftrightarrow Y).$$

This bound is based on the supports of itemsets X and Y and decreases monotonically with either support cardinality. Therefore, $X \diamond Y$ and all its possible extensions can be safely pruned when the potential gain given by this bound is lower than the gain of the current best rule. That is, the pruning condition is $rub(X \diamond Y) \leq \Delta_{\mathcal{D},T}(r^*)$.

Prior to search, all $I \in \mathcal{I}$ are ordered descending by $tub(\{I\})$, which determines the order of the depth-first search. This helps find rules with high compression gain as quickly as possible and thus increases the amount of pruning that can be performed.

Finally, the gain for any rule $X \diamond Y$ can be quickly bounded by an upper-bound on the bidirectional rule:

$$qub(X \diamond Y) = |\text{supp}(X)| L(Y | \mathcal{D}_{\mathbf{R}}) + |\text{supp}(Y)| L(X | \mathcal{D}_{\mathbf{L}}) - L(X \leftrightarrow Y).$$

Although this gives no guarantee for rule extensions and thus cannot be used to prune the search space, it is useful to quickly determine whether computing $\Delta_{\mathcal{D},T}(X \rightarrow Y)$ is needed; this computation can be skipped when $qub(X \diamond Y) \leq \Delta_{\mathcal{D},T}(r^*)$.

Depending on the dataset and current translation table, exhaustive search for the best rule may still be computationally too intensive. Therefore, we also propose two faster, approximate methods.

4.2 Iteratively finding good rules. The second algorithm, dubbed TRANSLATOR-SELECT, strongly resembles its exact counterpart: it also greedily adds rules to the table, but does not guarantee to find the best possible rule in each iteration. Instead of generating candidate rules on-the-fly, it *selects* them from a fixed set of

Algorithm 3 The TRANSLATOR-SELECT algorithm

Input: Two-view dataset \mathcal{D} , integer k , candidates C

- 1: $T \leftarrow \emptyset$
- 2: **repeat**
- 3: $R \leftarrow$ select k rules with highest $\Delta_{\mathcal{D},T}(r)$ from C
- 4: $used \leftarrow \emptyset$
- 5: **for** $i = 1 \dots k$ **do**
- 6: consider R_i as $X \diamond Y$
- 7: **if** $X \cap used = \emptyset \wedge Y \cap used = \emptyset$ **then**
- 8: **if** $L(\mathcal{D}, T \cup \{X \diamond Y\}) < L(\mathcal{D}, T)$ **then**
- 9: $T \leftarrow T \cup \{X \diamond Y\}$
- 10: $used \leftarrow used \cup X \cup Y$
- 11: **until** no rule added to T
- 12: **return** T

candidates. This set consists of two-view frequent itemsets, i.e. all itemsets Z for which $|\text{supp}(Z)| > \text{minsup}$, $Z \cap \mathcal{I}_{\mathbf{L}} \neq \emptyset$ and $Z \cap \mathcal{I}_{\mathbf{R}} \neq \emptyset$. These candidates are given as input, and can be mined using any standard frequent itemset mining algorithm that is modified to ensure that each itemset contains items from both views.

TRANSLATOR-SELECT(k), presented in Algorithm 3, selects the top- k rules with regard to compression gain $\Delta_{\mathcal{D},T}$ among all possible rules that can be constructed from the candidate itemsets. Three rules can be constructed for each candidate itemset: one for each direction. When k is set to 1, this implies that the single best rule among the candidates is chosen in each iteration, similar to Algorithm 2. To further speed-up the process, it is possible to choose a larger k , so that multiple rules are selected in each iteration. The selected rules are added to the translation table one by one, but rules that contain an itemset that overlaps with an itemset of a rule previously added in the current iteration are discarded (to this aim, the set of *used* items is maintained). The reason for this is that the compression gain of such a rule has decreased, and it can therefore no longer be assumed to be part of the top- k for the current round.

4.3 Greedily finding good rules. Our third method, called TRANSLATOR-GREEDY, employs single-pass filtering: given a dataset and a candidate set of frequent itemsets (ordered descendingly first by length, then by support in case of equality), it iteratively considers all itemsets one by one. For each itemset that is considered, compression gain is computed for each of the three possible rules, one for each direction. The corresponding rule with the largest gain is added if that gain is strictly positive. If there is no such rule for an itemset, it is discarded and never considered again. This extremely greedy procedure strongly resembles the se-

lection mechanism of KRIMP.

5 Experiments

In this section we empirically evaluate our proposed formalization and algorithms. We analyze the performance of the three methods, compare them to existing methods, and present examples of obtained rules. Due to space limitations, part of our detailed experimental evaluation is provided as an appendix.¹

Datasets. Except for **Mammals** and **Elections**, all datasets were obtained from the LUCS/KDD,² UCI,³ and MULAN⁴ repositories. **Mammals** contains presence records of mammal species in Europe and is a natively Boolean real-world dataset [8], **Elections** contains information about the candidates that participated in the 2011 Finnish parliamentary elections.⁵ It is a good example of a natural two-view dataset, where one looks for associations between candidate profiles and their political views. Basic properties of the datasets are given in Table 1, further details on pre-processing can be found in Appendix A.1.

Implementation. We implemented TRANSLATOR in C++. The source code, datasets and the splits required to be able to reproduce the results will be made publicly available upon publication of this manuscript.

5.1 Comparison of search strategies. We first compare the three different variants of the TRANSLATOR algorithm. As candidate sets for both TRANSLATOR-SELECT and TRANSLATOR-GREEDY we use all closed frequent two-view itemsets up to a given minimum support threshold. Furthermore, SELECT(k) is evaluated for both $k = 1$ and $k = 25$.

For the first batch of experiments we set the lowest possible minimum support threshold, i.e. $minsup = 1$ (threshold not used by EXACT). Consequently, for these experiments we use only datasets with a moderate numbers of items. The results, presented in the top half of Table 1, show large variations in both compression ratio and runtime, which both heavily depend on the characteristics of the dataset. We observe that using compression as stopping criterion results in relatively few rules: in all cases, there are much fewer rules than there are transactions in the dataset. Together with the observation that compression ratios up to 54% are attained, this implies that rules that generalize well can

be found. On the other hand, some datasets can hardly be compressed, indicating that there only few cross-view associations and/or that they do not cover large areas of the data. This is an advantage of the compression-based translation approach that we advocate: if there is little or no structure connecting the two views, this will be reflected in the attained compression ratios. Note, however, that also other properties of the data influence compression. For example, dense data generally results in better compression than sparse data.

The four method instances all yield similar compression ratios and numbers of rules. However, TRANSLATOR-EXACT needs to dynamically construct and explore large parts of the search space in each iteration, and this results in relatively long runtimes. The main problem is that the pruning strategies are only effective in the first few iterations. After that, the gain in compression that a single rule can achieve decreases significantly, so that a much larger part of the search space needs to be explored. This is demonstrated by taking a closer at the construction of translation tables (see Appendix A.2).

SELECT and GREEDY do not suffer from the aforementioned problem, as they generate a candidate set once and only perform candidate testing. SELECT tests all candidates in each iteration, GREEDY tests each candidate exactly once. TRANSLATOR-GREEDY is clearly the fastest, and often approximates the best solution quite well. However, there are exceptions to this. For **Wine**, for example, the compression ratios obtained by EXACT and SELECT are 10% lower (= better) than that obtained by GREEDY.

We now shift our focus to the bottom half of Table 1, which presents results obtained on the larger datasets. We do not have results for the exact method because it takes too long to finish on these datasets. Moreover, we fix $minsup$ such that the number of candidates remains manageable (between 10K and 200K).

We again observe strongly varying results dependent on the data. Unsurprisingly, the GREEDY method is much faster than the SELECT alternatives, but in some cases this also results in poor compression. For example, on **House** it only achieves a compression ratio of 71.45%, compared to 49.26% obtained by SELECT(1).

As expected, there is a trade-off between runtime and solution quality. TRANSLATOR-EXACT attains the best compression rates, but can only be used on small datasets. TRANSLATOR-SELECT provides a good trade-off between compression and runtime. Depending on the dataset, choosing a larger k can be useful to speed-up the search. For example, on **Crime** compression remains practically the same while runtime decreases from 5h15m to 1h27m. When the dataset is very large,

¹See supplementary material.

²<http://cgi.csc.liv.ac.uk/~frans/KDD/Software/LUCS-KDD-DN/DataSets/dataSets.html>

³<http://archive.ics.uci.edu/ml/>

⁴<http://mulan.sourceforge.net/>

⁵<http://blogit.hs.fi/hsnnext/hsn-vaalikone-on-nyt-avointa-tietoa>

Table 1: Datasets and comparison of TRANSLATOR-EXACT, TRANSLATOR-SELECT, and TRANSLATOR-GREEDY. The densities of \mathcal{D}_L and \mathcal{D}_R are denoted as d_L and d_R , respectively. For each experiment, we report the number of obtained rules $|T|$, the compression ratio $L\%=L(\mathcal{D},T)/L(\mathcal{D},\emptyset)$, and the runtime t .

Dataset	<i>minsup</i>	$ \mathcal{D} $	$ \mathcal{I}_L $	$ \mathcal{I}_R $	d_L	d_R	EXACT			SELECT(1)			SELECT(25)			GREEDY		
							$ T $	L%	t	$ T $	L%	t	$ T $	L%	t	$ T $	L%	t
Abalone ³	1	4 177	27	31	0.19	0.13	88	54.81	3h22m	86	54.86	28m	86	54.95	10m51s	114	57.75	19s
Car ²	1	1 728	15	10	0.27	0.30	12	94.18	1m13s	9	94.67	28s	9	94.67	20s	12	95.27	3s
ChessKRvK ²	1	28 056	24	34	0.17	0.09	320	94.89	2d	311	94.94	17h19m	315	94.95	6h22m	314	95.60	3m21s
Nursery ²	1	12 960	19	13	0.26	0.31	28	98.36	3h19m	27	98.36	1h47m	27	98.36	1h15m	19	98.83	4m46s
Tictactoe ²	1	958	15	14	0.33	0.36	61	85.18	35m	64	85.20	8m16s	66	84.86	3m31s	73	90.97	7s
Wine ²	1	178	35	33	0.20	0.21	38	67.99	1h22m	27	69.15	15s	30	69.10	8s	48	79.98	< 1s
Yeast ³	1	1 484	24	26	0.17	0.19	49	81.99	46m	32	82.73	2m16s	32	82.73	2m15s	38	83.00	4s
Adult ²	4885	48 842	44	53	0.18	0.13	—	—	—	8	54.29	50m	8	54.29	29m	19	55.50	7m
CAL500 ⁴	20	502	78	97	0.24	0.07	—	—	—	59	86.45	36m	60	86.48	13m	92	88.88	40s
Crime ²	200	2 215	244	294	0.20	0.19	—	—	—	144	87.45	5h15m	146	87.47	1h27m	183	88.51	2m7s
Elections	47	1 846	82	867	0.06	0.03	—	—	—	80	93.28	36m	83	93.27	12m	132	94.49	28s
Emotions ⁴	40	593	430	12	0.17	0.50	—	—	—	22	97.35	20m	24	97.34	14m	37	97.54	54s
House ³	8	435	26	24	0.35	0.33	—	—	—	37	49.26	14m	37	49.27	8m	50	71.45	23s
Mammals	773	2 575	95	94	0.17	0.17	—	—	—	55	68.23	58m	56	68.31	29m	39	85.85	1m3s

TRANSLATOR-GREEDY may be the best choice, but this may come at the expense of good compression.

5.2 Comparison with other approaches. Although association rule mining, redescription mining, and KRIMP are all related to our work, they also all consider rather different problems. We confirm these differences by means of an extensive empirical comparisons that we present in Appendix A.3; here we provide a summary of the results.

Using a standard association rule mining [1] method to obtain unidirectional, cross-view rules results in thousands of rules (up to 153 609 for **House**). In contrast, our algorithm identifies only small rule sets (containing at most 256 rules, for **ChessKRvK**) and also finds bidirectional rules; these ‘summaries’ are much easier to interpret by a domain expert.

Turning the redescriptions found by the REREM algorithm [5] into translation rules results in very poor compression and even inflation (compression ratios above 100% for eight datasets), despite comparable average maximum confidences. This is a result of redundancy within the results, but also of not exploiting unidirectional associations.

Finally, directly turning the itemsets mined by KRIMP into bidirectional translation rules generally results in a strong increase of the total encoded size (typically doubling or tripling the original size of the translations). Considering these itemsets as candidates for TRANSLATOR-GREEDY still results in poor compression rates. This demonstrates that the identification of cross-view associations requires a different formalization than

that of within-view associations.

From the experimental comparison of our approach to these three techniques, we conclude that translation tables have very different properties from the results of related methods. TRANSLATOR provides better results to the problem considered: smaller sets of rules that provide a better characterization of the associations across the two data views.

5.3 Example rules. We now turn to a qualitative assessment of the rules found by our algorithm, for which we present examples from **Elections** in Figure 2.

The four rules shown clearly conform to the common understanding of the Finnish political landscape. The first rule highlights views on defense, finance, development aid and nuclear energy that are commonly ascribed to the Green party. The second rule conveys that candidates for Change 2011, a Finnish party known for being critical towards immigration, think that current immigration policy is too loose. Observe, however, that the rule is not bidirectional, implying that there are also candidates for other parties that have this opinion. This shows that having both bidirectional and unidirectional rules is useful. Furthermore, the rules are generally easy to interpret by domain experts.

6 Conclusions

We introduced the task of finding small and non-redundant sets of associations that provide insight in how the two sides of two-view datasets are related. To this end, we proposed a novel, compression-based approach that uses rules to translate one view into the

(3) $party = \text{'Green League'}$ \leftrightarrow

Question: A government may decide to cut government spending. Which would be your first choice for cuts? **Answer:** Defense. \wedge

Q: Which of the following statements best describes your views regarding Finland financial support to other euro countries in the crisis? **A:** Supporting the euro is in the interest of Finland itself. \wedge

Q: Which of the following statements most closely matches your vision regarding the global Financial Transaction Tax (FTT) proposed by the EU? **A:** The EU should adopt an FTT, even if the rest of the world does not participate in the system. \wedge

Q: Which of the following statements best describes your views regarding development aid? **A:** Finland must increase its commitment to the development to 0.7 percent during the next legislature. \wedge

Q: Should authorization be granted for the replacement of the two nuclear reactors at the Loviisa power plant? **Importance:** high

(17) $party = \text{'Change 2011'}$ \rightarrow

Q: What do you think of the current Finnish immigration policy? **A:** Too loose.

(21) $gender = \text{'female'}$ \leftarrow

Q: Arms Act was tightened in the autumn of 2010. What should the new parliament do in that domain? **A:** Keeping small arms in the home should be banned. \wedge

Q: Child Benefit is paid for each child living in Finland until the age of 17. What should be done about Child Benefit? **Importance:** high

(26) $party = \text{'National Coalition'} \wedge education\ level = \text{'higher education'}$ \leftarrow

Q: Which of the following statements best describes your views regarding Finland financial support to other euro countries in the crisis? **A:** Supporting the euro is in the interest of Finland itself. \wedge

Q: Should Finland apply for NATO membership? **A:** Yes, but not at the beginning of the legislature.

Figure 2: Example rules mined from Elections with TRANSLATOR-SELECT(1). The numbers, (#), indicate the iterations in which the rules were selected.

other and vice versa. These translation rules can be either unidirectional or bidirectional, and a set of rules together forms a translation table. Our approach generalizes existing methods such as association rule mining and redescription mining, but also avoids redundancy by mining a *set of patterns* rather than individual patterns. For this we use a formalization based on the Minimum Description Length principle.

We presented three TRANSLATOR algorithms for inducing translation tables. The exact variant is parameter-free and iteratively adds the optimal rule to the table, while the second variant iteratively selects the best rule from a fixed set of candidates and is therefore substantially faster. Nevertheless, in practice it approximates the best possible compression ratio very well.

The third variant greedily selects rules in a single pass over a set of candidates and is the fastest of the three, but does not always find a good solution.

The experiments demonstrate that only modest numbers of rules are needed to characterize any cross-view associations in the two-view data. In general, having both bidirectional and unidirectional rules proves useful; the obtained rules are easy to inspect, non-redundant, and provide valuable insight in the data.

References

- [1] R. AGRAWAL, T. IMIELINSKI, AND A. N. SWAMI, *Mining association rules between sets of items in large databases*, in SIGMOD Conference, 1993, pp. 207–216.
- [2] F. BONCHI, M. VAN LEEUWEN, AND A. UKKONEN, *Characterizing uncertain data using compression*, in SDM, SIAM / Omnipress, 2011, pp. 534–545.
- [3] L. DE RAEDT AND A. ZIMMERMANN, *Constraint-based pattern set mining*, in SDM, SIAM, 2007.
- [4] C. FALOUTSOS AND V. MEGALOOIKONOMOU, *On data mining, compression, and kolmogorov complexity*, Data Min. Knowl. Discov., 15 (2007), pp. 3–20.
- [5] 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 (2012), pp. 284–303.
- [6] P. D. GRÜNWARD, *The Minimum Description Length Principle*, MIT Press, 2007.
- [7] D. LEMAN, A. FEELDERS, AND A. J. KNOBBE, *Exceptional model mining*, in ECML/PKDD (2), 2008, pp. 1–16.
- [8] A. J. MITCHELL-JONES ET AL., *The Atlas of European Mammals*, Ac. Press, 1999.
- [9] E. OMIECINSKI, *Alternative interest measures for mining associations in databases*, IEEE Trans. Knowl. Data Eng., 15 (2003), pp. 57–69.
- [10] L. PARIDA AND N. RAMAKRISHNAN, *Redescription mining: Structure theory and algorithms*, in AAAI, M. M. Veloso and S. Kambhampati, eds., 2005, pp. 837–844.
- [11] A. SIEBES, J. VREEKEN, AND M. VAN LEEUWEN, *Item sets that compress*, in SDM, J. Ghosh, D. Lambert, D. B. Skillicorn, and J. Srivastava, eds., SIAM, 2006.
- [12] G. TSOUMAKAS, I. KATAKIS, AND I. P. VLAHAVAS, *Mining multi-label data*, in Data Mining and Knowledge Discovery Handbook, Springer, 2010, pp. 667–685.
- [13] G. I. WEBB, *Discovering significant rules*, in KDD, T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, eds., ACM, 2006, pp. 434–443.
- [14] C.-W. WU, B.-E. SHIE, V. S. TSENG, AND P. S. YU, *Mining top-k high utility itemsets*, in KDD, Q. Yang, D. Agarwal, and J. Pei, eds., 2012, pp. 78–86.
- [15] M. J. ZAKI, S. PARTHASARATHY, M. OGIHARA, AND W. LI, *New algorithms for fast discovery of association rules*, in KDD, 1997, pp. 283–286.

A Experiments complement

A.1 Data pre-processing. The LUCS/KDD repository provides Boolean datasets, the datasets from the other two repositories were pre-processed to make them Boolean: numerical attributes were discretized using five equal-height bins and each categorical attribute-value was converted into an item. For CAL500, the *genre*, *instruments* and *vocals* attributes are used as right-hand side, the rest as left-hand side. In Emotions, all audio features form the left-hand side, while the right-hand side consists of the different emotion labels. For the other repository datasets, the attributes were split such that the items were evenly distributed over two views of similar density.

The Mammals dataset contains presence records of mammal species in Europe and is a natively Boolean real-world dataset [8]. We vertically split the dataset into two views of similar sizes and densities.

The Elections dataset contains information about the candidates that participated in the 2011 Finnish parliamentary elections. This dataset was collected from www.vaalikone.fi, the “election engine” of the Finnish newspaper *Helsingin Sanomat*. The left-hand side contains candidate properties such as party, age, and education, while the answers provided to 30 multiple-choice questions and the assigned importances form the right-hand side. We created an item for each attribute-value. Those items that occurred in more than half of the transactions were discarded because they would result in many rules of little interest.

A.2 Construction of a translation table. Here we zoom in on TRANSLATOR-SELECT(1), the search strategy that provides the best trade-off in terms of compression and runtime, and the House dataset. For this combination we examine the changes in encoded lengths and coverage while rules are iteratively added to the translation table. Figure 3 (left) shows how the numbers of uncovered ones ($|U|$) and errors ($|E|$) evolve, for both sides. Figure 3 (right) shows how the encoded lengths evolve, i.e. the encoded length of the left-to-right translation $L(\mathcal{D}_{L \rightarrow R} | T)$, the encoded length of the right-to-left translation $L(\mathcal{D}_{R \leftarrow L} | T)$, the length of the translation table $L(T)$, and the total encoded length of the bidirectional translation $L(\mathcal{D}_{L \leftrightarrow R}, T)$, which is the sum of the three parts.

As expected, the number of uncovered items quickly drops as rules are added to the translation table, while the number of errors slowly rises. As new rules are added to the translation table, the encoded lengths of both sides decrease accordingly. We note as a general trend, that compression gain per rule decreases quite quickly. This is also what we observed with the exact

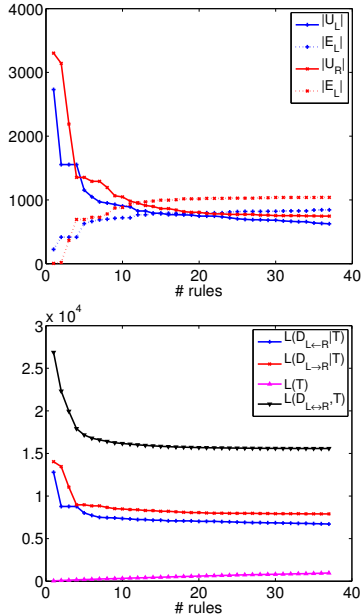


Figure 3: Evolution of the number of uncovered and erroneous items (top), and encoded lengths (bottom) during the construction of a translation table for House with TRANSLATOR-SELECT(1).

search strategy, and what severely limited the power of the pruning scheme. As a consequence, the exact search strategy is most attractive when one is only interested in few rules.

A.3 Comparison with other approaches. Although association rule mining, redescription mining, and KRIMP are all clearly related to our work, they also all solve rather different problems. To emphasize these differences, we will now empirically demonstrate that translation tables are indeed an answer to a substantially different question than each of the aforementioned methods.

First, we consider association rule mining [1], for which we need to choose minimum confidence and support thresholds before we can start mining. To ensure that we can find similar cross-view associations as with our methods, we use the lowest conf^+ and supp values for any rules found in our translation tables as respective thresholds (per dataset). Using these tuned thresholds, we mined all cross-view association rules of either direction using an adapted miner that only mines rules spanning the two views. As shown in Table 2, this

Table 2: Comparing TRANSLATOR to Association rules, REREMi and KRIMP. We report the number of rules $|T|$, maximum confidences conf^+ averaged over the pattern set, and compression ratio L%.

Dataset	TRANSLATOR-SELECT(1)			Asso. rules		REREMi			KRIMP		KRIMP+GREEDY	
	$ T $	conf^+	L%	$ T $	conf^+	$ T $	conf^+	L%	$ T $	L%	$ T $	L%
Abalone	86	0.602	54.86	42755	0.356	35	0.730	76.16	352	330.46	113	60.93
Adult	8	0.790	54.29	9360	0.766	12	0.765	102.58	312	86.59	84	55.90
CAL500	59	0.630	86.45	116306	0.605	25	0.610	104.40	204	272.73	28	92.58
Car	9	0.595	94.67	1978	0.401	5	0.677	115.28	109	271.91	7	99.68
ChessKrvK	256	0.688	95.05	5054	0.421	14	0.340	134.18	1619	816.34	245	96.83
Crime	145	0.734	87.45	67530	0.775	41	0.750	93.66	742	307.33	102	89.99
Elections	80	0.577	93.28	30096	0.462	19	0.619	101.51	792	445.90	31	97.16
Emotions	22	0.726	97.35	69866	0.884	30	0.663	101.26	524	342.93	25	98.33
House	37	0.718	49.26	153609	0.615	26	0.814	59.64	95	200.03	31	56.25
Mammals	55	0.863	68.28	32825	0.860	43	0.833	72.95	157	97.25	67	72.79
Nursery	27	0.566	98.36	3504	0.508	4	0.430	105.15	232	265.48	3	99.88
Tictactoe	64	0.491	85.20	31717	0.335	14	0.568	99.08	165	212.74	49	93.76
Wine	27	0.793	69.15	8434	0.719	20	0.764	81.28	57	165.24	21	75.82
Yeast	32	0.737	82.73	4347	0.679	15	0.557	121.16	127	395.80	36	90.83

resulted in several thousands of association rules per dataset, i.e. up to several orders of magnitude more than are selected for the translation tables. Also, average conf^+ is generally higher for the rules that TRANSLATOR selects. The large advantage of our approach is that we obtain *small, non-redundant sets* of rules, that are consequently easier to interpret by a domain expert. On top of that, we win expressiveness by using both unidirectional and bidirectional rules.

Second, we mined redescrptions with the REREMi algorithm [5], restricted to monotone conjunctions. This algorithm selects (bidirectional) redescrptions based on ad-hoc pruning, driven primarily by accuracy. Table 2 shows that REREMi finds rules with average conf^+ values that are on par to those of TRANSLATOR. TRANSLATOR is slightly better, with 0.68 ± 0.1 over all datasets, versus 0.65 ± 0.13 for REREMi. Furthermore, the identified redescrptions yield poor compression ratios, sometimes even inflating the data as a result of redundancy within the result set. Again, the purpose of the method is different from ours: Redescription Mining aims to find individual bidirectional rules of high accuracy, but these are likely to be redundant and do not explain all associations across the two views of the data (and certainly not unidirectional ones).

Third, we perform an indirect comparison to KRIMP. Since KRIMP uses itemsets and we use rules, a direct comparison is impossible. However, we can still show that the itemsets found by KRIMP do not capture the same associations as the rules discovered by TRANSLATOR. For this we devise two ways to transform a set of itemsets into a translation table. Note that this necessarily implies that we use the task and problem as formalized in Section 3 for this comparison.

For the first variant, KRIMP code tables mined

from the joint two-view datasets are directly interpreted as bidirectional rules and put in a translation table. Then, compression is computed using the translation scheme introduced in this paper. For the second variant, denoted KRIMP+GREEDY, the itemsets obtained with KRIMP were considered as candidates and used as input for the TRANSLATOR-GREEDY method. KRIMP was used with pruning enabled, and closed frequent itemsets were used as candidates, with *minsup* set to the values mentioned in Table 1. Table 2 shows the results obtained with both variants.

The results clearly demonstrate that KRIMP aims at finding associations that are very different from those that TRANSLATOR identifies. KRIMP finds many more associations, and when treated as translation table the complete set of associations results in extremely bad compression: compression rates range up to 816.34%, implying that the translation is inflated to more than eight times its original encoded size. The rightmost columns show that even when the KRIMP itemsets are considered as candidates and fed to the TRANSLATOR-GREEDY method, the resulting translation tables are generally larger and yield worse compression ratios. This demonstrates that the associations found by KRIMP are not a good solution to the task considered in this paper.

Overall, we conclude that translation tables have substantially different properties from the results of the related methods considered in this paper, and that TRANSLATOR provides better results to the problem considered: smaller sets of rules that provide a better characterization of the associations across the two data views.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FI-00014 University of Helsinki, FINLAND

JULKAISUSARJA A

SERIES OF PUBLICATIONS A

Reports are available on the e-thesis site of the University of Helsinki.

- A-2008-1 I. Autio: Modeling Efficient Classification as a Process of Confidence Assessment and Delegation. 212 pp. (Ph.D. Thesis)
- A-2008-2 J. Kangasharju: XML Messaging for Mobile Devices. 24+255 pp. (Ph.D. Thesis).
- A-2008-3 N. Haiminen: Mining Sequential Data – in Search of Segmental Structures. 60+78 pp. (Ph.D. Thesis)
- A-2008-4 J. Korhonen: IP Mobility in Wireless Operator Networks. 186 pp. (Ph.D. Thesis)
- A-2008-5 J.T. Lindgren: Learning nonlinear visual processing from natural images. 100+64 pp. (Ph.D. Thesis)
- A-2009-1 K. Hätönen: Data mining for telecommunications network log analysis. 153 pp. (Ph.D. Thesis)
- A-2009-2 T. Silander: The Most Probable Bayesian Network and Beyond. 50+59 pp. (Ph.D. Thesis)
- A-2009-3 K. Laasonen: Mining Cell Transition Data. 148 pp. (Ph.D. Thesis)
- A-2009-4 P. Miettinen: Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms. 164+6 pp. (Ph.D. Thesis)
- A-2009-5 J. Suomela: Optimisation Problems in Wireless Sensor Networks: Local Algorithms and Local Graphs. 106+96 pp. (Ph.D. Thesis)
- A-2009-6 U. Köster: A Probabilistic Approach to the Primary Visual Cortex. 168 pp. (Ph.D. Thesis)
- A-2009-7 P. Nurmi: Identifying Meaningful Places. 83 pp. (Ph.D. Thesis)
- A-2009-8 J. Makkonen: Semantic Classes in Topic Detection and Tracking. 155 pp. (Ph.D. Thesis)
- A-2009-9 P. Rastas: Computational Techniques for Haplotype Inference and for Local Alignment Significance. 64+50 pp. (Ph.D. Thesis)
- A-2009-10 T. Mononen: Computing the Stochastic Complexity of Simple Probabilistic Graphical Models. 60+46 pp. (Ph.D. Thesis)
- A-2009-11 P. Kontkanen: Computationally Efficient Methods for MDL-Optimal Density Estimation and Data Clustering. 75+64 pp. (Ph.D. Thesis)
- A-2010-1 M. Lukk: Construction of a global map of human gene expression - the process, tools and analysis. 120 pp. (Ph.D. Thesis)
- A-2010-2 W. Hämäläinen: Efficient search for statistically significant dependency rules in binary data. 163 pp. (Ph.D. Thesis)
- A-2010-3 J. Kollin: Computational Methods for Detecting Large-Scale Chromosome Rearrangements in SNP Data. 197 pp. (Ph.D. Thesis)
- A-2010-4 E. Pitkänen: Computational Methods for Reconstruction and Analysis of Genome-Scale Metabolic Networks. 115+88 pp. (Ph.D. Thesis)
- A-2010-5 A. Lukyanenko: Multi-User Resource-Sharing Problem for the Internet. 168 pp. (Ph.D. Thesis)
- A-2010-6 L. Daniel: Cross-layer Assisted TCP Algorithms for Vertical Handoff. 84+72 pp. (Ph.D. Thesis)

- A-2011-1 A. Tripathi: Data Fusion and Matching by Maximizing Statistical Dependencies. 89+109 pp. (Ph.D. Thesis)
- A-2011-2 E. Junttila: Patterns in Permuted Binary Matrices. 155 pp. (Ph.D. Thesis)
- A-2011-3 P. Hintsanen: Simulation and Graph Mining Tools for Improving Gene Mapping Efficiency. 136 pp. (Ph.D. Thesis)
- A-2011-4 M. Ikonen: Lean Thinking in Software Development: Impacts of Kanban on Projects. 104+90 pp. (Ph.D. Thesis)
- A-2012-1 P. Parviainen: Algorithms for Exact Structure Discovery in Bayesian Networks. 132 pp. (Ph.D. Thesis)
- A-2012-2 J. Wessman: Mixture Model Clustering in the Analysis of Complex Diseases. 118 pp. (Ph.D. Thesis)
- A-2012-3 P. Pöyhönen: Access Selection Methods in Cooperative Multi-operator Environments to Improve End-user and Operator Satisfaction. 211 pp. (Ph.D. Thesis)
- A-2012-4 S. Ruohomaa: The Effect of Reputation on Trust Decisions in Inter-enterprise Collaborations. 214+44 pp. (Ph.D. Thesis)
- A-2012-5 J. Sirén: Compressed Full-Text Indexes for Highly Repetitive Collections. 97+63 pp. (Ph.D. Thesis)
- A-2012-6 F. Zhou: Methods for Network Abstraction. 48+71 pp. (Ph.D. Thesis)
- A-2012-7 N. Välimäki: Applications of Compressed Data Structures on Sequences and Structured Data. 73+94 pp. (Ph.D. Thesis)
- A-2012-8 S. Varjonen: Secure Connectivity With Persistent Identities. 139 pp. (Ph.D. Thesis)
- A-2012-9 M. Heinonen: Computational Methods for Small Molecules. 110+68 pp. (Ph.D. Thesis)
- A-2013-1 M. Timonen: Term Weighting in Short Documents for Document Categorization, Keyword Extraction and Query Expansion. 53+62 pp. (Ph.D. Thesis)
- A-2013-2 H. Wettig: Probabilistic, Information-Theoretic Models for Etymological Alignment. 130+62 pp. (Ph.D. Thesis)
- A-2013-3 T. Ruokolainen: A Model-Driven Approach to Service Ecosystem Engineering. 232 pp. (Ph.D. Thesis)
- A-2013-4 A. Hyttinen: Discovering Causal Relations in the Presence of Latent Confounders. 107+138 pp. (Ph.D. Thesis)
- A-2013-5 S. Eloranta: Dynamic Aspects of Knowledge Bases. 123 pp. (Ph.D. Thesis)
- A-2013-6 M. Apiola: Creativity-Supporting Learning Environments: Two Case Studies on Teaching Programming. 62+83 pp. (Ph.D. Thesis)
- A-2013-7 T. Polishchuk: Enabling Multipath and Multicast Data Transmission in Legacy and Future Internet. 72+51 pp. (Ph.D. Thesis)
- A-2013-8 P. Luosto: Normalized Maximum Likelihood Methods for Clustering and Density Estimation. 67+67 pp. (Ph.D. Thesis)
- A-2013-9 L. Eronen: Computational Methods for Augmenting Association-based Gene Mapping. 84+93 pp. (Ph.D. Thesis)
- A-2013-10 D. Entner: Causal Structure Learning and Effect Identification in Linear Non-Gaussian Models and Beyond. 79+113 pp. (Ph.D. Thesis)