

Annalisa Appice Michelangelo Ceci
Corrado Loglisci Giuseppe Manco
Elio Masciari Zbigniew W. Ras (Eds.)

LNAI 7765

New Frontiers in Mining Complex Patterns

First International Workshop, NFMCP 2012
Held in Conjunction with ECML-PKDD 2012
Bristol, UK, September 2012, Revised Selected Papers

 Springer

Lecture Notes in Artificial Intelligence 7765

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

Annalisa Appice Michelangelo Ceci
Corrado Loglisci Giuseppe Manco
Elio Masciari Zbigniew W. Ras (Eds.)

New Frontiers in Mining Complex Patterns

First International Workshop, NFMCP 2012
Held in Conjunction with ECML-PKDD 2012
Bristol, UK, September 24, 2012
Revised Selected Papers



Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Annalisa Appice
Michelangelo Ceci
Corrado Loglisci
Università degli Studi di Bari "Aldo Moro", Dipartimento di Informatica
Via Orabona 4, 70126 Bari, Italy
E-mail: {annalisa.appice, michelangelo.ceci, corrado.loglisci}@uniba.it

Giuseppe Manco
Elio Masciari
Institute for High Performance Computing and Networks (ICAR)
National Research Council (CNR)
Via Pietro Bucci 41C, 87036 Rende, Italy
E-mail: {manco, masciari}@icar.cnr.it

Zbigniew W. Ras
University of North Carolina, Department of Computer Science
9201 University City Boulevard, Charlotte, NC 28223, USA
and Warsaw University of Technology, Institute of Computer Science
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
E-mail: ras@uncc.edu

ISSN 0302-9743
ISBN 978-3-642-37381-7
DOI 10.1007/978-3-642-37382-4
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349
e-ISBN 978-3-642-37382-4

Library of Congress Control Number: 2013934109

CR Subject Classification (1998): H.2.8, H.2, H.3, I.2.6

LNCS Sublibrary: SL 7 – Artificial Intelligence

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

New Frontiers in Mining Complex Patterns (NFMCP 2012)

Data mining and knowledge discovery can be considered today as mature research fields, with numerous algorithms and studies available to extract knowledge from data in different forms. Although most existing data mining approaches look for patterns in tabular data, there are also numerous studies where the focus is on complex data (e.g., multi-table data, XML data, Web data, time series and sequences, graphs and trees).

This book sets out to explore emerging technologies and applications where complex patterns in expressive languages are principally extracted from new prominent data sources such as blogs, event or log data, medical data, spatio-temporal data, social networks, mobility data, sensor data and streams, and so on. The individual contributions of this book illustrate advanced data mining techniques that preserve the informative richness of complex data and allow for efficiently and effectively identifying complex information units present in such data.

The papers presented in this book are revised and significantly extended versions of those accepted for presentation at the First International Workshop on New Frontiers in Mining Complex Patterns (MCP 2011), held in Bristol, UK, on September 24, 2012, in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2012).

The book is composed of four parts and a total of 15 chapters.

Part I gives a view of mining rich (relational) datasets by illustrating some complex situations and the related complexity. It contains two chapters. Chapter 1 proposes a learning paradigm where users can write (or adapt) their operators, according to the problem, the data representation, and the way the information should be navigated. The chapter explains the necessary steps for using data instances, background knowledge, rules, programs, and operators in the same functional language. Chapter 2 presents a feature selection method to improve the performance of relational learning without affecting the learned hypothesis and evaluates the approach on four datasets with the popular system Aleph and the state-of-the-art relational learner nFOIL.

Part 2 analyzes issues posed by mining complex patterns from miscellaneous data. It consists of four chapters. Chapter 3 investigates the use of temporal data mining in a declarative framework to analyze the log files of computer networks. It describes the analysis of log files based on XML. Chapter 4 analyzes data aggregation and correlation in a relational scenario where data are characterized by very large fluctuations that are neither attributable to noise nor outliers. Chapter 5 explores the use of machine learning techniques in geographical ethnomusicology in order to predict the distribution of music from around the world.

Chapter 6 investigates the concept of object-driven action rules and presents a new pair-based way of examining temporal and object-driven systems. The chapter presents a case study in the analysis of distortions of speech disorders in children.

Part 3 gives a general overview of mining complex patterns from trajectory and sequence data by illustrating issues and solutions. It contains three chapters. Chapter 7 presents a complete framework to cluster trajectory data streams. Chapter 8 describes a new approach to extract patterns from a complex sequences including both dimensional items and itemsets. Chapter 9 illustrates an approach to cluster moving object trajectories whose data are constrained in a (road-) network.

Finally, Part 4 presents technologies and applications where complex patterns are discovered from graphs and networks. It contains six chapters. Chapter 10 explores the relationship between the graph structure and the distribution of attribute values. Chapter 11 presents a probabilistic graph framework to model the uncertainty inherent in real-world domains by means of probabilistic edges whose value quantifies the likelihood of the edge existence or the strength of the link it represents. Chapter 12 describes a technique to automatically extract conceptual graphs from text and illustrates the method in a scenario concerning social networks on sociopolitical and economic topics. Chapter 13 considers a dynamic network scenario, where nodes/relationships can be added or removed and relationships can change in their type over time and presents a method to discover evolution chains that express the temporal evolution of the network. Chapter 14 describes an approach to quickly, capillary, and effectively ease information spreading in a multi-social-network context. Chapter 15 illustrates the problem of discovering predictive performance models in process mining. It describes an ensemble-based clustering method, where multiple predictive clustering trees are learnt and integrated into an overall (predictive) clustering model.

We would like to thank all the authors who submitted papers for publication in this book and all the workshop participants and speakers. We are also grateful to the members of the Program Committee and external referees for their excellent work in reviewing submitted and revised contributions with expertise and patience. A special thanks is due to both the ECML PKDD Workshop Chairs and to the members of the ECML PKDD organization team who made the event possible. Last but not the least, we thank Alfred Hofmann of Springer for his continuous support.

January 2013

Annalisa Appice
Michelangelo Ceci
Corrado Loglisci
Giuseppe Manco
Elio Masciari
Zbigniew W. Ras

Organization

Program Chairs

Annalisa Appice	University of Bari “Aldo Moro”, Bari, Italy
Michelangelo Ceci	University of Bari “Aldo Moro”, Bari, Italy
Corrado Loglisci	University of Bari “Aldo Moro”, Bari, Italy
Giuseppe Manco	ICAR-CNR, Rende, Italy
Elio Masciari	ICAR-CNR, Rende, Italy
Zbigniew W. Ras	University of North Carolina, Charlotte, USA & Warsaw University of Technology, Poland

Program Committee

Francesco Bonchi	Yahoo! Research Barcelona, Spain
Saso Dzeroski	Jozef Stefan Institute, Slovenia
Floriana Esposito	University of Bari “Aldo Moro”, Italy
Dimitrios Gunopulos	University of Athens, Greece
Mohand-Saïd Hacid	University Claude Bernard Lyon 1, France
Dino Ienco	IRSTEA Montpellier, UMR TETIS, France
Donato Malerba	University of Bari “Aldo Moro”, Italy
Stan Matwin	University of Ottawa, Canada
Dino Pedreschi	University of Pisa, Italy
Fabrizio Riguzzi	University of Ferrara, Italy
Eirini Spyropoulou	University of Bristol, UK
Jerzy Stefanowski	Poznan University of Technology, Poland
Maguelonne Teisseire	IRSTEA Montpellier, UMR TETIS, France
Shusaku Tsumoto	Shimane University, Japan
Herna L. Viktor	University of Ottawa, Canada
Alicja Wiczorkowska	Polish-Japanese Institute of IT, Poland
Djamel Zighed	Université Lumière, Lyon 2, France

Additional Reviewers

Nicola Di Mauro	Diego Pennacchioli
Fabio Fumarola	Gianvito Pio
Valerio Grossi	Salvatore Rinzivillo
Massimo Guarascio	Ettore Ritacco
Lucrezia Macchia	Daniela Stojanova
Krystyna Napierala	Aneta Trajanov

Table of Contents

Mining Rich (Relational) Datasets

Learning with Configurable Operators and RL-Based Heuristics	1
<i>Fernando Martínez-Plumed, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana</i>	
Reducing Examples in Relational Learning with Bounded-Treewidth Hypotheses	17
<i>Ondřej Kuželka, Andrea Szabóová, and Filip Železný</i>	

Mining Complex Patterns from Miscellaneous Data

Mining Complex Event Patterns in Computer Networks	33
<i>Dietmar Seipel, Philipp Neubeck, Stefan Köhler, and Martin Atzmueller</i>	
Learning in the Presence of Large Fluctuations: A Study of Aggregation and Correlation	49
<i>Eric Paquet, Herna Lydia Viktor, and Hongyu Guo</i>	
Retracted: Machine Learning as an Objective Approach to Understanding Music	64
<i>Claire Q and Ross D. King</i>	
Pair-Based Object-Driven Action Rules	79
<i>Ayman Hajja, Alicja A. Wieczorkowska, Zbigniew W. Ras, and Ryszard Gubrynowicz</i>	

Mining Complex Patterns from Trajectory and Sequence Data

Effectively Grouping Trajectory Streams	94
<i>Gianni Costa, Giuseppe Manco, and Elio Masciari</i>	
Healthcare Trajectory Mining by Combining Multidimensional Component and Itemsets	109
<i>Elias Egho, Chedy Raïssi, Dino Ienco, Nicolas Jay, Amedeo Napoli, Pascal Poncelet, Catherine Quantin, and Maguelonne Teisseire</i>	
Graph-Based Approaches to Clustering Network-Constrained Trajectory Data	124
<i>Mohamed Khalil El Mahrsi and Fabrice Rossi</i>	

Mining Complex Patterns from Graphs and Networks

Finding the Most Descriptive Substructures in Graphs with Discrete and Numeric Labels	138
<i>Michael Davis, Weiru Liu, and Paul Miller</i>	
Learning in Probabilistic Graphs Exploiting Language-Constrained Patterns	155
<i>Claudio Taranto, Nicola Di Mauro, and Floriana Esposito</i>	
Improving Robustness and Flexibility of Concept Taxonomy Learning from Text	170
<i>Fabio Leuzzi, Stefano Ferilli, and Fulvio Rotella</i>	
Discovering Evolution Chains in Dynamic Networks	185
<i>Corrado Loglisci, Michelangelo Ceci, and Donato Malerba</i>	
Supporting Information Spread in a Social Internetworking Scenario	200
<i>Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino</i>	
Context-Aware Predictions on Business Processes: An Ensemble-Based Solution	215
<i>Francesco Folino, Massimo Guarascio, and Luigi Pontieri</i>	

Erratum

Machine Learning as an Objective Approach to Understanding Music	E1
<i>Claire Q and Ross D. King</i>	
Author Index	231

Learning with Configurable Operators and RL-Based Heuristics*

Fernando Martínez-Plumed, Cèsar Ferri, José Hernández-Orallo,
and María José Ramírez-Quintana

DSIC, Universitat Politècnica de València, Camí de Vera s/n, 46022 València, Spain
{fmartinez, cferri, jorallo, mramirez}@dsic.upv.es

Abstract. In this paper, we push forward the idea of machine learning systems for which the operators can be modified and finetuned for each problem. This allows us to propose a learning paradigm where users can write (or adapt) their operators, according to the problem, data representation and the way the information should be navigated. To achieve this goal, data instances, background knowledge, rules, programs and *operators* are all written in the same functional language, Erlang. Since changing operators affect how the search space needs to be explored, heuristics are learnt as a result of a decision process based on reinforcement learning where each action is defined as a choice of operator and rule. As a result, the architecture can be seen as a ‘system for writing machine learning systems’ or to explore new operators.

Keywords: machine learning operators, complex data, heuristics, inductive programming, reinforcement learning, Erlang.

1 Introduction

The number and performance of machine learning techniques dealing with complex, structured data has considerably increased in the past decades. However, the performance of these systems is usually linked to a transformation of the feature space (possibly including the outputs as well) to a more convenient, flat, representation, which typically leads to incomprehensible patterns in terms of the transformed (hyper-)space. Alternatively, other approaches do stick to the original problem representation but rely on specialised systems with embedded operators that are only able to deal with specific types of data.

Despite all these approaches and the vindication of more general frameworks for data mining [6], there is no general-purpose machine learning system which can deal with *all* of these problems *preserving* the problem representation. There

* This work was supported by the MEC projects CONSOLIDER-INGENIO 26706 and TIN 2010-21062-C02-02, GVA project PROMETEO/2008/051, and the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA), and funded by the Ministerio de Economía y Competitividad in Spain. Also, F. Martínez-Plumed is supported by FPI-ME grant BES-2011-045099.

are of course several paradigms using, e.g., distances or kernel methods for structured data [13,9] which can be applied to virtually any kind of data, provided we can define similarity functions to compare the individuals. However, this generality comes at the cost of losing the original problem representation and typically losing the recursive character of many data structures.

Other paradigms, such as inductive programming (ILP [23], IFP [16] or IFLP [12]), are able to tackle any kind of data thanks to the expressive power of first-order logic (or term rewriting systems). However, each system has a predefined set of operators (e.g., *lgg* [24], inverse entailment [22], splitting conditions in a decision tree, or others) and an embedded heuristic. Even with the help of background knowledge it is still virtually impossible to deal with, e.g., an XML document, if we do not have the appropriate operators to delve into its structure.

In this paper we present and explore a general rule-based learning setting where operators can be defined and customised for each kind of problem. While one particular problem may require generalisation operators, another problem may require operators which add recursive transformations to explore the structure of the data. A right choice of operators can embed transformations on the data but can also determine the way in which rules are generated and transformed, so leading to (apparently) different learning systems. Making the user or the problem adapt its own operators is significantly different to the use of feature transformations or specific background knowledge. In fact, it is also significantly more difficult, since operators can be very complex things and usually embed the essence of a machine learning system. A very simple operator, such as *lgg*, requires several lines of code in almost any programming language, if not more. Writing and adapting a system to a new operator is not always an easy task. As a result, having a system which can work with different kinds of operators at the same time is a challenging proposal beyond the frontiers of the state of the art in machine learning.

In addition, machine learning operators are tools to explore the hypothesis space. Consequently, some operators are usually associated to some heuristic strategies (e.g., generalisation operators and bottom-up strategies). By giving more freedom to the kind of operators a system can use, we lose the capacity to analyse and define particular heuristics to tame the search space. This means that heuristics must be overhauled, as *decisions* about the operator that must be used at each particular state of the learning process.

We therefore propose a setting where operators can be written or modified by the user. Since operators are defined as functions which transform patterns, we clearly need a language for defining operators which can integrate the representation of the examples, patterns and operators. We will argue that functional programming languages, with reflection and higher-order primitives, are appropriate for this, and we will choose a powerful and relatively popular programming language in this family, Erlang [1]. A not less important reason for using a functional language is that operators can be understood by the users and properly linked with the data structures used in the examples and background knowledge, so making the specification of new operators easier. The language also sets the

general representation of examples as equations, patterns as rules and models as sets of rules.

From here, we devise a flexible architecture which works with populations of rules and programs, which *evolve* as in an evolutionary programming setting or a learning classifier system [14]. Operators are applied to rules and generate new rules, which are combined with existing or new programs. With appropriate operators and using some optimality criteria (based on coverage and simplicity) we will eventually find some good solutions to the learning problem. However, without heuristics, the number of required iterations gets astronomically high. This issue is addressed with a reinforcement learning (RL) approach, where the application of an operator over a rule is seen as a decision problem, for which learning also takes place, guided by the optimality criteria which feed a rewarding module. Interestingly, different problems using the same operators can reuse the heuristics. As a result, the architecture can be seen as a ‘system for writing machine learning systems’ or to explore new operators.

The paper is organised as follows. Section 2 makes a short account of the many approaches and ideas which are related to this proposal. Section 3 introduces how operators are expressed and applied. Section 4 describes the RL-based heuristics used to guide the learning process. Section 5 includes some examples which illustrate how operators are defined and how solutions are reached. Section 6 closes the paper.

2 Previous Work

The system we present in this paper is related to different areas of machine learning: learning from complex data, reinforcement learning, Learning Classifier Systems, evolutionary techniques, meta-learning, etc. In this section we summarise some of the previous works in these fields somehow related to our proposal.

Inductive programming [16], inductive logic programming (ILP) [23] and some of the related areas such as relational data mining [8] are arguably the oldest attempts to handle complex data. They can be considered *general* machine learning systems, because any problem can be represented, preserving its structure, with the use of the Turing-complete languages underneath: logic, functional or logic-functional. Apart from their expressiveness, the advantage of these approaches is the capability of capturing complex problems in a comprehensible way. ILP, for instance, has been found especially appropriate for scientific theory formation tasks where the data are structured, the model may be complex, and the comprehensibility of the generated knowledge is essential. Learning systems using higher-order (see, e.g., [19]) were one of the first approaches to deal with complex structures, which were usually flattened in ILP. Despite the power of higher-order functions to explore complex structure, this approach has never become mainstream.

All these systems are based on the use of different fixed operators. For instance, Plotkin’s lgg [24] operator works well with a specific-to-general search.

The ILP system Progol [22] combines the Inverse Entailment with general-to-specific search through a refinement graph. The Aleph system [26] is based on Mode Direct Inverse Entailment (MDIE). In inductive functional logic programming, the FLIP system [12] includes two different operators: inverse narrowing and a consistent restricted generalisation (CRG) generator. In any case, the set of operators configures and delimits the performance of each learning system. Also, rules that are learned on a first stage can be reused as background knowledge for subsequent stages (incremental learning). Hybrid approaches that combine Genetic Algorithms and ILP have also been introduced as in [29].

As an evolution of ILP into the fields of (statistical) (multi-)relational learning or related approaches, many systems have been developed to work with rich data representations. In [4], for example, we can find an extensive description of the current and emerging trends in the so-called ‘structured machine learning’ where the authors propose to go beyond supervised learning and inference, and consider decision-theoretic planning and reinforcement learning in relational and first-order settings.

Structured Prediction (SP) is one example of learning from complex data context, where not only the input is complex but also the output. This has led to new and powerful techniques, such as Conditional Random Fields (CRFs) [18], which use a log-linear probability function to model the conditional probability of an output y given an input x where Markov assumptions are used in order to make inference tractable. Other well-known Global Model is SVM for Interdependent and Structured Output spaces (SVM-ISO, also known as SVM^{struct}) as a SP evolution of [13] (Kernels) or [9] (distances). [30]. Also, *hierarchical classification* can be viewed as a case of SP where taxonomies and hierarchies are associated with the output [17].

Some of these previous approaches use special functions (probabilistic distributions, metrics or kernels) explicitly defined on the individual space. These methods either lack a model (they are instance-based methods) or the model is defined in terms of the transformed space. A recent proposal which has tried to re-integrate the distance-based approach with the pattern-based approach is [11], (leading, e.g., to Newton trees [21]).

There have been several approaches applying planning and reinforcement learning to structured machine learning [28]. While the term Relational Reinforcement Learning (RRL) [7,28] seems to come to mind, it offers state-space representation that is much richer than that used in classical (or propositional) methods, but its goal is not structured data. Other related approaches are, for instance, incremental models [3,20] which try to solve the combinatorial nature of the very large input/output structured spaces since the structured output is built incrementally. These methods can be applied to a wide variety of techniques such as parsing, machine translation, sequence labelling and tree mapping.

Finally, there is an approach, somewhat in between genetic algorithms and reinforcement learning, known as *Learning Classifier Systems (LCSs)* [15]. LCSs employ two biological metaphors: evolution and learning which are respectively embodied by the genetic algorithm, and a reinforcement learning-like mechanism

appropriate for the given problem. Both mechanisms rely on what is referred to as the *environment* of the system (the source of input data). The architecture of our system will resemble in some ways the LCS approach.

Learning to learn is one of the (required) features of our setting and is related to the area of meta-learning [2]. Learning at the metalevel is concerned with accumulating experience on the performance of multiple applications of a learning system. A more integrated approach resembling meta-learning and incremental learning is [25], where the authors present the *Optimal Ordered Problem Solver* (OOPS), an optimally fast way of incrementally solving each task in the sequence by reusing successful code from previous tasks.

3 Configuring Rule Operators

After this review of related work, we still perceive a lack of flexibility in the way in which different problems can be handled, especially when structured learning is required. As we have mentioned in Section 1, in this paper we set the goal of constructing a system which can be configured with different (possibly user-defined) operators, and where the heuristics are also learned from previous applications of operators for the same or similar problems. As a long-term goal, this can be roughly seen as a general system for designing customised systems for applications with complex data.

In order to achieve the above-mentioned goals, we need to use configurable operators, instead of hard-wired operators. Changing hard-wired operators requires the modification of dozens of lines of code and usually entails a re-writing (or complete overhauling) of heuristics. Instead, in our approach the heuristics will be substituted by a reinforcement learning approach, which will determine which pair of operator and rule will be chosen at each state of the system.

Additionally, we will represent operators in the same language already used for examples, models and background knowledge. The advantages of using the same representation language (in this case, rules expressed as unconditional / conditional equations) has been previously shown by the fields of ILP, IFP and IFLP (except for operators). Hence, we look for a flexible language, with powerful features for defining operators and able to represent all other elements (theories and examples) in an understandable way. For this reason we use Erlang, a functional language with reflection mechanisms which allows us to interact easily with the meta-level representation of how rules and programs are transformed by operators.

3.1 Notation

Let Σ be a set of *function symbols* together with their arity and \mathcal{X} a countably set of *variables*, then $\mathcal{T}(\Sigma, \mathcal{X})$ denotes the set of *terms* built from Σ and \mathcal{X} . The set of variables occurring in a term t is denoted $Var(t)$. A term t is a *ground term* if $Var(t) = \emptyset$.

An equation is an expression of the form $l = r$ where l and r are terms. l is called the left hand side (*lhs*) of the equation and r is the right hand side (*rhs*). \mathcal{R} denotes the space of all (conditional) functional rules ρ of the way l [when G] $\rightarrow T, r$ where l and r are the lhs and the rhs of ρ (respectively), $G = \{g_1, g_2, \dots, g_m \mid m \geq 0\}$ is a set of conditions or Boolean expressions called guards, and $T = b_1, \dots, b_n$, the tail of ρ , is a sequence of equations. If $G = \emptyset$, then ρ is said to be an unconditional rule. Let $\mathcal{P} = 2^{\mathcal{R}}$ be the space of all possible functional programs formed by sets of rules $\rho \in \mathcal{R}$. Given a program $p \in \mathcal{P}$, we say that term t reduces to term s with respect to p , $t \rightarrow_p s$, if there exists a rule l [when G] $\rightarrow T, r \in p$ such that a subterm of t at occurrence u matches l with substitution θ , all conditions $h_i\theta$ holds, for each equation $b_{i_l} = b_{i_r} \in T$, $b_{i_l}\theta$ and $b_{i_r}\theta$ have the same normal form (that is, $b_{i_l}\theta \rightarrow_p^* b$, and $b_{i_r}\theta \rightarrow_p^* b$ and b can not be further reduced) and s is obtained by replacing in t the subterm at occurrence u by $r\theta$.

An example e is a rule without condition nor tail, that is e is of the form $l \rightarrow r$, being r in normal form and both l and r are ground. We say that an example $l \rightarrow r$ is covered by a program p (denoted by $p \models \{l \rightarrow r\}$) if l and r have the same normal form with respect to p . A functional program $p \in \mathcal{P}$ is a solution of a learning problem defined by a set of positive examples E^+ , a (possibly empty) set of negative examples E^- and a background theory B if it covers all positive examples, $B \cup p \models E^+$ (posterior sufficiency or completeness), and does not cover any negative example, $B \cup p \not\models E^-$ (posterior satisfiability or consistency). Our system has the aim of obtaining complete solutions, but their consistency is not a mandatory property, so approximate solutions are possible. The function $Cov^+ : 2^{\mathcal{R}} \rightarrow \mathbb{N}$ calculates the positive coverage of a program $p \in 2^{\mathcal{R}}$ and it is defined as $Cov^+(p) = Card(\{e \in E^+ : B \cup p \models e\})$, where $Card(S)$ denotes the cardinality of the set S . Additionally, the function $Cov^- : 2^{\mathcal{R}} \rightarrow \mathbb{N}$ calculates the negative coverage of a program $p \in 2^{\mathcal{R}}$ and it is defined as $Cov^-(p) = Card(\{e \in E^- : B \cup p \models e\})$.

As we can see in Figure 1, our system works with two sets: a set of rules $R \subseteq \mathcal{R}$ and a set of programs $P \subseteq \mathcal{P}$, where each program $p \in P$ is composed by rules belonging to R . Initially, the set of rules R is populated with the positive evidence E^+ and the set of programs P is populated with as many unitary programs as there are rules in R .

3.2 Operators

The definition of customised operators is one of the key concepts of our proposal. The idea is to transform the set of rules R using a set of *operators* O (provided by the user or existing in the system). An operator $o \in \mathcal{O}$ is then a function $o : 2^{\mathcal{R}} \rightarrow 2^{\mathcal{R}}$ where $O \in \mathcal{O}$ will be the set of operators defined by the user.

An operator can be seen as a piece of code (as complex as the user may wants) which performs modifications over the *lhs* or *rhs* of a rule and which is written in the same functional language as the system (Erlang) to take advantage of its high-order and reflection capabilities. The main idea is that, when the user wants to deal with a new problem, he/she can define his/her own set of *operators*, especially suited for the data structures of the problem. This feature allows our system to adapt to the problem at hand.

Depending on the operators the user provides to the system, it could well behave as a decision tree or, more precisely, as a coverage-based rule learning system (if we implement operators that apply some conditions on the rules), or as a bottom-up concept covering algorithm (if we provide generalisation operators). That is, the system may behave very differently by changing the operators.

Let us see an example. Given a rule $FName(Arguments) \rightarrow RHS$, where $Arguments$ is a list, imagine that we want to define an operator for obtaining the head of $Arguments$ and return it as the rhs of a new rule. This operator could be defined as:

$$\begin{aligned} takeHead(FName(Arguments) \rightarrow RHS) [when\ Arguments\ is\ a\ List] \\ \Rightarrow (FName(Arguments) \rightarrow head(Arguments)). \end{aligned}$$

where \Rightarrow represents the rule transformation relation defined by the operators. The codification in Erlang could be as follows:

```
Operator_takeHead(Rule) ->
(1)  {function,_,FName,_,{clause,_,Arguments,Guards,RHS}} = Rule,
(2)  {cons,_,L1,L2} = Arguments,
(3)  {function,_,FName,_,{clause,_,Arguments,Guards,L1}}.
```

where identifiers with a capital letter followed by any combination of uppercase and lowercase letters and underscores are Erlang variables, and other static (or constants) literals are Erlang atoms. In line 1, the `Rule` is parsed and transformed into a valid Erlang *abstract syntax tree* (AST) in order to easily access to its components: the Erlang forms `FName`, `Arguments`, `Guards` and `RHS`. Next, the operator decomposes `Arguments` into the Erlang meta-expression for lists (line 2), and finally, line 3 returns the new AST constructed by replacing the `RHS` part by `L1` in the AST obtained in line 1. For simplicity, we have omitted some further code for checking the arity and type of `Arguments`.

Our system also has a special kind of operators, called *combiners*, that only apply to programs. The *Program Generator* module (Figure 1) applies a combiner to the last rule ρ' generated by the *Rule Generator* module and the population of programs P . Thus, a combiner $c \in C$ can be formally described as a function $c : \mathcal{P} \times \mathcal{P} \rightarrow \mathcal{P}$ that transforms programs into programs.

By default, our system provides two simple combiners (although other possibilities are considered): *addition*, joins the new rule ρ' generated with the best program (in terms of optimality) to the population P ; and *union* which joins the two best programs (also in terms of optimality) in P .

4 LR-Based Heuristics

The freedom given to the user concerning the definition of their own operators implies the impossibility of defining specific heuristics to explore the search space. This means that heuristics must be overhauled, as decisions about the operator that must be used at each particular state of the learning process. For this, we have developed a model-based reinforcement learning approach, where the application of an operator over a rule is seen as a decision problem, for which

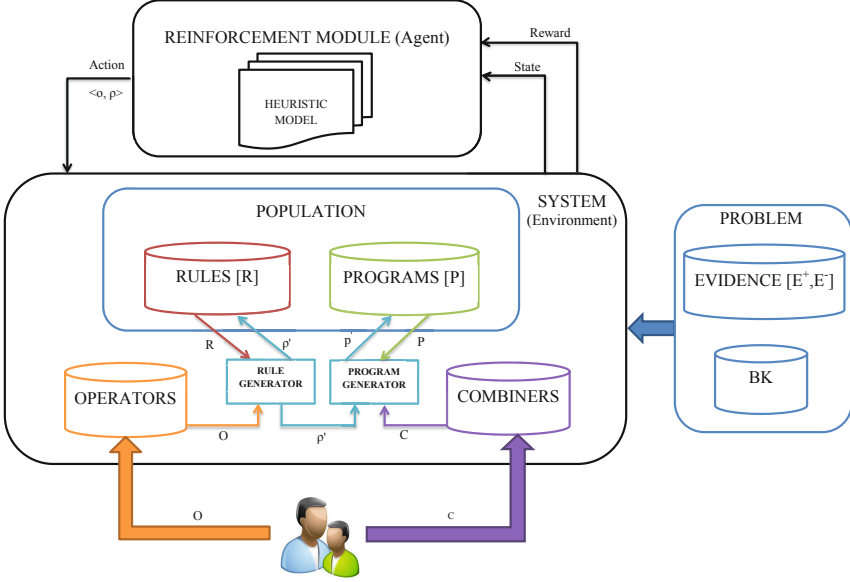


Fig. 1. Prototype System Architecture

learning also takes place, guided by the optimality criteria which feed a rewarding module. Below we will describe our approach.

4.1 State of the System

To guide the learning process we need a picture of the system in each step of the process (before and after applying an action) in terms of the quality of the set of rules and programs generated until now. Formally, we define a state at each iteration t of the system as a tuple $\sigma_t = \langle R, P \rangle$ which represent the population of rules R and programs P in t . The probable infinite number of states makes the abstraction of states necessary. How to do this? As we want to find a good solution to the learning problem, we describe each state σ_t by a tuple of features $s_t = \langle \phi_1, \phi_2, \phi_3, \phi_4, \phi_5 \rangle$ from which to extract relevant information in t :

1. *Global optimality* (ϕ_1): This feature shows the average optimality of all programs in P_t . In turn, the optimality of each program p consists of four factors:
 - *Positive Coverage* measures the proportion of positive examples covered by the program:

$$PosCov(p) = \frac{Cov^+(p)}{Card(E^+)} \quad (1)$$

- *Negative Coverage* measures the proportion of negative examples covered by the program:

$$NegCov(p) = \frac{Cov^-(p)}{Card(E^-)} \quad (2)$$

- *Program Length Ratio* measures the cardinality of p w.r.t. the cardinality of the positive evidence:

$$ProgLength(p) = \frac{Card(p)}{Card(E^+)} \quad (3)$$

- *Applied Operators Ratio*, the idea is to penalise programs which have used a large number of operators:

$$OperatorsRate(p) = \frac{\sum_{\rho \in p} Card(PrevOperators(\rho))}{Card(O) \cdot Card(p)} \quad (4)$$

where $PrevOperators(\rho)$ is the list of previous operators applied to obtain the rule ρ . The optimality of a program p is computed by weighting the four factors according to its importance, in a way inspired by the *MDL/MML principle* [31]:

$$Opt(p) = w_1 \cdot PosCov(p) - w_2 \cdot NegCov(p) - w_3 \cdot ProgLength(p) - w_4 \cdot OperatorsRate(p) \quad (5)$$

by default, $w_1 = 0.4$, $w_2 = 0.3$, $w_3 = 0.1$ and $w_4 = 0.2$.

Finally, the *Global optimality* factor is then calculated as the average of the optimalities of all programs in the system:

$$OptGlobal(P_t) = \frac{1}{Card(P_t)} \sum_{p \in P_t} Opt(p) \quad (6)$$

2. *Average Size of Rules* (ϕ_2): measures the average size of all the rules in R_t . In particular, we compute the size of a rule ρ as in [12]:

$$Size(\rho) = 1 + n_v/2 + n_c + n_f \quad (7)$$

with n_v , n_c and n_f being, respectively, the number of variables, constants and functors of only the rhs of r .

3. *Average Size of programs* (ϕ_3): measures the average cardinality of all the programs in P_t in terms of the number of rules.
4. *Best Rule Optimality* (ϕ_4): is the optimality of the best rule (as unitary program) generated until now.
5. *Best Program Optimality* (ϕ_5): is the optimality of the best program generated until now.

4.2 Decisions

For each iteration of the system, we have to select the rule and operator to produce new rules. Depending on the problem to solve, the number of required iterations to learn a problem could be astronomically high. To address this issue we need a particular heuristic to tame the search space and make good decisions about the choice of rule and operator, in which the application of an operator to a rule is seen as a decision problem.

For that, we model the decision process as a typical reinforcement learning task. Formally, our decision problem is a four-tuple $\langle \mathcal{S}, \mathcal{A}, \tau, \omega \rangle$ where: \mathcal{S} is an infinite state space; \mathcal{A} is a finite actions space ($\mathcal{A} = \mathcal{O} \times \mathcal{R}$); $\tau : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is a transition function between states and $\omega : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. These components are defined below:

- **States.** Each state is described by five features as we have seen in section 4.1.
- **Actions.** An action is a tuple $\langle o, \rho \rangle$ with $\rho \in R$ and $o \in O$ that represents the operator o to be applied to the rule.
- **Transitions.** Transitions are deterministic. A transition τ evolves the current sets of rules and programs by applying the operators selected (together with the rule) and the combiners.
- **Rewards.** The optimality criteria seen above is used to feed the rewards. In particular, we use the result returned by equation (5) as reward.

With all these elements, the aim of our decision process is to find a policy $\pi : S \rightarrow A$ that maximises

$$V^\pi(s_t) = \sum_{i=0}^{\infty} \gamma^i w_{t+i} \quad (8)$$

for all s_t , where $\gamma \in [0, 1]$ is the *discount parameter* which determines the importance of the future rewards ($\gamma = 0$ only considers current rewards, while $\gamma = 1$ strives for a long-term high reward).

At each point in time, the reinforcement learning policy can be in one of the states s_t of S and selects an action $a_t = \pi(s_t) \in A$ to execute. Executing such action a_t in s_t will change the state into $s_{t+1} = \tau(s_t, a_t)$, and the policy receives a reward $w_t = \omega(s_t, a_t)$. The policy does not know the effects of the actions, i.e. τ and ω are not known by the policy and need to be learned. This is the typical formulation of reinforcement learning [27] but using features to represent the states.

In our setting, for the reinforcement learning module, we use a hybrid between model-free value-function methods (which search for action that maximises values) and model-based methods (which generalise τ and ω) [27]. Our approach uses the *state-value* function ($Q(s, a)$, which returns q values) generalising it with a regression model, actually a Linear Regression, where $s \in \mathcal{S}$, $a \in \mathcal{A}$, and finally, the quality values $q \in \mathbb{R}$.

A model $M : S \times \mathcal{A} \rightarrow \mathbb{R}$ calculates the optimality or q -value for each state and action. By using $a_t = \arg \max_{a \in \mathcal{A}} \{M(s_t, a_i)\}$ we get the best action for state s_t . Once we have the action, it is carried out to obtain a new state $s_{t+1} = \tau(s_t, a_t)$.

In order to train the model we need to provide different states and actions as an input, and quality values as output. We use q values for the state-value function as in Q-learning [32], so to train the model we use a matrix $Q = |S| \times |O| \times |R|$ where S is the set of states reached so far, O is the set of operators and R is the set of rules generated. Both sets, S and R , grow in each step of the system (the number of operators is constant), therefore, the matrix also grows in terms of the number of rows (states) and columns (actions). In Table 1 we can see an example of a Q -matrix that can be used to train our model. Before the system starts, this matrix is initialised with one row (state s_0) with q values equal to 1 for every action (combinations of operators and rules). In this way, the model trained with this matrix will have the same probability of selection for all possible actions at the initial steps of the algorithm.

Table 1. Q -matrix example

state (s)					action (a)		q
ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	o	ρ	
1.223	1.473	3.431	1.88	1.99	2	12	0.78
1.301	1.511	3.431	1.88	1.99	5	27	0.65
...							

Once the system has started, at each step, the Q -matrix is updated (as we will see below) and the model can be retrained periodically.

To update each q value in the Q -matrix at each step we use the following formula, as in Q -learning:

$$Q[s_t, a_t] \leftarrow Q[s_t, a_t] + \alpha \times \left[w_{t+1} + \gamma \max_{a_{t+1}} M(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \quad (9)$$

where the max future value is obtained by the model instead of a Q -matrix. α ($\alpha \in [0, 1]$) is the *learning rate* which determines to what extent the newly acquired information will override the old information ($\alpha = 0$ makes the agent not to learn anything, while $\alpha = 1$ makes the agent consider only the most recent information); and $\gamma \in [0, 1]$ is the discount parameter. By default, $\alpha = 0.5$ and $\gamma = 0.5$.

Using our *Reinforcement Learning* approach, populations of rules and programs are updated at each step of the algorithm. First, the *Rule Generator* process (Figure 1) gets the operator o and the rule ρ returned as an action $a = \langle o, \rho \rangle$ by the *Reinforcement Learning Module* (policy). This process applies the operator over the rule obtaining a new rule ρ' (if the operator is not suitable for the rule selected, the process returns the same rule) which is added to R . The way in which the set programs is evolved is by the *Program Generator* process. This takes the new rule generated ρ' (if appropriate) as input, the set of programs P and the set of combiners C and generates a new program p' (which is added to P) applying the combiners over the previous inputs.

4.3 Stopping Criterion

The process is limited to a maximum number of iterations which is also determined by the user or when the prototype finds the best solution (a program which covers all the positive evidence and does not cover any negative examples) with a minimum optimality value, whichever comes first.

5 Examples

In this section, we describe three different examples where we illustrate how operators are defined and used to iteratively approach the solution. We also show more details about our system¹ and how it solves these problems².

¹ Available at <http://users.dsic.upv.es/~flip/SystemMetaRL.rar>

² Available at <http://users.dsic.upv.es/~flip/SystemMetaRLProblems.rar>

5.1 Sequence Processing

Let us start with a toy example of the kind used in structured prediction, where not only the input is structured but also the output. Consider the problem of learning a transformation over the words formed by a given alphabet. More precisely, suppose we have a set of instances where both the input and output are lists (i.e., strings). Consider the very particular case where we have a small alphabet of a non-empty finite set of symbols $\Sigma = \{a, t, c, g, u\}$ and the transformation just replaces t with u . Instances would look like this: $trans([t, c, g, a, t]) \rightarrow [u, c, g, a, u]$.

The first thing we need to define is the basic *replacement functions* for the symbols in the alphabet. This is done in the background knowledge, with functions like: $f_{at}(a) \rightarrow t$; $f_{cg}(c) \rightarrow g$; ... Typically, all the combinations can be defined or only some of them if some replacements are not possible.

According to the data structure of examples (a string), we need a way to navigate the structure and apply local or global changes. In order to do this we need to define appropriate operators. The first operator, *applyMap* is a mechanism to convert a rule into another rule which introduces the higher-order function *map*, which applies a parametrised function to the whole list. The definition of this operator is written in Erlang, but it can be informally defined as follows: $applyMap(trans(X) \rightarrow Y) \Rightarrow trans(X) \rightarrow map(V_F, X)$, where X and Y stand for any list and V_F is a function variable (a higher-order variable).

In order to introduce a replacement function, we need more operators, such as *addBK_f*, which fills the gap V_F by introducing the function f from the BK. Note that at this moment it seems a matter of taste whether we define one operator for each replacement function or a single stochastic operator for all of them, but the difference is important for heuristics. An example of one of each of these operators is: $addBK_f(trans(X) \rightarrow map(V_F, X)) \Rightarrow trans(X) \rightarrow map(f, X)$. Finally, we need a way of generalising input (and output) strings. This is performed by the *genPat* operator: $genPat(trans(X) \rightarrow Y) \Rightarrow trans(V_S) \rightarrow Y$, where V_S is a string variable.

For this toy example there is a simple sequence of operator applications which turns a simple example into a general solution. For instance, given the instance $trans([t, c, g, a, t]) \rightarrow [u, c, g, a, u]$, we have this sequence.

$$\begin{aligned} genPat(trans([t, c, g, a, t]) \rightarrow [u, c, g, a, u]) &\Rightarrow trans(V_S) \rightarrow [u, c, g, a, u] \\ applyMap(trans(V_S) \rightarrow [u, c, g, a, u]) &\Rightarrow trans(V_S) \rightarrow map(V_F, V_S) \\ addBK_{f_{tu}}(trans(V_S) \rightarrow map(V_F, V_S)) &\Rightarrow trans(V_S) \rightarrow map(f_{tu}, V_S) \end{aligned}$$

This latter equation $trans(V_S) \rightarrow map(f_{tu}, V_S)$ is the solution for this toy example. Given the simplicity and the relatively small number of operators, the effect of the coverage mechanisms and the heuristics is not critical, and the system solves this problem (with five positive and five negative examples) in 9.58 seconds using 58 iterations.

5.2 Bunch of Keys

We will continue with a more complex problem, a well-known multi-instance classification problem. Consider the problem of determining whether a key in a bunch of keys can open a door [19]. More precisely, for each bunch of keys either no key opens the door or there is at least one key which opens the door. Each instance is given by a bunch of keys, where each key has several features, so there is a two-level structure (sets of lists). While this is a prototypical multiple-instance problem, it is similar to a number of important practical problems, e.g., drug activity prediction [5].

We model a *Bunch* of keys as a set of keys. Each key, in turn, is modeled as a list capturing four of its properties: the company that makes it (*Abloy*, *Chubb*, *Rubo*, *Yale*), its number of prongs (an integer), its length (*Short*, *Medium*, *Long*) and its width (*Narrow*, *Normal*, *Broad*). A training example (a bunch with two keys which does open the door) may look like this: $opens([[abloy, 3, medium, narrow], [chubb, 6, medium, normal]]) = true$.

Given a set of such examples, we want to learn the function $opens : Bunch \rightarrow \{True, False\}$. For this, we need a function $setExists(Key, Bunch)$ which evaluates (*True* or *False*) whether there exists a *Key* in a *Bunch*. This function will belong to the background knowledge. We also need to provide the system with a set of operators. We again need an operator which incorporates conditions on the right hand side of a rule: $addBK(opens(X) = True) \Rightarrow opens(X) \rightarrow setExists([], X)$.

This incorporates an empty list of conditions. Now we need operators to add conditions. We will have one operator for each attribute value. For instance, the operator for inserting a condition for keys with *abloy* is: $KCond(opens(X) \rightarrow setExists(C, X)) \Rightarrow opens(X) \rightarrow setExists([abloy|C], X)$.

Finally, we need a generalisation operator which introduces a variable instead of a list: $genPat(opens(X) = Y) \Rightarrow opens(V_L) \rightarrow Y$.

If the system and operators are provided, given the original evidence for this example (five *True* instances and four *False* instances), it will return the following definition: $opens(X) \rightarrow setExists([abloy, medium], X)$, which means that *a bunch of keys opens the door if and only if it contains an abloy key of medium length*, which is the proposed solution for this classical example. The system solves this problem in 17.88 seconds using 60 iterations.

5.3 Web Categorisation

The last example corresponds to a *web* classification problem with a higher level of difficulty. It was originally proposed in [10]. The evidence of the problem is modelled with 3 parameters described as follows: *Structure* (the graph of links between pages is represented as ordered pairs where each node encodes a linked page), *Content* (the content of the web page is represented as a set of attributes with the keywords, the title, etc.), and *Connections* (the information derived from connections to a web server which is encoded by means of a numerical attribute with the daily number of connections).

The goal of the problem is to categorise which web pages are about sports. A training example looks like this: $sportsWeb(Structure, Content, Connections) \rightarrow$

true where the *Structure* attribute may be for instance $\{[\{olympics, games\}, [swim]], \{[swim], [win]\}, \{[win], [medal]\}\}$ and is interpreted in the following way: the first component of the list stands for the current web page with keywords “olympics” and “games”. This page links to another page which has “swim” as its only keyword. There are other two connections. The *Content* may be $\{[\{olympics, 30\}, \{held, 10\}, \{summer, 40\}]\}$, which represents the frequency (number of occurrences) of the most relevant words in the web page. Finally, *Connections* is just an integer attribute which represents the number of connections.

Given the structure of the data, we need to add functions to the background knowledge to navigate this structure. We define *graphExists(Edge, Graph)* which checks whether an edge is in a graph, and *setExists(Key, List)* which tests whether the keyword Key belongs to the list. Again, we also need to provide the system with a set of operators. As in previous cases, we can reuse a generic operator to select some function from the background knowledge (one for each function) in order to replace the right hand side of the rules: $addBK_{graph}(sportsWeb(S, C, U) \rightarrow True) \Rightarrow sportsWeb(S, C, U) \rightarrow graphExists(\{\}, \{\}, S)$, which introduces an empty condition about a connection between pages. We can similarly define an operator for introducing a condition over the sets.

Another useful operator takes some type constants and add adds them to the condition of the *setExists* function (first attribute) and another operator which generate a node and adds it as a node to search in the graph attribute of the function *graphExists*:

$$linkl_{football}(sportsWeb(S, C, U) \rightarrow graphExists(\{X, Y\}, S)) \\ \Rightarrow sportsWeb(S, C, U) \rightarrow graphExists(\{[football|X], Y\}, S).$$

Note that this operator is parametrised for the different attribute values. Finally, we need a generalisation operator for each input pattern of the rules: $genPat_1(sportsWeb(S, C, U) \rightarrow True) \Rightarrow sportsWeb(V_S, C, U) \rightarrow True$. There are also some other operators to generalise the second and third arguments.

Our system found the following correct program which defines the *sportsWeb* function:

$$\{sportsWeb(V_S, V_C, V_U) \rightarrow graphExists(\{[final], [match]\}, V_S). \\ sportsWeb(V_S, V_C, V_U) \rightarrow setExists(\{[athens]\}, V_C). \\ sportsWeb(V_S, V_C, V_U) \rightarrow setExists(\{[europe]\}, V_C). \}$$

which means that *if the word ‘athens’ or ‘europe’ appears in Content, and Structure contains the link {[final], [match]} then this is a sport web page.* The system solves this problem (with seven positive examples and 2 negative examples) in 19.02 seconds using 42 iterations.

6 Conclusions and Future Work

The increasing interest in learning from complex data has led to a more integrated view of this area, where the same (or similar) techniques are used for a

wide range of problems using different data and pattern representations. This general view has not been accompanied by general systems which otherwise need to be modified when the original data representation and structure changes. In fact, the most general approach can still be found in ILP (or the more general area of inductive programming). However, each system is still specific to a set of embedded operators and heuristics.

In this paper, we have proposed that more general systems can be constructed by not only giving power to data and background knowledge representation but also to a flexible operator redefinition and the reuse of heuristics across problems and systems. This carries a computational cost. In order to address this issue we rely on the definition of customised operators, depending on the data structures and problem at hand. This can be done by the user, using a language for expressing operators. A generalised operator choice entails generalised heuristics, since the use of different operators precludes the system to use specialised heuristics for each of them. The choice of the wight pair of operator and rule has been reframed as a decision process, as a *reinforcement learning* problem.

We have included some illustrative examples with a first system implementing the general architecture, and we have seen where the flexibility stands out. Our immediate future work is focused on the reuse of operators and heuristics (RL models) across different problems.

Overall, we are conscious that our approach entails some risks, since a general system which can be instantiated to behave virtually like any other system by a proper choice of operators is an ambitious goal. We think that for cocomplex problems that cannot be solved by the system with its predefined operators, the system can be used to investigate which operators are more suitable. In more general terms, this can be used as a system testbed, where we can learn and discover some new properties, limitations and principles for more general machine learning systems that can be used in the future.

References

1. Armstrong, J.: A history of erlang. In: Proceedings of the Third ACM SIGPLAN Conf. on History of Programming Languages, HOPL III, pp. 1–26. ACM (2007)
2. Brazdil, P., Giraud-Carrier: Metalearning: Concepts and systems. In: Metalearning. Cognitive Technologies, pp. 1–10. Springer, Heidelberg (2009)
3. Daumé III, H., Langford, J.: Search-based structured prediction (2009)
4. Dietterich, T., Domingos, P., Getoor, L., Muggleton, S., Tadepalli, P.: Structured machine learning: the next ten years. *Machine Learning* 73, 3–23 (2008)
5. Dietterich, T.G., Lathrop, R., Lozano-Perez, T.: Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence* 89, 31–71 (1997)
6. Džeroski, S.: Towards a general framework for data mining. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 259–300. Springer, Heidelberg (2007)
7. Dzeroski, S., De Raedt, L., Driessens, K.: Relational reinforcement learning. *Machine Learning* 43, 7–52 (2001), 10.1023/A:1007694015589
8. Dzeroski, S., Lavrac, N. (eds.): *Relational Data Mining*. Springer (2001)
9. Estruch, V., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Similarity functions for structured data. an application to decision trees. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* 10(29), 109–121 (2006)

10. Estruch, V., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Web categorisation using distance-based decision trees. *ENTCS* 157(2), 35–40 (2006)
11. Estruch, V., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Bridging the Gap between Distance and Generalisation. *Computational Intelligence* (2012)
12. Ferri-Ramírez, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Incremental learning of functional logic programs. In: Kuchen, H., Ueda, K. (eds.) *FLOPS 2001*. LNCS, vol. 2024, pp. 233–247. Springer, Heidelberg (2001)
13. Gärtner, T.: *Kernels for Structured Data*. PhD thesis, Universität Bonn (2005)
14. Holland, J.H., Booker, L.B., Colombetti, M., Dorigo, M., Goldberg, D.E., Forrest, S., Riolo, R.L., Smith, R.E., Lanzi, P.L., Stolzmann, W., Wilson, S.W.: What is a learning classifier system? In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) *IWLCS 1999*. LNCS (LNAI), vol. 1813, pp. 3–32. Springer, Heidelberg (2000)
15. Holmes, J.H., Lanzi, P., Stolzmann, W.: Learning classifier systems: New models, successful applications. *Information Processing Letters* (2002)
16. Kitzelmann, E.: Inductive programming: A survey of program synthesis techniques. In: Schmid, U., Kitzelmann, E., Plasmeijer, R. (eds.) *AAIP 2009*. LNCS, vol. 5812, pp. 50–73. Springer, Heidelberg (2010)
17. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: *Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997*, pp. 170–178. Morgan Kaufmann Publishers Inc., San Francisco (1997)
18. Lafferty, J., McCallum, A.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *ICML 2001*, pp. 282–289 (2001)
19. Lloyd, J.W.: Knowledge representation, computation, and learning in higher-order logic (2001)
20. Maes, F., Denoyer, L., Gallinari, P.: Structured prediction with reinforcement learning. *Machine Learning Journal* 77(2-3), 271–301 (2009)
21. Martínez-Plumed, F., Estruch, V., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Newton trees. In: Li, J. (ed.) *AI 2010*. LNCS, vol. 6464, pp. 174–183. Springer, Heidelberg (2010)
22. Muggleton, S.: Inverse entailment and Progol. *New Generation Computing* (1995)
23. Muggleton, S.H.: Inductive logic programming: Issues, results, and the challenge of learning language in logic. *Artificial Intelligence* 114(1-2), 283–296 (1999)
24. Plotkin, G.: A note on inductive generalization. *Machine Intelligence* 5 (1970)
25. Schmidhuber, J.: Optimal ordered problem solver. *Maching Learning* 54(3), 211–254 (2004)
26. Srinivasan, A.: *The Aleph Manual* (2004)
27. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press (1998)
28. Tadepalli, P., Givan, R., Driessens, K.: Relational reinforcement learning: An overview. In: *Proc. of the Workshop on Relational Reinforcement Learning* (2004)
29. Tamaddoni-Nezhad, A., Muggleton, S.: A genetic algorithms approach to ILP. In: Matwin, S., Sammut, C. (eds.) *ILP 2002*. LNCS (LNAI), vol. 2583, pp. 285–300. Springer, Heidelberg (2003)
30. Tsochantaris, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: *ICML* (2004)
31. Wallace, C.S., Dowe, D.L.: Refinements of MDL and MML coding. *Comput. J.* 42(4), 330–337 (1999)
32. Watkins, C., Dayan, P.: Q-learning. *Machine Learning* 8, 279–292 (1992)

Reducing Examples in Relational Learning with Bounded-Treewidth Hypotheses

Ondřej Kuželka, Andrea Szabóová, and Filip Železný

Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, 16627 Prague, Czech Republic
{kuzelon2,szaboand,zelezny}@fel.cvut.cz

Abstract. Feature selection methods often improve the performance of attribute-value learning. We explore whether also in relational learning, examples in the form of clauses can be reduced in size to speed up learning *without affecting the learned hypothesis*. To this end, we introduce the notion of safe reduction: a safely reduced example cannot be distinguished from the original example *under the given hypothesis language bias*. Next, we consider the particular, rather permissive bias of bounded treewidth clauses. We show that under this hypothesis bias, examples of arbitrary treewidth can be reduced efficiently. The bounded treewidth bias can be replaced by other assumptions such as acyclicity with similar benefits. We evaluate our approach on four data sets with the popular system Aleph and the state-of-the-art relational learner nFOIL. On all four data sets we make learning faster for nFOIL, achieving an order-of-magnitude speed up on one of the data sets, and more accurate for Aleph.

1 Introduction

Reducing the complexity of input data is often beneficial for learning. In attribute-value learning, a wide range of *feature selection* methods is available [1]. These methods try to select a strict subset of the original example features (attributes) while maintaining or even improving the performance of the model learned from it with respect to that learned from the original feature set. For binary classification tasks with Boolean features, the REDUCE [2] algorithm has been proposed that removes so called *irrelevant* features. For any model learned with the original feature set, a model with same or better fit on the learning examples may be expressed without the irrelevant features. In the later work [3], the REFER algorithm extended REDUCE to the multiple-class learning setting.

In inductive logic programming—an important framework for relational learning [4]—examples are not expressed as tuples of feature values but rather take the form of logical constructs such as first-order clauses. Feature-selection methods are thus not applicable to simplify such learning examples. Here we are interested to see whether also first-order clausal examples can somehow be reduced while guaranteeing that the set of logical formulas which can be induced from such reductions would not be affected.

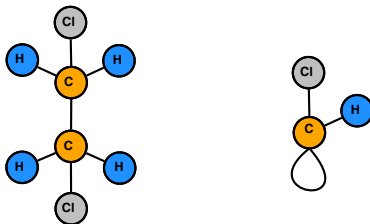


Fig. 1. A learning example and its reduction

An obvious approach would be to look for θ -reductions [5] of the input clauses. A θ -reduction of a clause is a smaller, but subsumption-equivalent (and thus also logically equivalent) clause. We have explored an approach based on θ -subsumption before in [6], achieving learning speed-up factors up to 2.63. However, the main problem of θ -reduction is that finding it is an NP-hard problem, rendering the approach practically unfeasible in domains with large examples such as those describing protein structures [7].

Here we follow the key idea that the complexity curse can be avoided by sacrificing part of the generality of θ -reduction. In particular, we will look for reductions which may not be equivalent to the original example in the logical sense, but which are equivalent *given the language bias* of the learning algorithm. In other words, if the learning algorithm is not able to produce a hypothesis covering the original example but not covering its reduction (or vice versa), the latter two may be deemed equivalent. For instance, consider the clausal example $\leftarrow \text{atom}(\mathbf{a1}) \wedge \text{carbon}(\mathbf{a1}) \wedge \text{bond}(\mathbf{a1}, \mathbf{a2}) \wedge \dots$, whose entire structure is shown in the left of Figure 1. Assume that all terms in hypotheses are variables and hypotheses must have *treewidth* at most 1 or be *acyclic*. Then the learning example is equivalent to the simpler one shown in the right of the figure.

Our first main contribution is a formal framework for example reduction based on the given language bias for hypotheses. In this framework, we prove two propositions which can be used for showing that certain procedures which transform learning examples always produce reductions equivalent to the original examples under the given bias.

Our second main contribution is the application of the above framework to the specific bias of *bounded treewidth* clauses. We show that in this case, interestingly, learning examples can be reduced in polynomial time, and moreover, that, in some cases, they can be reduced even more than they would be using the NP-hard θ -reduction.

Intuitively, a clause viewed as a graph (not necessarily a tree) has a small treewidth if it can be recursively decomposed into small subgraphs that have small overlap [8]. As the previous paragraph indicates, the benefits gained from the bounded treewidth assumption are significant. Notably though, the price we pay for them is not too high in that the bias would be over-restrictive. First remind that, as a hypothesis bias, it only constrains the learned clauses, and not the learning examples. Second, low treewidth is in fact characteristic of clauses induced in typical ILP experiments. In [9] we observed that all clauses learned

by the ILP system Progol in all the conducted experiments had treewidth 1 although this had not been stipulated by the language bias. Similarly, in experiments in another study [10], all clauses learned by the systems nFOIL and kFOIL were of treewidth 1 after the removal of the variable formally identifying the learning example. These observations become plausible when viewing the exemplary chemical-domain clauses shown in the leftmost column of Table 1, which have the respective treewidths 1 and 2.

A salient feature of our approach is its general application scope. Indeed, example reduction can take place independently of the type of ILP learner employed subsequently. While we evaluate the approach in the standard ILP setting of learning from entailment, it is also relevant to propositionalization [11], which aims at the construction of feature-based descriptions of relational examples. Interestingly, propositionalization can simultaneously benefit from both the relational example reduction step employed before the construction of features, and any feature selection algorithm applied subsequently on the constructed feature set. In this sense, our approach is complementary to standard feature selection methods.

The rest of the paper is structured as follows. In the next section we review the preliminaries for the study, namely θ -subsumption and reduction, their correspondence to the constraint satisfaction problem model, and the concepts of tree decomposition and treewidth. Section 3 presents the framework for safe reduction of relational examples under a given hypothesis language bias. Section 4 instantiates the framework to the language bias of bounded-treewidth clauses and shows that reduction under this bias can be conducted effectively. We experimentally evaluate our method in Section 5 and conclude in Section 6.

2 Preliminaries: Logic, Constraint Satisfaction, Treewidth

A first-order-logic clause is a universally quantified disjunction of first-order-logic literals. For convenience, we do not write the universal quantifiers explicitly. We treat clauses as disjunctions of literals and as sets of literals interchangeably. We will sometimes use a slightly abused notation $a(x, y) \subseteq a(w, x) \vee a(x, y)$ to denote that a set of literals of one clause is a subset of literals of another clause. The set of variables in a clause A is written as $vars(A)$ and the set of all terms by $terms(A)$. Terms can be variables or constants. A substitution θ is a mapping from variables of a clause A to terms of a clause B . The next definition introduces the concepts of θ -subsumption and θ -equivalence [5].

Definition 1 (θ -subsumption). *Let A and B be clauses. The clause A θ -subsumes B (denoted by $A \preceq_{\theta} B$), if and only if there is a substitution θ such that $A\theta \subseteq B$. If $A \preceq_{\theta} B$ and $B \preceq_{\theta} A$, we call A and B θ -equivalent (written $A \approx_{\theta} B$).*

The notion of θ -subsumption was introduced by [5] as an incomplete approximation of implication. Let A and B be clauses. If $A \preceq_{\theta} B$ then $A \models B$ but the other direction of the implication does not hold in general. However, it does hold for non-self-resolving function-free clauses.

Example 1. Let us have clauses $A = a(X, Y) \vee a(Y, Z)$ and $B = a(c, d) \vee a(d, e) \vee a(f, d)$. Then $A \preceq_\theta B$ because, for $\theta = \{X/c, Y/d, Z/e\}$, we have $A\theta = a(c, d) \vee a(d, e) \subseteq B$.

Definition 2 (θ -Reduction). *Let A be a clause. If there is another clause R such that $A \approx_\theta R$ and $|R| < |A|$ then A is said to be θ -reducible. A minimal such R is called θ -reduction of A .*

Constraint satisfaction [12] with finite domains represents a class of problems closely related to the θ -subsumption problems and to relational-structure homomorphisms. In fact, as shown by [13], these problems are almost identical although the terminology differs.

Definition 3 (Constraint Satisfaction Problem). *A constraint satisfaction problem is a triple $(\mathcal{V}, \mathcal{D}, \mathcal{C})$, where \mathcal{V} is a set of variables, $\mathcal{D} = \{D_1, \dots, D_{|\mathcal{V}|}\}$ is a set of domains of values (for each variable $v \in \mathcal{V}$), and $\mathcal{C} = \{C_1, \dots, C_{|\mathcal{C}|}\}$ is a set of constraints. Every constraint is a pair (s, R) , where s (scope) is an n -tuple of variables and R is an n -ary relation. An evaluation of variables θ satisfies a constraint $C_i = (s_i, R_i)$ if $s_i\theta \in R_i$. A solution is an evaluation that satisfies all constraints.*

The CSP representation of the problem of deciding $A \preceq_\theta B$ has the following form [14]. There is one CSP variable X_v for every variable $v \in \text{vars}(A)$. The domain of each of these CSP variables contains all terms from $\text{terms}(B)$. The set of constraints contains one k -ary constraint $C_l = (s_l, R_l)$ for each literal $l = \text{pred}_i(t_1, \dots, t_k) \in A$. We denote by $I_{var} = (i_1, \dots, i_m) \subseteq (1, \dots, k)$ the indexes of variables in arguments of l (the other arguments might contain constants). The scope s_l of the constraint C_l is $(X_{t_{i_1}}, \dots, X_{t_{i_m}})$ (i.e. the scope contains all CSP variables corresponding to variables in the arguments of literal l). The relation R_l of the constraint C_l is then constructed in three steps. First, a set L_l is created which contains all literals $l' \in B$ such that $l \preceq_\theta l'$ (note that checking θ -subsumption of two literals is a trivial linear-time operation). Then a relation R'_l is constructed from the arguments of these literals such that it contains a tuple (t'_1, \dots, t'_k) if and only if $l' = \text{pred}(t'_1, \dots, t'_k) \in L_l$. Finally, the relation R_l of the constraint C_l is then the projection of R'_l on indexes I_{var} (only the elements of tuples which correspond to variables in l are retained).

Next, we exemplify this transformation process.

Example 2 (Converting θ -subsumption to CSP). Let us have clauses A and B as follows

$$A = \text{hasCar}(C) \vee \text{hasLoad}(C, L) \vee \text{shape}(L, \text{box})$$

$$B = \text{hasCar}(c) \vee \text{hasLoad}(c, l_1) \vee \text{hasLoad}(c, l_2) \vee \text{shape}(l_2, \text{box}).$$

We now show how we can convert the problem of deciding $A \preceq_\theta B$ to a CSP problem. Let $\mathcal{V} = \{C, L\}$ be a set of CSP-variables and let $\mathcal{D} = \{D_C, D_L\}$ be a set of domains of variables from \mathcal{V} such that $D_C = D_L = \{c, l_1, l_2\}$. Further, let $\mathcal{C} = \{C_{\text{hasCar}(C)}, C_{\text{hasLoad}(C, L)}, C_{\text{shape}(L, \text{box})}\}$ be a set of constraints with scopes

(C) , (C, L) and (L) and with relations $\{(c)\}$, $\{(c, l_1), (c, l_2)\}$ and $\{(l_2)\}$, respectively. Then the constraint satisfaction problem given by \mathcal{V} , \mathcal{D} and \mathcal{C} represents the problem of deciding $A \preceq_\theta B$ as it admits a solution if and only if $A \preceq_\theta B$ holds.

The Gaifman (or primal) graph of a clause A is the graph with one vertex for each variable $v \in \text{vars}(A)$ and an edge for every pair of variables $u, v \in \text{vars}(A)$, $u \neq v$ such that u and v appear in a literal $l \in A$. Similarly, we define Gaifman graphs for CSPs. The Gaifman graph of a CSP problem $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ is the graph with one vertex for each variable $v \in \mathcal{V}$ and an edge for every pair of variables which appear in a scope of some constraint $c \in \mathcal{C}$. Gaifman graphs can be used to define treewidth of clauses or CSPs.

Definition 4 (Tree decomposition, Treewidth). *A tree decomposition of a graph $G = (V, E)$ is a labeled tree T such that*

- *Every node of T is labeled by a non-empty subset of V .*
- *For every edge $(v, w) \in E$, there is a node of T with label containing v, w .*
- *For every $v \in V$, the set of nodes of T with labels containing v is a connected subgraph of T .*

The width of a tree decomposition T is the maximum cardinality of a label in T minus 1. The treewidth of a graph G is the smallest number k such that G has a tree decomposition of width k . The treewidth of a clause is equal to the treewidth of its Gaifman graph. Likewise, the treewidth of a CSP is equal to the treewidth of its Gaifman graph.


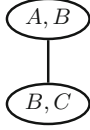
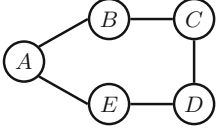
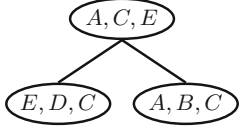
An illustration of Gaifman graphs of two exemplar clauses and their tree-decompositions is shown in Table 1. Note that tree decomposition is not unique.

It is easy to check that if a clause A has treewidth bounded by k then also the CSP representation of the problem of deciding $A \preceq_\theta B$ has treewidth bounded by k for any clause B . Constraint satisfaction problems with treewidth bounded by k can be solved in polynomial time by the k -consistency algorithm¹ [16]. If the k -consistency algorithm returns *false* for a CSP problem \mathcal{P} then \mathcal{P} is guaranteed to have no solutions. If it returns *true* then the problem may or may not have some solutions. Finally, if the k -consistency algorithm returns *true* and \mathcal{P} has treewidth bounded by k then \mathcal{P} is guaranteed to have a solution. It is known that due to the equivalence of CSPs and θ -subsumption, the problem of deciding θ -subsumption $A \preceq_\theta B$ can be solved in polynomial time when clause A has bounded treewidth.

Proposition 1. *We say that clause A is k -consistent w.r.t. clause B (denoted by $A \triangleleft_k B$) if and only if the k -consistency algorithm executed on the CSP representation of the problem of deciding $A \preceq_\theta B$ returns *true*. If A has treewidth at most k and $A \triangleleft_k B$ then $A \preceq_\theta B$.*

¹ In this paper we follow the conventions of [15]. In other works, e.g. [16], what we call k -consistency is known as *strong $k + 1$ -consistency*.

Table 1. An illustration of Gaifman graphs and tree-decompositions of clauses

Clause	Gaifman graph	Tree decomposition
$\leftarrow \text{atm}(A, h) \wedge$ $\text{bond}(A, B, 1) \wedge \text{atm}(B, c) \wedge$ $\text{bond}(B, C, 2) \wedge \text{atm}(C, o)$		
$\leftarrow \text{bond}(A, B, 1) \wedge$ $\text{bond}(B, C, 1) \wedge \text{bond}(C, D, 1) \wedge$ $\text{bond}(D, E, 1) \wedge \text{bond}(E, A, 1)$		

Proof. Follows directly from the solubility of CSPs with bounded treewidth by the k -consistency algorithm [15] and from the equivalence of CSPs and θ -subsumption shown earlier in this section.

3 Safe Reduction of Learning Examples

The learning task that we consider in this paper is fairly standard. We are given labelled learning examples encoded as first-order-logic clauses and we would like to find a classifier predicting the class labels of examples as precisely as possible. This task could be solved by numerous relational-learning systems. We aim at finding a reduction procedure that would allow us to reduce the number of literals in the examples while guaranteeing that the coverage of any hypothesis from a pre-fixed hypothesis language \mathcal{L} would not be changed.

There are several settings for logic-based relational learning. We will work within the *learning from entailment setting* [17].

Definition 5 (Covering under Learning from Entailment). *Let \mathcal{H} be a clausal theory and e be a clause. Then we say that \mathcal{H} covers e under entailment if and only if $\mathcal{H} \models e$.*

The basic learning task is to find a clausal theory \mathcal{H} that covers all positive examples and no negative examples and contains as few clauses as possible.

Definition 6 (Safe Equivalence and Safe Reduction under Entailment). *Let e and \hat{e} be two clauses and let \mathcal{L} be a language specifying all possible hypotheses. Then \hat{e} is said to be safely equivalent to e if and only if $\forall \mathcal{H} \in \mathcal{L} : (\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \hat{e})$. If e and \hat{e} are safely equivalent and $|\hat{e}| < |e|$ then \hat{e} is called safe reduction of e .*

Clearly, if we have a hypothesis $\mathcal{H} \in \mathcal{L}$ which splits the examples to two sets X and Y then this hypothesis \mathcal{H} will also split the respective set of safely reduced examples to the sets \widehat{X}, \widehat{Y} containing the safely reduced examples from the sets X and Y , respectively. Also, when predicting classes of test-set examples, any deterministic classifier that bases its decisions on the queries using the covering relation \models will return the same classification even if we replace some of the examples by their safe reductions. The same is also true for propositionalization approaches that use the \models relation to construct boolean vectors which are then processed by attribute-value-learners.

In this paper, we focus on hypothesis languages in the form of *non-resolving* clausal theories. Recall that we do not put any restrictions on the learning examples. The only restrictions are those put on hypotheses. A *non-resolving* clausal theory is a set of clauses such that no predicate symbol which appears in the head of a clause appears also in the body of any clause. The main reason why we start with non-resolving clausal theories is that logical entailment $\mathcal{H} \models A$, for a non-resolving clausal theory \mathcal{H} and a clause A , can be checked using θ -subsumption. If there is a clause $H \in \mathcal{H}$ such that $H \preceq_{\theta} A$ then $\mathcal{H} \models A$, otherwise $\mathcal{H} \not\models A$.

We start by defining x -subsumption and x -equivalence which are weaker versions of θ -subsumption and θ -equivalence. The notions of x -subsumption and x -equivalence will be central tools used in this section.

Definition 7 (x -subsumption, x -equivalence). *Let X be a possibly infinite set of clauses. Let A, B be clauses not necessarily from X . We say that A x -subsumes B w.r.t. X (denoted by $A \preceq_X B$) if and only if $(C \preceq_{\theta} A) \Rightarrow (C \preceq_{\theta} B)$ for every clause $C \in X$. If $A \preceq_X B$ and $B \preceq_X A$ then A and B are called x -equivalent w.r.t. X (denoted by $A \approx_X B$). For a given set X , the relation \preceq_X is called x -subsumption on X and the relation \approx_X is called x -equivalence on X .*

For example, the set X can consist of clauses having treewidth bounded by k or having hypertreewidth [18] bounded by l or having at most m variables etc. When it is clear from the context, we omit the phrase *w.r.t. X* from A x -subsumes B w.r.t. X . The x -equivalence w.r.t. the set X is closely related to safe equivalence w.r.t. a set $\mathcal{L} \subseteq 2^X$ containing only non-resolving clausal theories composed of clauses from X . If two learning examples are x -equivalent w.r.t. the set of clauses X then they are also safely equivalent w.r.t. the set of clausal theories \mathcal{L} .

Example 3. Let us have the following two clauses: $C = e(A, B) \vee e(B, C) \vee e(C, A)$ and $D = e(A, B) \vee e(B, C) \vee e(C, D) \vee e(D, A)$. For these clauses, it holds $C \preceq_X D$ and $D \preceq_X C$ w.r.t. the set of clauses with treewidth at most 1. On the other hand, $C \not\preceq_X D$, $D \not\preceq_X C$ for sets of clauses with treewidth at most k where $k > 1$. This is because the treewidth of C and D is 2.

The next proposition states basic properties of x -subsumption and x -equivalence.

Proposition 2. *Let X be a set of clauses. Then x -subsumption w.r.t. X is a transitive and reflexive relation on clauses and x -equivalence w.r.t. X is an equivalence relation on clauses.*

Proof. These properties of x -subsumption and x -equivalence can be shown very easily.

1. Transitivity of x -subsumption: Let $A \preceq_X B$ and $B \preceq_X C$. We need to show that then necessarily also $A \preceq_X C$, i.e. that for any clause $D \in X$ such that $D \preceq_\theta A$ it also holds $D \preceq_\theta C$. This is straightforward because if $D \preceq_\theta A$ then $D \preceq_\theta B$ (from $A \preceq_X B$) and also $D \preceq_\theta C$ (from $B \preceq_X C$).
2. Reflexivity of x -subsumption: obvious.
3. x -equivalence is an equivalence relation: Reflexivity and transitivity of x -equivalence follow from reflexivity and transitivity of x -subsumption. It remains to show that x -equivalence is also symmetric but that follows immediately from $(A \approx_X B) \Leftrightarrow (A \preceq_X B \wedge B \preceq_X A)$.

Definition 7 provides no efficient way to decide x -subsumption between two clauses as it demands θ -subsumption of an infinite number of clauses to be tested in some cases. The next proposition provides a necessary condition for x -subsumption. It will be the basic tool that we exploit in this section to develop methods for safely reducing learning examples.

Proposition 3. *Let X be a set of clauses. If \preceq_X is x -subsumption on X and \triangleleft_X is a relation such that:*

1. *If $A \triangleleft_X B$ and $C \subseteq A$ then $C \triangleleft_X B$.*
2. *If $A \in X$, ϑ is a substitution and $A\vartheta \triangleleft_x B$ then $A \preceq_X B$.*

Then $(A \triangleleft_X B) \Rightarrow (A \preceq_X B)$ for any two clauses A, B (not necessarily from X).

Proof. We need to show that if $A \triangleleft_x B$ then $(C \preceq_\theta A) \Rightarrow (C \preceq_\theta B)$ for all clauses $C \in X$. First, if $A \triangleleft_x B$ and $C \not\preceq_\theta A$ then the proposition holds trivially. Second, $C \preceq_\theta A$ means that there is a substitution ϑ such that $C\vartheta \subseteq A$. This implies $C\vartheta \triangleleft_X B$ using the condition 1. Now, we can use the second condition which gives us $C \preceq_X B$ (note that $C \in X$ and $C\vartheta \triangleleft_X B$). Finally, we get $C \preceq_\theta B$ using Definition 7 because $C \in X$.

Proposition 3 can be used to check if two learning examples e and \hat{e} are equivalent w.r.t. hypotheses from a fixed hypothesis language. It can be therefore used to search for safe reductions of learning examples. This is formalized in the next proposition. Note that this proposition does not say that e and \hat{e} are equivalent. It merely says that they are equivalent when being used as learning examples in the *learning from entailment* setting with hypotheses drawn from a fixed set.

Proposition 4. *Let \mathcal{L} be a hypothesis language containing only non-resolving clausal theories composed of clauses from a set X and let \triangleleft_X be a relation satisfying conditions 1 and 2 from Proposition 3 on the set X . If e and \hat{e} are learning examples (not necessarily from X), $e \triangleleft_X \hat{e}$ and $\hat{e} \triangleleft_X e$ then for any $\mathcal{H} \in \mathcal{L}$ it holds $(\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \hat{e})$. Moreover, if $|\hat{e}| < |e|$ then \hat{e} is a safe reduction of e under entailment.*

Proof. First, $e \triangleleft_X \hat{e}$ and $\hat{e} \triangleleft_X e$ imply $e \approx_X \hat{e}$ (where \approx_X denotes x -equivalence on the set X). Then for any non-resolving clausal theory $\mathcal{H} \in \mathcal{L}$ we have $(\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \hat{e})$ because for any clause $A \in X$ we have $(A \preceq_\theta e) \Leftrightarrow (A \preceq_\theta \hat{e})$ (from $e \approx_X \hat{e}$). This together with $|\hat{e}| < |e|$ means that \hat{e} is a safe reduction of e under entailment w.r.t. hypothesis language \mathcal{L} .

We will use Propositions 3 and 4 for showing that certain procedures which transform learning examples always produce safe reductions of these examples. Specifically, we will use them to show that k -consistency algorithm can be used for computing safe reductions of learning examples w.r.t. hypothesis sets composed of clauses with bounded treewidth.

We start with two simpler transformation methods for which Propositions 3 and 4 are not actually needed. For the first transformation method, we assume to have a fixed hypothesis language $\mathcal{L}_\mathcal{U}$ consisting of non-resolving clausal theories which contain only constants from a given set \mathcal{U} . The transformation then gets a clause A on its input and produces a new clause \tilde{A} by *variabilizing* constants in A which are not contained in \mathcal{U} . It is easy to check that for any such A and \tilde{A} it must hold $A \approx_X \tilde{A}$ w.r.t. the set of clauses containing only constants from \mathcal{U} . Therefore A and \tilde{A} are safely equivalent w.r.t. \mathcal{L} . We can think of the constants not used in a hypothesis language \mathcal{L} as identifiers of objects whose exact identity is not interesting for us. Such constants can appear e.g. when we describe molecules and we want to give names to atoms in the molecules with no actual meaning.

Another simple transformation which produces safely equivalent clauses is based on θ -reduction. In this case the set of clauses X can be arbitrary. The transformation gets a clause A on its input and returns its θ -reduction. The x -equivalence of the clause A and its θ -reduction follows from the fact that θ -subsumption is an x -subsumption w.r.t. the set of all clauses.

Importantly, transformations which produce x -equivalent clauses w.r.t. a set X can be chained due to transitivity of x -subsumption. So, for example, if we have a hypothesis language $\mathcal{L}_\mathcal{U}$ consisting of non-resolving clausal theories which contain only constants from a pre-fixed set \mathcal{U} and we want to safely reduce a clause A then we can first variabilize it and then reduce it using θ -reduction.

Example 4. Let us have an example

$$e = \text{edge}(a, b, 1) \vee \text{edge}(b, a, 2) \vee \text{edge}(b, c, 2) \vee \text{edge}(c, d, 1) \vee \text{edge}(d, a, 2)$$

and a hypothesis language \mathcal{L} containing arbitrary non-resolving clausal theories with the set of allowed constants $\mathcal{U} = \{1, 2\}$. We variabilize e and obtain clause

$$\tilde{e} = \text{edge}(A, B, 1) \vee \text{edge}(B, A, 2) \vee \text{edge}(B, C, 2) \vee \text{edge}(C, D, 1) \vee \text{edge}(D, A, 2).$$

Now, e and \tilde{e} are safely equivalent w.r.t. to hypotheses from \mathcal{L} . Next, we obtain a safe reduction of e by computing θ -reduction of \tilde{e} which is $\hat{e} = \text{edge}(A, B, 1) \vee \text{edge}(B, A, 2)$.

4 Reduction under the Bounded Treewidth Assumption

Next, we describe a transformation method which assumes the hypothesis languages to consist only of clauses with bounded treewidth. Unlike the exponential-time method based on θ -reduction, this method runs in time polynomial in the size of the reduced clause (though, with a multiplicative factor exponential in the fixed maximum treewidth of allowed hypotheses). Interestingly, it does not need any restrictions (e.g. bounded treewidth) on the learning examples which are reduced. The reduction method is based on x -subsumption on the set X_k of clauses with treewidth at most k . We start by showing that x -subsumption w.r.t. X_k can be checked using the k -consistency algorithm. We do this by showing that the k -consistency relation \triangleleft_k on clauses satisfies the conditions from Proposition 3.

Proposition 5. *Let X_k be a set containing only clauses with treewidth at most k . For any two clauses A, B , if $A \triangleleft_k B$ (i.e. if A is k -consistent w.r.t. B) then $A \preceq_X B$ w.r.t. the set X_k .*

Proof. We will show that the conditions of Proposition 3 are satisfied by \triangleleft_k from which the validity of the proposition will follow. First: If $C \subseteq A$ and $A \triangleleft_k B$ then $C \triangleleft_k B$, which is obviously true. Second: If C is a clause with treewidth bounded by k and $C\theta \triangleleft_k D$ then $C \preceq_\theta D$. Checking $C\theta \triangleleft_k D$ is equivalent to checking k -consistency of the original CSP representation of $C \preceq_\theta D$ problem where we added additional constraints to enforce consistency with the substitution θ . If this *restricted* problem is still k -consistent then also for the original problem it must have held $C \triangleleft_k D$ and consequently $C \preceq_\theta D$ because C has treewidth at most k (using Proposition 1).

We leave the question open whether x -subsumption w.r.t. the set of clauses with treewidth at most k also implies k -consistency.

The safe reduction method based on k -consistency works as follows. We suppose that there is a set \mathcal{U} of constants which are allowed in the hypothesis language \mathcal{L}_k and that the hypotheses in \mathcal{L}_k consist only of clauses with treewidth at most k . The method gets a clause A and variabilizes all constants not contained in \mathcal{U} . The result is a clause \tilde{A} which is also safely equivalent to A w.r.t. the hypothesis language \mathcal{L}_k . This clause is then reduced by so-called *literal-elimination algorithm* which is based on the k -consistency algorithm, always produces a clause \hat{A} which is safely equivalent to A w.r.t. \mathcal{L}_k as Proposition 6 shows. Moreover, it runs in time polynomial in the size of the reduced clause (though, with a multiplicative factor exponential in the fixed maximum treewidth of allowed hypotheses).

Literal-elimination algorithm:

1. Given a clause A for which the x -reduction should be computed.
2. Set $A' := A$, $CheckedLiterals := \{\}$.
3. Select a literal L from $A' \setminus CheckedLiterals$. If there is no such literal, return A' and finish.

4. If $A \triangleleft_k A' \setminus \{L\}$ then set $A' := A' \setminus \{L\}$, else add L to *CheckedLiterals*.
5. Go to step 3.

Proposition 6. *Let \mathcal{L}_k be a set of non-resolving hypotheses containing only clauses with treewidth at most k . Let C be a clause and \widehat{C}_θ be the maximal θ -reduction of a subset of the literals in C . We can find a clause \widehat{C}_k such that $C \approx_X \widehat{C}_k$ w.r.t. to \mathcal{L}_k and $|\widehat{C}_k| \leq |\widehat{C}_\theta|$ in time $\mathcal{O}(|C|^{2k+3})$ by the "literal-elimination algorithm".*

Proof. First, it follows from transitivity of k -equivalence that $\widehat{C}_k \approx_k C$. What remains to be shown is that the resulting clause \widehat{C}_k will not be θ -reducible. Let us assume, for contradiction, that $|\widehat{C}_k|$ is θ -reducible. When \widehat{C}_k is θ -reducible, there must be a literal $l \in \widehat{C}_k$ such that $\widehat{C}_k \preceq_\theta \widehat{C}_k \setminus \{l\}$. θ -subsumption implies k -consistency (for clauses of arbitrary treewidth) therefore it also holds $\widehat{C}_k \triangleleft_k \widehat{C}_k \setminus \{l\}$. However, then l should have been removed by the literal-elimination algorithm which is a contradiction with \widehat{C}_k being output of it. As for the running time of the algorithm, k -consistency can be checked in time $\mathcal{O}(|C|^{2k+2})$ [15] and it is invoked exactly $|C|$ times by the above procedure which gives us the runtime $\mathcal{O}(|C|^{2k+3})$.

We can find even smaller safely equivalent clauses w.r.t. \mathcal{L}_k for a clause C by a *literal-substitution algorithm* with just a slightly higher runtime $\mathcal{O}(|C|^{2k+4})$. This algorithm first runs the *literal-elimination algorithm* and then tries to further reduce its output C' as follows: For each pair of literals $l, l' \in C'$ it constructs a substitution $\theta : \text{vars}(l) \rightarrow \text{vars}(l')$ and checks if $C'\theta \triangleleft_k C'$ and if so, it sets $C' \leftarrow C'\theta$. It is easy to check that it always holds $C \approx_X C'$ w.r.t. the set of clauses with treewidth at most k . The algorithm runs in time $\mathcal{O}(|C|^{2k+4})$ as it performs $\mathcal{O}(|C|^2)$ k -consistency checks.

The clauses with bounded treewidth are not the only ones for which efficient safe reduction can be derived. For example, it is possible to derive a completely analogical safe reduction w.r.t. acyclic clauses, which can have arbitrary high treewidth but despite that admit a polynomial-time θ -subsumption checking algorithm. The only difference would be the use of generalized arc-consistency algorithm [16] instead of the k -consistency test.

5 Experimental Evaluation of Safe Reduction

We experimentally evaluate usefulness of the *safe reduction of learning examples* with real-world datasets and two relational learning systems – the popular system Aleph and the state-of-the-art system nFOIL [19]. We implemented *literal-elimination* and *literal-substitution* algorithms for treewidth 1, i.e. for tree-like clausal theories. We used the efficient algorithm AC-3 [20] for checking 1-consistency². We forced nFOIL and Aleph to construct only clauses with

² Note again the terminology used in this paper following [15]. In CSP-literature, it is often common to call 2-consistency what we call 1-consistency.

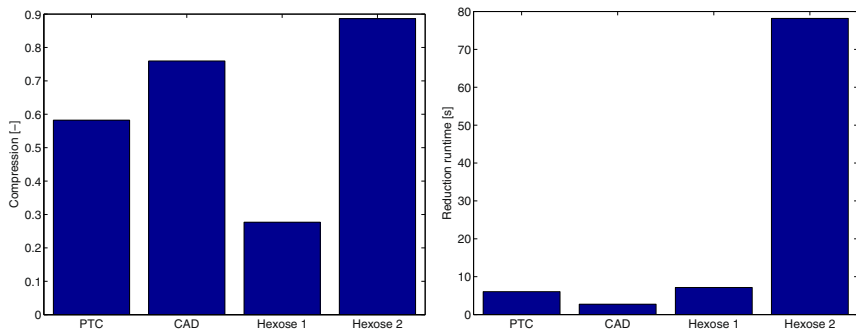


Fig. 2. **Left:** Compression rates achieved by *literal-substitution algorithm* on four datasets (for treewidth 1). **Right:** Time for computing reductions of learning examples on four datasets (for treewidth 1).

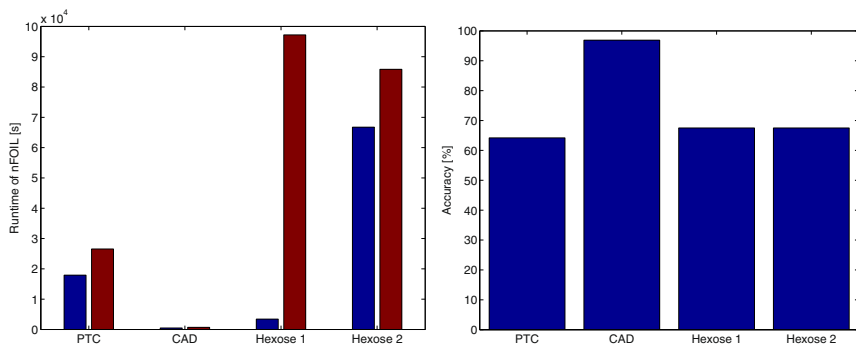


Fig. 3. **Left:** Runtime of nFOIL on reduced (blue) and non-reduced (red) datasets. **Right:** Predictive accuracies of nFOIL on four datasets estimated by 10-fold cross-validation.

treewidth 1 using their *mode declaration* mechanisms. We used three datasets in the experiments: *predictive toxicology challenge* [21], *CAD* [22] and *hexose-binding proteins* [7]. The PTC dataset contains descriptions of 344 molecules classified according to their toxicity for male rats. The molecules are described using only *atom* and *bond* information. The CAD dataset contains descriptions of 96 class-labelled product-structure designs. Finally, the hexose-binding dataset contains 80 hexose-binding and 80 non-hexose-binding protein domains. Following [7] we represent the protein domains by atom-types and atom-names (each atom in an amino acid has a unique name) and pair-wise distances between the atoms which are closer to each other than some threshold value. We performed two experiments with the last mentioned dataset for cut-off set to 1 Angstrom and 2 Angstroms.

We applied the *literal-elimination* algorithm followed by *literal-substitution* algorithm on the three datasets. The compression rates (i.e. ratios of number

of literals in the reduced learning examples divided by the number of literals in the original non-reduced examples) are shown in the left panel of Figure 2. The right panel of Figure 2 then shows the time needed to run the reduction algorithms on the respective datasets. We note that these times are generally negligible compared to runtimes of nFOIL and with the exception of Hexose ver. 2 also to runtimes of Aleph.

5.1 Experiments with nFOIL

We used nFOIL to learn predictive models and evaluated them using 10-fold cross-validation. For all experiments with the exception of the hexose-binding dataset with cut-off value 2 Angstroms, where we used beam-size 50, we used beam-size 100. From one point of view, this is much higher than the beam-sizes used by [19], but on the other hand, we have the experience that this allows nFOIL to find theories which involve longer clauses and at the same time have higher predictive accuracies. The runtimes of nFOIL operating on reduced and non-reduced data are shown in the left panel of Figure 3. It can be seen that the reduction was beneficial in all cases but that the most significant speed-up of more than an order of magnitude was achieved on Hexose data. This could be attributed to the fact that nFOIL constructed long clauses on this dataset and the covering test used by it had not probably been optimized. So, in principle, nFOIL could be made faster by optimizing the efficiency of its covering test. The main point, however, is that we can speed-up the learning process for almost any relational learning algorithm merely by preprocessing its input. The right panel of Figure 3 shows nFOIL’s predictive accuracies (estimated by 10-fold cross-validation). The accuracies were not affected by the reductions. The reason is that (unlike Aleph) nFOIL exploits learning examples only through the entailment queries.

5.2 Experiments with Aleph

We performed another set of experiments using the relational learning system Aleph. Aleph restricts its search space by bottom-clauses. After constructing a bottom-clause it searches for hypotheses by enumerating subsets of literals of the bottom-clause. When we reduce learning examples, which also means reduction of bottom-clauses, we are effectively reducing the size of Aleph’s search space. This means that Aleph can construct longer clauses earlier than if it used non-reduced examples. On the other hand, this also implies that, with the same settings, Aleph may run longer on reduced data than on non-reduced data. That is because computing coverage of longer hypotheses is more time-consuming. Theories involving longer clauses may often lead to more accurate predictions. For these reasons, we measured not only runtime and accuracy, but also the average number of learnt rules and the average number of literals in these rules on reduced and non-reduced data.

We ran Aleph on reduced and non-reduced versions of the datasets and evaluated it using 10-fold cross-validation. We used the literal elimination algorithm

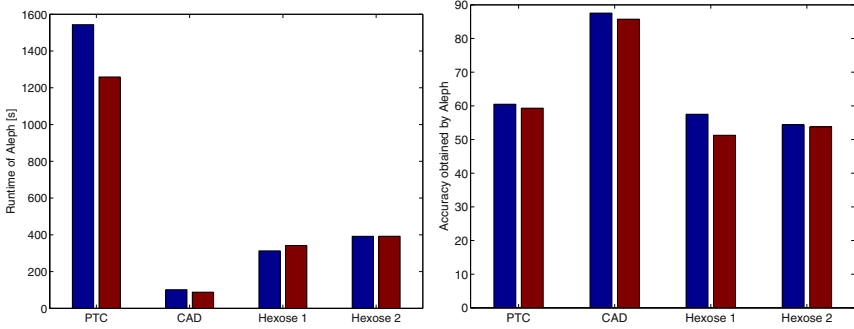


Fig. 4. Left: Runtime of Aleph on reduced (blue) and non-reduced (red) datasets. **Right:** Predictive accuracies of Aleph on reduced (blue) and non-reduced (red) datasets estimated by 10-fold cross-validation.

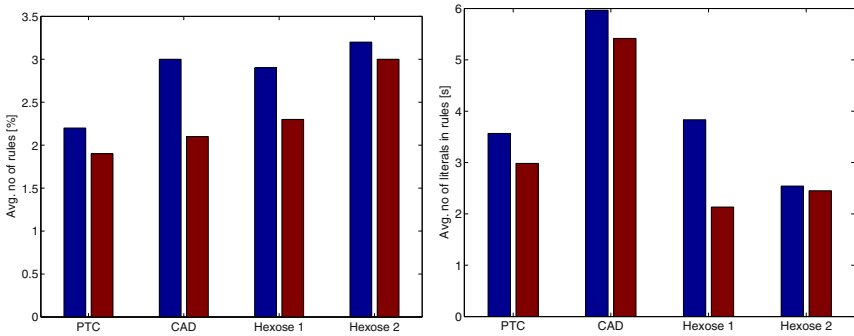


Fig. 5. Left: Average number of rules generated by Aleph on reduced (blue) and non-reduced (red) datasets. **Right:** Average number of literals in rules generated by Aleph on reduced (blue) and non-reduced (red) datasets.

for reducing examples. We set the maximum number of explored *nodes* to 50000, the *noise* parameter to 1% of the number of examples in the respective datasets. The runtime in the performed experiments was higher for reduced versions of datasets PTC and CAD, the same for Hexose 2 and lower for Hexose 1 than for their non-reduced counterparts (see left panel of Figure 4). The accuracies were higher for reduced versions of all four datasets (see right panel of Figure 4). Similarly, the average number of rules, as well as the average number of literals in the rules, was higher for the reduced versions of all four datasets (see Figure 5). These results confirm the expectation that Aleph should be able to construct longer hypotheses on reduced datasets which, in turn, should result in higher predictive accuracies.

6 Conclusions

We have introduced a novel concept called *safe reduction*. We have shown how it can be used to safely reduce learning examples (without affecting learnability) which makes it possible to speed-up many relational learning systems by merely preprocessing their input. The methods that we have introduced run in polynomial time for hypothesis languages composed of clauses with treewidth bounded by a fixed constant. The bounded-treewidth assumption, while arguably appropriate in ILP, can be replaced by other kinds of assumptions such as acyclicity.

Acknowledgements. This work was supported by the Czech Grant Agency through project 103/11/2170 *Transferring ILP techniques to SRL*.

Appendix: The k -Consistency Algorithm

In this section, we briefly describe the k -consistency algorithm. The description is based on the presentation by Atserias et al. [15]. Let us have a CSP $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ where \mathcal{V} is the set of variables, \mathcal{D} is the set of domains of the variables and \mathcal{C} is the set of constraints. A partial solution ϑ is an evaluation of variables from $\mathcal{V}' \subseteq \mathcal{V}$ which is a solution of the sub-problem $\mathcal{P}' = (\mathcal{V}', \mathcal{D}, \mathcal{C})$. If ϑ and φ are partial solutions, we say that φ extends ϑ (denoted by $\vartheta \subseteq \varphi$) if $Supp(\vartheta) \subseteq Supp(\varphi)$ and $V\vartheta = V\varphi$ for all $V \in Supp(\vartheta)$, where $Supp(\vartheta)$ and $Supp(\varphi)$ denote the sets of variables which are affected by the respective evaluations ϑ and φ .

The k -consistency algorithm then works as follows:

1. Given a constraint satisfaction problem $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ and a positive integer k .
2. Let H be the collection of all partial solutions ϑ with $|Supp(\vartheta)| < k + 1$.
3. For every $\vartheta \in H$ with $|Supp(\vartheta)| \leq k$ and every $V \in \mathcal{V}$, if there is no $\varphi \in H$ such that $\vartheta \subseteq \varphi$ and $V \in Supp(\varphi)$, remove ϑ and all its extensions from H .
4. Repeat step 3 until H is unchanged.
5. If H is empty return *false*, else return *true*.

References

1. Liu, H., Motoda, H., Setiono, R., Zhao, Z.: Feature selection: An ever evolving frontier in data mining. *Journal of Machine Learning Research - Proceedings Track 10*, 4–13 (2010)
2. Lavrac, N., Gamberger, D., Jovanoski, V.: A study of relevance for learning in deductive databases. *J. Log. Program.* 40(2-3), 215–249 (1999)
3. Appice, A., Ceci, M., Rawles, S., Flach, P.A.: Redundant feature elimination for multi-class problems. In: *ICML*, vol. 69 (2004)
4. Raedt, L.D.: *Logical and Relational Learning: From ILP to MRDM (Cognitive Technologies)*. Springer-Verlag New York, Inc. (2008)
5. Plotkin, G.D.: A note on inductive generalization. *Machine Intelligence 5*, 153–163 (1970)

6. Kuželka, O., Železný, F.: Seeing the world through homomorphism: An experimental study on reducibility of examples. In: Frasconi, P., Lisi, F.A. (eds.) ILP 2010. LNCS, vol. 6489, pp. 138–145. Springer, Heidelberg (2011)
7. Nassif, H., Al-Ali, H., Khuri, S., Keirouz, W., Page, D.: An inductive logic programming approach to validate hexose binding biochemical knowledge. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 149–165. Springer, Heidelberg (2010)
8. Erickson, J.: CS 598: Computational Topology, course notes, University of Illinois at Urbana-Champaign (2009)
9. Kuželka, O., Železný, F.: Block-wise construction of acyclic relational features with monotone irreducibility and relevancy properties. In: ICML 2009: the 26th Int. Conf. on Machine Learning (2009)
10. Kuželka, O., Železný, F.: Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning* 83, 163–192 (2011)
11. Krogel, M.-A., Rawles, S., Železný, F., Flach, P.A., Lavrač, N., Wrobel, S.: Comparative evaluation of approaches to propositionalization. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 197–214. Springer, Heidelberg (2003)
12. Dechter, R.: *Constraint Processing*. Morgan Kaufmann Publishers (2003)
13. Feder, T., Vardi, M.Y.: The computational structure of monotone monadic smp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.* 28(1), 57–104 (1998)
14. Maloberti, J., Sebag, M.: Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning* 55(2), 137–174 (2004)
15. Atserias, A., Bulatov, A., Dalmau, V.: On the power of k -consistency. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 279–290. Springer, Heidelberg (2007)
16. Rossi, F., van Beek, P., Walsh, T. (eds.): *Handbook of Constraint Programming*. Elsevier (2006)
17. De Raedt, L.: Logical settings for concept-learning. *Artif. Intell.* 95(1), 187–201 (1997)
18. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences* 64(3), 579–627 (2002)
19. Landwehr, N., Kersting, K., Raedt, L.D.: Integrating naïve bayes and FOIL. *Journal of Machine Learning Research* 8, 481–507 (2007)
20. Mackworth, A.: Consistency in networks of relations. *Artificial Intelligence* 8(1), 99–118 (1977)
21. Helma, C., King, R.D., Kramer, S., Srinivasan, A.: The predictive toxicology challenge 2000-2001. *Bioinformatics* 17(1), 107–108 (2001)
22. Žáková, M., Železný, F., Garcia-Sedano, J.A., Masia Tissot, C., Lavrač, N., Křemen, P., Molina, J.: Relational data mining applied to virtual engineering of product designs. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 439–453. Springer, Heidelberg (2007)

Mining Complex Event Patterns in Computer Networks

Dietmar Seipel¹, Philipp Neubeck², Stefan Köhler³, and Martin Atzmueller⁴

¹ University of Würzburg, Department of Computer Science

² Google Germany GmbH, Munich

³ Infosim GmbH & Co. KG, Würzburg

⁴ University of Kassel, Knowledge and Data Engineering Group

Abstract. More and more ubiquitous and mobile computer networks are becoming available, which leads to a massive growth in the amount of traffic and according log messages. Therefore, sophisticated approaches for network management and analysis are necessary for handling and managing networks efficiently.

In this paper, we show how to use temporal data mining in a declarative framework for analysing log files for computer networks. From a sequence of network management protocol messages, we derive temporal association rules, which state frequent dependencies between the occurring events. We also present methods for extendable and modular parsing of text messages and their analysis in log files based on XML.

1 Introduction

With the advent of mobile, dynamic, and more and more ubiquitous devices, computer networks – providing the technical infrastructure, that is, the links between the individual computational nodes – are becoming more widespread at a rapid pace. In such contexts, e.g., in TCP/IP-based environments, network management is a critical issue for ensuring continuous quality of service and stability of the network. With the growing adoption and size of the networks, there is an increasing amount of *alarm events* and *error messages* indicating exceptional situations. Usually, their detection is based on analysing logs and searching for faults or other exceptional events. An appropriate management needs to apply sophisticated approaches both for handling the amount of data and for determining patterns at the appropriate level of abstraction.

In this paper, we consider a declarative framework for analysing events in networks. More specifically, we focus on the analysis of network events using temporal data mining techniques. The main contributions are the integrated analysis and the preprocessing and postprocessing options of the mined patterns. The presented techniques allow for the detection of relations between different events in order to identify complex event patterns. From a log file, we could, e.g., automatically extract the rule “Every *link down* event is preceded by a *lineprotocol down* event in less than 10 seconds”. Thus, complex patterns consisting of sequences of events are identified and can then be applied, e.g., for managing and optimizing the system.

Approaches for the temporal analysis of event sequences and for temporal data mining have become increasingly prominent in recent years. Especially, mining alarm patterns has attracted significant attention, e.g., [10, 5, 14]: Most similarly to our setting,

de Aguiar et al. [14] present an approach for alarm pattern mining. However, in contrast to their approach, we present a method and case study that focuses more on message and sequence analysis. It considers grouping equivalences, resolving word-based, most specific event types, and filtering rules using background knowledge, e.g., already known relations.

Laxman et al. [11] provide a general overview of the field of temporal data mining. They discuss the problem setting and introduce several algorithms. Furthermore, Achar et al. [1] provide a unified view on apriori-based algorithms for frequent episode discovery. Mannila et al. [12, 10] provide classic instantiations of the problem and propose the WINEPI and MINEPI algorithms for mining frequent sequences and episodes; these algorithms are among the core ingredients of our presented approach. However, we have extended these approaches by flexible filtering, mainly concerning overlapping sequences and repetitions, and additional message and sequence analysis. Wu et al. [19] as well as Tatti and Cule [17] consider extensions of episode mining algorithms focusing on complete or strictly closed episodes. As discussed below, this remains as one of the possible future extensions of our work.

For the analysis of event logs, the Simple Logfile Clustering Tool (SLCT) [18], for example, analyses text messages and determines frequent line patterns. The patterns can be reused for defining event types by the analyst. The approach is iterative and requires human interaction; an automatic approach is not mentioned. It is also important to note, that SLCT does not consider temporal relations, but only absolute frequencies of patterns. In contrast, our approach combines sequence and message analysis. Instead of a simple event type, we assign several key/value pairs to each event.

The rest of the paper is structured as follows: Section 2 summarizes steps for pre-processing the log files. Section 3 describes the framework for temporal data mining including some examples for the presented approach. Section 4 presents a more detailed case study, and Section 5 explains the practical relevance of our approach. Finally, Section 6 concludes with a summary and discusses possible future extensions.

2 Log File Processing

A common source of log information are facilities, such as *syslog* under Unix-like operating systems and *Windows Event Logs* under Microsoft Windows. Several log facilities collect events with a text field, which is used in many ways and not further standardised. This section is concerned with analysing this unstructured text field. The other fields, like the timestamp, sender, log facility, and priority of the events will be ignored for now. Below, we give examples from a syslog file, which has 20.000 lines and occupies 6 MB of space; it is an Excel file in CSV format (values separated by “:”) covering the events of about 2 days. A small selection of text messages shows the diversity of the events in the file:

```
07.430: %SYS-5-CONFIG_I:
    Configured from console by mdoess.k5 onvty0 (23.80.40.147)
%CRYPTO-6-AUTOGEN: Generated new 768 bit key pair
%SSH-5-ENABLED: SSH 1.99 has been enabled
%LINEPROTO-5-UPDOWN: Line protocol on Interface
    FastEthernet0/26, changed state to down
```

```

%LINK-3-UPDOWN:
  Interface FastEthernet0/26, changed state to down
14.272: %OSPF-5-ADJCHG:
  Process 1, Nbr 23.80.248.135 on Serial0/2/1
  from FULL to DOWN, Neighbor Down: Interface down or detached
13.522: %IPPHONE-6-REGISTER_NEW:
  ephone-1:SEP00146A62D078 IP:23.80.250.62
  Socket:1 DeviceType:Phone has registered.
13.743: %IPPHONE-6-UNREGISTER_NORMAL:
  ephone-1:SEP00146A62D078 IP:23.80.250.62
  Socket:1 DeviceType:Phone has unregistered normally.
System: SNMP configuration change.
SNMP access control 2 access type. 0x0003
Control Manager: 7035: Y068DPK1\A0681634:
  The control statement "start" was sent successfully
  to the service "Eventlog to Syslog".
Control Manager: 7036: Service "McAfee McShield"
  now is in the status "stopped".
Control Manager: 7036: Service "McAfee McShield"
  now is in the status "executed".

```

In the listing above, the lines have been broken; the continuation lines are indented. The separators “:” either separate fields – as in, e.g., the last two lines – or attribute value pairs – as in, e.g., IP:23.80.250.62. We have developed parsers that can distinguish between these two uses of “:”.

2.1 Removing Digits from Event Messages

The simplest analysis involves manually reviewing the file with a standard file viewer and searching for a message pattern.

We apply an automatic transformation, which improves the overview of the file considerably without much effort: The central step is to replace all consecutive occurrences of digits by a single placeholder character, e.g., ‘9’, because in most cases numbers identify values (like a time or address). The reduced messages can then be sorted and duplicates can be dropped. This reduces the 20.000 lines to 1.048 different lines (patterns). For example, we find the patterns:

```

%SSH-9-ENABLED: SSH 9.9 has been enabled
%LINK-9-UPDOWN: Interface FastEthernet9/9, changed state to up

```

2.2 Extracting Event Types and List Patterns

We apply the following incremental approach for extracting event types by grouping event messages: The log file itself is the initial group. In each step, we take a large group, identify a frequent pattern, write a new parser rule and split the group accordingly. The splitting can be done automatically according to the type, which we store in each event. Repeating this step, creates a tree structure of patterns. The nodes near the root represent rather general patterns, whereas the leafs represent the most specific ones. In the case of the example log file, we can obtain the tree in Figure 1. The nodes *cisco* and *cisco/updown* correspond to the previously mentioned patterns. The first number of an inner node, counts the events matched by this node and none of its children, the second one counts the events matched by the whole subtree. Accordingly, leafs are marked only with a single number.

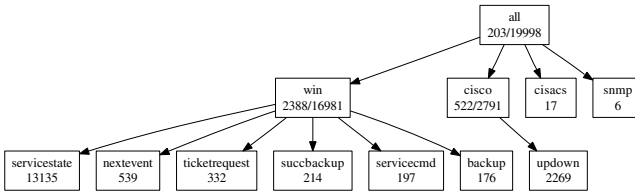


Fig. 1. Hierarchy of Event Types

At this stage, reviewing the groups reveals, that the *win* group with 2.388 matching events and 847 patterns is the most inhomogeneous; all the other groups are homogeneous already. The used file clearly has a strong dominance of Windows events. In log files with more diverse events, we would find more formats of different vendors and logging facilities at the top layer.

While creating the parsers for this example, we also found patterns of syntax elements like lists of key/value pairs or the dotted–decimal notation of IP addresses. These patterns can be handled by parser modules, which are used orthogonally to the tree structure. For example, we find a list pattern in many Windows messages. In the two patterns below, the lists follow the word *System*:

```

Security: 9: ... System: login attempt from: ME
    account: lock workstation: Y9 error code: 9xC9A
Security: 9: ... System: user logout: user
    name: Y9 domain: Y9 login type: 9
  
```

The list pattern has the form “ $a_1 : v_1 a_2 : v_2 \dots$ ”, where a_i and v_i are key/value pairs. After defining a parser for this format, we have implemented a very generic transformation rule for extracting all such lists from Windows event messages.

In order to build an extendable parser, we have formulated each parser element using transformation rules in the PROLOG–based XML transformation language FNTRANSFORM [16]. The rules are applied repeatedly, until no more rules match. This permits transformations to refine the result of previous transformations. We have also developed more refined parsing techniques based on *extended definite clause grammars*, which can be used for elegantly specifying and parsing more complex structures. Such grammar rules have been applied, for example, to electronic dictionaries in [15].

2.3 A Modular Event Format

Text messages are intended for human readers, and only a few formatting standards exist. Over time, many formats have evolved, all of which need special processing. So we cannot expect a parser to handle all of them. Instead, parser elements for new types of messages have to be created, and the parser has to be easily *extendable*. The following message, for example, taken from the mentioned log file, is expressed by the XML element below:

```
'1215009055419000', '31087: Jul 2 16:30:54:', 'unknown',
 '%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/26,
  changed state to up', '23', '5', ...
```

```
<event source="syslog" timestamp="1215009055419000"
  message="%LINEPROTO-5-UPDOWN: Line pro..."
  sl_facility="23" sl_priority="5" ... />
  <content type="cisco" severity="5" mnemonic="UPDOWN"
    facility="LINEPROTO" text="Line protocol on Interf..." />
  <content type="updown" new_state="up"
    iface="FastEthernet0/26" />
</event>
```

The attributes of the *event* element describe general values provided by the logging facility. Each parsing step creates an additional *content* subelement storing the extracted information. In the example above, the first *content* element of type *cisco* contains the fields common to all cisco events. The second *content* element provides the special values of this kind of line protocol message. This modular format allows for representing general patterns. Furthermore, we can refine these patterns by defining additional parsing rules. Such general patterns are defined, e.g., by hardware or software vendors like Cisco or Microsoft.

3 Temporal Data Mining Workflow

In this section, we will describe our declarative framework for temporal data mining. For the data reduction phase (preprocessing), declarative programming in PROLOG was very suitable, since the methods can be flexibly adapted and extended. Our approach also allows flexible options for analysis, considering, e.g., the inclusion of background knowledge. In a prototypical implementation, we have also formulated the central temporal data mining algorithms in PROLOG; since the performance was no problem, we have postponed a possible reimplementaion in a standard procedural language. In Section 3.5 we also apply text mining approaches for reducing the number of words.

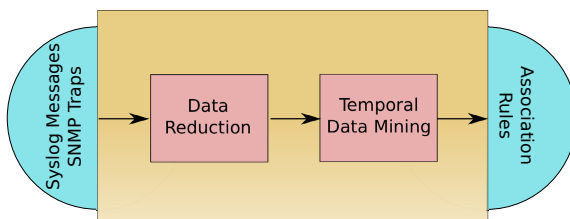


Fig. 2. Temporal Data Mining Workflow

We have decided to store the events uniformly in XML. In the case of parsed events, several properties are available. Unparsed events reveal only the basic values like the timestamp, the text message, and the sender address.

3.1 Combining Sequence Analysis and Message Analysis

The event sequence analysis of [10, 12] requires to assign an event type to each event. The event types could be identified by integers, and in the case of text messages, each type would describe a set of messages, i.e. the type describes a pattern. So, defining event types, requires approximately as much work as implementing the parser functions.

The approach presented in this paper combines sequence and message analysis. Generalising simple event types, we assign several key/value pairs to each event; the set of these pairs forms the *event data*. For unparsed text messages, we follow the approach of SLCT [18] and split the message into words. Every word is a value and the key is the word's position in the message. For parsed messages, more meaningful data, like the event types of [10, 12], can be added instead. We define a selection predicate using only the words of each text message. It selects the required attributes *timestamp* and *message* with expressions of the PROLOG-based XML query language FNQUERY [16], and then splits the message into words. Finally, the words are enumerated in their order, and we consider each position/word pair as an event on its own by inheriting the timestamp of the event; we call these events *subevents*. Because every original event had a unique timestamp, we can still reconstruct the event from its subevents. Subevents will show similarly strong correlations in a temporal analysis as were found in the non-temporal analysis of SLCT.

Since words describe the attributes of a message, temporal relations between attributes of events correspond to relations between subevents. Applying temporal analysis on the subevents will reveal these relations, although at no step manual classification of events was necessary. We augment the event sequence with unique timestamps. An *event pair* $P = (T, E)$ consists of the event's timestamp T and the event data E . Moreover, an *episode* $\alpha = (P_1, \dots, P_n)$ is a sequence of event pairs P_i .

3.2 Sliding Windows

The WINEPI algorithm is based on sliding a window over the event sequence. At each position of the window, it considers the set (or multiset) of visible events; the order of the events is irrelevant. For our analysis, we are only interested in the content of each window and not in its time bounds. A call *window*(*Win*) providing all possible instances of the sliding window in the variable *Win* as a list of events would be nice, but has some deficiencies. Firstly, we are not interested in empty windows. These are unneeded for frequency analysis, and the total number of windows can be computed from the sequence's bounds. Secondly, depending on the distribution of the events, many consecutive windows share the same content. Furthermore, the number of windows – but not the number of different contents – depends heavily on the timestamps' resolution. To avoid these issues, we have implemented a slightly different call *window*(*Win*, *Rep*), which provides all windows in the variable *Win*; several identical successive windows are returned only once. *Rep* specifies the repetition of the window *Win*. The rest of the algorithm, i.e. the candidate generation and the controlling loop, is equivalent to the standard apriori algorithm. Per option *injective*, one can decide between injective and not injective episodes, i.e. an episode may contain the same event type several times.

3.3 Frequent Episodes

We can now apply the frequent episodes analysis on events. For a first analysis, a rather small window size of 10 seconds has turned out to be appropriate. After having filtered (cf. Section 3.7) the most frequent episodes with a small temporal extent, we can select larger window sizes. The minimum frequency is difficult to preselect, because a single episode can occur in several windows. For example, a single occurrence of an event is counted in 100 windows, if the timestamps are given in 10–ths of a second and the window size is 10 seconds. But an episode spanning 10 seconds occurs in exactly one window. We decided to use a minimum frequency of 0.01 and selected appropriate settings in the following sections to obtain comparable results.

Using the words of the original text messages as event types, only three frequent injective and parallel episodes are found in the log file with 20.000 messages:

```
1-episodes: [ '___ "stopped".' ]-0.03799, [ '___ "executed".' ]-0.038002
2-episodes: [ '___ "stopped".' , '___ "executed".' ]-0.0376735
```

Here, ___ stands for the text 'Control Manager: 7036: Service "McAfee McShield" now is in the status'. Inspecting the frequencies of the 1– (single events) and 2–episodes (combinations of events) reveals, that the start and stop messages of the antivirus software almost always occur close to each other.

3.4 Episode Rules

Such temporal relations are described by episode rules, which are similar to association rules. We can apply the algorithm for association rules without modification.

Given the list of frequent itemsets (or frequent episodes), we search for an itemset A and a subset B of A , such that the *confidence* $conf = f(A)/f(B)$ is larger than a given minimum confidence threshold; here, $f(I)$ is the frequency of an itemset I . The result is an association rule $B \rightarrow A \setminus B$. If we apply the filtering on the set of frequent episodes, which we found above, then the following episode rules with a confidence of at least 0.99 are found. For brevity, we have replaced some text passages with an underscore – as in the previous subsection.

```
( '___ "stopped".' -> '___ "executed".' ): [0.99167, 0.0376735]
( '___ "executed".' -> '___ "stopped".' ): [0.99136, 0.0376735]
```

In the sequence of subevents, we find too many and a lot of redundant episode rules. We are interested in the most general rules, where a rule $r : A \rightarrow B$ is *more general* than a rule $s : C \rightarrow D$, if $A \subseteq C$ and $D \subseteq B$; the frequency is not important here. For example, the rule $\{a\} \rightarrow \{c, d\}$ is more general than $\{a, b\} \rightarrow \{c\}$. The less general rule s can be deduced from the more general rule r , because of the trivial rules $C \rightarrow A$ and $B \rightarrow D$; this means, that s provides no additional information except the confidence, which is not already provided by r . We ignore the confidence here, because we are interested in the compliance with the rather high confidence threshold and not the exact value. Of course, other variations are conceivable.

3.5 Data Reduction Based on Most Specific or Equivalent Words

Searching for frequent episodes in the sequence of subevents, i.e. on the words of the messages, produces a huge number of frequent episodes. Many of these episodes have no temporal extent, but occur in a single message, because the subevents of a message show a strong correlation. All subsets of a frequent message's words will be identified as frequent episodes. That is just too much to calculate and provides no new information. The main reason of this problem is the number of words per message: the 20.000 messages of the example file produce 259.735 words.

Therefore, in this section, we will present two methods for reducing the number of words or event data before applying frequent episode mining. Prior to any other reduction or analysis, we can drop all infrequent words, as they appear neither in frequent itemsets nor in frequent episodes. Dropping words with a frequency below 0.05 lowers the number of words from 259.735 down to 222.026.

Most Specific Words. Text messages are highly redundant, especially if they contain regular sentences. All redundant words can be dropped and the remaining words will suffice to identify the type of the message. Such relations are described by association rules treating the messages as transactions and the words as items. The interpretation of a rule $a \rightarrow b$ is that in a message containing the word a , we will also find the word b . If this rule has a high confidence, then we can drop the word b . Afterwards, b can also be reconstructed in all messages containing a .

The effect of this reduction is, that each message is described by as few words as possible. These words are the *most specific* ones, and unspecific words occurring in different types of messages are removed. For example, the following message is reduced to the three words/position pairs (1, '%LINK'), (5, 'FastEthernet0/26') and (9, up). From the first pair all remaining words can be derived.

```
%LINK-3-UPDOWN: Interface FastEthernet0/26, changed state to up
```

Applying this reduction on the log file lowers the word count from 222.026 to 20.793. Afterwards, a temporal analysis on the remaining words can only discover relations between such very specific types of messages, and more general patterns are lost.

Equivalent Words. Another approach is to find *equivalent words* in the messages. Of two equivalent words, only one has to be kept, the other one does not carry any further information and can be dropped, or the two words can be merged into one *compound word*. Two words are equivalent, if they always occur together. In data mining, we can only determine such relations with a certain frequency, of course.

At first, we search for frequent wordsets in the messages, i.e. itemsets in transactions. We only consider frequent 1- and 2-itemsets. Similar to finding association rules, the next step is finding association equivalences with a high confidence. From a frequent 2-itemset $\{a, b\}$ we deduce the association equivalence $a \leftrightarrow b$ with the confidence $conf(a \leftrightarrow b) = \min\{f(a)/f(b), f(b)/f(a)\}$; here, $f(i)$ is the frequency of an item i .

According to [18], applying the apriori algorithm for finding frequent sets of words in messages results in exponentially (i.e., for 2-itemsets: quadratically) many candidates, nearly all of which are infrequent. It is more appropriate to deduce candidates from each message and to check these candidates' frequency, which occur at least once. For two parameters $MinIt \leq MaxIt$, we compute all k -itemsets with $k \leq MaxIt$ by applying the apriori algorithm only after the $MinIt$ -th iteration. For calculating only the 1- and 2-itemsets without utilising the apriori algorithm, we set $MinIt = MaxIt = 2$. From the frequent 1- and 2-itemsets, we calculate all equivalences with high confidence.

The 20.000 messages reveal about 480 word equivalences with a confidence of at least 0.95. Using these equivalences, the 222.026 frequent words can be reduced to 48.318 words. For example, the words of the antivirus message from above are combined into two groups. The first group describes, that the service "McAfee McShield" has changed its state, and the second indicates the new state, the service changed to. The listing below shows the groups in detail. This example shows, that the grouping by association equivalences can be as accurate as a manual type definition.

```
[ (1, Control), (2, Manager), (3, 7036), (4, Service), (5, "McAfee"),
  (6, McShield), (7, now), (8, is), (9, in), (10, the), (11, status) ]
[ (12, "stopped") ]
[ (12, "executed") ]
```

In contrast to the reduction to the most specific words, which removes the words describing general message types, the reduction based on equivalent words keeps keywords for different degrees of abstraction, and more general relations can be discovered.

Frequent Episode Mining. After reducing the number of words to a more feasible number, we can apply the frequent episode algorithm and the rule discovery on the sequence of words or word groups. With the same settings as before (window size of 10 seconds, minimum frequency of 0.01, and injective episodes), we find three rules, which provide exactly the same information as the rules discovered before from the sequence of messages. For clarity we omit some words of the groups, which have already been listed above:

```
( (12, "stopped") -> (12, "executed"), [(1, Control) ___] ):0.99
( (12, "executed") -> (12, "stopped"), [(1, Control) ___] ):0.99
( [(1, Control) ___] -> (12, "stopped"), (12, "executed") ):0.98
```

We notice a deficiency of using parallel episodes for finding relations between subevents. The rules do not describe the temporal extent of the affected events, and accordingly, the reference to a relation between occurrences of complete text messages is unclear.

Let $\{A\} \rightarrow \{B, C\}$ be the third rule with the actual words replaced by the variables A, B and C . Then, we cannot tell, if this rule describes primarily three different occurrences of messages or only two, because the events may coincide with each other. A single message is not covered by this rule, because B and C exclude each other. In the case of two messages it is unclear, if A and B belong to a single message, or instead A and C . These rules do not comprise any temporal extent, and especially, do not describe any ordering, because we have used parallel episodes. It is important to keep in mind

that the episodes discovered by the sliding window approach only identify frequent occurrences of events in a time span given by the window size. Actually, there can be several occurrences in a single window.

3.6 Minimal Occurrences of Episodes

For comparison, we discuss another approach using minimal occurrences and serial episodes based on the MINEPI algorithm in this section. Every episode α is now additionally augmented with the set μ_α of its minimal occurrences, i.e. the minimal intervals containing the episode.

1-Episodes. In analogy to the apriori algorithm, we begin by determining all 1-episodes. The minimal occurrences of a 1-episode (A) are given by the occurrences of the event A . If A occurs at time S , then the point interval $[S, S]$ is a minimal occurrence of (A).

Candidate Generation. From a set of frequent serial episodes, larger episodes can be created analogously to the apriori algorithm. Two serial episodes $\alpha = \alpha' \cdot \delta$ and $\beta = \delta \cdot \beta'$ can be concatenated to an episode $\gamma = \alpha' \cdot \delta \cdot \beta'$. I.e., the suffix δ of α is also a prefix of β , and δ appears only once in the concatenation. According to [12], the minimal occurrences μ_γ of γ can be calculated from μ_α and μ_β by merging pairs of intervals: Let $[S_A, E_A] \in \mu_\alpha$ and $[S_B, E_B] \in \mu_\beta$ be minimal occurrences of α and β respectively, such that $S_A < S_B$ and $E_A < E_B$. If there is no other, later minimal occurrence $[S'_A, E'_A] \in \mu_\alpha$, where $S_A < S'_A$, with the same property, then $[S_A, E_B]$ is a minimal occurrence of γ . For example, for the sequence $\langle (1, a), (2, b), (3, a), (4, c), (5, b), (6, c), (7, d) \rangle$ and the episodes $\alpha = (a, b, c)$ and $\beta = (b, c, d)$, we get $\mu_\alpha = \{[1, 4], [3, 6]\}$, $\mu_\beta = \{[5, 7]\}$, $\gamma = (a, b, c, d)$, and $\mu_\gamma = \{[3, 7]\}$. We enforce the intervals $[S_A, E_A]$ and $[S_B, E_B]$ to be different, because we allow non-injective episodes. This can be seen in the following simple example. In the case of injective episodes, this restriction can be left out. In the sequence $\langle (1, a), (2, a) \rangle$, for example, the episode $\alpha = \beta = (a)$ has the set $\mu_\alpha = \mu_\beta = \{[1, 1], [2, 2]\}$ of minimal occurrences, and the combination $\gamma = (a, a)$ has only a single minimal occurrence: $\mu_\gamma = \{[1, 2]\}$. Beginning with the 1-episodes, we iteratively combine k -episodes to candidate $(k + 1)$ -episodes. From these candidates only the frequent ones are selected. In this process, we find all frequent episodes.

Rules. We use serial episodes and minimal occurrences in this section. Therefore, we have to adapt the previous rule algorithm, which uses parallel episodes and relative frequencies (based on windows). Different kinds of rules can be derived from serial episodes; we describe only the two most understandable kinds.

Forward rules have the form $\alpha \rightarrow \beta$, where α and β are serial episodes. Their interpretation is: if α has a minimal occurrence $[S, E]$, then the concatenation $\gamma = \alpha \cdot \beta$ has a minimal occurrence $[S, F]$ (with $E \leq F$). Less formally spoken, an occurrence of α is followed by an occurrence of β . The maximal extent of all occurrences is given by the same upper bound. That is, we do not use two different time bounds for each rule, but only one upper bound for the whole rule. This reduces the parameters for the algorithm, and nonetheless, rules with a smaller temporal extent are still found. *Backward rules*

$\beta \leftarrow \alpha$ mean: if α has a minimal occurrence $[S, E]$, then the concatenation $\gamma = \beta \cdot \alpha$ has a minimal occurrence $[R, E]$ (with $R \leq S$). In other words, an occurrence of α is preceded by an occurrence of β . In both cases, the confidence of such rules r is easily calculated as $conf(r) = f(\gamma)/f(\alpha)$, as in the case of association rules.

In the 20.000 lines of the example log file, we discover the same relation as before. The upper bound was set to 10 seconds, the minimum support to a 1/10 of the word count (20.793), and the minimum confidence was set to 0.90.

```
(12, "stopped") -> (12, "executed")
(12, "stopped") -> (1, Control) __ (11, Status)
```

But this time, the rules are more expressive. We can interpret both rules together, because their left hand sides are identical. They describe that a message including the word *stopped* is usually followed by a message including the words of the right hand sides (Control __ executed) within 10 seconds. Again, we have omitted some words of the word groups described in Section 3.5.

Analysing the Minimal Occurrences. The minimal occurrences of the concatenation (i.e., γ from above) of a rule's left and right hand side indicate the rule's temporal expansion, which can actually be much smaller than the given upper bound.

Therefore, we analyse the lengths of the minimal occurrences. Several thousands of values can be visualised in a histogram. But a histogram does not necessarily reveal dense regions, if the values differ slightly as is the case with distances between events. It seems more appropriate to apply a clustering algorithm on the length values. We have selected the common *hierarchical clustering* method. Beginning with a cluster for each value, clusters too close to each other are merged repeatedly. For our experiments, we have used the distance between the arithmetic means of two clusters to decide their proximity. In the case of the above rule, the results are trivial. The two events occur in immediate succession, i.e. all minimal occurrences have a length of two 10-ths of a second. Further interesting results that we have obtained with MINEPI will be reported in a more detailed case study in Section 4.

3.7 Filter Rules for Event Messages

In order to discover additional correlations, it is reasonable to filter already known relations, which would clutter up the results unnecessarily. A first filter would replace the previously shown two messages of the antivirus software with a single *stop/start* message. Analysing the reduced event sequence will then reveal, that these messages repeat approximately every 5 minutes. Therefore, we need other filter rules, which are the combination of repeating events and the deletion of a rule's consequent.

For a replacement rule $replace(Episode, Replace, WinSize)$ the events matching the serial episode *Episode* are removed from the sequence, and the replacement *Replace* is inserted at the time of the first event. *WinSize* determines the time span in which the episode has to occur. Moreover, each event type in the episode can match a more specific type. For example, the event type or event data X matches the data Y , where

```
X = [(1, "Service"), (2, "McAfee)],
Y = [(1, "Service"), (2, "McAfee"), (3, "stopped)].
```

4 Case Study

We have developed an interactive workflow, which incorporates the presented methods. In the following list, we briefly repeat each step's functionality:

1. Read the log file – provided in CSV format – and create the initial event sequence.
2. Apply the event filters thereby removing already discovered correlations from the sequence.
3. Reduce the number of words by dropping infrequent data and by identifying the most specific words and the equivalent words.
4. Create the sequence of subevents.
5. Decide suitable parameters, and then apply a data mining algorithm (WINEPI or MINEPI) to find frequent parallel or serial episodes.
6. Derive episode rules. In the case of serial episodes from the MINEPI algorithm, we display the extent of the minimal occurrences in a histogram and determine clusters, which can help to find temporal properties of the rules.

In an extensive experiment, we did several successive runs of the data mining algorithm MINEPI (steps 5 and 6), discovered new relations each time, and formulated appropriate filter rules. Depending on the results of the previous run, we decided the parameters for the next run. As input we used again a log file with 20.000 messages. In the following, we describe the settings and results of each run of MINEPI. For clarity, we omit all but some important keywords from the discovered rules; the gaps are marked with an underscore.

First Run. The algorithm MINEPI (injective, serial episodes) is applied with the following parameters: *Upper Bound*: 10 seconds, *Min. Support*: 0.1, *Confidence*: 0.9. It discovers the following relation: A stop message of service McAfee is followed immediately (less than 1 second) by a start message of the same service. They occur always together and in this order:

```
[ (1, Control), (2, Manager) _ (5, "McAfee) _ (12, "stopped") ] <->
[ (1, Control), (2, Manager) _ (5, "McAfee) _ (12, "executed") ]
```

The double arrow describes that from either message the existence of the other can be deduced.

The first run reveals the relation which we have already described in the previous sections. We integrate this relation in a filter rule. The joint occurrence of the messages is replaced by a single stop/start message. 6.320 such occurrences are found in the log file. These settings reveal no other relations, therefore, we have to increase the upper bound or lower the minimum support in the next iteration.

Second Run. The algorithm MINEPI (not only injective, serial episodes) is applied with the following parameters: *Upper Bound*: 6 minutes, *Min. Support*: 0.1, *Confidence*: 0.9. It discovers the following relation: A McAfee stop/start message is repeated in about 6 minutes with a probability of 0.99.

```
[ (1, Control), (2, Manager) _ (5, "McAfee) _ (7, stop/start) ] <->
[ (1, Control), (2, Manager) _ (5, "McAfee) _ (7, stop/start) ]
```

Without lowering the minimum support, we had to increase the upper bound to 6 minutes in order to find something interesting. The reported relation indicates a periodic repetition of this message. Investigating the lengths of the minimal occurrences reveals a single cluster, i.e. the messages repeat after an average gap of 5 minutes. We can filter these chains using a repetition rule, which removes 6.241 messages.

Third Run. In this run, four relations were found using the algorithm MINEPI (injective, serial episodes) with the following parameters: *Upper Bound:* 10 seconds, *Min. Support:* 0.01, *Confidence:* 0.9. It discovers the following relation: A change of an interface's state causes a change to the protocol's state of this interface. The algorithm reported the following rules:

```
[ (1, %LINK), (2, 3), (3, UPDOWN), (4, Interface),
  (6, changed), (7, state), (8, to), (9, up) ] ->
[ (1, %LINEPROTO), (2, 5), (3, UPDOWN), (4, Line),
  (5, protocol), (6, on), (7, Interface), (9, changed),
  (10, state), (11, to), (12, up) ]

[ (1, %LINEPROTO), (2, 5), (3, UPDOWN), (4, Line),
  (5, protocol), (6, on), (7, Interface), (9, changed),
  (10, state), (11, to), (12, down) ] <-
[ (1, %LINK), (2, 3), (3, UPDOWN), (4, Interface),
  (6, changed), (7, state), (8, to), (9, down) ]
```

The start-up of a link is followed by the start-up of the according line protocol. The reversed arrow of the second rule describes that the shut-down of a link is preceded by the shut-down of the line protocol. The rules don't talk about the words at position 5 and 8 in the link and the line protocol message respectively. These words describe the affected interface. That means, the algorithm correctly identified an abstract pattern and categorisation of messages. The same relation is reported with messages of a slightly other format, too. We will handle these in the same way.

The event filter can remove the line protocol message using two deletion rules. The current implementation of the event filter does not support variables to describe that only pairs of messages should be filtered, which affect the same interface. 810 line protocol messages were removed all together.

In the following, we explain three of the relations discovered during the third run.

1. *Discovered relation:* The assignment of special permissions and the successful login of a specific user always occur together within 1 second and in this order:

```
[ (1, Security) __ (5, Besondere), (6, Rechte), (7, bei),
  (8, neuer), (9, Anmeldung), (10, Benutzername),
  (11, Y068SPWDK102$), (12, Domaene), (13, Y068DPK1) __ ] <->
[ (1, Security) __ (5, Erfolgreiche), (6, Netzwerkeranmeldung),
  (7, Benutzername), (8, Y068SPWDK102$), (9, Domaene'),
  (10, Y068DPK1) __ ]
```

Interestingly, this relation is found for one specific user. That means, that this user logs in very frequently. Nonetheless, this rule should be effective for any user. 123 messages of the second kind could be removed.

2. *Discovered relation:* We have no useful interpretation of the following relation with a time bound of less than 1 second; 132 messages of the first kind were removed:

```
[ (1, find), (2, message), (3, file), (4, key), (5, for),
  (6, "SYSTEM\\_\\TsmVssPlugin") ] <->
[ (1, TsmVssPlugin) _ (7, Status), (8, Success) ]
```

3. *Discovered relation:* The service *Volumeschattenkopie* (i.e., a backup) starts if and only if the control manager has sent the command to do so in advance. The algorithm reported the following equivalence in two separate rules:

```
[ (1, Control), (2, Manager) _ (6, Der),
  (7, Steuerbefehl), (8, "starten"), (9, wurde) _ (11, an),
  _ (14, "Volumeschattenkopie"), (15, gesendet) ] <->
[ (1, Control), (2, Manager) _ (5, "Volumeschattenkopie"),
  _ (9, im), (10, Status), (11, "executed") ]
```

We decided to remove the second message, i.e. the execution, and keep the command. The event filter reports 89 deletions. Later, we found the same relation between the same messages but translated to english. These occurred 57 times.

Fourth Run. The algorithm MINEPI (injective, serial episodes) is applied with the following parameters: *Upper Bound:* 10 minutes, *Min. Support:* 1/130, *Confidence:* 0.9. It discovers the following relation: The service *Volumeschattenkopie* completes in less than 8 minutes.

```
[ (1, Control), (2, Manager) _ (8, "starten") _
  (11, an) _ (14, "Volumeschattenkopie"), (15, gesendet) ] <->
[ (1, Control), (2, Manager) _ (5, "Volumeschattenkopie") _
  (11, "Beendet") ]
```

86 messages were removed. The results of this run include a lot of clutter caused by bursts of logouts.

5 Practical Relevance

New technology and vendors are integrated daily in a large network. This leads to new log messages, and based on the connection of the elements new event types occur in a network. In practice the amount of log messages (several millions over a day) and the simultaneous occurrence of different events leads to a mix up in the received event messages of different events and makes it very difficult to extract *best-practice rules* for daily operation.

With our presented approach, we expect to simplify the life of a network administrator, and in the best case to even automate the generation of rules in the used fault management system. We use the proposed approach to group and aggregate messages to dedicated events as a first stage in our practical implementation.

Thus, from millions of lines of log messages, which are nearly impossible to investigate, we come to a much smaller set of events with dedicated log messages. The network operator only has to investigate the small amount of discovered and grouped events and to decide, if the discovered correlation should be part of the rule set of his used fault management system. This simplifies the process of rule discovery dramatically, and good feedback is already provided. Our ultimate goal would be an automatic generation of rules based on the network events.

6 Conclusions

We have presented two selected algorithms for the discovery of episode rules, and we have given insights into the possible results, providing the basis for future research about these rules.

As a next step, we are planning to integrate the presented solution into the commercial network management solution StableNet, which covers fault, performance and configuration management in one product. StableNet has been used by large enterprise and telco customers for several years. The integration gives us the possibility to verify the results in large data networks. We are expecting that the discovery of rules will simplify the setup of fault management in StableNet dramatically. At the moment, rules have to be defined by an experienced user and refined later. We are expecting to simplify and speed up this process with the presented approach, such that we can react earlier to changes in the network.

The used temporal data mining algorithms can be further extended to also support the discovery of serial episodes using window-based frequency and parallel episodes using minimal occurrences. Furthermore, we can consider complete or strictly closed episodes [19, 17]. The combination of non-injective serial episodes and subevents is still an issue, because several subevents share the same timestamp. At the moment, we have solved this by enforcing the minimal occurrences to contain only one event per timestamp and accordingly only one event per message, but some interesting rules require message patterns consisting of several words.

For example, consider the messages `login` of `X` and `logout` of `X` for some user `X`. Then, we can usually find serial episodes like $((1, \text{login}), (1, \text{logout}), (1, \text{login}), (1, \text{logout}))$ or $((1, \text{login}), (3, A), (1, \text{logout}), (3, B))$, but not like the serial non-injective episode $((1, \text{login}), (3, A), (1, \text{logout}), (3, A))$, because the first two and the last two words occur at the same time respectively. But, the latter would be required for the rule $((1, \text{login}), (3, A)) \rightarrow ((1, \text{logout}), (3, A))$.

While applying the filter rules, the event filter could at the same time determine the confidence and the outliers of each filter rule. The confidence would then indicate the quality of the filter rules and the incorporated knowledge with respect to the latest events. The outliers indicate particularly interesting events. Furthermore, causal analysis of event sequences, c.f. [6], complementing the association and correlation analysis, would be interesting as well, since it could directly provide more actionable knowledge. Additionally, the integration and extension of subgroup discovery methods [2–4] for temporal sequences is another interesting option to consider.

References

1. Achar, A., Laxman, S., Sastry, P.: A Unified View of the Apriori-Based Algorithms for Frequent Episode Discovery. *Journal of Knowledge and Information Systems* 31(2), 223–250 (2012)
2. Atzmueller, M., Lemmerich, F.: Fast Subgroup Discovery for Continuous Target Concepts. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) *ISMIS 2009*. LNCS, vol. 5722, pp. 35–44. Springer, Heidelberg (2009)

3. Atzmueller, M., Puppe, F.: A Knowledge-Intensive Approach for Semi-Automatic Causal Subgroup Discovery. In: Berendt, B., Mladenič, D., de Gemmis, M., Semeraro, G., Spiliopoulou, M., Stumme, G., Svátek, V., Železný, F. (eds.) *Knowledge Discovery Enhanced with Semantic and Social Information*. SCI, vol. 220, pp. 19–36. Springer, Heidelberg (2009)
4. Atzmueller, M., Puppe, F., Buscher, H.-P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 647–652 (2005)
5. Chen, J., He, H., Williams, G., Jin, H.: Temporal Sequence Associations for Rare Events. In: Dai, H., Srikant, R., Zhang, C. (eds.) *PAKDD 2004*. LNCS (LNAI), vol. 3056, pp. 235–239. Springer, Heidelberg (2004)
6. Chuah, E., Lee, G., Tjhi, W., Kuo, S., Hung, T., Hammond, J., Minyard, T., Browne, J.C.: Establishing Hypothesis for Recurrent System Failures from Cluster Log Files. In: *Proc. 9th IEEE International Conference on Dependable, Autonomic and Secure Computing*, pp. 15–22 (2011)
7. Casas-Garriga, G.: Discovering Unbounded Episodes in Sequential Data. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) *PKDD 2003*. LNCS (LNAI), vol. 2838, pp. 83–94. Springer, Heidelberg (2003)
8. Hand, D.J., Smyth, P., Mannila, H.: *Principles of Data Mining*. MIT Press (2001)
9. Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Series in Data Management Systems. Morgan Kaufmann (2000)
10. Klemettinen, M., Mannila, H., Toivonen, H.: Rule Discovery in Telecommunication Alarm Data. *Journal of Network and Systems Management* 7(4), 395–423 (1999)
11. Laxman, S., Sastry, P.S.: A Survey of Temporal Data Mining. *Sadhana, Academy: Proceedings in Engineering Sciences* 31, 173–198 (2006)
12. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of Frequent Episodes in Event Sequences. *Journal of Data Mining and Knowledge Discovery* 1, 259–289 (1997)
13. Méger, N., Rigotti, C.: Constraint-Based Mining of Episode Rules and Optimal Window Sizes. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) *PKDD 2004*. LNCS (LNAI), vol. 3202, pp. 313–324. Springer, Heidelberg (2004)
14. Pflieger de Aguiar, L., de Almeida, V.A.F., Meira, W.: Mining Redundant Industrial Alarm Occurrences with Association Rules Extraction and Complex Networks Modeling. *Journal of Computational Methods in Science and Engineering* 11, 15–28 (2011)
15. Schneiker, C., Seipel, D., Wegstein, W., Prätör, K.: Declarative Parsing and Annotation of Electronic Dictionaries. In: *Proc. 6th International Workshop on Natural Language Processing and Cognitive Science, NLPCS (2009)*
16. Seipel, D.: Processing XML-Documents in PROLOG. In: *Proc. 17th Workshop on Logic Programmierung, WLP (2002)*
17. Tatti, N., Cule, B.: Mining Closed Strict Episodes. *Journal of Data Mining and Knowledge Discovery* 25(1), 34–66 (2012)
18. Vaarandi, R.: A Data Clustering Algorithm for Mining Patterns from Event Logs. In: *Proc. IEEE Workshop on IP Operations and Management (2003)*
19. Wu, J., Wan, L., Xu, Z.: Algorithms to Discover Complete Frequent Episodes in Sequences. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) *PAKDD Workshops 2011*. LNCS, vol. 7104, pp. 267–278. Springer, Heidelberg (2012)

Learning in the Presence of Large Fluctuations: A Study of Aggregation and Correlation

Eric Paquet^{1,2}, Herna Lydia Viktor², and Hongyu Guo¹

¹ National Research Council, 1200 Montreal Road, Ottawa, Ontario, K1A 0R6, Canada

² School of Electrical Engineering and Computer Science, University of Ottawa, 800 King
Edward, Ottawa, Ontario, K1N 6N5, Canada
{eric.paquet, hongyu.guo}@nrc-cnrc.gc.ca,
hlviktor@eecs.uottawa.ca

Abstract. Consider a scenario where one aims to learn models from data being characterized by very large fluctuations that are neither attributable to noise nor outliers. This may be the case, for instance, when predicting the potential future damages of earthquakes or oil spills, or when conducting financial data analysis. It follows that, in such a situation, the standard central limit theorem does not apply, since the associated Gaussian distribution exponentially suppresses large fluctuations. In this paper, we present an analysis of data aggregation and correlation in such scenarios. To this end, we introduce the Lévy, or stable, distribution which is a generalization of the Gaussian distribution. Our theoretical conclusions are illustrated with various simulations, as well as against a benchmarking financial database. We show which specific strategies should be adopted for aggregation, depending on the stability exponent of the Lévy distribution. Our results indicate that the correlation in between two attributes may be underestimated if a Gaussian distribution is erroneously assumed. Secondly, we show that, in the scenario where we aim to learn a set of rules to estimate the level of stability of a stock market, the Lévy distribution produces superior results. Thirdly, we illustrate that, in a multi-relational database mining setting, aggregation using average values may be highly unsuitable.

Keywords: Aggregation in Relational Learning, Correlation-based Analysis and Covariance, Lévy Distribution, Stable Distribution.

1 Introduction

Aggregation is an important step when pre-processing data, prior to building a data mining model. This step is crucial when considering complex data that represents interactions between several potentially heterogeneous entities. For instance, in social network analysis the frequency of a particular relationship is often represented by an aggregation based on the number of occurrences. The same observation holds in multi-relational database mining and in spatial data exploration, where aggregation is needed to link multiple tables together [1, 2]. Similarly, data obtained from data streams are frequently summarized into manageable sized buckets or windows, prior to mining.

Often, during a data mining exercise, it is implicitly assumed that large-scale data fluctuations must be either associated with noise or outliers, or that a concept drift has occurred. The most striking consequence of such an assumption is that, once the noisy data and the outliers have been eliminated, the remaining data may be characterized in two ways. That is, firstly, their typical behaviour (i.e. their mean) and secondly, by the characteristic scale of their variations (i.e. their variance). Fluctuation above the characteristic scale is thus being assumed to be highly unlikely, or assumed to indicate that a model has become outdated. However, there are many categories of data which are characterized by large-scale fluctuations. For instance, surprisingly, supermarket ketchup sales have been shown to be typified by such large-scale fluctuations [3]. Further, financial data and earthquake-related data are also examples of data exhibiting this behaviour [4]. This issue is highly relevant when aiming to build models that predict the potential damages caused by catastrophic events, such as financial market turbulences and tsunamis. The large-scale fluctuations do not originate from noise or outliers, but constitute an intrinsic and distinctive feature of the data. Mathematically speaking, small fluctuations are modelled with the central limit theorem and the Gaussian distribution, while large fluctuations are modelled with the generalized central limit theorem and the Lévy distribution. This paper studies the aggregation of data presenting large-scale fluctuations, to determine their properties, the best approaches for their aggregation and the impact of such behaviour on their correlation.

Our main contributions are as follows. Firstly, we provide a theoretical analysis which shows the importance of taking the data distribution characteristics, when learning involves aggregation and correlation, into account. Secondly, we introduce the Lévy (or also called stable) distribution as a mechanism to allow machine learning methods to compute meaningful aggregated information and to correctly evaluate the correlation when learning from data presenting large fluctuations. Thirdly, we demonstrate the proposed method's applicability in typical machine learning and data mining problems. Here, we discuss the analysis of financial data in order to build rules to classify the volatility of the market as well as the case of multi-relational database learning involving aggregation.

This paper is organized as follows. In Section 2, we review the fundamental assumptions behind aggregation, namely the central limit theorem and the Gaussian distribution. In Section 3, we introduce a more general distribution for the aggregate, the Lévy distribution, for which the Gaussian distribution is a particular case. We explain how this distribution may be estimated from the empirical data and present some useful properties of the Lévy distribution. Then, we study the rank ordering statistics of the Lévy distribution in order to determine if there are some dominant terms in the distribution. We introduce the multivariate stable distribution in order to generalize, in Section 4, the concepts of covariance and correlation to stable distributions. In Section 5, we present various simulations in order to illustrate the theoretical results obtained in the previous sections, as well as their consequences for aggregation. We show that our methodology is applicable to real world financial data, within two classification settings. The last section presents our conclusions and directions for future work.

2 Aggregation, Central Limit Theorem and Gaussian Distribution

In this section, we review the basic assumptions on which aggregation is based. Despite the fact that these assumptions are quite general, they do not cover all possible data distributions, for instance, the Lévy distribution. Importantly, as will be shown in Section 5, data associated with catastrophic events stock market crashes, other financial turmoil (such as a housing market collapse) and earthquakes often follows the Lévy distribution and need special care during data pre-processing and model building. Consequently, we aim to understand their strengths as well as their limitation in order to be able to address them in the following sections.

Aggregation is based on the standard central limit theorem which may be stated as follows. The sum of N normalized independent and identically distributed random variables of zero mean and finite variance σ^2 is a random variable with a probability distribution function converging to the Gaussian distribution with variance σ^2 . This implies that aggregation, in the sense of a sum of real numbers, has a Gaussian distribution irrespectively of the original distribution of its individual data. This is a very powerful theorem because the Gaussian distribution may be characterized with solely two numbers, namely its mean μ and its variance σ^2 which are the first two moments of the distribution. In practice, this implies that an aggregation, such as a sum, may be fully characterized by its mean and its variance; this is why aggregation is so powerful. All the other moments of the Gaussian distribution are equal to zero.

Aggregation has often been used, during data pre-processing, to handle one to many and many to many relationships in data. For example, consider the scenario where one is building a classification model against a relational database that contains one to many relationships from the so-called target table to the other (background) tables. Existing methods such as Relaggs and MRC proceed by using the standard SQL aggregation operations such as *average*, *sum*, *minimum* and *maximum*, in order to link these tables together [5, 6]. Further, in data streams a number of techniques employ some form of aggregation, when selecting the data window to be used for model building [7]. It follows that aggregation is highly suitable in domains where the central limit theorem holds. However, it should be used with care, especially in cases where large-scale fluctuations are common. This is often the case in many data stream applications that involve monitoring, e.g. medical monitoring or security screening of CCTV. In these cases, we are more interested in finding the exception that is typified by a large fluctuation (e.g. a security breach), rather than the rule. In the next section, we consider a generalization of the central limit theorem which requires the introduction of the Lévy or stable distribution.

3 Aggregation with an Underlying Lévy Distribution or Stable Distribution

In this section, we review the Lévy distribution, we show how it may be estimated from the underlying empirical data and we analyze its properties. Finally, we

introduce the multivariate Lévy distribution in order to extend the notions of covariance and correlation to data distributed in this way.

3.1 Definition of the Lévy Distribution

One may associate to a probability distribution $L(x)$ its Fourier transform or characteristic function $L(k)$:

$$L(k) = \int_{-\infty}^{\infty} \exp(ikx) L(x) dx \Leftrightarrow L(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-ikx) L(k) dk \quad (1)$$

Stable or Lévy distributions are distributions for which the individual data as well as their sum are identically distributed [8]. This fact implies that the convolution of the individual data is equal to the distribution of the sum or, equivalently, that the characteristic function of the sum is equal to the product of their individual characteristic functions. The Lévy distribution does not have a closed form and is more easily defined from its characteristic function:

$$L_{\alpha,\mu,\beta,\gamma}(k) = \mathbf{E}\left(\exp(ikX)\right) = \exp\left[-i\mu k - \gamma^\alpha |k|^\alpha \left(1 - i\beta \operatorname{sgn}(k) W(\alpha, k)\right)\right] \quad (2)$$

where $\mathbf{E}(\bullet)$ is the expectation and where

$$W(\alpha, k) \triangleq \begin{cases} \tan \frac{\pi\alpha}{2} & \alpha \neq 1 \\ -\frac{2}{\pi} \ln |k| & \alpha = 1 \end{cases} \quad (3)$$

The Lévy distribution is characterized by four parameters, as opposed to the Gaussian distribution which is characterized by only two. The parameters are the stability exponent α , the scale parameter γ , the asymmetry parameter β and the localisation parameter μ . While the tail of the Gaussian distribution is exponentially suppressed, the tail of the Lévy distribution decays as a power law (heavy tail) which depends on its stability exponent α :

$$L_\alpha(x) \sim \frac{C_\pm}{|x|^{1+\alpha}} \Bigg|_{x \rightarrow \pm\infty} \quad (4)$$

Eq. (4) shows that extreme values are much more likely for the Lévy distribution than they are for the Gaussian distribution. The reason for this is that the Gaussian distribution fluctuates around its mean, the scale of the fluctuations being characterized by its variance (the fluctuations are exponentially suppressed) while the Lévy distribution may produce fluctuations far beyond the scale parameter. This behaviour is due to the tail power decay law. It should be noted that the Lévy

distribution reduces to the Gaussian distribution when $\alpha = 2$ and when the asymmetry parameter is equal to zero; then one has $\sigma^2 = \frac{1}{2}\gamma$. Finally, the moments

of the Lévy distribution, $m_n = \int_{-\infty}^{\infty} x^n P(x) dx$, may be finite if $n \leq \alpha$ while they are infinite if $n > \alpha$. This implies that a Lévy distribution with $1 \leq \alpha < 2$ has a finite mean, but an infinite variance while a distribution with $\alpha < 1$ has both an infinite mean and an infinite variance. As we will see in the following sections, these properties have grave consequences from the aggregation point of view. This is because the concept of a mean and variation becomes unsuitable when the data follows this distribution.

3.2 Estimation of the Lévy Distribution from the Empirical Data

We explain how the parameters of the Lévy distribution may be estimated from the empirical data and how the validity of the Lévy distribution hypothesis may be asserted. Although various approaches have been proposed in the literature, one of the most efficient is the one presented by Paulson, Holcomb and Leitch (PHL) [9] in which the following objective function is minimized against the parameters of the Lévy distribution

$$\min_{\alpha, \beta, \gamma, \mu} \left\| \hat{L}(k) - L_{\alpha, \mu, \beta, \gamma}(k) \right\|_{\text{PHL}} \quad (5)$$

where the PHL-norm is defined as

$$\left\| \hat{L}(k) - L_{\alpha, \mu, \beta, \gamma}(k) \right\|_{\text{PHL}} \triangleq \int_{-\infty}^{\infty} \left| \hat{L}(k) - L_{\alpha, \mu, \beta, \gamma}(k) \right|^2 \exp(-k^2) dk \quad (6)$$

Because the integration domain is not bounded, Eq. (5) is more readily solved with a Gauss-Hermite quadrature.

3.3 Rank Ordering Statistics

More insight about the Lévy distribution may be obtained from its rank ordering statistics

$$F(y_n) dy_n = (N - n + 1) \binom{N}{n} \left(1 - \int_{y_n}^{\infty} P(x) dx \right)^{N-n} P(y_n) dy_n \left(\int_{y_n}^{\infty} P(x) dx \right)^{n-1} \quad (7)$$

which gives the probability that the largest value of order n be y_n ; for instance, y_1 is the maximum. From Eq. (7), it may be demonstrated that the maximum of likelihood for the statistics of order n associated with the stable distribution is:

$$y_n^{\text{ML}} = \left[\frac{(\alpha N + 1)\gamma}{\alpha n + 1} \right]^{1/\alpha} \quad (8)$$

When the stability exponent is inferior to one, the rank ordering statistics exhibit a strong hierarchical behaviour up to the point that the ordering statistic of order one (the maximum) completely dominates over all the other rank ordering statistics. This behaviour shall become more evident with the experimental results, presented in Section 5. A practical consequence associated with this behaviour is that the aggregation should be based on the maximum value which completely dominates the ordering statistics. This is the case, for instance, when a Stock Market Index crashes. The information obtained from the rank ordering statistics may be exploited in order to group the elements of the hierarchy according to their scale, or order of magnitude. Then, it may be shown that each scale is characterized by its own Gaussian distribution. Consequently, the Gaussian paradigm is applicable to the Lévy distribution in a multiscale framework and the Lévy distribution might be thought of as a multiscale generalization of the Gaussian distribution.

3.4 Multivariate Lévy Distribution: Definition and Estimation

We extend the Lévy distribution to the multivariate case, i.e. when we have more than one dimension or feature. Such a multivariate distribution is required when one aims to study the correlation in between two stable stochastic variables. The multivariate Lévy characteristic function [8] of dimension d is defined as follow

$$L_{\alpha, \boldsymbol{\mu}}(\mathbf{k}) = \exp(-I_{\mathbf{X}}(\mathbf{k})) = \mathbf{E}[\exp(i\langle \mathbf{k}, \mathbf{X} \rangle)] = \exp\left[-i\langle \boldsymbol{\mu}, \mathbf{k} \rangle - \int_{S^d} \psi_{\alpha}(\langle \mathbf{k}, \mathbf{s} \rangle) \Delta(ds)\right] \quad (9)$$

where as usual $\mathbf{E}(\cdot)$ is the expectancy, where

$$\psi_{\alpha}(u) \triangleq \begin{cases} |u|^{\alpha} \left(1 - i \operatorname{sgn}(u) \tan \frac{\pi\alpha}{2}\right) & \alpha \neq 1 \\ |u| \left(1 + i \frac{\pi}{2} \operatorname{sgn}(u) \ln |u|\right) & \alpha = 1 \end{cases} \quad (10)$$

and where the Euclidian inner product, the frequency vector and the stochastic data vector are defined as

$$\langle \mathbf{k}, \mathbf{X} \rangle \triangleq \sum_{i=1}^d k_i X_i, \quad \mathbf{k} = [k_1, \dots, k_d]^T, \quad \mathbf{X} = [X_1, \dots, X_d]^T \quad (11)$$

As opposed to the univariate case, three parameters are required. This is in contrast to the four scalar parameters that are required for the one-dimensional stable distribution. The first two are the stability exponent α and the localisation vector $\boldsymbol{\mu}$. The information about the scale and the asymmetry, which in the one-dimensional case was captured by two scalar parameters, is now encapsulated in a unique parameter $\Delta(ds)$ which is a measure, or a partition, defined on the hypersphere S^d

(or the sphere in two dimensions). In order to estimate such a distribution from the empirical data, we follow an approach introduced by Nolan et al. [10]. At first, the means and the stability exponents associated with each dimension are estimated independently. Then, the estimated mean vector and the stability exponent of the multivariate distribution are given by:

$$\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \dots, \hat{\mu}_d)^\top, \quad \bar{\alpha} = \frac{1}{d} \sum_{k=1}^d \hat{\alpha}_k \quad (12)$$

The empirical multivariate characteristic function is obtained from a discrete formulation of Eq. (9). The discrete equation associated with the expectation is:

$$L(\mathbf{k}) = \mathbf{E}[\exp(i\langle \mathbf{k}, \mathbf{X} \rangle)] \Rightarrow \hat{L}(\mathbf{k}_i) = \frac{1}{M} \sum_{j=1}^M \exp(i\langle \mathbf{k}_i, \mathbf{X}_j \rangle) \quad (13)$$

where $\{\mathbf{X}_j\}_{j=1, \dots, M}$ is the set of all the empirical multivariate data while the discrete equation associated with the right part of Eq. (9) is:

$$\hat{I}(\mathbf{k}_i) = -\ln \hat{L}(\mathbf{k}_i) = \sum_{j=1}^n \psi_{\bar{\alpha}}(\langle \mathbf{k}_i, \mathbf{s}_j \rangle) \Delta_j \quad (14)$$

If a symmetric grid is assumed for both the hypersphere and the frequency domain, the weights on the hypersphere $\{\Delta_i\}_{i=1, \dots, n}$ may be estimated from the following constrained objective function

$$\min \quad \|\mathbf{c} - \mathbf{A} \boldsymbol{\Delta}\|^2 \quad \therefore \quad \boldsymbol{\Delta} \geq 0 \quad (15)$$

where

$$\mathbf{c} = \begin{bmatrix} \text{Re}(\hat{I}(\mathbf{k}_1)), \dots, \text{Re}(\hat{I}(\mathbf{k}_m)), \\ \text{Im}(\hat{I}(\mathbf{k}_1)), \dots, \text{Im}(\hat{I}(\mathbf{k}_m)) \end{bmatrix}^\top \quad (16)$$

$$\boldsymbol{\Delta} = [\Delta_j], \quad \mathbf{A} = [A_{i,j}]$$

where Re and Im stand for the real and imaginary part of a complex number and where

$$A_{i,j} = \begin{cases} \text{Re}(\psi_{\bar{\alpha}}(\langle \mathbf{k}_i, \mathbf{s}_j \rangle)) & i, j = 1, \dots, m \\ \text{Im}(\psi_{\bar{\alpha}}(\langle \mathbf{k}_i, \mathbf{s}_j \rangle)) & i, j = m + 1, \dots, 2m \end{cases} \quad (17)$$

Then, we use this grid in order to define the generalization of the covariance for the stable distribution.

4 Generalization of the Covariance: The Covariation

We extend the concept of covariance to stable distributions. This is important, in practice, because we often need to determine if two variables are correlated or not. This is the case, for instance, when aiming to protect data privacy, where the goal is to determine if an

attribute may be inferred from another [11]. If the data are distributed according to a Lévy distribution, the concept of covariance must be generalized with the concept of covariation [12]. The covariation in between two stochastic stable variables is defined as

$$\llbracket X_1, X_2 \rrbracket_\alpha = \int_{S^2} k_1 k_2 \langle \alpha-1 \rangle \Delta(ds) \quad \therefore x \langle \alpha-1 \rangle \triangleq |x|^{\alpha-1} \operatorname{sgn}(x) \quad (18)$$

where the measure on the bidimensional sphere is associated with the bivariate distribution of the vector formed from the concatenation of the two stochastic variables involved in the convolution $\Delta(ds) \Leftrightarrow \mathbf{X} = [X_1, X_2]^T$.

Such a measure may be estimated with the method presented in the previous section and with the grid introduced in Eq. (15). The covariation reduces to the covariance when the distribution is Gaussian, i.e. when the stability exponent is equal to two and the asymmetry is zero. The correlation belongs to the interval $[0, 1]$ where zero indicates an absence of correlation while the unity indicates a strong correlation. As shown in the next section, the covariance and the correlation tend to be much stronger if the stability exponent is less than two. Practically, if one incorrectly assumes a Gaussian distribution from the start, one may strongly underestimate the real correlation between two variables. Such an underestimation might have severe consequences [13].

For instance, consider the scenario where private attributes should be identified and protected. Our earlier work shows that aggregation potentially introduces new privacy violations. That is, potentially harmful attributes obtained with aggregation are often different from the ones obtained from non-aggregated databases which means that, even when privacy is enforced on non-aggregated data, it is not automatically enforced on the corresponding aggregated data [14]. Suppose that an absence of correlation between two aggregated attributes such as Age and Income is assumed, due to an erroneous assumption that the data distribution is Gaussian, rather than Lévy. However, in reality, these two attributes may be highly correlated. This incorrect assumption would result in an absence of protection for sensitive attributes, when they do indeed need to be protected against induced attacks.

5 Experimental Results

In this section, we present various simulations and experiments against real-world financial databases which illustrate our previous theoretical results. All experiments were performed using Mathematica 8.0 on a Dell Precision M6400. In the following, one should keep in mind that $\alpha = 2$ corresponds to a Gaussian distribution.

Table 1. Estimation of the exponent of the Lévy distributions associated with various Stock Exchange. Excerpted from [15]

Index	Period	α
FTA W Jap	86.01-93.09	1.808
TOPIX	75.01-91.02	1.519
MSCI Japan Net	80.01-93.09	1.463
Nikkei 225	80.01-93.09	1.626

We begin by analyzing some results on financial data as reported by Lévy Véhel and Walter (LVW) [15]. The importance of stable distribution is not only theoretical; as a matter of fact, it has far reaching consequences for financial data. With the pioneer work of Mandelbrot, it became increasingly apparent that financial data may be characterized with stable distributions. For instance, let us consider Table 1 which shows the results obtained for various Stock Market Indexes in Europe and in Japan by LVW [15]. The stability exponent was estimated with the method presented in Section 3 and the null hypothesis was asserted with the Kolmogorov-Smirnov test. As shown by the data, all these indexes clearly have a stable distribution (confidence level of 99%) and the value of the stability exponent is typically in between 1.6 and 1.8, which is clearly not in the Gaussian regime.

Table 2. Covariation of two shares (Thompson and Michelin) and the CAC 40 Index for various values of the stability exponent. The acquisition period is from 87.07.09 to 95.05.31. Excerpted from [15].

$\llbracket X_1, X_2 \rrbracket_\alpha$	α			
	2.0	1.7	1.5	1.3
THOMPSON, CAC 40	0.042	0.157	0.390	0.975
MICHELIN, CAC 40	0.042	0.159	0.326	0.993
CAC 40, CAC 40	0.036	0.128	0.300	0.750

Table 2 shows the covariations in between the Michelin and the Thompson titles as well as with the CAC 40 Stock Exchange Index. The covariations may be estimated with the method presented in Sections 4. The calculation was repeated for various values of the stability exponent; the real one being around 1.7. Table 2 shows that, if a Gaussian distribution is incorrectly assumed, the correlation (covariation) tends to be underestimated. For instance, the covariation in between the Michelin share and the CAC 40 Index is 0.042 with the false assumption of a Gaussian distribution for the data while in reality it is 0.159 for a stability exponent of 1.7. Here, the covariation tends to be stronger, when the stability exponent is smaller.

5.1 Market Value Volatility Classification

In the next experiment, we consider a binary classification problem. In this simulation, the level of stability of the market (i.e. whether a market is volatile or stable) is assumed to be inferred from a set of rules based on the enterprise value indicator. The enterprise value is an economic indicator reflecting the market value of a business in its entirety. It is defined as the sum of the claims of all the security-holders, namely debt holders, preferred shareholders, minority interest, common equity holders, and others. The enterprise value is one of the fundamental metrics used in business valuation, financial modelling, accounting and portfolio analyses, amongst others. A rule assessing the volatility of the market may be stated as follows. If the typical fluctuation of the enterprise value indicator is greater than a critical level, then the market is likely to become volatile. On the other hand, if the typical fluctuation of the indicator is less than the critical value, the market should remain stable.

In order to simulate this problem, we created ten synthetic data sets. Each data set is constituted of 100 000 enterprise value indicators. The indicators are distributed according to a truncated Lévy distribution. The truncation is necessary in order to have a realistic simulation, because the indicators may only take values within a certain interval. That is, the value that an enterprise is worth is always bounded in that its value cannot be infinite. In this case, the typical fluctuation may be assimilated to the scale parameter γ of the Lévy distribution. In order to demonstrate the importance of taking into account the underlying distribution, we estimate the typical fluctuation with two approaches. In the first approach, we assume that the central limit theorem applies and that the typical variation may be assimilated to the standard deviation of the data. In the second approach, we fit a truncated Lévy distribution to the synthetic data and assimilate the typical fluctuation to the scale parameter of the fitted Lévy distribution.

The Lévy distribution used in order to generate the data is parameterized as follows. The stability exponent which characterized extreme fluctuations has a value of 0.5. The scale parameter, which refers to the typical variations of the indicator, has a value of 4. For the purpose of this simulation, the asymmetry parameter is equal to zero and the localization parameter is equal to one, while the distribution is defined on the interval one to hundred. The critical value of the indicator, i.e. the value where the market becomes volatile, is set to 5.

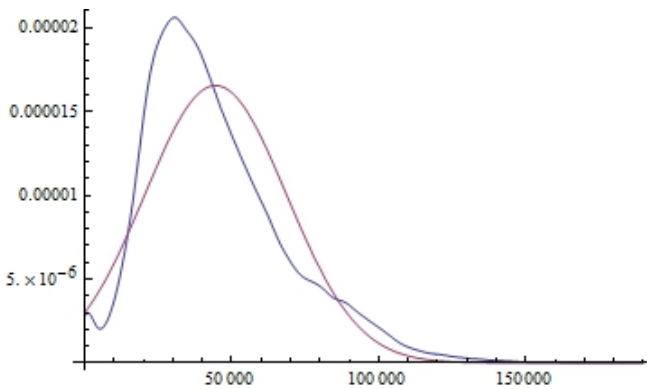


Fig. 1. Distribution of the Balance attribute (PKDD 1999 discovery challenge financial database) in **blue** and of the best fitting normal distribution in **red** as obtained with the maximum of likelihood method. (The normal distribution poorly represents the actual distribution.) The abscissa is the Balance attribute and the ordinate is the probability density function.

If one assumes an underlying Gaussian distribution and calculates the typical variation with the standard deviation for each data set, one obtains a value in between 19.38 and 19.54 (recall that the correct value is 4) with an accuracy of 0%. On the other hand, if one fits a truncated Lévy distribution to each data set and read the typical variation from the scale parameter of the fitted distribution, one obtains a value in between 3.96 and 4.04 (recall again that the correct value is 4) with an accuracy of 100%. The validity of the fitting is asserted by applying both the Kolmogorov-Smirnov and the Cramér-von Mises tests. In both cases, the null hypothesis that the data are distributed according to the fitted distribution is not rejected at the 5% level.

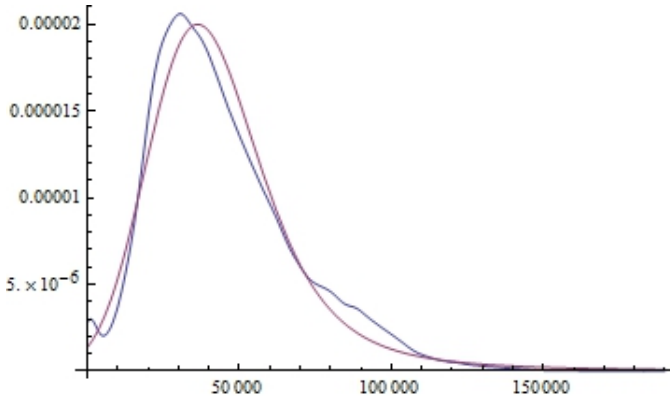


Fig. 2. Distribution of the Balance attribute in blue and of the best fitting stable distribution in red; the later is obtained with the maximum of likelihood method. The abscissa is the Balance attribute and the ordinate is the probability density function.

Consequently, if one assumes an underlying Gaussian distribution for the data, one wrongly classifies the market as highly volatile ($19.38 > 5$). When fitting a Lévy distribution, one correctly concludes the market is unlikely to be volatile ($4.04 < 5$).

5.2 PKDD 1999 Discovery Challenge Financial Database

Next, we consider the PKDD 1999 discovery challenge financial database, which has been widely used as a benchmark in the multi-relational classification domain. This database was offered by a Czech bank and contains data describing the level of risk of a customer to default on a loan [16]. The database consists of eight tables. The Account table contains the account number, as well as the information regarding the district a person falls in and the frequency of payment. Other tables include the Demographic profile, the client's Disposition in terms of type, Credit Card information and Client descriptions, including gender and the location in which they reside. The Order table details the number of money transfers and the Loan table describes the payments of loans. Very often, this database is used to classify whether a Loan is at risk or not. This dataset contains the data about 682 Loans, of which 76 had bad outcomes. In this paper, we are especially interested in situations where an unforeseen event may expose more Loans to risk and we are thus interested in determining the upper limits of the minimum payments associated with high risk Loans. To this end, we aim to determine the interplay in between the remaining Balance and the Amounts to be paid, in order to determine their importance for determining the risk level of a Loan. Specifically, we are interested in determining the rules to identify the Loans that are likely to become high risk (the minority class), e.g. in a housing market collapse caused by sudden financial turmoil.

To this end, we turn our attention to the Transaction table, which contains the details of all Transactions associated with a Loan. We apply a number of feature selection algorithms to this database, including the Gain ratio, Chi Squared and Correlation based Feature Selection (CFS) measures [11]. Our results indicate that the

Amount and Balance attributes are always selected as being features that are strongly related to the outcome of a Loan, with and without aggregation. In this setting, a single Loan has many Transactions associated with it.

Table 3. Parameters of the fitted stable distribution associated with the Balance attribute

α	β	μ	γ
1.6232	1.0	47321.3	14013.9

The Transaction table contains, amongst others, the Amount and Balance attributes that both contain 54694 entries. Recall that, to accommodate one to many relationships, existing techniques that learn from multi-relational databases, such as Relaggs and MRC, use SQL aggregation functions [5, 6]. To this end, for each Loan, the original Transaction table is transformed into storing the *average*, *minimum*, *maximum* and *sum* of the Amount and Balance, prior to model building. Implicitly, it is assumed that the data has a normal distribution. For example, the SQL average function computes the average of a set of values by dividing the sum of those values by the count of values that is not null. Next, we explore whether this implicit assumption holds for these two attributes.

Table 4. Mean, standard deviation, skewness and kurtosis as obtained from the fitted Gaussian and Lévy distributions associated with the Balance attribute

Distribution	Mean	Std Dev.	Skewness	Kurtosis
Normal	44534.2	24109.7	0	0
Lévy	47321.3	∞	∞	∞

Figure 1 shows the distribution of the Balance attribute as well as the best fitting Gaussian distribution. The parameters of this distribution are obtained with the maximum likelihood method. It follows that the Gaussian distribution offers a poor fit to the Balance attribute distribution. We attempted to fit numerous distributions to the Balance, such as the Student distribution, the Weibull distribution, amongst others. However, the best fit was obtained with the Lévy distribution, as illustrated in Fig. 2. The parameters corresponding to this distribution, which were obtained with the maximum likelihood method, are shown in Table 3.

Table 5. Mean, standard deviation, skewness and kurtosis as obtained from data generated from the fitted stable distribution associated with the Balance attribute. Each generated data set consists of 54694 entries.

Distr.	Mean	Std Dev.	Skewness	Kurtosis
1	47419.2	99020.1	145.408	27467.7
2	47996.1	158948	168.972	33756.5
3	47041.7	46051.2	18.1492	677.382

Table 4 shows the mean, the standard deviation, the skewness and the kurtosis calculated from the normal and the stable distributions associated with the Balance attribute distribution. Since the stability exponent α is smaller than two, it is not

possible to evaluate the standard deviation and the skewness from the stable distribution, because the statistical moments needed for the calculation are infinite. Nevertheless, the parameters of the stable distribution provide a measure of the standard deviation and of the skewness through the scale parameter γ and β the asymmetry parameter. This implies that, when the underlying distribution is stable, the scale and the asymmetry should not be estimated directly from the data, but from the parameters of the fitted distribution.

In order to further stress the importance of not directly estimating these parameters from the data, we have generated three data sets. These datasets consisted of 54694 entries as in the original database with the stable distribution parameterized by Table 4. Table 5 shows the results. Since the stability exponent is greater than one (1.62), the estimation of the mean from the data is consistent from one set to the next. However, because the stability exponent is less than two (1.62), it is not possible to estimate any moment greater or equal than two which means that the standard deviation, the skewness and the kurtosis are not consistent from one synthetic data set to the next. In other words, any measure based on the statistical moments greater or equal than two is a random number which will vary from one realization of the data set to the other as shown in Table 5. These findings suggest that when correlation evaluation is sensitive, one should carefully select the right equation for correlation computation [11].

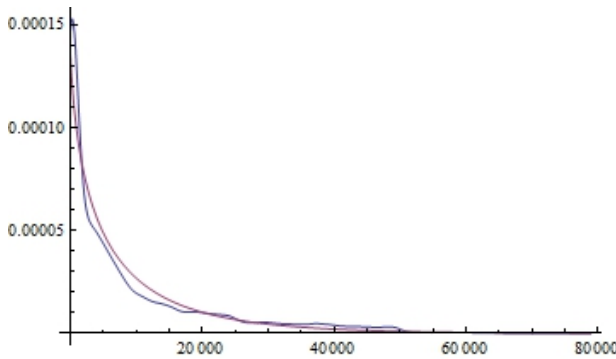


Fig. 3. Distribution of the Amount attribute in blue and of the best fitting Weibull distribution in red. The abscissa is the Amount attribute and the ordinate is the probability density function.

Next, we turn our attention to the Amount attribute, which also contains 54,694 values. Here, the best fit was obtained with a Weibull distribution [17], another extreme value distribution

$$W_{\xi < 0, m, a}(x) = \exp \left[- \left(\frac{m + \frac{a}{|\xi|} - x}{a} \right)_+^{\frac{1}{|\xi|}} \right] \tag{19}$$

as illustrated in Fig. 3. This distribution also has a heavy tail, which implies that the normal distribution is not a good fit in the case where one is interested in studying extreme values. In this application, it follows that we are indeed very interested in transactions with high amounts, in order to alert the financial institution of e.g. the

possibility of money laundering activities. The same observation holds for small amounts being paid, which may point to customers who are at risk to shortly default on their loans. Again, using aggregation functions to handle the one to many relationship between the Loan and the Transaction table, may lead to a data mining algorithm failing to find rules that detect such cases.

In summary, our analysis indicates that care should be taken when employing aggregation when building models against relational databases. In many cases, assuming that the normal distribution holds is not correct. In the case where we are interested in extreme values, averaging values may not be the best option. To further illustrate this point, we built a number of J48 decision trees and JRIP rule learners against a dataset which contains the Loan and Transaction tables. The resulting dataset contains 682 tuples with aggregated values for the balances and amounts paid. We trained the classifiers against this dataset, in order to explore the importance of each aggregate when aiming to predict a loan's outcome. We followed the standard 10 fold cross validation approach. The accuracy of a default classifier is 88.8% and it fails to identify *any* bad Loans.

Next, we used an aggregate based on the minimum values of the Amounts being paid and remaining Balances, i.e. in order to predict the upper limits of the minimum Amounts and Balances being paid, before a high risk loan defaults. This aggregate produce a J48 decision tree which is 93.8% accurate and a JRIP rule learner which is 94.5% accurate. We were thus able to learn sets of highly accurate rules that identify the upper threshold values for the Balances and Amounts associated with bad Loans. Specifically, JRIP created two out of three high coverage rules that are 100% accurate against the bad Loans. Importantly, using the average values does not produce highly accurate predictive models, with an accuracy of only 89.1%. Rather, when using the average, this aggregate fails to correctly classify any bad Loans. This confirms our observation that the averages are not useful to typify the outcomes of high risk Loans.

6 Conclusions

The development of new data mining models for catastrophic event prediction, including stock market volatility detection, estimating the damages caused by oil spills, and forecasting the extend of tsunamis and other natural disasters, are an important and urgent research topic. In this communication, we have analyzed data aggregation and the data covariance (covariation), of such data, where the underlying distributions are not Gaussian, but Lévy. We have shown that such data may be aggregated with the mean, but not with the variance. This is due to the fact that the variance becomes infinite and its estimate tends to fluctuate randomly when evaluated on a finite size aggregate. We have also shown that the estimation of the mean converges rather slowly when the stability exponent is small. In this case, both the mean and the variance are infinite and their estimate on a finite size aggregate tends to fluctuate randomly. In these circumstances, the aggregate is better characterized with its upper limit which tends to dominate by many orders of magnitude over the other elements of the aggregate, both from a sum and rank ordering statistics point of view. We have shown that financial data may be characterized with stable distributions with a stability exponent typically around 1.7. The calculation of the covariations in between Stocks and Stock Market Indexes has shown that the covariance (covariation) tends to be underestimated if a Gaussian distribution is wrongly

assumed. We have shown that, for a well-known benchmarking financial database, some attribute values follow a stable distribution rather than the normal distribution. Further, our results show that the use of average values is not suitable in such situations. As we also mentioned in Section 2, our approach is highly relevant for data stream mining, where we are interested in finding exceptions and large fluctuations in fast evolving data. We aim to explore this research issue in our future work.

References

1. Knobbe, A.J., Siebes, A., Marseille, B.: Involving Aggregate Functions in Multi-Relational Search. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 145–168. Springer, Heidelberg (2002)
2. Malerba, D.: A relational perspective on spatial data mining. *Int. J. Data Mining. Modelling and Management* 1(1), 103–118 (2008)
3. Groot, R.D.: Lévy distribution and long correlation times in supermarket sales. *Physica A: Statistical Mechanics and its Applications* 353, 501–514 (2005)
4. Walter, C.: Lévy-stability-under-addition and fractal structure of markets: implications for the investment management industry and emphasized examination of MATIF notional contract. *Mathematical and Computer Modelling* 29(10-12), 37–56 (1999)
5. Krogel, M.A., Wrobel, S.: Facets of aggregation approaches to propositionalization. In: *The 13th International Conference on Inductive Logic Programming, ILP 2003* (2003)
6. Guo, H., Viktor, H.L.: Multirelational classification: A multiple view approach. *Knowledge and Information Systems* 17, 287–312 (2008)
7. Zliobaite, I., et al.: Next challenges for adaptive learning systems. *ACM SIGKDD Explorations Newsletter* 14(1), 9 (2012)
8. Samorodnitsky, G., Taqqu, M.S.: *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. Chapman & Hall, New York (1994)
9. Paulson, A.S., Holcomb, E., Leitch, R.: The estimation of the parameters of the stable law. *Biometrika* 62(1), 163–170 (1977)
10. Nolan, J.P., Panorska, A.K., McCulloch, J.H.: Estimation of spectral measures. *Mathematical and Computer Modelling* 34(9-11), 1113–1122 (2001)
11. Guo, H., Viktor, H.L., Paquet, E.: Privacy Disclosure and Preserving in Learning with Multi-relational Databases. *Journal of Computing Science and Engineering* 5(3), 183–196 (2011)
12. Cheng, B., Rachev, S.: Multivariate Stable Future Prices. *Mathematical Finance* 5, 133–153 (1995)
13. Tao, Y., Pei, J., Li, L., Xiao, X., Yi, K., Xing, Z.: Correlation hiding by independence masking. In: *IEEE 26th International Conference on Data Engineering, ICDE*, pp. 964–967 (2010)
14. Jafer, Y., Viktor, H.L., Paquet, E.: Aggregation and privacy in multi-relational databases. In: *Tenth Annual International Conference on Privacy, Security and Trust, PST*, pp. 67–74 (2012)
15. Lévy Véhel, J., Walter, C.: *Les marchés fractals (“The fractal markets”)*. Presses Universitaires de France, Paris (2002)
16. Berka, P.: Guide to the Financial Data Set. In: Siebes, A., Berka, P. (eds.) *PKDD 2000 Discovery Challenge* (2000)
17. Rinne, H.: *The Weibull Distribution: A Handbook*. Taylor & Francis Group, Boca Raton (2009)

Retracted: Machine Learning as an Objective Approach to Understanding Music

Claire Q¹ and Ross D. King²

¹ Aberystwyth University, UK
ceq08@aber.ac.uk

² University of Manchester, UK
ross.king@manchester.ac.uk

Abstract. Traditional research into the arts has generally been based around the subjective judgment of human critics. We propose an alternative approach based on the use of objective machine learning programs. To illustrate this methodology we investigated the distribution of music from around the world: geographical ethnomusicology. To ensure that the knowledge obtained about geographical ethnomusicology is objective and operational we cast the problem as a machine learning one: predicting the geographical origin of pieces of music. We collected 1,142 pieces of music from 73 countries, and described them using 2 sets of standard audio descriptors using MARSYAS. To predict the location of origin of the music we developed a method designed to deal with the spherical surface topology based upon a modified k-nearest-neighbour. We also investigated the utility of *a priori* geographical knowledge in the predictions: a land and sea mask, and a population distribution overlay. The best-performing prediction method achieved a median land distance error of 1,506km, with comparable random trials having mean of medians 3,190km - this is significant at $P < 0.001$.

1 Introduction

1.1 An Objective Approach to Understanding Art

We hold the strong philosophical position that we do not fully understand a phenomenon unless we can make a machine that reproduces it: “What I cannot create, I do not understand” (written on Richard Feynman’s blackboard at the time of his death). The advantage of this approach to understanding is that it is wholly objective and fully operational. This type of approach to understanding a phenomenon is taken by the AI community to be working towards understanding intelligence – they aim to develop intelligent machines. If they succeed in doing so it will be possible to objectively determine their success. This approach to understanding contrasts strikingly with that of the traditional research into understanding art, which has almost always been based around the subjective judgment of human critics. Often great insight is gained by this subjective approach, but it also has to be granted that there are limitations to the results relying upon the peculiarities of the listener. We propose to extend the objective

approach to understanding phenomena to art – in this case, music. Specifically we will use the success of predictive machine learning programs as a measure of objective success in understanding a phenomenon. To illustrate and demonstrate our proposed objective approach to art we investigated the distribution of music from around the world: geographical ethnomusicology. To ensure that any understanding is objective the problem is cast as a machine learning one. This removes personal opinions and expectations in the listener because all decisions are made by machine.

1.2 Geographical Ethnomusicology

The world contains a vast variety of types of music. This music arose as the result of complex geographical, historical, and prehistorical processes. One way to better understand these processes is to analyse the current geographical distribution of music. The study of this distribution is termed Geographical Ethnomusicology. The problem of determining the geographical origin of a piece of music is complicated. Musical forms are rarely pure. Over time they have influenced each other, and many forms of music have travelled far from their point of origin. In particular the influence of western music is nearly ubiquitous. The influence of other forms of music are also widely distributed, for example: Arabic musical influence spread all around the Indian Ocean, across North Africa, to Spain, to central Asia, etc.; more recently reggae has spread from Jamaica, to the UK, Brazil, Mauritius, etc. The question we wish to answer is given these complications, how well can a computer predict the geographical origin of a piece of music?

It could be argued that unsupervised spatial clustering methods such as Kohonen nets [1] would be best suited to such a task. However, the problem with such clustering methods is that there is generally no objective measure of success. Such methods could find groups of similar music in terms of their audio descriptors, but they would not necessarily extract those features most suited to predicting a location. This contrasts with supervised methods, where the labels on known examples (classes or numbers) enable the objective measuring of whether a method is working or not - does it predict well or badly? As we know the geographical location of origin of the music (to some degree) in our corpus we should exploit this information. We therefore cast the problem as that of training a machine learning program to be able to predict the geographical origin of pieces of music, i.e. the computer learns a functional relationship between the audio content and its geographic origin on the globe. This predictive task is possible to some extent by human musicologists.

1.3 Related Work

A large amount of research has been recently done on the development of audio features (attributes) for the computational analysis of music, e.g. [2, 3]. These attributes have typically been used to perform automatic classification and clustering to identify similar pieces of music (especially for recommendation

systems), e.g. to identify mood, genre, emotive content, and various other purposes for which it would be impossible to provide an exhaustive list [4]. In addition to audio attributes other meta-data have been utilised via, for example, web searches and social tags, but also MIDI, score reading and lyric mining [5–8]. Various machine learning and statistical methods have been used upon these attributes: Support Vector Machines, k-Nearest-Neighbour, Neural Nets etc. with good success for certain applications [9]. Despite these advances little computational work has been done on computational ethnomusicology. The work of Liu *et al.* demonstrated the applicability of computational music analysis techniques to non-western music [10]. Gomez *et al.* applied these techniques to classifying music as western or non-western with success, and also found some important features relating to the latitude and longitude of origin of a piece [11, 12]. Tzanetakis’ work on computational ethnomusicology [13] sought to illustrate the potential application of music information retrieval (MIR) to ethnomusicology. This allows the analysis of large corpuses of music to obtain automatically features that would take copious time to transcribe by hand. Though the features are from signal processing they relate closely to how humans perceive music. One example is the spectral centroid, which is mathematically simple (it is the weighted mean of the frequencies in the signal) and yet is strongly correlated with human perception of ‘brightness’ in sound [14, 15].

Spatial Statistics. The problem of predicting the geographical origin of music is closely related to the field of spatial statistics [16]. The most common application of spatial statistics has been in statistical geography, and in epidemiology. A classic example is the work of John Snow who in 1855 provided strong evidence that cholera was waterborne via statistical means: a geographical dot map showing the occurrences of cholera were clustered around a particular water pump. On a larger scale, geographic information systems including global positioning systems have in recent years have created the discipline geospatial information studies, in which large databases of geographic information are analysed using geospatial relationships such as adjacency, containment and distance. Our work can be considered part of the latter category as a distance measure is the eventual output which measures success. Recent work in Self Organising Maps refers to the formation of geospatial shapes to avoid the edge problem, one example being GeoSOM [17]. Though we have already mentioned why clustering is unsuited to our prediction task, the difference in topology between a flat map of latitude and longitude and a spherical representation that wraps around is an important distinction.

2 Method

2.1 Music Collection

Our corpus was built from a personal collection of 1,142 tracks covering 73 countries.¹ The music used is traditional, ethnic or ‘world’ only, as classified

¹ The music used is subject to copyright, but the processed data is not, and the data is to be made publicly available.

by the publishers of the product on which it appears. We have not included any Western music as is naturally hard to place since its influence is global – what we seek are the aspects of music that most influence location. Thus, being able to specify a location with strong influence to the music is paramount. This will form the target function for the learning algorithm. To determine the geographical location of origin we manually collected the information from the CD sleeve notes, and when this information was inadequate we searched other information sources. There are most certainly other options as demonstrated by Govaerts *et. al.* but these have varying levels of accuracy and indeed their ground truth for the experiment was ‘personal knowledge’ or ‘by looking up the origin’ [18]. We did not wish to confound the ability of the predictor with incorrect location information. The location data is limited in precision to the country of origin - we did not have time to try to find out more about each track. In many cases the level of detailed precision possible for musical origin is arguably not much smaller than perhaps a region of a country, except in certain cases where a community has been extremely isolated.

The country of origin was determined by the artist’s or artists’ main country of residence. Of course, many artists live in different places throughout their lives, but our aim was to determine the major influence. For example, if a Malian writing Mali music lives in retirement in Paris, we consider the music Malian. We recognise that some music is less linked to countries than cultures, but countries at least have true geographical locations that are measurable - by virtue of having defined borders - allowing our machine learning approach to give objective output. We have taken the position of each country’s capital city by latitude and longitude as the absolute point of origin in the beginning. The assumption here is that the political capital is also the cultural capital of the country. This assumption also utilises coarse a priori knowledge about population - most, but not all, countries have a highly populous capital city. Using the capital takes into account country-level population distribution in a simple way without resorting to time-consuming investigations into the exact place each artist spent most time. In the population distribution task we altered this to the centre of population, or population centroid, of each country, which is a fairer measure that takes into account skewed population distribution and capitals with low populations. Countries are linked to artists, not tracks.

It is clear that the country of *production* may have no bearing on the origin, since many world music CDs are made in Germany or the US despite the music coming from e.g. Kenya. The artist in question is usually the composer where known, or else the performer where the music is traditional to their home country. If several artists have made a contribution to the music, all substantial (not merely ‘featuring’) contributions are taken into account when deciding if it should be included. Any track that had ambiguous origin – whether because the artist’s own origin was ambiguous, or many artists from different countries collaborated, or the track is a deliberate fusion of styles – was removed from the dataset. There are no “right answers” in such cases. For example, Bhangra music is a fusion of Indian Punjab culture, UK culture and hip-hop. Therefore to try to

determine a single geographical location for a Bhangra track would be nonsensical as it has multiple sites. One could suggest that the geographical midpoint of all the influences is the right answer, but this does not fully encapsulate the data and would result in a position for Bhangra music near Volgograd!

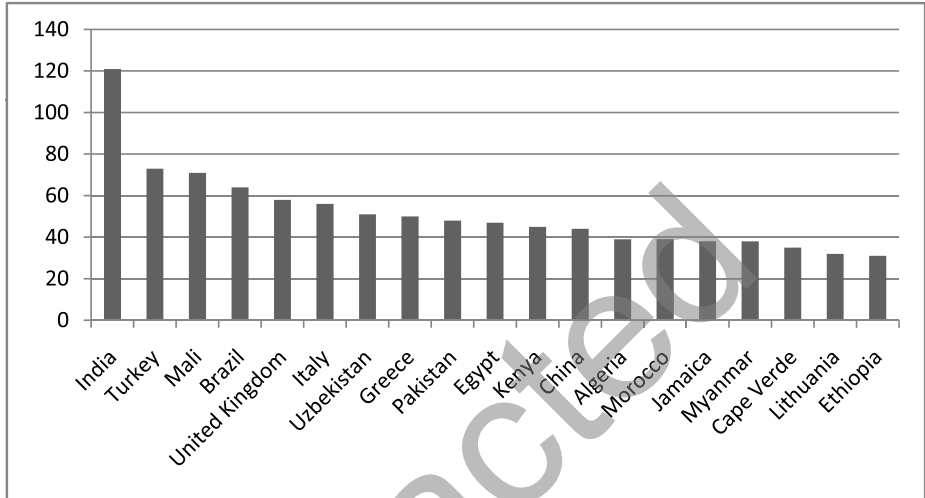


Fig. 1. Partial sample of music distribution by country

Figure 1 shows the distribution of tracks per country, for the 20 best represented countries. It can be seen that some countries are much better represented than others, which will affect the performance - more examples from a country gives more data about that country’s music and thus better predictions.

2.2 Audio Features

The program MARSYAS [19] was used to extract audio descriptors from the wave files. We first used the default MARSYAS settings in single vector format (68 features/attributes) to estimate the performance with basic timbral information covering the entire length of each track.

MARSYAS Features. The Marsyas features available are summarised in table 1.

Each of the default features is an indicator of timbre, which is one of the main ways (another being attack-decay-sustain-release models) [14] to distinguish musical instruments. Since instrumentation is also a major difference between cultural music traditions, these are appropriate to the task. No feature weighting or pre-filtering was applied. All features were transformed to have a mean of 0, and a standard deviation of 1. We also investigated the utility of adding chromatic

Table 1. MARSYAS features

Feature	Explanation	Grouping
Time Zero-crossings	The number of times a signal changes sign, <i>i.e.</i> how often it crosses the horizontal zero line.	Default timbral
Spectral Centroid	A measure of the “centre of mass” of the power spectrum.	Default timbral
Spectral Rolloff	Describes the amount of skew in the power spectrum.	Default timbral
Spectral Flux	Is an indicator of the amount of spectral variance based upon differences between adjacent spectral windows.	Default timbral
Mel-Frequency Cepstral Coefficients	coefficients for a mel-frequency (tailored to human auditory response) power cepstrum - representing the short-term power spectrum.	Default timbral
chroma	detects frequency matches for each musical note of the Western scale (and its octaves)	Chromatic feature
Spectral Measure	Flatness quantifies how tone-like, as opposed to noise-like, a sound is.	Non-default timbral
Spectral Crest Factor	Peak to average ratio of amplitude. Variance in loudness across frequency.	Non-default timbral
Line Spectral Pair	Describe the two resonance frequencies of the vocal tract when open or closed.	Non-default misc
Linear Prediction Cepstral Coefficients	Prediction as MFCC, but linear rather than Mel-scale	Non-default timbral

attributes. These describe the notes of the scale being used. This is especially important as a distinguishing feature in geographical ethnomusicology – unlike in Western music which largely conforms to one tuning system. The chromatic features provided by MARSYAS are 12 per octave – Western tuning, but it may be possible to tell something from how similar to or different the music is from Western tuning.

2.3 Geographic Representation

The problem of predicting a point on the surface of a sphere is made more complicated as the standard coordinate system (latitudes and longitudes), which are the natural targets for regression, have a complicated relationship to surface area where the predicted point will be. This is illustrated in the standard Mercator projection of the globe where countries near the poles are unnaturally large. Area is not preserved equally on such projections. In general there is no

perfect flat projection - a compromise is made for some parameter or parameters in favour of some desired quality, perhaps straight lines of latitude and longitude, perhaps equal area, even such complicated solutions as the butterfly map by Cahill later re-imagined by Waterman. None of these addresses the true mathematical problem fully - the Earth is not flat, and should therefore ideally be treated as close as possible to its true topology. In considering the sparsity of our data, we choose a sphere as an approximate representation for the globe, though with more precision still it is an oblate spheroid with certain peaks and troughs across the surface.

2.4 Spherical k-Nearest Neighbour Prediction Method

We decided to cast the problem as a regression problem (predicting a point) rather than a classification problem (predicting a country) because the large number of countries, and low number of examples per country, would complicate classification. Most regression methods assume either that either only one real number is to be predicted, or if multiple real numbers are to be predicted that they are independent. Perhaps the simplest approach to running regression with spherical coordinates is to side-step this difficulty and use a k-Nearest Neighbour method to predict points [1]. A Euclidean (in attribute space) k-NN algorithm was run using the musical features as axes. For each track the nearest k neighbours were found, the geodesic mean of their locations was taken, and the result compared to the true origin. To adopt this method to predict geographical location we used spherical geometry and took the average positions on an idealised sphere of Earth radius, using standard geodesic distance calculations. The results can be measured in terms of its great-circle distance from the true location (capital city) of the piece under consideration.

Finding the geodesic midpoint of the k nearest neighbours: with λ as latitude and ϕ as longitude (both in radians), convert to cartesian coordinates on a unit sphere:²

$$x = \cos(\lambda)\cos(\phi) \quad (1)$$

$$y = \cos(\lambda)\sin(\phi) \quad (2)$$

$$z = \sin(\lambda) \quad (3)$$

Take means of the nearest neighbour points per dimension $\bar{x}, \bar{y}, \bar{z}$. Find the longitude $\bar{\phi}$ of the midpoint:

$$\bar{\phi} = \arctan\left(\frac{\bar{y}}{\bar{x}}\right) \quad (4)$$

² In practice a four-quadrant inverse tangent function is necessary for equations 4-5 to cover all cases. The function we used is known as atan2 or arctan2 in most programming languages.

Find the latitude $\bar{\lambda}$ of the midpoint

$$\bar{\lambda} = \arctan\left(\frac{\bar{z}}{\sqrt{\bar{x}^2 + \bar{y}^2}}\right) \quad (5)$$

2.5 Utilising *a priori* Background Knowledge

We investigated the utility of using *a priori* knowledge to improve the predictions.

Land and Sea

The first piece of knowledge used is that music is produced on land. To utilise this we applied the NASA LandMask projected onto the idealised sphere of the Earth. This gives the terrain type for each square *degree* latitude by longitude. This varies in true size from about 110km wide to exactly 0 at each pole for longitude, whereas the latitude separation at 1 degree remains roughly 69km apart, excepting differences for the oblate spheroidal shape of the Earth. Because the landmask is given in latitude-longitude squares, a line-drawing algorithm must be employed to traverse a spherical distance across the earth, ensuring that the great-circle calculation is performed between each step, so that the correct next square is chosen. The total land contained in the path of error is weighted against the total water coverage. Different weightings were investigated but we settled with the simplest option: land 1:0 water such that only the land part counts. The reasoning behind this is that though human migration is slow across land, it is comparatively very fast across water. For example, Brazilian music is close musically to Portuguese music because of migration patterns despite the size of the Atlantic Ocean. Different weightings will be tested in future.

Population Centroids and Population Density

The first change is to recenter each country to its population centroid, which were collated from the GPWv3 per administrative area data [20] scaled up to per country via the method detailed by Greg Hamerly (<http://cs.ecs.baylor.edu/hamerly/>). The same dataset also provides population density grids at several resolutions. We chose to use the coarsest – per square degree – since it matches what we have for the land mask and is thus more easily comparable. Population density (as opposed to count) is chosen because it avoids the problem of varying size of square degree on the earth’s surface owing to the diminishing distance between longitudinal degrees as either pole is approached, as explained above.

The algorithm for applying this mask is as follows:

1. The k nearest neighbours are determined based upon musical features.
2. For the group of nearest neighbours, we find the geodesic midpoint
3. The geodesic distance to the furthest neighbour from that midpoint gives us a radius of a geodesic circle which contains all the neighbours.
4. We calculate a new midpoint for this circle which is weighted by the population density in each of the points, at a resolution of one square degree.

The weighted midpoint is found as in equations 1-5 but each dimensional mean is calculated from weighted coordinates such that

$$\bar{x} = \sum \frac{x_i \rho}{k} \quad (6)$$

$$\bar{y} = \sum \frac{y_i \rho}{k} \quad (7)$$

$$\bar{z} = \sum \frac{z_i \rho}{k} \quad (8)$$

where ρ is the population density for the square degree at the point (λ, ϕ) and k is the number of neighbours.

3 Results

3.1 kNN Performance

Using the default MARSYAS features the best predictive performance we achieved was a 2,827km median distance, and a 2,125km median land distance from the true position. This was achieved with $k=10$.

This result (like all above results presented) is significant at $P\text{-value} < 0.001$ (see below). We chose the median as our main statistic, rather than the mean, as it is more robust to outliers. The results for the means were also each time significant at $P\text{-value} < 0.001$.

Table 2 shows the breakdown of results varying k (nearest neighbours), the feature set (default or default with chromatic features added, 68 and 116 features respectively), mapping is whether the land measure is used to find the closest landmass (spherical means a pure distance measure using spherical geometry, population is as spherical but weighted by population) and median is the median distance from the true answer for each combination.

Whilst demonstrably better than random this result could be improved upon. What we show here is that even with one of the simplest possible algorithms the information contained in the features is enough to indicate some geographic information. The land proportion is, of course always smaller than the total distance, but we argued that this is a fairer measure of error. This indicates that some measure of population distribution would be useful *a priori*.

It is interesting that for land measures the $k=2$ method performed best before addition of chromatic features, but the $k=10$ version performed best with the extra features. We hypothesise that for very similar music the chromatic features do not aid the similarity measure already gained by timbral features, yet when more dissimilar music is encountered, chromatic features come into their own as a coarser measure of similarity. Since some countries were underrepresented to the point of having fewer than k member tracks, this follows. However, the simple spherical method medians show no such inversion.

Table 2. Median and mean distance from true location per k, featureset and algorithm

k	Features	Mapping	Median (km)	Mean
10	default	spherical	2869	3776
5	default	spherical	2913	3717
3	default	spherical	2975	3820
2	default	spherical	2980	3849
10	def+chrom	spherical	3013	3774
5	def+chrom	spherical	2999	3715
3	def+chrom	spherical	3012	3772
2	def+chrom	spherical	3269	3902
10	default	population	3591	4251
5	default	population	3042	3803
3	default	population	2987	3825
2	default	population	2933	3830
10	def+chrom	population	3572	4249
5	def+chrom	population	3140	3855
3	def+chrom	population	2997	3721
2	def+chrom	population	3107	3853
10	default	land	2125	2675
5	default	land	2024	2610
3	default	land	1966	2665
2	default	land	1850	2583
10	def+chrom	land	1506	2694
5	def+chrom	land	1550	2613
3	def+chrom	land	2087	2639
2	def+chrom	land	1996	2627

3.2 kNN with Population Distribution

Figures 2-3 compare the standard spherical approach against the population method.

Values for the mean performance were also calculated but are not shown. For lower k values the use of population distribution leads to an improvement to results. However, once k is greater than 3 the population method median degrades, and once it is greater than 4 the mean is similarly worsened. Experiments with higher k have noisier data since the additional neighbours are more likely to be from a different country. When more dissimilar music is encountered it is likely that instead of improving a result within a country the method selects more densely populated countries instead.

3.3 Statistical Significance

The determination of whether a music geographical prediction method is performing better than random is difficult using traditional statistical methods, because of the complicated geographical distribution of the music and countries.

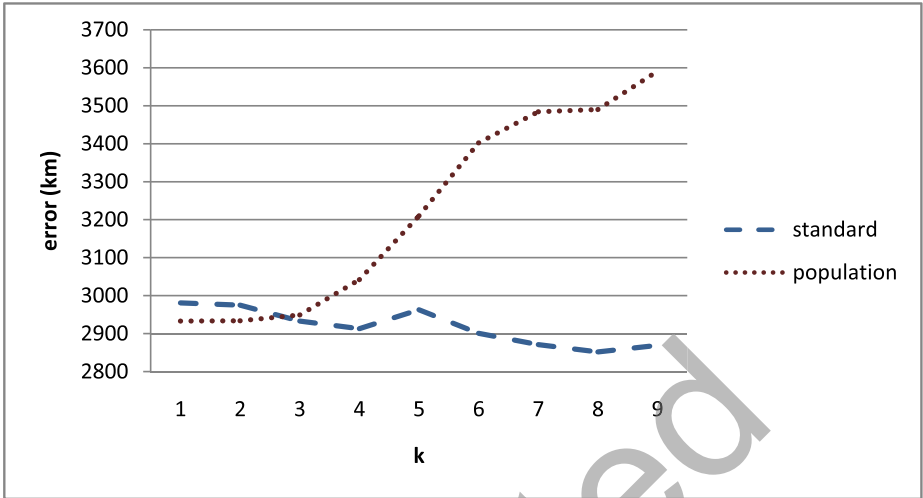


Fig. 2. Default features performance as median for all k, standard (spherical) measure and population measure

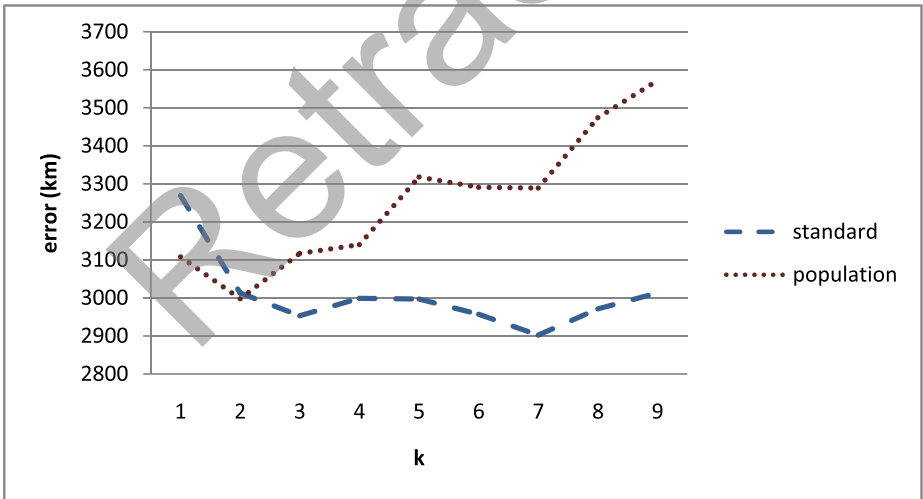


Fig. 3. Default + chromatic features performance as median for all k, standard (spherical) measure and population measure

Therefore, we used a computationally-intensive statistical method based on re-sampling, programming 1,000 random trials for each k and method combination, and measuring our distance means against the distribution of random means to ensure the statistical significance of our results. An example distribution can be seen in Figure 4.

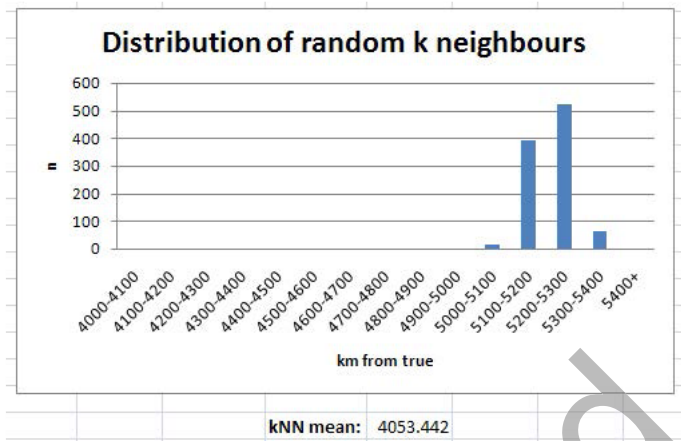


Fig. 4. Example random distribution. Our means were all so far left of the distribution that they were significant at $p < 0.001$.

3.4 Performance by Country

The algorithm performed much better on some countries than others – even with the same number of tracks available, suggesting that some countries are more musically diverse than others. An additional problem is the relative size of countries affecting the level of precision required, for example, Russian music is much more geographically diverse than that of Croatia.

Figure 5 (courtesy of Google Maps) shows half of the estimates (for clarity and to avoid overloading of positions) for Greek music, taken evenly from the distribution for Greece. From this it is clear that the prediction range for a country can be quite tightly distributed – the furthest estimate is 3,652km but there is a close cluster central to the image that reflects the more general skew in the distribution of estimates for all countries.

4 Discussion and Future Work

We have introduced the problem of predicting the geographical origin of music. There is great scope for further research and improvement in prediction performance.

With a larger corpus with both more tracks from each country, and more countries represented, the prediction results will inevitably improve. If one is only interested in predicting location (as opposed to understanding the historical/pre-historical reasons for musical distribution) the problem is in many ways similar to statistical analysis of text, where organisations such as Google have now indexed so many pieces of text that they can solve many problems that were once thought to require solving deep problems in computational linguistics.



Fig. 5. Greek music: predictions of location

More geographical information could be utilised. It would be better to have access to the exact location of the origin of the music, rather than just the capital or population centroid, as most countries have strong regional variations in style. Some cultures change drastically over small areas, some are unchanged over large expanses, and this needs to be learnt by the prediction method.

Better representations. The music could be better represented for computational analysis. It is a truism within machine learning that the hard part is getting the features correct, and with the correct features almost any learning algorithm will work. For example, extra features such as the fine chromatic feature used by Gomez *et al.* could be applied. [11] It would also be useful to explore the possibility of filtering and pre-selection of descriptors.

Many other forms of machine learning could be applied: neural-networks, support vector machines, decision trees, etc. It is also generally possible to improve the performance of individual methods by combining them together to form consensus predictions [1].

It is difficult to know how good our prediction results are as there are no previously published related comparisons. It would therefore be very interesting to compare the results of the machine learning programs with that of human performance in predicting musical origin.

The motivation for this work is to better understand the diversity of world music. To do this we have to go beyond just the prediction of location, but to analyse what features of the music are responsible for these predictions. This is now the main focus of our research.

References

1. Duda, R.O., Hart, P.E.: *Pattern Classification and Scene Analysis*. Wiley, New York (1973)
2. Tzanetakis, G., Cook, P.: Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10(5), 293–302 (2002)
3. Grachten, M., Schedl, M., Pohle, T., Widmer, G.: The ISMIR Cloud: a Decade of ISMIR Conferences at Your Fingertips. In: *International Symposium of Music Information Retrieval*, pp. 63–68 (2009)
4. Laurier, C., Grivolla, J., Herrera, P.: Multimodal Music Mood Classification Using Audio and Lyrics. In: *Seventh International Conference on Machine Learning and Applications*, pp. 688–693 (2008)
5. Widmer, G., Dixon, S., Knees, S., Pampalk, E., Pohle, T.: From sound to sense via feature extraction and machine learning: Deriving high-level descriptors for characterising music. In: *Sound to Sense: Sense to Sound: A State-of-the-Art, S2S2 Consortium*, Florence (2005)
6. Turnbull, D.R., Barrington, L., Lanckriet, G., Yazdani, M.: Combining audio content and social context for semantic music discovery. In: *SIGIR 2009*, pp. 387–394. ACM, New York (2009)
7. Knees, P., Pohle, T., Schedl, M., Widmer, G.: A music search engine built upon audio-based and web-based similarity measures. In: *SIGIR 2007*, pp. 447–454. ACM, New York (2007)
8. Mckay, C., Fujinaga, I.: Automatic genre classification using large high-level musical feature sets. In: *Int. Conf. on Music Information Retrieval*, pp. 525–530 (2004)
9. Mckinney, M., Breebaart, J.: Features for Audio and Music Classification. In: *International Symposium on Music Information Retrieval*, pp. 151–158 (2003)
10. Liu, Y., Xiang, Q., Wang, Y., Cai, L.: Cultural style based music classification of audio signals. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 57–60 (2009)
11. Gomez, E., Herrera, P.: Comparative Analysis of Music Recordings from Western and Non-Western traditions by Automatic Tonal Feature Extraction. *Empirical Musicology Review* 3(3), 140–156 (2008)
12. Gomez, E., Haro, M., Herrera, P.: Music and Geography: Content Description of Musical Audio from Different Parts of the World. In: *International Symposium on Music Information Retrieval*, pp. 753–758 (2009)
13. Tzanetakis, G., Kapur, A., Schloss, A., Wright, M.: Computational Ethnomusicology. *Journal of Interdisciplinary Music Studies* 1(2), 1–24 (2007)
14. Lichte, W.H.: Attributes of Complex Tones. *Journal of Experimental Psychology* 28, 455–480 (1941)
15. Grey, J.M.: Multidimensional perceptual scaling of musical timbres. *Journal of the Acoustic Society of America* 61(5), 1270–1277 (1977)
16. Ripley, B.D.: *Spatial Statistics*. Wiley, New York (2004)

17. Wu, Y., Takatsuka, M.: The Geodesic Self-Organizing Map and its Error Analysis. In: Twenty-Eighth Australasian Conference on Computer Science, vol. 38, pp. 343–351 (2005)
18. Govaerts, S., Duval, E.: A Web-based Approach to Determine the Origin of an Artist. In: International Symposium on Music Information Retrieval, pp. 261–266 (2009)
19. Tzanetakis, G., Cook, P.: MARSYAS: a framework for audio analysis. *Organised Sound* 4(3), 169–175 (2000)
20. Gridded Population of the World, Version 3 (GPWv3): SEDAC, Columbia University, <http://sedac.ciesin.columbia.edu/gpw>

Retracted

Pair-Based Object-Driven Action Rules

Ayman Hajja¹, Alicja A. Wieczorkowska²,
Zbigniew W. Ras^{1,3}, and Ryszard Gubrynowicz²

¹ University of North Carolina, Dept. of Computer Science,
9201 University City Blvd., Charlotte, NC 28223, USA

² Polish-Japanese Institute of Information Technology,
Koszykowa 86, 02-008 Warsaw, Poland

³ Warsaw University of Technology, Institute of Computer Science,
Nowowiejska 15/19, 00-665 Warsaw, Poland
{ahajja, ras}@uncc.edu, alicja@poljap.edu.pl,
rgubryn@pjwstk.edu.pl

Abstract. Action rules, as proposed by Raś and Wieczorkowska in [11], can be defined as actionable tasks that describe possible transitions of objects from one state to another with respect to a distinguished attribute. Recently, a new specialized case of action rules, namely object-driven action rules, has been introduced by Ayman et al. in [4]. Object-driven action rules are action rules that are extracted from information systems with temporal and object-based nature. By object-based nature, we refer to systems that contain multiple observations for each object. A typical example of an object-based system would be a system of patients recording multiple visits; each patient is considered a distinct object. In this paper, we will further investigate the concept of object-driven action rules by proposing a new pair-based way of examining object-driven systems, which we believe is more intuitive for temporal and object-driven systems. The focus of this paper will be on our proposed pair-based approach, along with the modifications required to extract action rules and calculate their properties.

Keywords: action rules, object-driven action rules, temporal data, hypernasality.

1 Introduction and Background

Action Rules, as proposed by Raś and Wieczorkowska in [11], describe possible transitions of objects from one state to another with respect to a specific attribute, called the decision attribute. Action rules have been successfully applied in many domain areas including business [11], medical diagnosis and treatment [16], [17], and music automatic indexing and retrieval [6], [9].

System users are mainly interested in actionable tasks that trigger state transitions that move objects from a less desirable state to a more desirable state; action rules specify the actions needed to be taken to reach that desired goal.

In this paper, we will introduce a novel approach for extracting action rules from object-driven and temporal systems. There has been considerable research on the varied methodologies for extracting action rules from information systems [5,7,13]. However, adapted action rules systems that are designed for datasets with particular nature is to some extent new. In [4], Ayman et al. proposed an adapted action rules extraction method for information systems of temporal and object-based nature. In this work, we will extend the approach presented in [4].

2 Object-Driven Action Rules Revisited

The drive behind the introduction of object-driven action rules in [4] was to bring forth an adapted approach to extract action rules from systems of temporal and object-driven nature.

In [4], we proposed an object-independency assumption that suggests extracting patterns from subsystems defined by unique objects, and then aggregating similar patterns amongst all objects. The motivation behind this approach is based on the fact that same-object observations share similar features that are not shared with other objects, and these features are possibly not explicitly included in our dataset. Therefore, by individualizing objects prior to calculating action rules, variance is reduced, and over-fitting is potentially avoided. In addition to the object-independency assumption, temporal information is exploited by taking into account only the state transitions that occurred in the valid direction.

In this section, we will start with providing the necessary background concerning action rules. A complete description of the motivation and the concept of the pair-based approach will be presented next, along with the modifications required to extract object-driven action rules and calculate their properties.

2.1 Action Rules

The notion of action rules was first proposed by Z. W. Raś and A. Wieczorkowska in [11]. Action rules describe possible transition of objects from one state to another with respect to a specific attribute, called the decision attribute. The goal of action rules is to provide system users with actionable tasks that can be directly applied to objects listed in information systems to reach a desired goal.

Let $S = (X, A, V)$ denotes an information system [8], where:

1. X is a nonempty, finite set of instances (objects),
2. A is a nonempty, finite set of attributes;
 $a : X \rightarrow V_a$ is a function for any $a \in A$, where V_a is called the domain of a ,
3. $V = \bigcup \{V_a : a \in A\}$.

By a decision table, we mean an information system that makes a clear explicit distinction between attributes in A , and will therefore label each attribute as either a *decision attribute*, or a non-decision attribute, called *condition attribute*.

The decision attribute(s), which normally but not necessarily is a single attribute, is the attribute that we are interested in most. For system users, the eventual goal would be to change the decision attribute from less desirable to more desirable state. For example, a company would be interested in moving clients' states of loyalty from lower to higher.

All non-decision, or condition, attributes are further partitioned into two mutually exclusive sets; the first one is the *stable* attributes set, and the second one is the *flexible* attributes set. By stable attributes set we mean the set that contains attributes that we have no control over; their values cannot be changed by the users of our system. An example of a *stable* attribute is the place where the person was born. On the other hand, values of *flexible* attributes can be influenced and changed; an example of a *flexible* attribute is the patient's prescribed medications. In this paper, A_{St} , A_{Fl} , and $\{d\}$ will represent the set of stable attributes, the set of flexible attributes, and the decision attribute, respectively. Hence, the set of attributes A can be redefined as $A = A_{St} \cup A_{Fl} \cup \{d\}$.

An *atomic action set* is an expression that defines a change of state for a single distinct attribute. For example, $(a, a_1 \rightarrow a_2)$ is an atomic action set which defines a change of state for the attribute a from a_1 to a_2 , where $a_1, a_2 \in V_a$. Clearly, in this case, the attribute a is a flexible attribute, since it changes its state from a_1 to a_2 . In the case when there is no change, we omit the right arrow sign, so for example, (b, b_1) means that the value of attribute b remains b_1 , where $b_1 \in V_b$.

An *action set* is defined as follows:

1. If t is an atomic action set, then t is an action set.
2. If t_1, t_2 are action sets and \wedge is a 2-argument functor called composition, then $t_1 \wedge t_2$ is a candidate action set.
3. If t is a candidate action set and for any two atomic action sets $(a, a_1 \rightarrow a_2), (b, b_1 \rightarrow b_2)$ contained in t we have $a \neq b$, then t is an action set.
4. No other sets are called action sets.

The *domain* $Dom(t)$ of an action set t is the set of attributes of all atomic action sets contained in t . For example, $t = (a, a_1 \rightarrow a_2) \wedge (b, b_1)$ is an action set that consists of two atomic action sets, namely $(a, a_1 \rightarrow a_2)$ and (b, b_1) . Therefore, the domain of t is $\{a, b\}$.

Action rules are expressions that take the following form: $r = [t_1 \Rightarrow t_2]$, where t_1, t_2 are action sets. The interpretation of the action rule r is that by applying the action set t_1 , we would get, as a result, the changes of states in action set t_2 . We also assume that $Dom(t_1) \cup Dom(t_2) \subseteq A$, and $Dom(t_1) \cap Dom(t_2) = \emptyset$.

For example, $r = [(a, a_1 \rightarrow a_2) \wedge (b, b_2)] \Rightarrow (d, d_1 \rightarrow d_2)$ means that by changing the state of the attribute a from a_1 to a_2 , and by keeping the state of the attribute b as b_2 , we would observe a change in the attribute d from the state d_1 to d_2 , where d is commonly referred to as the *decision attribute*.

Standard interpretation N_s of action sets in S is defined as follows:

1. If $(a, a_1 \rightarrow a_2)$ is an atomic action set, then
 $N_s((a, a_1 \rightarrow a_2)) = [\{x \in X : a(x) = a_1\}, \{x \in X : a(x) = a_2\}]$.
2. If $t_1 = (a, a_1 \rightarrow a_2) \wedge t$ and $N_s(t) = [Y_1, Y_2]$, then
 $N_s(t_1) = [Y_1 \cap \{x \in X : a(x) = a_1\}, Y_2 \cap \{x \in X : a(x) = a_2\}]$.

Let us define $[Y_1, Y_2] \cap [Z_1, Z_2]$ as $[Y_1 \cap Z_1, Y_2 \cap Z_2]$ and assume that $N_s(t_1) = [Y_1, Y_2]$ and $N_s(t_2) = [Z_1, Z_2]$. Then, $N_s(t_1 \wedge t_2) = N_s(t_1) \cap N_s(t_2)$.

If t is an action set and $N_s(t) = [Y_1, Y_2]$, then the support of t in S is defined as $supp(t) = \min\{card(Y_1), card(Y_2)\}$.

Let $r = [t_1 \Rightarrow t_2]$ be an action rule, $supp(t_1) > 0$, $N_s(t_1) = [Y_1, Y_2]$, and $N_s(t_2) = [Z_1, Z_2]$. Support $supp(r)$ and confidence $conf(r)$ of r are defined as:

$$supp(r) = \min\{card(Y_1 \cap Z_1), card(Y_2 \cap Z_2)\},$$

$$conf(r) = \left[\frac{card(Y_1 \cap Z_1)}{card(Y_1)} \right] * \left[\frac{card(Y_2 \cap Z_2)}{card(Y_2)} \right].$$

2.2 Action Rules Extraction

There have been a considerable amount of research on various methodologies for extracting action rules from information systems [5,7,13]. In general however, we can categorize all methodologies into two groups; the first one being when classification rules are required prior to the construction of action rules [12], [15], and the second, more recent approach, being when action rules are directly extracted from an information system [10].

To extract pair-based object-driven action rules, we used the algorithm described in [10]. The idea of the algorithm is to start by constructing all possible action sets that have occurred more than a pre-defined number, called the minimum support. Then, in accordance to our desired change in the decision attribute, action rules are formed.

Let t_a be an action set, where $N_s(t_a) = [Y_1, Y_2]$ and $a \in A$. We say that t_a is a *frequent action set* [10] if $card(Y_1) \geq \lambda_1$ and $card(Y_2) \geq \lambda_1$, where λ_1 is the minimum support. Another way of interpreting the frequent action sets would be that all frequent action sets have support greater than or equal to the minimum support λ_1 . By specifying λ_1 , we make sure that the extracted action rules have support greater than or equal to the minimum support λ_1 . Algorithm presented below is similar to [1].

To extract action rules, we start with generating atomic action sets that have support greater than or equal to the minimum support value λ_1 pre-defined by the user; we will refer to this set as *1-element frequent action set*. The term *frequent* will be used to indicate that an action set has support greater than or equal to the minimum support, and the term *k-element* will be used to indicate the number of elements (or atomic action terms) in an action set.

Both *frequent atomic action sets* and *1-element frequent action set* refer to exactly the same set, since from the definition of action sets, they consist of only one element.

After generating all frequent atomic action sets, we undertake the following two-step process initially for $k = 1$:

1. **Merge step:** Merge pairs (t_1, t_2) of k -element action sets into all $(k + 1)$ -element candidate action sets.
2. **Delete step:** Delete all $(k + 1)$ -element candidate action sets that are either not action sets, or contain a non-frequent k -element action set, or that have support less than the minimum support λ_1 .

We keep iterating the above two steps until we cannot generate new frequent action sets anymore. At this point, we have generated all $(k + 1)$ -element frequent action sets, which will allow us to generate action rules that are guaranteed to have support greater than or equal to the minimum support λ_1 . Last step is to further filter the desired action rules based on their confidence, where we only consider action rules with confidence greater than or equal to a pre-defined minimum confidence λ_2 . For example, from the frequent action set $t_1 = (a, a_1 \rightarrow a_2) \wedge (d, d_1 \rightarrow d_2)$, we can generate the following two action rules:

1. $r_1 = [(a, a_1 \rightarrow a_2) \Rightarrow (d, d_1 \rightarrow d_2)]$.
2. $r_2 = [(d, d_1 \rightarrow d_2) \Rightarrow (a, a_1 \rightarrow a_2)]$.

where both r_1 and r_2 have support greater than or equal to the minimum support λ_1 . However, we will only be interested in specific changes of the decision attribute, e.g. in changing the decision attribute d from state d_1 to d_2 . Therefore, we will only consider r_1 .

2.3 Temporal Constraint and Pair-Based Approach

As defined previously, temporal object-driven datasets consist of numerous unique objects, where each object is comprised of multiple instances that have assigned corresponding timestamps. Previously in [4], the object p based standard interpretation of an action set $t = (a, a_1 \rightarrow a_2)$ was defined as the pair of two sets $[Y_1, Y_2]$ where Y_1 is the set of instances of the object p that satisfy the left side, or condition side, of the action set, and Y_2 is the set of instances of the object p that satisfy the right side, or decision side, of the action set, with the addition that for every instance in Y_1 , there exist a matching instance in Y_2 that occurred after it. This definition resembles the definition of standard interpretation for classical action rules while restricting valid transitions to only one direction. In this paper however, we argue that the nature of the object-driven temporal dataset allows us to redefine the standard interpretation into a more intuitive pair-based structure which we believe is more appropriate for object-driven temporal systems.

Let us first assume that $I_s(p)$ denotes the set of all instances of the object p in an information system S . Also, the relation $\angle \subseteq I_s(p)$ is defined as:
 $(p_1, p_2) \in \angle$ iff p_2 has occurred after p_1 .

The pair-based standard interpretation $N_{s(p)}^{TC}$ in $S = (X, A, V)$ for an object p is redefined as:

1. If $(a, a_1 \rightarrow a_2)$ is an atomic action set, then
 $N_{s(p)}^{TC}((a, a_1 \rightarrow a_2)) = \{(p_1, p_2) \in \angle : a(p_1) = a_1, a(p_2) = a_2\}$
 where $\angle \subset I_s(p)$.
2. If $t_1 = (a, a_1 \rightarrow a_2) \wedge$ and $N_{s(p)}^{TC}(t) = Y_1$, then
 $N_{s(p)}^{TC}(t_1) = Y_1 \cap \{(p_1, p_2) \in \angle : a(p_1) = a_1, a(p_2) = p_2\}$
 where $\angle \subset I_s(p)$.

In other words, our standard interpretation will consist of all valid transitions from the left side of an action set to the right side, represented as pairs. The motivation behind this new interpretation is due to the fact that the instances within one object are not observed independently, which will allow us to relax the minimum assumption previously used. Our object-independency assumption states that the whole system consists of multiple independent subsystems, each one marked by a unique object. Although it confines the system to extract action rules only from instances of the same object, it provides more flexibility to be applied within unique objects.

If t is an action set and $N_{s(p)}^{TC}(t) = Y_1$, then the support of t in S is defined as: $supp_p^{TC} = card(Y_1)$.

Let $r = [t_1 \Rightarrow t_2]$ be an action rule, where $N_{s(p)}^{TC}(t_1) = Y_1, N_{s(p)}^{TC}(t_2) = Y_2$. The p^{th} support $supp_p^{TC}(r)$ and the p^{th} confidence $conf_p^{TC}(r)$ of r are defined as follows:

$$supp_p^{TC}(r) = card(Y_1 \cap Y_2),$$

$$conf_p^{TC}(r) = \left[\frac{card(Y_1 \cap Y_2)}{card(Y_d)} \right].$$

To define Y_d , let us first assume that $\zeta(Y)$ denotes the set of first elements of the set of pairs Y . For instance, if $Y = \{(p_1, p_3), (p_3, p_4), (p_1, p_2)\}$, then $\zeta(Y) = \{p_1, p_3\}$. We define $Y_d = \{(p_1, p_2) \in Y_1 : p_1 \in \zeta(Y_2)\}$. The interpretation of this definition means that to calculate the confidence of the action rule $r = (a, a_1 \rightarrow a_2) \Rightarrow (d, d_1 \rightarrow d_2)$, the pairs that we are considering are the ones that have first elements that satisfy $a = a_1$ and $d = d_1$. Since the transition from a_1 to a_2 could possibly trigger other states of decision attribute d , we are only interested in the states of our action rule.

After all object-driven action rules are extracted and their p^{th} support and p^{th} confidence are computed for all $p \in X$, we then calculate their total support $supp_X^{TC}(r)$ (called support) and total confidence $conf_X^{TC}(r)$ (called confidence) following the definition below:

Table 1. Information System S

	<i>objectID</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
x_0	1	a_1	b_1	c_1	d_1
x_1	1	a_2	b_1	c_1	d_1
x_2	1	a_2	b_2	c_2	d_2
x_3	1	a_1	b_2	c_1	d_1
x_4	1	a_2	b_1	c_1	d_2
x_5	2	a_1	b_2	c_1	d_2
x_6	2	a_2	b_1	c_1	d_1
x_7	3	a_1	b_2	c_1	d_1
x_8	3	a_2	b_2	c_1	d_2
x_9	3	a_1	b_1	c_1	d_1
x_{10}	3	a_2	b_1	c_1	d_2

$$supp_X^{TC}(r) = \sum_{p \in X} supp_p^{TC}(r),$$

$$conf_X^{TC}(r) = \sum_{p \in X} \left(\frac{supp_p^{TC}(r) * conf_p^{TC}(r)}{supp_X^{TC}(r)} \right).$$

If the denominator in the formula for calculating confidence is equal to zero, then the confidence is equal to zero by definition.

Example to Demonstrate Pair-Based Object Driven Support and Confidence: Here, we provide an example to demonstrate how we calculate the support and the confidence for the whole system S shown in Table 1. We assume that for all 3 objects in X their instances x_i , where $1 \leq i \leq 10$, have chronological order.

Referring to our information system S shown in Table 1, we calculate the support $supp_X^{TC}(r)$ and the confidence $conf_X^{TC}$ for the following rule:

$$r = [(a_1 \rightarrow a_2) \wedge (c, c_1) \Rightarrow (d, d_1 \rightarrow d_2)].$$

We first calculate the p^{th} standard interpretation for each object p (e.g. for a patient) for both the condition and the decision parts in the action rule r :

$$N_{s(1)}^{TC}((a, a_1 \rightarrow a_2) \wedge (c, c_1)) = \{(x_0, x_1), (x_0, x_2), (x_0, x_4), (x_3, x_4)\} \cap \\ \{(x_0, x_1), (x_0, x_3), (x_0, x_4), (x_1, x_3), (x_1, x_4), (x_3, x_4)\} \\ = \{(x_0, x_1), (x_0, x_4), (x_3, x_4)\},$$

$$N_{s(1)}^{TC}(d, d_1 \rightarrow d_2) = \{(x_0, x_2), (x_0, x_4), (x_1, x_2), (x_1, x_4), (x_3, x_4)\},$$

$$N_{s(2)}^{TC}((a, a_1 \rightarrow a_2) \wedge (c, c_1)) = \{(x_5, x_6)\},$$

$$N_{s(2)}^{TC}(d, d_1 \rightarrow d_2) = \phi ,$$

$$\begin{aligned} N_{s(3)}^{TC}((a, a_1, \rightarrow a_2) \wedge (c, c_1)) &= \{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\} \cap \\ &\{(x_7, x_8), (x_7, x_9), (x_7, x_{10}), (x_8, x_9), (x_8, x_{10}), (x_9, x_{10})\} \\ &= \{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\} , \end{aligned}$$

$$N_{s(3)}^{TC}(d, d_1 \rightarrow d_2) = \{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\} .$$

Using the temporal constraint and the object-driven assumptions, the pair-based support and confidence for each object is calculated as follows:

$$sup_1^{TC}(r) = card(\{(x_0, x_4), (x_3, x_4)\}) = 2 ,$$

$$conf_1^{TC}(r) = \left[\frac{card(\{(x_0, x_4), (x_3, x_4)\})}{card(\{(x_0, x_1), (x_0, x_4), (x_3, x_4)\})} \right] = \frac{2}{3} ,$$

$$sup_2^{TC}(r) = card(\phi) = 0 ,$$

$$conf_2^{TC}(r) = 0 ,$$

$$sup_3^{TC}(r) = card(\{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\}) = 3 ,$$

$$conf_3^{TC}(r) = \left[\frac{card(\{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\})}{card(\{(x_7, x_8), (x_7, x_{10}), (x_9, x_{10})\})} \right] = \frac{3}{3} = 1 .$$

Now we calculate the overall support and confidence for the whole system:

$$sup_X^{TC}(r) = 5, \quad conf_X^{TC}(r) = \left(\frac{2 * \frac{2}{3}}{5} \right) + \left(\frac{3 * 1}{5} \right) = \frac{4.33}{5} = .87 .$$

3 Experimental Data: Hypernasality Data Set

Distortions of the velopharyngeal closure, resulting in speech hypernasality or hyponasality, may cause speech disorders in children [3]. The patient's nasopharynx disorders have been examined in the Children's Memorial Health Institute in Warsaw for many years. The gathered data also include general information on the patient's condition if it can be of importance, e.g. cerebral palsy, neurology, or myopathy. This way a reach collection of complex data describing hypernasality was gathered, in close cooperation with one of the co-authors, Prof. Ryszard Gubrynowicz, who is a speech scientist and expert in this area; the data were collected when he was working in the Children's Memorial Health Institute.

3.1 Velum Malfunction in Children

Hypernasality can be examined by means of Czermak's mirror test of nasal air escape, see Figure 1. The child is asked to repeat several times a syllable composed of a plosive consonant and an open vowel, e.g. /pa/-/pa/-/pa/, and the sizes of the fogging circles appearing on the mirror are rated on 4-point scale, from 0 (no hypernasality) to 3 (most severe hypernasality). Therefore, *Czermak's mirror test* was used as a decision attribute in the nasality data set. All attributes, representing various medical conditions in the examined children, are listed in Table 2. More explanations about these attributes are given below.

Each patient was examined several times. Personal data were recorded (first name and last name, sex), and for each examination the age of the child was marked. Personal data were removed before further processing, and replaced with ID data, representing the patient's ID combined with the sequential number of this patient's visit.

During each visit, the articulation of selected vowels and consonants was recorded, and the recording date was marked (*recording date* attribute). The data stored in columns marked as *diagnosis* and *diagnosis2* describe patient's condition related to nasality; only one diagnosis is stored in each of these columns, so *diagnosis2* represents additional diagnosis, if there is more than one. The following diagnoses are described in these columns: R - cleft, RP - cleft palate, OR - after cleft palate surgery, WKP - congenital short velum, NO - hypernasality, NZ - hyponasality, BR - no diagnosis, PRP - submucous cleft palate, AT - after tonsillectomy, DKP - quite short palate, RJ - cleft uvula, III - hypertrophy of adenoids and possibly palatine tonsils, MP - hypertrophy of palatine tonsils, MPDz - cerebral palsy, AD - after adenotomy, ADT - after adenotonsillectomy, UK - larynx after injury/trauma, NS - hypoacusis, ORM - retarded speech development, NEU - neurology, ONR - after neurological surgery. If NO (hypernasality) is diagnosed and marked in the column *diagnosis*, it represents the most severe case of hypernasality. The numbers 0–3 in *diagnosis2* refer to sleep apnoea, i.e. temporary cessation of respiration during sleep. 0 means no apnoea, 3 - very often. Sleep apnoea is also represented as a separate attribute, but the values assessed for the same patient may differ significantly, so they were kept in both columns. Generally, physicians may differ in their opinions, this is why we must be prepared to deal with some inconsistencies in the data. More of



Fig. 1. Czermak's mirror fogging test, rating the degree of the patient's nasal air escape on a 4-point scale: none = 0; small = 1, medium = 2, large = 3 [3]

Table 2. Attributes in the Hypernasality Data Set. Expansions of acronyms are given in the text, see Section 3.1.

Attribute	Description
<i>ID</i>	Patient's ID, with the sequential number of his/her visit
<i>age</i>	Age [years, months]
<i>sex</i>	Sex {M, F}
<i>recording date</i>	Recording Date [yyyy.mm.dd]
<i>diagnosis</i>	Diagnosis {AD, ADT, AT, BR, III, myopathy, MPDz, NEU, NO, ONR, OR, ORM, RJ, RP, UK, WKP}
<i>comments</i>	Comments, details of the diagnosis
<i>diagnosis2</i>	Diagnosis {0, 1, 2, 3, DKP, RJ, WKP}
<i>sleep apnoea</i>	Sleep apnoea {0, 1, 2, 3}
<i>tonsils</i>	Hypertrophy of adenoids and possibly palatine tonsils {0, 1, 2, 3}
<i>Czermak's mirror test</i> - decision attribute	Mirror-fogging test {0, 1, 2, 3}
<i>yeaoui</i>	Measure of nasalization for vowels /I, e, a, o, u, i/ [0, 100]
<i>i - long</i>	Measure of nasalization for vowel /i/-long [0, 100]
<i>bdg</i>	Measure of nasalization for high pressure consonants /b, d, g/ [0, 100]
<i>motility</i>	Motility of the soft palate [0, 12]
<i>difference level F1 - F2</i>	The difference level of 1 st & 2 nd formant measured for /i/-long [-14, 26]

diagnostic details are given in the column *comments*, but these comments are not taken into account in the current version of our action rule software.

Other physical conditions recorded in the database include the degree of hypertrophy of adenoids and possibly palatine tonsils, and the degree of motility of the soft palate, represented as *tonsils* and *motility* attributes. The assessment of the patient's recorded speech is represented in the following attributes: *yeaoui* (vowels /I, e, a, o, u, i/ - a sequence of short vowel sounds spoken in isolation), *i - long* (long vowel /i/ - vowel of sustained phonation), and *bdg* (high pressure consonants /b, d, g/); SAMPA coding of phonetic alphabet is used [14]. These attributes describe the measure of nasalization (coefficient of nasalization), calculated from the analysis of mouth and nose signals (separately recorded), as the ratio of the nose signal level to the sum of the level of the nose and mouth signals for the phonemes indicated in each attribute. *difference level F1 - F2* describes the vocal tract's first 2 resonances as the difference level of the 1st and the 2nd formant, measured for /i/-long.

The best diagnosis we are interested in is when the parameters' values are in normal ranges. Our decision attribute is Czermak's mirror test, so its values are most important in our research. The most desired value of our decision attribute is when it is equal to 0. The diagnosis is worse when Czermak's test value equals 2, next worse case is when Czermak's test value equals 3, and this is the most severe case. The lower the Czermak's test value, the better

the diagnosis is. Therefore, we are interested in action rules indicating how to decrease the Czermak's test value. The goal of our system is to find action rules which purpose is to provide hints referring to doctor's interventions. They show how values of certain attributes need to be changed (through various medical procedures, according to the physician's order), so the patient's condition will get improved.

4 Application of Object-Driven Action Rules

In this work, we derived a new set of attributes in accordance to [4]. In addition to our attributes shown in Table 2, for each of the following four attributes: *yeaoui*, *i - long*, *bdg*, and *motility*, two new attributes were derived, resulting in eight new attributes. The two derived attributes are the difference, and the rate of change for every two consecutive instances, which we calculated as follows:

1. The difference of values for *yeaoui*, *i - long*, *bdg* and *motility* for every two consecutive visits is calculated, thus constituting the following new attributes: $yeaoui_1$, $i_1 - long$, bdg_1 and $motility_1$. For example, the value of bdg_1 equals to the value of *bdg* for the $(k + 1)^{th}$ visit minus the value for the k^{th} visit.
2. The rate of change a_2 for every two consecutive visits is defined as:

$$a_2 = \arctan \left(\frac{a_1}{\text{age difference in months}} \right)$$

where a_1 is the difference of values of the attribute a for the two visits.

After calculating the derived attributes, we used the Rough Set Exploration System [2] to discretize our real-valued attributes wrt. our decision attribute. Next, our temporal object-driven action rule discovery system, presented in Section 2.3, was applied to the discretized data.

Our decision attribute *Czermak's mirror test* was not discretized. Moreover, when a physician could not decide between two neighboring Czermak's test values, an intermediate value was assigned. Therefore, the decision values are $\{0, .5, 1, 1.5, 2, 2.5, 3\}$.

5 Results and Discussion

In this section we show a sample of the results after running our proposed pair-based approach to extract object-driven action rules from temporal systems. We show that by using pair-based approach, not only we were able to extract a dramatically larger set of action rules, but also we were able to extract action rules that provide more dramatic decrease of patient severity than the rules extracted in [4]. For an action rule to be eligibly used on a patient, the pre-conditions of the action rule and the patient's current condition have to match, meaning that only a subset of our patients will benefit from each particular action rule. Having said that, using our pair-based approach to extract action rules

will generate a significant amount of action rules that can be appropriately used for various sets of patients.

Rule 1. $r_1 = (\text{difference level } F1-F2, \geq 9.5 \rightarrow [6.5, 9.5])$
 $\Rightarrow (\text{Czermak's mirror test}, 3 \rightarrow 2); \text{ supp}(r_1) = 2, \text{ conf}(r_1) = 100\% .$

This rule means that by decreasing the difference between the first two formants of the vocal tract for /i/ - long, we would notice a decent shift of the Czermak's mirror test, decreasing from 3 to 2. In [4], we extracted a similar action rule that also indicated the importance of *difference level F1-F2* attribute. However, this action rule is exclusive to the work described in this paper.

Rule 2. $r_2 = (i_2 - \text{long}, \geq 5.5 \rightarrow < 5.5) \Rightarrow (\text{Czermak's mirror test}, 3 \rightarrow 2);$
 $\text{supp}(r_2) = 3, \text{ conf}(r_2) = 66.7\% .$

This rule means that decreasing the value of *i - long* in a short period of time, since *i₂ - long* is defined as the rate of change, will result in a similar decrease of the Czermak's mirror test from 3 to 2. Again, this rule affirms the importance of the attribute *i - long*.

Rule 3. $r_3 = (i_2 - \text{long}, \geq 5.5 \rightarrow < 5.5) \wedge (\text{bdg}, \geq 8.5)$
 $\Rightarrow (\text{Czermak's mirror test}, 2.5 \rightarrow 2); \text{ supp}(r_3) = 2, \text{ conf}(r_3) = 100\% .$

This rule is similar to Rule 1. It confirms the effect of decreasing the rate of change of the nasalization measured for /i/ - long, but also adds an additional condition concerning the nasality of /bdg/, that is, this rule only applies to patients suffering from high nasality for /bdg/ (≥ 8.5).

Rule 4. $r_4 = (\text{tonsils}, < 2) \wedge (i_2 - \text{long}, \geq 5.5 \rightarrow < 5.5) \wedge (\text{motility}, [4.5, 5.5])$
 $\Rightarrow (\text{Czermak's mirror test}, 2 \rightarrow 1.5); \text{ supp}(r_4) = 2, \text{ conf}(r_4) = 100\% .$

This rule states that when a patient is experiencing a little hypertrophied adenoids and possibly palatine tonsils (*tonsils* < 2), we can slightly improve his condition from Czermak's mirror test 2 to 1.5 by decreasing the rate of change in /i/ - long, and if the motility of the soft palate does not change.

Rule 5. $r_5 = (\text{bdg}, \geq 8.5 \rightarrow [6.5, 8.5]) \Rightarrow (\text{Czermak's mirror test}, 1 \rightarrow .5);$
 $\text{supp}(r_5) = 3, \text{ conf}(r_5) = 66.7\% .$

This rule states that by only decreasing the nasality of /bdg/, we would be able to shift the patients' Czermak's mirror test state from 1 to .5.

Rule 6. $r_6 = (i - \text{long}, \geq 9.5 \rightarrow [2.5, 7.5]) \Rightarrow (\text{Czermak's mirror test}, 1 \rightarrow .5);$
 $\text{supp}(r_6) = 2, \text{ conf}(r_6) = 100\% .$

Although the support of this action rule is not high, the rule is rather interesting. It states that by decreasing only one attribute; /i/ - long, there is a 100% chance that the Czermak's mirror test will shift from 1 to .5.

Rule 7. $r_7 = (\text{motility}, < 3.5 \rightarrow [4.5, 5.5]) \wedge (\text{diagnosis}, \text{OR})$
 $\Rightarrow (\text{Czermak's mirror test}, 1 \rightarrow 0); \text{supp}(r_7) = 3, \text{conf}(r_7) = 100\% .$

This rule states that if a patient has gone through a cleft palate surgery (OR), then increasing the motility of the soft palate would significantly improve the patient's condition, to the level where the patient is entirely cured, which will result in shifting the Czermak's mirror test value from 1 to 0.

Rule 8. $r_8 = (i_2 - \text{long}, \geq 5.5 \rightarrow < 5.5) \Rightarrow (\text{Czermak's mirror test}, .5 \rightarrow 0);$
 $\text{supp}(r_8) = 7, \text{conf}(r_8) = 71\% .$

This rule has a relatively high support. It states that decreasing the rate of change of $i - \text{long}$ from greater than or equal to 5.5, to less than 5.5, will result in curing a light hypernasality.

Rule 9. $r_9 = (i_2 - \text{long}, \geq 5.5 \rightarrow < 5.5) \wedge (\text{sleep apnoea}, < 2)$
 $\Rightarrow (\text{Czermak's mirror test}, .5 \rightarrow 0); \text{supp}(r_9) = 6, \text{conf}(r_9) = 83\% .$

This rule is a similar, but more specific action rule than rule 8. By expanding the condition side of action rules, we are able to generate action rules with higher confidence. Rule 8 states that by only decreasing the rate of change of $i - \text{long}$, we would have a 71% chance of shifting the Czermak's mirror test from .5 to 0. However, rule 9 states that by decreasing the rate of change of $i - \text{long}$ and maintaining a low value of sleep apnoea, we would have an 83% chance of shifting Czermak's mirror test from .5 to 0.

Rule 10. $r_{10} = (\text{tonsils}, \geq 2 \rightarrow < 2) \Rightarrow (\text{Czermak's mirror test}, .5 \rightarrow 0);$
 $\text{supp}(r_9) = 5, \text{conf}(r_9) = 100\% .$

This rule states, with absolute certainty (confidence 100%), that by decreasing the hypertrophied adenoids and possibly palatine tonsils that the patient is experiencing, the Czermak's mirror test will shift from .5 to 0. Although the improvement does not appear to be significant, the high support and high confidence make this rule highly valuable.

In our hypernasality dataset, most of the patients were experiencing slight to no hypernasality speech (Czermak's mirror test .5 or 0). As a consequence, the last three action rules had a much higher support compared to the others.

6 Summary and Conclusions

In this paper we presented a new approach to examine temporal and object-driven information systems. We proposed a novel pair-based extraction approach that extends the work proposed by Ayman et al. in [4]. In addition to extracting stronger action rules in the same domain of hypernasality speech treatment, we were able to extract a dramatically larger set of action rules that is highly diversified and can be applied to various cases of patients.

One of the authors is still collaborating with physicians, so the outcome of this research can be implemented and tested in practice. This confirms that the obtained rules are in concordance with experience, and they help speech scientists to recapitulate their practical knowledge.

Acknowledgments. The authors would like to thank Dr. Danuta Chojnacka-Wądołowska and Dr. Cecylia Konopka from Children's Memorial Health Institute in Warsaw for their help with data collection and providing medical diagnoses.

This project was partially supported by the Research Center of PJIIT, supported by the Polish Ministry of Science and Higher Education. It is also based upon work supported by the National Science Foundation under Grant No. OISE-0730065.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Database. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
2. Bazan, J.G., Szczuka, M.: The rough set exploration system. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005)
3. Gubrynowicz, R., Chojnacka-Wądołowska, D., Konopka, C.: Assessment of Velum Malfunction in Children Through Simultaneous Nasal and Oral Acoustic Signals Measurements. Archives of Acoustics 32(1), 165–175 (2007)
4. Hajja, A., Wieczorkowska, A., Raś, Z.W., Gubrynowicz, R.: Object-driven Action Rules and their Application to Hypernasality Treatment. In: Proceedings of ECML-PKDD Workshop on New Frontier in Mining Complex Patterns, Bristol, UK, September 24–28, pp. 104–115 (2012)
5. He, Z., Xu, X., Deng, S., Ma, R.: Mining Action Rules from Scratch. Expert Systems with Applications 29(3), 691–699 (2005)
6. Raś, Z.W., Wieczorkowska, A.A. (eds.): Advances in Music Information Retrieval. SCI, vol. 274. Springer, Heidelberg (2010)
7. Paul, R., Hoque, A.S.: Mining Irregular Association Rules Based on Action and Non-action Type Data. In: Proceedings of the Fifth International Conference on Digital Information Management, ICDIM, pp. 63–68 (2010)
8. Pawlak, Z.: Information systems - theoretical foundations. Information Systems Journal 6, 205–218 (1981)
9. Raś, Z., Dardzińska, A.: From Data to Classification Rules and Actions. International Journal of Intelligent Systems 26(6), 572–590 (2011)
10. Raś, Z.W., Dardzińska, A., Tsay, L.S., Wasyluk, H.: Association action rules. In: IEEE International Conference on Data Mining Workshops, pp. 283–290 (2008)
11. Raś, Z.W., Wieczorkowska, A.: Action-Rules: How to increase profit of a company. In: Zighed, D.A., Komorowski, J., Zytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 587–592. Springer, Heidelberg (2000)
12. Raś, Z.W., Wyrzykowska, E., Wasyluk, H.: ARAS: Action Rules discovery based on Agglomerative Strategy. In: Raś, Z.W., Tsumoto, S., Zighed, D.A. (eds.) MCD 2007. LNCS (LNAI), vol. 4944, pp. 196–208. Springer, Heidelberg (2008)

13. Rauch, J., Šimůnek, M.: Action Rules and the GUHA Method: Preliminary Considerations and Results. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS, vol. 5722, pp. 76–87. Springer, Heidelberg (2009)
14. SAMPA - computer readable phonetic alphabet, <http://www.phon.ucl.ac.uk/home/sampa/>
15. Tsay, L.-S., Raś, Z.W.: Action rules discovery system DEAR_3. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 483–492. Springer, Heidelberg (2006)
16. Wasyluk, H., Raś, Z.W., Wyrzykowska, E.: Application of Action Rules to HEPAR Clinical Decision Support System. *Experimental and Clinical Hepatology* 4(2), 46–48 (2008)
17. Zhang, X., Raś, Z.W., Jastreboff, P.J., Thompson, P.L.: From Tinnitus Data to Action Rules and Tinnitus Treatment. In: *Proceedings of 2010 IEEE Conference on Granular Computing*, Silicon Valley, CA, pp. 620–625. IEEE Computer Society (2010)

Effectively Grouping Trajectory Streams

Gianni Costa, Giuseppe Manco, and Elio Masciari

ICAR-CNR

{costa,manco,masciari}@icar.cnr.it

Abstract. Trajectory data streams are huge amounts of data pertaining to time and position of moving objects. They are continuously generated by different sources exploiting a wide variety of technologies (e.g., RFID tags, GPS, GSM networks). Mining such amount of data is a challenging problem, since the possibility to extract useful information from this peculiar kind of data is crucial in many application scenarios such as vehicle traffic management, hand-off in cellular networks, supply chain management. Moreover, spatial data streams pose interesting challenges for their proper representation, thus making the mining process harder than for classical point data. In this paper, we address the problem of trajectory data streams clustering, that revealed really intriguing as we deal with a kind of data (trajectories) for which the order of elements is relevant. We propose a complete framework starting from data preparation task that allows us to make the mining step quite effective. Since the validation of data mining approaches has to be experimental we performed several tests on real world datasets that confirmed the efficiency and effectiveness of the proposed technique.

1 Introduction

The trajectory streams clustering challenge. Data Clustering is one of the most important mining techniques exploited in the knowledge discovery process[12]. Clustering huge amounts of data is a difficult task since the goal is to find a suitable partition in a unsupervised way (i.e. without any prior knowledge) trying to maximize the similarity of objects belonging to the same cluster while minimizing the similarity among objects in different clusters. Many different clustering techniques have been defined in order to solve the problem from different perspective, i.e. partition based clustering (e.g. *K-means*[20]), density based clustering (e.g. *DBScan*[7]), hierarchical methods (e.g. *BIRCH*[31]) and grid-based methods (e.g. *STING* [28]). Moreover, clustering methods have been exploited in a wide variety of application scenarios ranging from transactional data, text documents, XML data, etc. The main problem when clustering data is the high degree of uncertainty both in the data selection phase and in the definition of clusters, moreover due to complexity matter some algorithms do not scale-up very well when the size of the dataset becomes really huge.

Trajectory data streams are intrinsically quite difficult to be analyzed due to their *ordering* that makes the clustering task quite complex. This difficulty propagates through the knowledge discovery process affecting the quality of the

obtained results. particular we can have imprecise information about: *a) the data*, i.e. we are unsure of what exactly we observe or measure (every phenomenon in real life scenarios exhibits a spatial variability). As an example, consider the symptoms of natural disasters like hurricanes, such as sudden temperature and pressure variations: a hurricane monitoring system collecting data from different devices should detect the symptoms to plan a proper reaction to the event. Moreover, trajectory data are collected *incrementally* as the source devices generate new data points; *b) the extracted rules*, e.g. we are unsure about the conclusions we can draw from even perfect data, i.e. an unusual data is an outlier or is it the symptom that something is changing in the structure of data (concept drift)?

This work follows our first proposal for trajectory clustering shown in [21,22]. Although our previous works gave some contribution to the trajectory clustering goal the techniques previously implemented are quite different since: 1) they are not designed for dealing with data streams; 2) we exploited an approach based on trajectory partitioning. We point out that, in this paper we tackle the clustering problem from a different point of view w.r.t. existing approaches (by ourselves and the other approaches in literature) both in the clustering definition (we work on the original trajectories) and the acquisition paradigm (we work on trajectory streams). The features of the presented approach guarantee more flexibility and better performances as will be shown in the experimental section.

Plan of the paper. In Section 2 we briefly describe some related works. In Section 3 we formalize the trajectory clustering problem and our pre-processing strategy exploiting lifting schemes. In Section 4 we show our fourier based algorithm for trajectory clustering. In Section 5 we describe our experimental evaluation. Finally in section 6 we draw our conclusions.

2 Related Work

Mining trajectory data is an active research area and many interesting proposals exist in the literature. In [29] an algorithm for sequential pattern mining is introduced. The algorithm *TrajPattern* mines patterns by a process that identifies the top- k most important patterns with respect to a suitable measure called *NM*. The algorithm exploits a min-max property since the well known Apriori property no longer holds for that measure. A general formal statement for the trajectory pattern mining problem is provided in [10], in that work, trajectory patterns are characterized in terms of both space (introducing the concept of regions of interest) and time (considering the duration of movements). In [17] *TRACCLUS* a density-based trajectory clustering algorithm working over the entire set of sub-trajectories is presented. Clustering is performed using a two-phase algorithm that first partitions the trajectory according to the *MDL* principle and then clusters the trajectory segments using a line-segment clustering algorithm. The algorithm is parametric w.r.t. weights assigned to sub-trajectories participating in the cluster definition, namely ε and *MinLens*. In [18] a technique for defining and maintaining micro clusters of moving objects is defined. In [14] a filter-refinement approach has been used for discovering convoys in trajectory

databases. In particular various trajectory simplification techniques are studied along with different query processing strategies. Other proposal for trajectory clustering are described in [9,3]. As a matter of fact all of these algorithms cannot efficiently handle incremental trajectory data. They are not suitable for incremental data since clusters are re-calculated from scratch every time. In [16] a partition and detect framework is presented for trajectory outlier detection. In particular a hybrid approach distance-based and density-based is exploited to identify anomalies. Interesting approaches exploiting Edit distance are proposed in [27] and [5]. In particular in [27] a distance measure called *LCSS* is proposed while in [5] the proposed distance measure is *EDR*. In both approaches the simple edit distance is improved in order to take into account possible errors such as noise, shifts, or different sampling choices among different trajectories. A traditional approach for comparing sequences is known in literature as *Time Warping* [30], which mainly consists in considering every possible stretching and narrowing of the two signals, and choosing the best matching. Essentially, time-warping corresponds to a tree-edit over sequences. Hence, it is quite expensive (quadratic in complexity), and in most cases the resulting structural similarity of two trajectories does not necessarily correspond to a similar shape of the associated signals. An approach exploiting dynamic time warping for comparing time series is presented in [15].

Data stream clustering has been widely studied and some proposal are available such as CluStream[1] that studies clustering dynamic data streams. However all the proposed methods handles only incremental data but not trajectory streams. Instead, the only proposal so far available for trajectory streams clustering is described in [19]. It works in two steps: online micro-cluster maintenance and offline macro-cluster creation. For online part, when a new bunch of trajectories arrives, each trajectory is simplified into a set of directed line segments in order to find clusters of trajectory subparts, this simplification is similar to the one for the static trajectories presented in [17]. When new data arrive, micro-clusters are updated incrementally to reflect the changes. For offline part, when a user requests to see current clustering result, macro-clustering is performed on the set of micro-clusters rather than on all trajectories over the whole time span. There exists also a class of approaches to trajectory data that can be considered orthogonal to our work but deserves to be mentioned. In [4], the authors examine an alternative representation for motion data, called non-materialized trajectory, which addresses the problem of effectively representing trajectory data by taking advantage of the a priori knowledge about the motion that can be gathered in a transport network. In [11] the problem of compressing spatio-temporal trajectories is studied in order to provide that the most common queries can still be answered approximately after the compression step has taken place. Finally, various compression algorithms for position data streams are presented in [13]. In [23], the authors propose SCUBA for evaluating continuous queries over spatio-temporal data streams. The key idea of SCUBA is to group moving objects and queries based on common spatio-temporal properties at runtime into moving clusters to optimize query execution and thus

facilitate scalability. SCUBA exploits shared cluster-based execution by abstracting the evaluation of a set of spatio-temporal queries as a spatial join first between moving clusters. This cluster-based filtering prunes true negatives. In [6], an algorithm for computing the approximate k-median of huge number of moving objects is presented. In order to process distributed location streams with redundancy and inconsistency, the authors propose a method based on min-wise hash and location summarization. With this method, redundant updates of distributed location streams can be filtered out, while the true location could be derived from inconsistent ones. Consequently, globally uniform samples can be obtained.

Main Differences of Our Method w.r.t. Existing Methods. Our approach differs from the above mentioned approaches both for the representation of trajectories and their mining. First of all, we work on the original trajectories with no loss of information (we will exploit the lifting of trajectories that is a lossless operation [25]). This feature makes our approach different also from the stream based approaches so far proposed in literature. Moreover, the spectral analysis allows us to catch both global and (eventually) local similarity among trajectories. Finally, the algorithm we designed is incremental so we can easily perform cluster computation and maintenance.

3 Background

In this paper we tackle the problem of clustering large corpus of trajectory data streams. While for transactional data a tuple is a collection of features, a trajectory is an ordered set (i.e., a sequence) of timestamped points. Trajectory data are usually recorded in a variety of different formats, and they can be drawn from a continuous domain. We assume a standard format for input trajectories, as defined next.

Definition 1 (Trajectory). *Let P and T denote the set of all possible (spatial) positions and all timestamps, respectively. A trajectory is defined as a finite sequence s_1, \dots, s_N , where $N \geq 1$ and each s_i is a pair (p_i, t_i) where $p_i \in P$ and $t_i \in T$.*

3.1 Data Pre-processing

Trajectory data are usually two dimensional, thus when a high accuracy is required we cannot disregard any point. Therefore, there is a need for a multi-resolution analysis to take into account both the spatial dimensions in the pre-processing step. We first give some preliminary definition on multi-resolution analysis.

Definition 2. *Let $\{S_m\}$ be a set of subspace of $L^2(\mathcal{R})$ and $m \in \mathcal{Z}$ such that the following hold: 1) $S_m \in S_{m+1}$; 2) $\bigcup_{j \in \mathcal{Z}} S_m = L^2(\mathcal{R})$; 3) $\bigcap_{j \in \mathcal{Z}} S_m = \emptyset$; 4) $x(t) \in S_m \Leftrightarrow x(\frac{t}{2}) \in S_{m-1}$ then $\{S_m\}$ is a multi-resolution system.*

The choice of $\{S_m\}$ defines the analysis being performed in particular if we choose orthogonal subspace we have *orthogonal multi-resolution* analysis. We exploit here L space since it has a proper norm.

Trajectory data multi-resolution analysis aims at representing each trajectory being analyzed (and then elaborated using a mathematical transform) using a reliable set of coefficient. In order to perform this step allowing a *perfect* (i.e. lossless) reconstruction of trajectories we adopt the so called *Lifting Scheme* approach.

3.2 Lifting Schemes

An effective approach for multi-resolution analysis is the *lifting scheme*. It was originally introduced for filtering signal and due to its intuitive features and more important because of its ability to exactly reconstruct the original input sequences it has been widely used also as a support for image compression in wavelet based systems. Moreover it is well suited as a preprocessing step for non separable transforms. In order to perform the proper lifting we need to define a filtering function. A filtering function \mathcal{F} , is a function that transform an input sequence I into an output sequence O according to an optimization function. Most widely used filters are Least Mean Square, Regression or Kalman filtering. In our implementation we exploited the Least Mean Square filter since it works by minimizing the least mean squares of the error signal, i.e. the difference between the computed and the actual signal.

Performing lifting steps. Given a filtering function \mathcal{F} and a input trajectory Tr_x , the trajectory can be split in two subsequences Tr_{x_o} and Tr_{x_e} that are respectively the sequence of odd and even indices. The trajectory lifting is performed by iteratively updating the subsequences with their predicted version in order to obtain two shorter sequences that are representative of the *original* sequence (say it Tr'_x) and the *trend* sequence (say it Tr_h). Obviously, when performing this step some errors could arise (say it Tr_e). More formally, let \mathcal{P} and \mathcal{U} be two filtering functions exploited for prediction and update, a generic lifting step works as follows:

$$\begin{aligned} - Tr_e(k) &= Tr_{x_o}(k) - \mathcal{P}(Tr_{x'}(k)); \\ - Tr_h(k) &= Tr_{x_e}(k) - \mathcal{U}(Tr_{x_e}(k)). \end{aligned}$$

By iterating the above steps, we obtain a succinct representation of the original trajectory with no loss of information. The proposed lifting scheme will be used for implementing the non separable transform based clustering described in next section. Indeed, when the trajectory size becomes unpractical we will reduce it by a lifting step without decreasing the information quality. Furthermore this step allow us to make trajectories equally lengthened so the successive transforms will better compare them without needing any alignment or interpolation operation. To better clarify the proposed pre-elaboration steps we provide a toy example. Consider the trajectories depicted in Fig. 1(a) regarding buses movements in the Athens metropolitan area. Applying the lifting scheme defined

above on a sample trajectory will produce the sequence in Fig. 1(b) where the solid (blue) stars represent the *trend* sequence, while the empty (red) stars represent the error sequence. It is easy to see that the number of points taken into account after lifting is really smaller than the original trajectory and this feature is particularly useful when considering (real life) longer trajectories having more complex shapes. Moreover, lifted trajectories can be updated as new data points arrive: indeed, incremental computation is a key requirement for streaming algorithms. Furthermore, we can make the lifted trajectories equally sized in order to enhance the clustering performances.

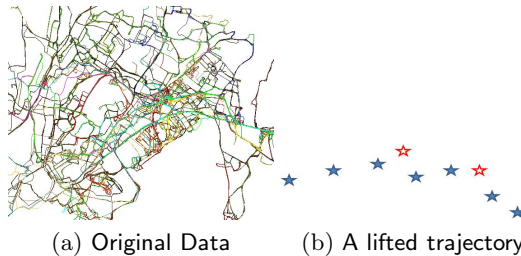


Fig. 1. Trajectory Lifting Example

4 Exploiting Fourier Transforms for Spatial Quincunx Lattices Based Clustering

In this section we exploit non separable transforms in order to effectively manage two dimensional trajectories. Non separable transforms allow us to consider the whole trajectory taking into account both dimensions in the computation thus avoiding any approximation due to mono-dimensional transform composition. We will exploit them in a suitable way in order to catch similarity among trajectories. The first step to be performed for any mathematical transform is to define the basis function and the features of the search space where data reside. After a deep investigation of several trajectory datasets we found that the best representation for the trajectory search space is a *Quincunx Lattice*¹. Indeed, optimal sampling scheme in the two-dimensional space is the hexagonal lattice. Unfortunately, hexagonal lattice is quite unwieldy in terms of hardware and software implementations. An appealing compromise is the quincunx lattice that is a sublattice of the square lattice. The quincunx lattice has a diamond tessellation which is closer to optimal hexagon tessellation than square lattice, and it can be easily generated by down sampling conventional digital images without any hardware change. Indeed, for this reason, quincunx lattice is widely adopted in many application dealing with spatial images[32]. Based on the above mentioned consideration exploiting a quincunx lattice will allow us to represent compactly

¹ It is always possible to find a basis that allows this representation for the search space.

while preserving the representation accuracy even for trajectories laying close to the edges of the search space. The first step is the construction of the algebra (details are not relevant for this paper, more details in [24]) that induces the spatial signal spectrum defined below:

$$\begin{aligned} - u_k &= \cos\left(\frac{k+1/2}{n/2} \cdot \pi\right), 0 < k < n/2 \\ - v_l &= \cos\left(\frac{l+1/2}{n/2} \cdot \pi\right), 0 < l < n/2 \\ - w_{k,l,\pm} &= \pm \frac{1}{2} \sqrt{(1+u_k) \cdot (1+v_l)} \end{aligned}$$

Once obtained the spatial signal spectrum we can easily define the distance between two trajectories by considering their spectra. We recall that we assume that each trajectory point is an impulse in the overall signal associated to the trajectory. In particular, given two trajectories Tr_1 and Tr_2 and their spectra $Q(Tr_1)$, $Q(Tr_2)$, we compute their angular distances as $d_{\alpha\beta}^{\widehat{\gamma}} = \arccos(u_{1k}) - \arccos(u_{2k})$ and $d_{\gamma\delta}^{\widehat{\gamma}} = \arccos(v_{1l}) - \arccos(v_{2l})$. For each pair of $w_{k,l,\pm}$ we compute the modulo distance as $d_w = \sqrt{w_{1k,l,\pm}^2 - w_{2k,l,\pm}^2}$. Finally, we define the overall *Quincunx* based distance as:

$$dist_Q(Tr_1, Tr_2) = \sqrt{\sum_{k=1}^{n^2} d_w(k)^2 \cdot \cos(\mu(d_{\alpha\beta}^{\widehat{\gamma}}) - \mu(d_{\gamma\delta}^{\widehat{\gamma}}))}$$

where $\mu(d_{\alpha\beta}^{\widehat{\gamma}})$ is the average angle distance between each pair of u_k and $\mu(d_{\gamma\delta}^{\widehat{\gamma}})$ is the mean between each pair of v_l . The distance so far defined is able to catch dissimilarity between trajectories since it considers the difference in angular distances between the two trajectories (the cos argument) while taking into account the overall extension of the spatial signal (the modulo part). This is a crucial feature since when using trajectory simplification techniques defined in literature we could lose relevant information about trajectories.

Clustering Trajectory Streams. We briefly recall the notion of trajectory cluster.

Definition 3 (Trajectory Cluster). *Given a set of trajectories T , a cluster is a subset $C \subseteq T$ such that the distance between each pair of trajectories in C is minimum, and the distance between each pair of trajectories $Tr_i \subseteq C$ and $Tr_j \notin C$ is maximized w.r.t. the chosen metric.*

Our algorithm described below is tailored for clustering evolving streams of trajectories, thus the dataset W being mined is defined as a sliding window over the continuous stream. W moves forward by a certain amount. Each window W either contains the same number of trajectories (*count-based* or physical window), or contains all trajectories arrived in the same period of time (*time-based* or logical window). The window is maintained by adding the new slide (δ^+) and dropping the expired one (δ^-). Therefore, the successive instances of W are referred as W_1, W_2, \dots . The number of trajectories that are added to (and

Method: IncrementalClusteringMaintenance**Input:**A trajectory stream T .**Output:**A trajectory clustering C_T .**Vars:**A new slide S_i of the input trajectories;An expiring slide S_{exp} of the input trajectories;A set of lifted trajectories T_L 0: $C_T = \emptyset$ 1: **For Each** New Slide S_i 2: $lift(T_L, S_i)$;3: $C_{T_i} = mineNewCluster(T_L)$;4: $mergeClusters(C_T, C_{T_i})$ 5: **For Each** trajectory $t \in T_L$ 6: $annotate(t)$;7: **For Each** Expiring Slide S_{exp} 8: $deleteOldestTrajectories(C_T)$;9: $compactClusters(C_T)$;**Fig. 2.** The incremental clustering algorithm

removed from) each window is called its slide size. In this paper, for the purpose of simplicity, we assume that all slides have the same size, and also each window consists of the same number of slides. Thus, $n = |W| \div |S|$ is the number of slides (a.k.a. panes) in each window, where $|W|$ denotes the window size and $|S|$ denotes the size of the slides. The incremental clustering algorithm is reported in Fig. 2.

As a new window slide S_i is loaded we compute the lifting of trajectories contained in S_i . We compute the clustering of the lifted trajectories (*mineNewCluster*) by running *k-means++*, a stable *k-means* improvement that has been proposed in [2], using a distance-based probabilistic algorithm $O(\log(k))$ that makes it competitive with optimal clustering and avoid the initial cluster assignment problem. To keep fresh clusters and to avoid concept drift we merge the existing clusters with the new computed one (*mergeClusters*) by minimizing $dist_Q$ between each pair of trajectories to be merged. More in detail, given the centers of the candidate clusters we assign the center of the new cluster as the trajectory that minimize the $dist_Q$ w.r.t previous centers, then we eventually discard trajectories that are now closer to a different existing cluster than the one being created. The result of this step is a cluster assignment that is continuously updated, thus it results impervious to the eventual concept drift. After clustering computation we annotate the trajectories (by their timestamps) in the current window slide to allow successive delta maintenance (function *annotate*). Finally, as a window slide expires we discard the oldest trajectories and recompact the clusters that could results loosely compact after trajectory deletions (we perform *compactClusters* by recomputing cluster centers w.r.t. $dist_Q$ minimization as in *mergeclusters*). In this respect, we recompute the clustering including the new trajectories and assigning new cluster centers (that are

carefully chosen by the clustering algorithm we exploit), this will allow us to take into account the eventual concept drift. Note that no threshold information are provided by the user since the provided distance is a metric thus it preserves the k -means++ features.

5 Experimental Results

In this section, we present the experiments we performed to assess the effectiveness of the proposed approach in clustering trajectories. To this purpose, a collection of tests is performed, and in each test some relevant groups of homogeneous trajectories (*trajectory classes*) are considered. The direct result of each test is a similarity matrix representing the degree of similarity for each pair of trajectories in the data set. The evaluation of the results relies on some *a priori* knowledge about the trajectory classes being used that was obtained by domain experts or available from the datasets providers.

We set up two classes of experiments: 1) we tested our algorithm (we refer to it in the following as $Fourier_{2D}$) in a *static* context, i.e. we considered datasets that can fit in a single window. The comparison in this case is made against TRACCLUS [17], since TRACCLUS is a parametric algorithm we report here the best parameters assignment as suggested by the authors in [17]. This set of tests is intended to evaluate the accuracy of the clustering, and we choose TRACCLUS since it is (at the best of our knowledge) the most accurate system available for static trajectory clustering and 2) we tested the *dynamic* streaming performances by tuning the windows size and measuring the effectiveness and the efficiency against TCMM [19] that is the most accurate proposal available for clustering trajectory data streams at the best of our knowledge.

Static Performance Evaluation. We performed several experiments on a wide variety of real datasets. More in detail we analyzed the following data: 1) *School Bus*: it is a dataset consisting of 145 trajectories of 2 school buses collecting (and delivering) students around Athens metropolitan area in Greece for 108 distinct days; 2) *Animals*, it is a dataset containing the major habitat variables derived for radio-telemetry studies of elk, mule deer, and cattle at the Starkey Experimental Forest and Range in northeastern Oregon².

In order to perform a simple quantitative analysis we produce for each test a similarity matrix, aimed at evaluating the resulting intra-cluster similarities (i.e., the average of the values computed for trajectories belonging to the same cluster), and to compare them with the inter-cluster similarities (i.e., the similarity computed by considering only trajectories belonging to different classes). To this purpose, values inside the matrix can be aggregated according to the cluster of membership of the related elements: given a set of trajectories belonging to n prior classes, a similarity matrix S about these trajectories can be summarized by a $n \times n$ matrix CS , where the generic element $CS(i, j)$ represents the average similarity between cluster i and cluster j .

² <http://www.fs.fed.us/pnw/starkey/data/tables/index.shtml>

$$CS(i, j) = \begin{cases} \frac{\sum_{x, y \in C_i, x \neq y} DIST(x, y)}{|C_i| \times (|C_i| - 1)} & \text{iff } i = j \\ \frac{\sum_{x \in C_i, y \in C_j} DIST(x, y)}{|C_i| \times |C_j|} & \text{otherwise} \end{cases}$$

where $DIST(x, y)$ is the chosen distance metric ($TRACCLUS$ metric or the $dist_Q$).

The above definition is significant since we normalize the different metrics pertaining to the different approaches, this will allow us to compare performance in the ideal setting for both approaches.

The higher are the values on the diagonal of the corresponding CS matrix w.r.t. those outside the diagonal, the higher is the ability of the similarity measure to separate different classes. In the following we report a similarity matrix for each dataset being considered, as it will be clear the reported results show that our technique is quite effective for clustering the datasets being considered and outperforms $TRACCLUS$.

Measuring Effectiveness for School Bus. For this dataset our prior knowledge is the set of trajectories related to the two school buses that define the two clusters in the dataset. Our algorithm is able to exactly detect the two clusters. We present the results using the two classes but we point out that our technique is able to (eventually) further refine the cluster assignment identifying the micro-clusters represented by common sub-trajectories. As it is easy to see in Fig. 3(a) and (b) $Fourier_{2D}$ outperforms $TRACCLUS$ by allowing a perfect assignment to the proper class to each trajectory.

$TRACCLUS$	Bus 1	Bus 2
Bus 1	0.9790	0.8528
Bus 2	0.8528	0.9915

(a)

$Fourier_{2D}$	Bus 1	Bus 2
Bus 1	1	0.6250
Bus 2	0.6250	1

(b)

Fig. 3. $TRACCLUS$ and $Fourier_{2D}$ similarity matrices for Bus dataset

The presence of several turns made by the buses makes the feature of $Fourier_{2D}$ well suited for this dataset since it takes into account all the angular values of the trajectory.

Measuring Effectiveness for Animals. In this case we considered as a class assignment the different trajectories traversed by elk, mule deer, and cattle. Also in this case our approach correctly identify the three cluster present in the dataset. We point out that it is worth studying animal data because the trajectories are in unrestricted space rather than on well known road network. In this case there were 3 main classes as it is shown in Fig. 4(a) and (b). Also in this case $Fourier_{2D}$ outperforms $TRACCLUS$. As we can see, differences among

the various classes are marked with higher precision by $Fourier_{2D}$. This is mainly due to the fact that our approach is quite discriminative since it takes into account both angular and modulo distances in the spectrum of a trajectory.

$Fourier_{2D}$	elk	mule deer	cattle
elk	0.9986	0.7759	0.7055
mule deer	0.7759	0.9889	0.7566
cattle	0.7055	0.7566	0.9920

(a)

$TRACCLUS$	elk	mule deer	cattle
elk	0.9885	0.7439	0.7108
mule deer	0.7439	0.9899	0.7223
cattle	0.7108	0.7223	0.9874

(b)

Fig. 4. $TRACCLUS$ and $Fourier_{2D}$ similarity matrices for *Animals* dataset

Quality Measures Evaluation. Distance minimization is a natural and widely used norm of similarity, but a devil’s advocate can point out that other clustering algorithms might not measure their effectiveness in terms of this metric or even the compactness and homogeneity of each cluster around its centroid. Thus, in this section we will attempt to measure the quality of the clusters produced by $Fourier_{2D}$ using very different criteria inspired by the nearest subclass classifiers that were previously used in a similar role in [26] and [8]. A first relevant evaluation measure is the error rate of a k -Nearest Neighbor classifier defined on the basis of the similarity measure. For a given trajectory, we can check whether the dominant class of the k most similar elements allows to correctly predict the actual class of membership. Thus, the total number of trajectories correctly predicted can be considered as a measure for evaluating the effectiveness of the similarity at hand. Formally, the error $e_k(S)$ of a k NN classifier exploiting a similarity matrix S can be defined as

$$e_k(S) = \frac{1}{N} \sum_{i=1}^N \gamma_k(i)$$

where N is the total number of trajectories, and $\gamma_k(i)$ is 0 if the predicted class of the i -th trajectory coincides with its actual class, and 1 otherwise. This value will measure the errors made in the cluster computation. Low values of the $e_k(S)$ index correspond to good results.

The above measure can be refined by evaluating the average number of elements, in a range of k elements, having the same class of the trajectory under consideration. Practically, we define q_k as the average percentage of trajectories in the k -neighborhood of a generic trajectory belonging to the same class of that trajectory. Formally:

$$q_k(S) = \frac{1}{N} \sum_{i=1}^N \frac{|\mathcal{N}_k(i) \cap Cl(i)|}{\min(k, n_i)}$$

where $Cl(i)$ represents the actual class associated with the i -th trajectory in the dataset, $n_i = |Cl(i)|$, and $\mathcal{N}_k(i)$ is the set of k trajectories having the lowest

distances from Tr_i , according to the similarity measure at hand. In principle, a Nearest Neighbor classifier exhibits a good performance when q_k is high, since high values of q_k indicate a great *purity* of the clustering. Furthermore, q_k provides a measure of the stability of a Nearest-Neighbor: high values of q_k make a k NN classifier less sensitive to increasing values k of neighbors considered. The sensitivity of the similarity measure can also be measured by considering, for a given group of trajectories x, y, z , the probability that x and y belong to the same class and z belongs to a different class, but z is more similar to x than y is. We denote this probability by $\varepsilon(S)$, which is estimated as

$$\varepsilon(S) = \frac{1}{N} \sum_{i=1}^N \left(\frac{1}{(n_i - 1)(N - n_i)} \sum_{Cl(j)=Cl(i), j \neq i} \sum_{Cl(k) \neq Cl(i)} \delta_S(i, j, k) \right)$$

where δ_S is 1 if $S(i, j) < S(i, k)$, and 0 otherwise. Once again, low values of $\varepsilon(S)$ denote a good performance of the similarity measure under consideration, since they indicate a low *ambiguity* in the clustering.

The quality values obtained by the various techniques are reported in the following Tables. As in the previous section we first show the Table for *Bus* dataset. For measures e_k and q_k we considered neighborhoods of size 70, i.e. the actual size of each class in the data set. As it is easy to see the results shown in Fig. 5 confirm the ones obtained by similarity matrix, thus assessing that for *Bus* dataset *Fourier*_{2D} outperforms *TRACCLUS*.

<i>Bus</i>			
ε	$e_{k=70}$	$q_{k=70}$	
<i>Fourier</i> _{2D}	0.0113	0.0235	0.9965
<i>TRACCLUS</i>	0.1811	0.0997	0.9658
<i>Animals</i>			
ε	$e_{k=50}$	$q_{k=50}$	
<i>Fourier</i> _{2D}	0.0792	0.1225	0.8945
<i>TRACCLUS</i>	0.1801	0.1887	0.6257

Fig. 5. Quality indices for our datasets

For *Animals* dataset, measures e_k and q_k are evaluated considering neighborhoods of size 50, i.e. the actual size of each class in the data set. Again the results shown in Fig. 5 confirm the ones obtained by similarity matrix, thus assessing that for *Animals* dataset *Fourier*_{2D} still outperforms *TRACCLUS*.

Measuring Performance on Streams. In previous section we assessed the accuracy of the approach for detecting clusters. In this section we will measure the performances of our algorithm when dealing with stressing trajectory streams. We report the results obtained for two huge real life datasets: 1) *Hurricanes* trajectory dataset. It is a dataset containing data about Atlantic hurricanes since 1851, it includes position in latitude and longitude, maximum sustained winds in knots, and central pressure in millibars³; 2) *GPS* trajectory dataset collected in

³ Available at <http://weather.unisys.com/hurricane/atlantic/>

(Microsoft Research Asia) *GeoLife* project [33] by 165 users in a period of over two years (from April 2007 to August 2009). This dataset recoded a broad range of users outdoor movements, including not only life routines like go home and go to work but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. Therefore, the dataset allows a severe test for our clustering approach.

We ran several tests varying the window size ($|W|$) thus using the *count-based* approach for comparison purposes w.r.t. TCMM. Moreover, in order to perform a better comparison against TCMM we computed the average SSQ as described in [19]. In particular, we find for each cluster its centroid and then compute the SSQ using our $dist_Q$ in the well known SSQ formula. In Fig. 6(a) and (b) we report the results for *Hurricanes* dataset w.r.t. varying window size expressed in MBytes while times are expressed as seconds, while in Fig. 6(c) and (d) we report the results obtained for *GPS* dataset. $Fourier_{2D}$ (the left red bar) outperforms $TCMM$ (the right green bar) both in terms of execution times and accuracy. The faster execution is due to the incremental computation of the lifted trajectories that allows us to save execution time as new point are added, while the accuracy result is due to the peculiar features of $dist_Q$ that takes into account all possible differences between trajectories as explained above even when incrementally maintained.

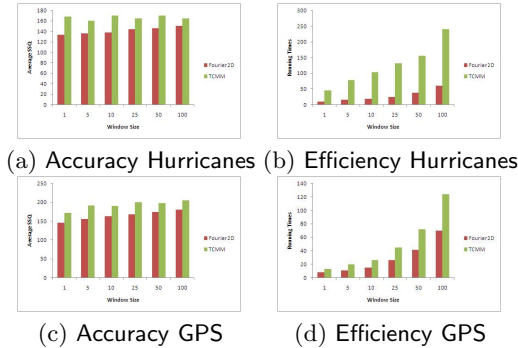


Fig. 6. Performance Comparison w.r.t. TCMM

Quality Measures Evaluation. The quality values obtained by the $Fourier_{2D}$ and $TCMM$ are reported in the following Tables. We first show the Table for *Hurricanes* dataset. For measures e_k and q_k we considered neighborhoods of size 100, i.e. the actual size of each class in the data set. As it is easy to see the results shown in Fig. 7 assess that for *Hurricanes* dataset $Fourier_{2D}$ outperforms $TCMM$.

Concerning *GPS* dataset for measures e_k and q_k we considered neighborhoods of size 75, i.e. the actual size of each class in the data set. Again the results shown in Table 7 confirm that also for *GPS* dataset $Fourier_{2D}$ outperforms $TCMM$.

<i>Hurricanes</i>			
	ε	$\varepsilon_{k=100}$	$q_{k=100}$
<i>Fourier_{2D}</i>	0.0097	0.0354	0.9877
<i>TMM</i>	0.1433	0.1025	0.9551
<i>GPS</i>			
	ε	$\varepsilon_{k=75}$	$q_{k=75}$
<i>Fourier_{2D}</i>	0.1633	0.2895	0.7781
<i>TMM</i>	0.2534	0.4033	0.6021

Fig. 7. Quality indices for our datasets

6 Conclusion

In this paper we addressed the problem of detecting clusters in trajectory data. The technique we have proposed is mainly based on the idea of representing a trajectory with its lifted version. Thereby, the similarity between two trajectories can be computed by analyzing their Fourier transforms in the two-dimensional case. Experimental results showed the effectiveness of the approach in detecting common clusters for trajectories and robustness to the eventual concept drift.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
2. Arthur, D., Vassilvitskii, S.: k-means++ the advantages of careful seeding. In: SODA, pp. 1027–1035 (2007)
3. Cadez, I.V., Gaffney, S., Smyth, P.: A general probabilistic framework for clustering individuals and objects. In: KDD, pp. 140–149 (2000)
4. Cao, H., Wolfson, O.: Nonmaterialized motion information in transport networks. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 173–188. Springer, Heidelberg (2005)
5. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp. 491–502. ACM, New York (2005)
6. Chong, Z., Ni, W., Xu, L., Xu, Z., Shu, H., Zheng, J.: Approximate k-median of location streams with redundancy and inconsistency. Int. J. of Software and Informatics 4(2), 165–182 (2010)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
8. Flesca, S., Manco, G., Masciari, E., Pontieri, L., Pugliese, A.: Fast detection of xml structural similarity. IEEE TKDE 17(2), 160–175 (2005)
9. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: KDD, pp. 63–72 (1999)
10. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: KDD, pp. 330–339 (2007)
11. Gudmundsson, J., Katajainen, J., Merrick, D., Ong, C., Wolle, T.: Compressing spatio-temporal trajectories. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 763–775. Springer, Heidelberg (2007)
12. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000)

13. Hönlé, N., Grossmann, M., Reimann, S., Mitschang, B.: Usability analysis of compression algorithms for position data streams. In: GIS, pp. 240–249 (2010)
14. Jeung, H., Yiu, M.L., Zhou, X., Jensen, C.S., Shen, H.T.: Discovery of convoys in trajectory databases. In: PVLDB, vol. 1(1), pp. 1068–1080 (2008)
15. Keogh, E.: Exact indexing of dynamic time warping. In: VLDB, pp. 406–417. VLDB Endowment (2002)
16. Lee, J.G., Han, J., Li, X.: Trajectory outlier detection: A partition-and-detect framework. In: ICDE, pp. 140–149 (2008)
17. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: SIGMOD (2007)
18. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: KDD, pp. 617–622 (2004)
19. Li, Z., Lee, J.-G., Li, X., Han, J.: Incremental clustering for trajectories. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part II. LNCS, vol. 5982, pp. 32–46. Springer, Heidelberg (2010)
20. Lloyd, S.: Least squares quantization in pcm. IEEE TOIT 28 (1982)
21. Masciari, E.: A complete framework for clustering trajectories. In: ICTAI, pp. 9–16 (2009)
22. Masciari, E.: Trajectory clustering via effective partitioning. In: Andreasen, T., Yager, R.R., Bulskov, H., Christiansen, H., Larsen, H.L. (eds.) FQAS 2009. LNCS, vol. 5822, pp. 358–370. Springer, Heidelberg (2009)
23. Nehme, R.V., Rundensteiner, E.A.: SCUBA: Scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 1001–1019. Springer, Heidelberg (2006)
24. Puschel, M., Rotteler, M.: Fourier transform for the directed quincunx lattice. In: ICASSP (2005)
25. Secker, A., Taubman, D.: Lifting-based invertible motion adaptive transform (limat) framework for highly scalable video compression. IEEE Trans. on Image Processing 12(12), 1530–1542 (2003)
26. Veenman, C.J., Reinders, M.J.T.: The nearest subclass classifier: A compromise between the nearest mean and nearest neighbor classifier. IEEE PAMI 27(9), 1417–1429 (2005)
27. Vlachos, M., Gunopoulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE, p. 673
28. Wang, W., Yang, J., Muntz, R.R.: Sting: A statistical information grid approach to spatial data mining. In: VLDB, pp. 186–195 (1997)
29. Yang, J., Hu, M.: TrajPattern: Mining sequential patterns from imprecise trajectories of mobile objects. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 664–681. Springer, Heidelberg (2006)
30. Yi, B., Jagadish, H.V., Faloutsos, C.: Efficient retrieval of similar time sequences under time warping. In: ICDE, pp. 201–208 (1998)
31. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: SIGMOD, pp. 103–114 (1996)
32. Zhang, X., Wu, X., Wu, F.: Image coding on quincunx lattice with adaptive lifting and interpolation. In: Data Compression Conf., pp. 193–202 (2007)
33. Zheng, Y., Zhang, L., Xie, X., Ma, W.Y.: Mining interesting locations and travel sequences from gps trajectories. In: WWW, pp. 791–800 (2009)

Healthcare Trajectory Mining by Combining Multidimensional Component and Itemsets

Elias Egho¹, Chedy Raïssi⁴, Dino Ienco^{2,3}, Nicolas Jay¹, Amedeo Napoli¹, Pascal Poncelet^{2,3}, Catherine Quantin⁵, and Maguelonne Teisseire^{2,3}

¹ Orpailleur Team, LORIA, Vandoeuvre-les-Nancy, France

`firstname.lastname@loria.fr`

² Irstea, UMR TETIS, 34093 Montpellier, France

`firstname.lastname@teledetection.fr`

³ LIRMM, Univ. Montpellier 2, Montpellier, France

`firstname.lastname@lirmm.fr`

⁴ INRIA, Nancy Grand Est, France

`firstname.lastname@inria.fr`

⁵ Department of Biostatistics and Medical Information

CHU of Dijon, Dijon, France

Abstract. Sequential pattern mining is aimed at extracting correlations among temporal data. Many different methods were proposed to either enumerate sequences of set valued data (i.e., itemsets) or sequences containing multidimensional items. However, in real-world scenarios, data sequences are described as events of both multidimensional items and set valued information. These rich heterogeneous descriptions cannot be exploited by traditional approaches. For example, in healthcare domain, hospitalizations are defined as sequences of multi-dimensional attributes (e.g. Hospital or Diagnosis) associated with two sets, set of medical procedures (e.g. { Radiography, Appendectomy }) and set of medical drugs (e.g. { Aspirin, Paracetamol }). In this paper we propose a new approach called MMISP (*Mining Multidimensional Itemset Sequential Patterns*) to extract patterns from a complex sequences including both dimensional items and itemsets. The novelties of the proposal lies in: (i) the way in which the data can be efficiently compressed; (ii) the ability to reuse and adopt sequential pattern mining algorithms and (iii) the extraction of new kind of patterns. We introduce as a case-study, experimented on real data aggregated from a regional healthcare system and we point out the usefulness of the extracted patterns. Additional experiments on synthetic data highlights the efficiency and scalability of our approach.

Keywords: Sequential Patterns, Multi-dimensional Sequential Patterns, Data Mining.

1 Introduction

Data warehouses are constituting a large source of data that can be used to extract information for expert analysis and decision makers [5]. In temporal data

warehouses, every bit of information is associated with a timeline describing a total order over events. This total ordering introduces complexity in the extraction process. Many efficient approaches were developed to mine these patterns (i.e., sequential patterns) like PrefixSpan [9], SPADE [17], ClosSpan [14],...etc. However, all these techniques and algorithms, without any exception, focus solely on sequences of set valued data (i.e., *itemsets*) and do not pay attention to real-world data that is described over multiple dimensions. To overcome this problem, Pinto et al. [10] introduced the notion of multi-dimensionality in sequences and proposed an efficient algorithm. Later works, like Zhang et al. [18] or Yu et al. [16] extended the initial Pinto's approach for different scenarios and use-cases. While in set valued approaches the events are represented by itemsets, in multi-dimensional temporal databases the events are defined over a fixed schema where all attributes appear in the extracted patterns. Furthermore, and this is particularly true in the data warehouse environment, background knowledge is usually available and can be represented as a hierarchy over the values of the attributes. Taking advantage of this observation, Plantevit et al. introduced M^3SP [11], an efficient algorithm that is able to incorporate different dimensions and their taxonomies in the sequential pattern mining process. The benefit of this approach is to extract patterns with the most appropriate level of granularity. Still, this ideal representation of data is uncommon in real-world applications where heterogeneity is usually elevated to a foundational concept. In this study, we focus on extracting knowledge from medical data warehouse representing information about patients in different hospitals. The successive hospitalizations of a patient can be expressed as a sequence of multidimensional attributes associated with a set of medical procedures and a set of medical drugs. Our goal is to be able to extract patterns that express patients stays along with combinations of procedures over time. This type of pattern is very useful to healthcare professionals to better understand the global behavior of patients over time. Unfortunately this kind of complex data cannot be mined by any traditional sequential pattern approach. In this paper, we propose a new method to extract patterns from sequences which include multidimensional items and itemsets at the same time. In addition, the proposed approach incorporates background knowledge in the form of hierarchies over attributes.

The remainder of this paper is organized as follows, Section 2 describes the existing work in the classical and multidimensional sequential patterns. Section 3 introduces the problem statement as well as a running example. The method for extracting multidimensional itemset frequent patterns is described in Section 4. Section 5 presents experimental results from both quantitative and qualitative point of views and Section 6 concludes the paper.

2 Related Work

Let \mathcal{I} be a finite set of *items*. An *itemset* X is a non-empty subset of \mathcal{I} . A *sequence* S over \mathcal{I} is an ordered list $\langle X_1 \cdots X_n \rangle$, where X_i ($1 \leq i \leq n$, $n \in \mathbb{N}$) is an itemset. A sequence $T = \langle Y_1 \cdots Y_m \rangle$ is a **subsequence** of $S = \langle X_1 \cdots X_n \rangle$,

denoted by $T \preceq S$, if there exist indices $1 \leq i_1 < i_2 < \dots < i_m \leq n$ such that $Y_j \subseteq X_{i_j}$ for all $j = 1 \dots m$ and $m \leq n$. S is said to be a **supersequence** of T . Let $S_{DB} = \{S_1, S_2 \dots S_n\}$ be a database of sequences. The support of a sequence s in D is the proportion of sequences of D containing s . Given a **minsup** threshold, the problem of frequent sequential pattern mining consists in finding the set FS of sequences whose support is not less than **minsup**. Following the first work of Agrawal and Srikant [1] and the Apriori algorithm, many studies have contributed to the efficient mining of sequential patterns. The main algorithms are PrefixSpan [9], SPADE [17], SPAM [3], PSP [8], DISC [4], PAID [15], FAST [12]. All of these algorithms aim to discover sequential patterns from a set of sequences of itemsets.

Usually, the information in a sequence is based on several dimensions. Pinto et al [10] propose the first work by including for mining multidimensional sequential patterns, by including dimensions in the first or the last itemset of the sequence. But this works only for dimensions that remain constant over time, such as gender of the patient. Among other proposals addressed in this area, Yu et al [16] consider multidimensional sequential pattern mining in the web domain. Here, dimensions are pages, sessions and days. They present two algorithms: AprioriMD and PrefixMDSpan.

in real world applications, each dimension can be represented at different levels of granularity, by using a taxonomy. The interest lies in the capacity of extracting more or less general/specific sequential patterns and overcome problems of excessive granularity and low support. Although Srikant and Agrawal [13] combined the use of hierarchy of values in the extraction of association rules and sequential patterns, their approach is not scalable in a multidimensional context. Han et al [7] proposed a method for mining multiple level association rules in large databases. But their approach could not extract patterns containing items from different levels in the taxonomy. Appice et al [2] proposed SPADA, an algorithm for discovering multi-level spatial association rules. Plantevit et al [11] proposed M^3SP , an algorithm taking both multilevel and multidimensional aspects into account. M^3SP is able to find sequential patterns with the most appropriate level of granularity. Egho et al [6] proposed an extension for M^3SP for extracting both general and specific sequences, they iteratively applied M^3SP , decreasing threshold by one objects at each step. Their proposition allows the extraction of more interesting sequences than using a single **minsup** threshold.

3 Problem Statement

In this section we list some preliminary definitions needed to formalize the problem. First of all, we introduce a motivating example from a real data set related to the PMSI (Program of medical information systems). This French nationwide information system describes hospital activities from both economical and medical points of view. In this system, each hospitalization is related to the recording of administrative, demographical and medical data. Let S_{DB} be a database of multidimensional itemsets data sequences. Figure 1 illustrates such a database.

Patients	Trajectories
P_1	$\langle\langle(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (UH_{Paris}, C_1, \{p_1\}, \{drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})\rangle\rangle$
P_2	$\langle\langle(UH_{Paris}, C_1, \{p_1\}, \{drug_2\}), (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})\rangle\rangle$
P_3	$\langle\langle(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2, drug_3\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2, drug_4\})\rangle\rangle$
P_4	$\langle\langle(UH_{Paris}, C_1, \{p_2\}, \{drug_1, drug_2\}), (UH_{Paris}, R_2, \{p_3\}, \{drug_2\}), (GH_{Lyon}, R_2, \{p_2\}, \{drug_3\})\rangle\rangle$

Fig. 1. An example of a database of patient trajectories

Definition 1. (*Dimensions and specialization down(d)*) A dimension (D, \leq) is a partially ordered set where D is the set of all items of dimension. For a given $d \in D$, $down(d)$ (resp. $up(d)$) denotes the set of all specializations $\{x \in D | x \leq d\}$ (resp. generalizations $\{x \in D | d \leq x\}$) of d .

Example 1. Figure 2 shows two dimensions (hospital and diagnosis). For hospital dimension, $D_{hospital} = \{T_{hospital}, UH, GH, UH_{Paris}, UH_{Nancy}, GH_{Paris}, GH_{Lyon}\}$ and $UH_{Paris} \in down(UH)$ as UH_{Paris} is a direct descendant of UH .

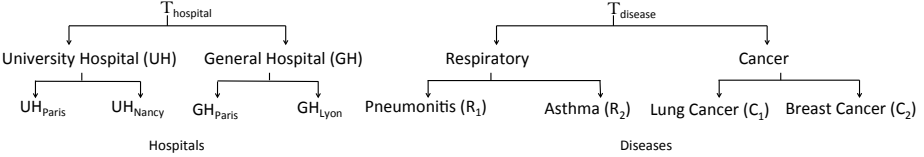


Fig. 2. Hospital and diagnoses taxonomies

By taking into account the multidimensional items and the sets of items, we define an event as follows.

Definition 2. (*Event*) An event $e = (d_1, \dots, d_n, itemset_{n+1}, \dots, itemset_{n+m})$ is a vector of n multidimensional items and m sets of items where $d_i \in D_i, i = 1, \dots, n$. Given two events $e = (d_1, \dots, d_n, itemset_{n+1}, \dots, itemset_{n+m})$ and $e' = (d'_1, \dots, d'_n, itemset'_{n+1}, \dots, itemset'_{n+m})$, e is more general than e' , denoted by $e' \leq_e e$, if and only if:

- $\forall i ; 1 \leq i \leq n ; d'_i \in down(d_i)$.
- $\forall j ; 1 \leq j \leq m ; itemset_{n+j} \subseteq itemset'_{n+j}$.

Example 2. $e' = (UH_{Paris}, C_1, \{p_1, p_2, p_3\}, \{drug_2, drug_3, drug_4\})$ is an event, where:

- UH_{Paris}, C_1 are two multidimensional items representing the two dimensions (hospital and diagnosis).
- $\{p_1, p_2, p_3\}, \{drug_2, drug_3, drug_4\}$ are two sets of items representing the medical procedures and the medical drugs.

The event $e = (UH, T_{disease}, \{p_1, p_2\}, \{drug_2, drug_3\})$ is more general than e' , $e' \leq_e e$, because of:

- $UH_{Paris} \in down(UH)$ and $C_1 \in down(T_{disease})$.
- $\{p_1, p_2\} \subseteq \{p_1, p_2, p_3\}$ and $\{drug_3, drug_4\} \subseteq \{drug_2, drug_3, drug_4\}$.

A multidimensional itemsets data sequence is composed of events.

Definition 3. (*Multidimensional Itemsets Sequence*) A multidimensional itemsets sequence $s = \langle e_1, e_2, \dots, e_l \rangle$ is an ordered list of events e_i . Given two multidimensional itemsets sequences $s = \langle e_1, e_2, \dots, e_l \rangle$ and $s' = \langle e'_1, e'_2, \dots, e'_l \rangle$, s is more general than s' , denoted by $s \leq_s s'$, if there exist indices $1 \leq i_1 < i_2 < \dots < i_l \leq l'$ such that $e_j \leq_e e'_{i_j}$ for all $j = 1 \dots l$ and $l \leq l'$.

Example 3. The multidimensional itemsets sequence $s = \langle (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2, drug_3\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2, drug_4\}) \rangle$ is a sequence of two events. It expresses the fact that a patient was admitted to the University Hospital of Paris UH_{Paris} for a lung cancer C_1 , underwent procedures p_1 and p_2 and was treated with $\{drug_1, drug_2, drug_3\}$, then he went to the General Hospital of Lyon GH_{Lyon} for pneumonitis R_1 where he underwent procedure p_2 and received $\{drug_2, drug_4\}$.

The sequence $s' = \langle (UH_{Paris}, Cancer, \{p_1\}, \{drug_1, drug_2\}) \rangle$ is more general than s , $s \leq_s s'$, because $(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2, drug_3\}) \leq_e (UH_{Paris}, Cancer, \{p_1\}, \{drug_1, drug_2\})$.

Definition 4. (*Patient Trajectory*) A patient trajectory is defined as a multidimensional itemsets sequence.

Example 4. In Table 1, the multidimensional itemsets sequence $s = \langle (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (UH_{Paris}, C_1, \{p_1\}, \{drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\}) \rangle$ represents the trajectory for the patient P_1 .

Let $\text{supp}(s)$ be the number of sequences that includes s in S_{DB} . Furthermore σ be a minimum support threshold specified by the end-user.

Definition 5. (*Most Specific Frequent Multidimensional Itemsets Sequence*) Let s be multidimensional itemsets sequence, we say that s is the most specific frequent multidimensional itemsets sequence in S_{DB} , if and only if: $\text{supp}(s) \geq \sigma$ and $\nexists s' \in S_{DB}$, where $\text{supp}(s) = \text{supp}(s')$ and $s \leq_s s'$.

The problem of mining multidimensional itemsets sequences is to extract the set of all most specific frequent multidimensional itemsets sequence in S_{DB} such as $\text{supp}(s) \geq \sigma$. By using the dimensions we can extract general or specific patterns and overcome problems of excessive granularities and low supports.

Example 5. Let $\sigma = 0.75$ (i.e. a sequence is frequent if it appears at least three times in S_{DB}). The sequence $s_1 = \langle (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\}) \rangle$ is frequent. $s_2 = \langle (UH, Cancer, \{p_1, p_2\}, \{drug_1, drug_2\}), (GH, Respiratory, \{p_2\}, \{drug_2\}) \rangle$ is also frequent. Nevertheless, s_2 is not kept since it is too general compared to s_1 .

4 Mining Multidimensional Itemsets Sequential Patterns

In this section, we present the MMISP (*Mining Multidimensional Itemsets Sequential Patterns*) algorithm for extracting multidimensional itemsets sequential patterns with different levels of granularity over each dimension. MMISP follows

a bottom-up approach by first focusing on extracting frequent multidimensional items that can exist at different level of granularity, then it considers the itemsets part of the events and compute the support of every item in S_{DB} for each itemset. After these two steps, frequent multidimensional items and frequent itemsets are combined to generate events. In the final step, the frequent events are mapped to a new representation and a standard sequential mining algorithm is applied to enumerate multidimensional itemsets sequential patterns.

In the next subsections, we provide the details of each step of our work and discuss the different challenges.

4.1 Generating Frequent Multidimensional Items

MMISP starts by processing the n multidimensional items of the events in the sequences. Basically it considers three types of dimensions: a temporal dimension D_t , a set of analysis dimension D_A and a set of reference dimension D_R . MMISP splits S_{DB} into blocks according to dimension D_R . Then, MMISP sorts each block according to the temporal dimension D_t . This is a classic way of partitioning the database and was introduced in [11]. The tuples of n multidimensional items appearing in an event are defined w.r.t. analysis dimensions D_A . The support of n multidimensional items is computed according to dimension of D_R . It is the ratio of the number of blocks supporting the n multidimensional items over the total number of blocks.

Date	Hospital	Diagnosis
1	UH _{Paris}	C ₁
2	UH _{Paris}	C ₁
3	GH _{Lyon}	R ₁

Block: *Patient*₁

Date	Hospital	Diagnosis
1	UH _{Paris}	C ₁
2	GH _{Lyon}	R ₁

Block: *Patient*₃

Date	Hospital	Diagnosis
1	UH _{Paris}	C ₁
2	UH _{Paris}	R ₂
3	GH _{Lyon}	R ₂

Block: *Patient*₄

Date	Hospital	Diagnosis
1	UH _{Paris}	C ₁
2	UH _{Paris}	C ₁
3	GH _{Lyon}	R ₁

Block: *Patient*₂

Fig. 3. Block partition of the database according to $D_R=\{\text{Patient}\}$

Example 6. In the running example, H (hospitals) and D (diseases) are the analysis dimensions, $Date$ is the temporal dimension, and P (patients) is the reference dimension. By using P (patients) to split the dataset, we obtain four blocks defined by *Patient*₁, *Patient*₂, *Patient*₃ and *Patient*₄ as shown in Figure 3.

To simplify our works we will represent the n multidimensional items of the event as follows:

Definition 6. (*multidimensional component*) Given a dimension (D, \leq) , a multidimensional component over D , denoted (mdc, \leq_{mdc}) , is a tuple (d_1, \dots, d_n) where $d_i \in D$, $i = 1, \dots, n$. For two given multidimensional components $mdc = (d_1, \dots, d_n)$ and $mdc' = (d'_1, \dots, d'_n)$, $mdc' \leq_{mdc} mdc$ denotes that mdc is more general than mdc' , if for every $i = 1, \dots, n$, $d'_i \in \text{down}(d_i)$.

Example 7. Let $(UH_{Paris}, Lung\ Cancer)$ and $(UH, Cancer)$ be two multidimensional components. $(UH_{Paris}, Lung\ Cancer) \leq_{mdc} (UH, Cancer)$ because $UH_{Paris} \in down(UH)$ and $Lung\ Cancer \in down(Cancer)$.

The first steps in MMISP is generation all the frequent multidimensional components. This generation is given by the product of all partially ordered sets of the dimensions. The result of this product is a semilattice which has a top element (T_1, \dots, T_m) and each node in this semilattice is a multidimensional component. Extracting only the frequent multidimensional components can be done by choosing `minsup` and building the iceberg semi-lattice. The iceberg semi-lattice is a semi-lattice where its elements have a support greater than `minsup`. Figure 4 shows iceberg semi-lattice generated by the product of the two partially ordered sets (hospital and diagnosis) in Figure 2 with `minsup` = $\frac{3}{4}$ patients.

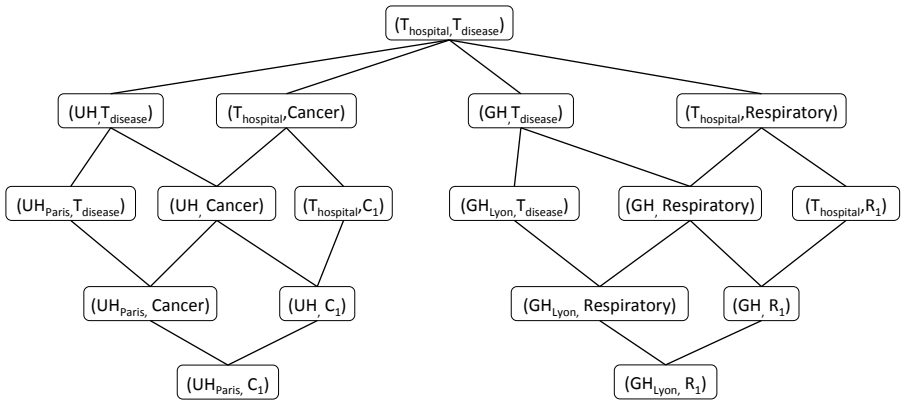


Fig. 4. Iceberg semilattice generated by the product of the two partially ordered sets (hospital and diagnosis) in Figure 2 with `minsup` = $\frac{3}{4}$ patient

Handling the product of several partial order sets is a cumbersome process. The result of a product is exponential in the number of partial order sets and the cardinality of each set. So, we present a simple and efficient algorithm to generate all frequent multidimensional components.

Following the previous partitioning, algorithm generates all the frequent multidimensional components as follows: firstly, we generate the most general multidimensional component, that is (T_1, \dots, T_n) . In our running example, we have two dimensions (hospital and disease), so the most general multidimensional component is $(T_{hospital}, T_{disease})$. Then, the algorithm generates all multidimensional components of the form $(T_1, \dots, T_{i-1}, d_i, T_{i+1}, \dots, T_n)$ where $d_i \in down(T_i)$. We take only the frequent multidimensional component which has support greater than σ . In the running example and for $\sigma = 75\%$ (3 blocks from 4), there are four new frequent multidimensional components: $(UH, T_{disease})$, $(GH, T_{disease})$, $(T_{hospital}, Respiratory)$ and $(T_{hospital}, Cancer)$.

The recursive generation of the new multidimensional components continues by using each previously generated frequent multidimensional component (a). This is done with an indexing method that identifies an integer z which is the position of the last dimension in a and is not top T . For example if $a=(UH, T_{Disease})$, z is equal to one, which is the first dimension (hospital) because the value for the hospital dimension (UH) and the second dimension (disease) has the value $T_{Disease}$.

For each dimension d_k in a , where $k \in [z, m]$, we replace d_k with each of its specialization from the set $down(d_k)$. For example, if $a=(UH, T_{Disease})$, we have $z=1$ and we can generate four new mdc_s : $\{(UH_{Paris}, T_{Disease}), (UH_{Nancy}, T_{Disease}), (UH, Respiratory), (UH, Cancer)\}$. The first and the second multidimensional components are generated by replacing UH by $down(UH) = \{UH_{Paris}, UH_{Nancy}\}$, the third and the fourth multidimensional components are generated by replacing $T_{Disease}$ by $down(T_{Disease}) = \{Respiratory, Cancer\}$.

At each step, we select only the frequent multidimensional components. For our previously example with $\sigma = 75\%$, $\{(UH_{Paris}, T_{Disease}), (UH, Cancer)\}$ are the new frequent multidimensional components generated by $(UH, T_{Disease})$.

Finally, from all frequent multidimensional components generated, we select only the most specific multidimensional component.

Definition 7. (*Most specific multidimensional component*) Let a be multidimensional component, we can say that, a is the most specific multidimensional component, if and only if $\nexists a'$ multidimensional component, where $supp(a) = supp(a')$ and $a' \leq_{mdc} a$.

Example 8. Figure 5 illustrates the generation of all frequent multidimensional components on the running example with $\sigma = \frac{3}{4}$. The most specific components are (UH_{Paris}, C_1) and (GH_{Lyon}, R_1) .

Table 1. The most specific frequent multidimensional components

Frequent multidimensional component
(UH_{Paris}, C_1)
(GH_{Lyon}, R_1)

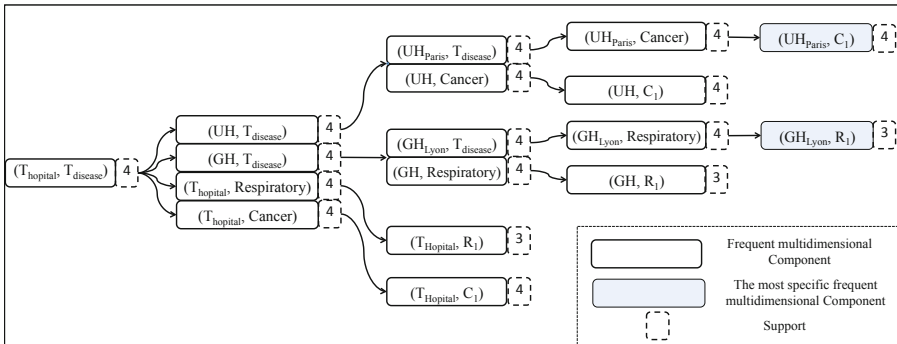


Fig. 5. Frequent multidimensional components generation

4.2 Generating Frequent Itemsets

In this step, MMISP focuses on m itemsets part of the events, $(d_1, \dots, d_n, \text{itemset}_{n+1}, \dots, \text{itemset}_{n+m})$. We will study separately each itemset in this part. Basically, this step aims at extracting the set of all items that are frequent in a sequence of length 1. Recall that, in level-wise approaches, either itemset-extension or sequence-extension can be considered. For example, if we have a sequence $s_1 = \langle \{1, 2, 3\} \rangle$, then $s_2 = \langle \{1, 2, 3\}\{4\} \rangle$ is an extended sequence of s_1 and $s_3 = \langle \{1, 2, 3, 4\} \rangle$ is an itemset-extended sequence of s_1 . In our context we only consider itemset-extension. This task can be easily done by adapting any standard sequential pattern algorithm to extract only the sequence of length 1.

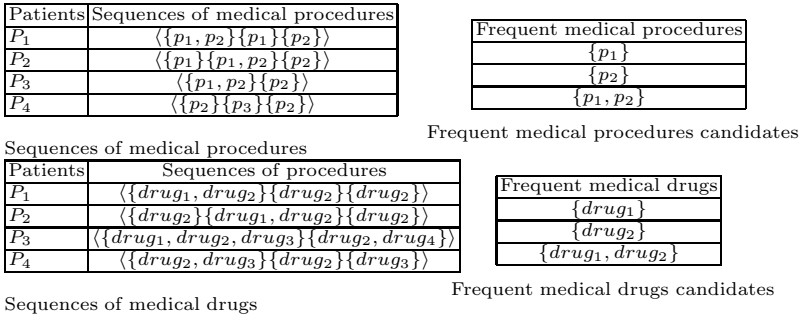


Fig. 6. The frequent itemset generated

Example 9. Figure 6 shows the sequences of medical procedures and medical drugs for patients, and also the frequent medical procedures and medical drugs candidates for $\sigma = \frac{3}{4}$.

4.3 Generating Frequent Events

Generating frequent events is achieved by combining frequent multidimensional components with frequent itemsets. This task has been done by building a prefix tree such that the first level in this tree is composed of the frequent multidimensional components and from the second level to leaves, each level is composed of the frequent itemset candidates for each itemset part in the vector of itemsets. More precisely, each branch in the tree represents an event. Then a scan is performed over the database to prune irrelevant events from the tree. For example, Figure 7 illustrates the tree before and after pruning infrequent events for $\sigma = \frac{3}{4}$.

4.4 Extracting Frequent Multidimensional Itemsets Pattern

Frequent sequences can then be mined by using any standard sequential pattern mining algorithm. As these algorithms require that the dataset to be mined is composed of pairs in the form (id, seq) , where id is a sequence identifier and seq is a sequence of itemsets, we transform the initial dataset as follows:

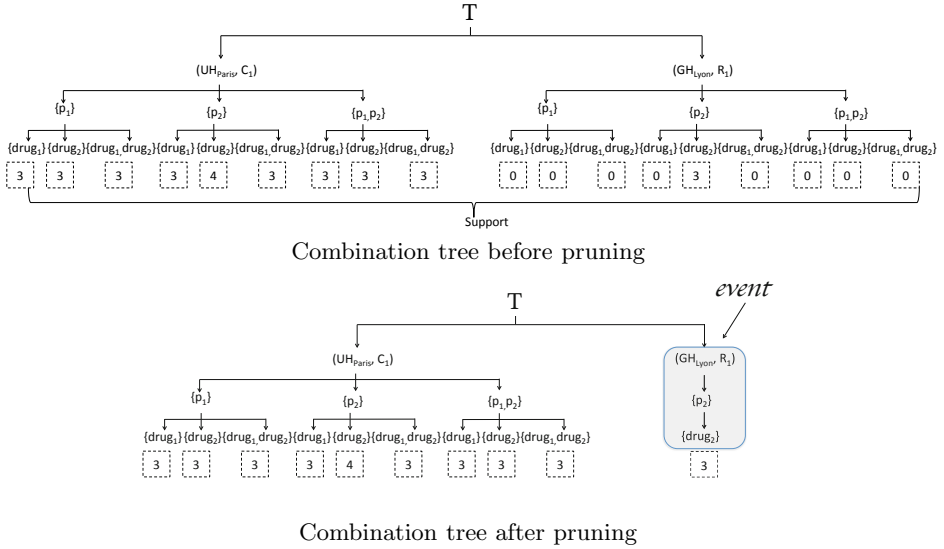


Fig. 7. An example of the tree for generating frequent events before and after the pruning

- Each branch in the prefix tree after pruning is assigned a unique id which will be used during the mining operation. This is illustrated in Table 2 .
- Each block (patient) is assigned a unique id of the form P_i .
- Every block b is transformed into a pair $(P_i, \mathbb{S}(p_i))$, where $\mathbb{S}(P_i)$ is built according to the date and the content of the blocks. The final result is reported in Table 3.

A standard sequence mining algorithm can be applied on the transformed database.

Table 2. Identification each branch (Event) in T

event-id	Frequent Event
e_1	$(UH_{Paris}, C_1, \{p_1\}, \{drug_1\})$
e_2	$(UH_{Paris}, C_1, \{p_1\}, \{drug_2\})$
e_3	$(UH_{Paris}, C_1, \{p_1\}, \{drug_1, drug_2\})$
e_4	$(UH_{Paris}, C_1, \{p_2\}, \{drug_1\})$
e_5	$(UH_{Paris}, C_1, \{p_2\}, \{drug_2\})$
e_6	$(UH_{Paris}, C_1, \{p_2\}, \{drug_1, drug_2\})$
e_7	$(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1\})$
e_8	$(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_2\})$
e_9	$(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\})$
e_{10}	$(GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})$

Table 3. Transformed database

id	Sequence data
P_1	$\langle \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\} \{e_2\} \{e_{10}\} \rangle$
P_2	$\langle \{e_2\} \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\} \{e_{10}\} \rangle$
P_3	$\langle \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\} \{e_{10}\} \rangle$
P_4	$\langle \{e_5\} \rangle$

Then, the extraction of frequent sequences can be carried out. With $\sigma = 0.75$, the pattern $\langle\{e_9\}\{e_{10}\}\rangle$ is frequent. This sequence corresponds to $\langle(UH_{Paris}, C_1 \{p_1, p_2\}, \{drug_1, drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})\rangle$ by using the identification in Table 2.

5 Experiments

We conduct experiments on both real and synthetic datasets. The algorithm is implemented in Java and the experiments are carried out on a MacBook Pro with a 2.5GHz Intel Core i5, 4GB of RAM Memory running OS X 10.6.8. The extraction of sequential patterns is based on the public implementation of CloSpan algorithm [14]. We use the implementation supplied by the IlliMine¹ toolkit.

In order to assess the effectiveness of our approach, we run several experiments on the PMSI dataset. This database includes the following informations for each stay: patient id and gender, hospital id, principal diagnosis and date of the stay, a set of associated diagnosis and a set of medical procedures. Our dataset contains 486 patients suffering from lung cancer and living in the East of France. The average length of data sequences is 27. The data is encoded using controlled vocabularies. In particular, diagnoses are encoded with the International Classification of Diseases (ICD10)². This classification is used as an input taxonomy for MMISP. The ICD10 can be seen as a tree with two levels. As illustrated in Figure 8, 3-characters codes such as C34 (Lung cancer) have specializations: C340 is cancer of the main bronchus, C341 is cancer of upper lobe etc.

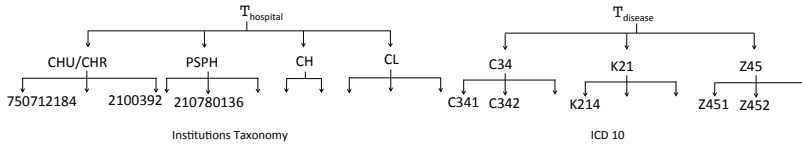


Fig. 8. Examples of taxonomies used in multilevel sequential pattern mining

Patients	Trajectories
P_1	$\langle\langle C341, 750712184, \{ZBQK002\}, \{D123, K573, C780\} \rangle, \langle Z452, 580780138, \{ZZQK002\}, \{C189\} \rangle, \dots \rangle$
P_2	$\langle\langle C770, 10000017, \{ZBQK002\}, \{C189\} \rangle, \langle C770, 210780581, \{ZZQK002, YYY030\}, \{D123, T573\} \rangle, \dots \rangle$
P_3	$\langle\langle H259, 210780110, \{YYYY030\}, \{D123, T573\} \rangle, \langle H259, 210780110, \{ZZQK002\}, \{D123, T573\} \rangle, \dots \rangle$
P_4	$\langle\langle R91, 210780136, \{YYYY030\}, \{D123, C780\} \rangle, \langle C07, 210780136, \{ZBQK002\}, \{C780\} \rangle, \dots \rangle$

Fig. 9. Care trajectories of 4 patients

Figure 9 shows an example of care trajectories described over two dimensions (diagnosis, hospital ID) coupled with two sets of medical procedures and associated diagnosis. For example $\langle C341, 750712184, \{ZBQK002\}, \{D123, K573, C780\} \rangle$ represents the stay of a patient in the University Hospital of Dijon (coded

¹ <http://illimine.cs.uiuc.edu/>

² <http://apps.who.int/classifications/apps/icd/icd10online/>

as 750712184) treated for a lung cancer (C341), where the patient underwent chest radiography (coded as ZBQK002) and during his treatment, he has the set of associated diagnosis $\{D123, K573, C780\}$.

The experiments extract multidimensional sequential patterns for describing and analyzing patient trajectories. For this experiment the support value is set to 15 (i.e. $\sigma = 0.03$). MMISP generates 156 different frequent trajectories. Figure 10 shows some results of the experiment. *Pattern 2* can be interpreted as follows: 40% of patients had a hospitalization in the University Hospital of Dijon (750712184) for any diagnosis (ALL), where they underwent a chest radiography (coded as ZBQK002) and an Electrocardiography (coded as DEQP003), with supplementary billing (coded as YYYY030); they had a malignant tumor of the lung as associated diagnosis. Then, the same patients had another stay for acute respiratory failure (J960), and they underwent tests with supplementary billing (coded as YYYY030). This second stay could occur in any hospital (ALL) and had the same associated diagnosis(C349).

Id	Support	Trajectory Patterns
1	53%	$((710780263, ALL, \{DEQP003\}, \{C349\}))$
2	40%	$((750712184, ALL, \{ZBQK002, YYYY030, DEQP003\}, \{C349\})(ALL, J960, \{YYYY030\}, \{C349\}))$
3	34%	$((710780263, ALL, \{ZBQK002, YYYY030, DEQP003\}, \{C349\})(710780263, ALL, \{ZBQK002, YYYY030, DEQP003\}, \{C349\}))$

Fig. 10. Some healthcare patients trajectories obtained by MMISP

In the second experiment, we study the scalability of the approach. We consider the number of extracted patterns and the running time with respect to two different parameters, the number of dimensions and the average length of itemsets in the event. The first batch of synthetic data generated contains 10000 sequences defined over (2, 3, 4 and 5) analysis dimensions. Each sequence contains 30 events and each event is described, in average, by 15 items in the itemset. Each dimension is defined over 5 levels of granularity between elements of each analysis dimension. Figure 11 reports the results according to different values of support threshold for different number of dimension in event. The running time

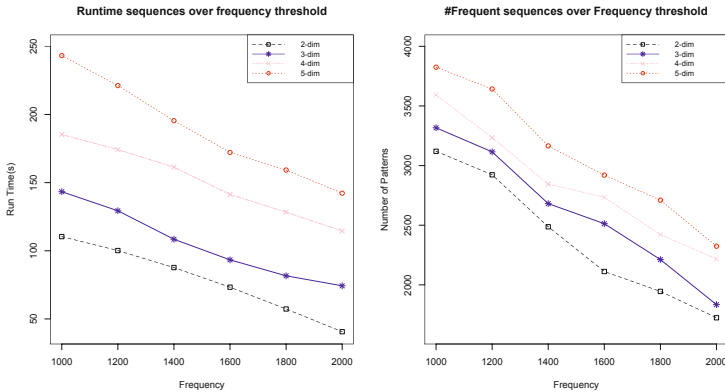


Fig. 11. Running Time (left) and Number of extracted pattern (right) obtained by MMISP with varying in the number of dimension

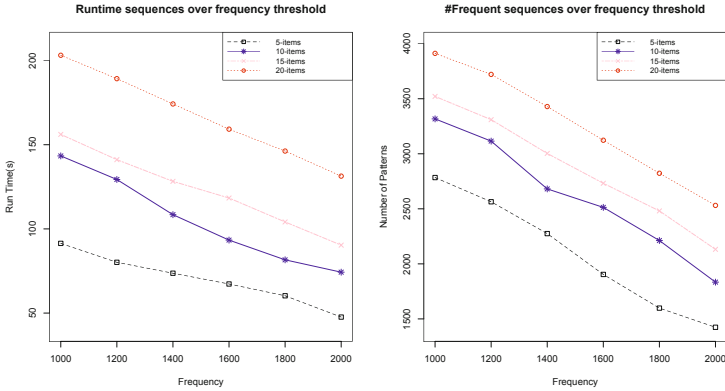


Fig. 12. Number of extracted pattern (right) and Running Time (left) obtained by MMISP with varying itemsets’ cardinalities

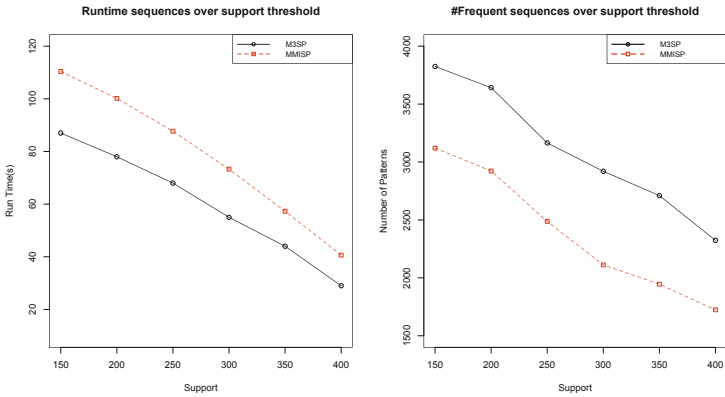


Fig. 13. Running Time (left) and Number of extracted pattern (right) obtained by MMISP and M^3SP over the synthetic dataset

increases for each newly added dimension. The second batch of generated synthetic data contains 10000 sequences with varying number of items 5, 10, 15 and 20. The sequences in the four generated data sets have an average cardinality of 30 events, by 3 dimensions. The dimensions are defined over 5 levels of granularity between elements of each dimension. Figure 12 reports the results according to different values of support threshold for different lengths of itemsets.

Another experiment is aimed at comparing the performance of MMISP with M^3SP on a synthetic dataset. In comparison we consider both the number of extracted patterns and the running time. The synthetic data generated contains 10000 sequences defined over two dimensions with one itemsets described by 5 items. Figure 13 reports the results according to different values of support threshold for both M^3SP and MMISP. MMISP is able to extract less patterns than M^3SP while from the point of view of time execution the two approaches

show comparable performances. The reduced size of the MMISP results is related to its ability in extracting a multidimensional itemsets sequential patterns.

6 Conclusion

In this paper, we propose a new approach to mine multidimensional itemset sequential patterns. Our approach is based on multidimensional items and the set of items. We provide formal definitions and propose a new algorithm MMISP to mine this new kind of pattern. We conduct experiments on both real and synthetic datasets. The method was applied on real-world data where the problem was to mine healthcare patients trajectories and gave potential interesting patterns for healthcare specialists.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, ICDE 1995, pp. 3–14. IEEE Computer Society, Washington, DC (1995)
2. Appice, A., Berardi, M., Ceci, M., Malerba, D.: Mining and Filtering Multilevel Spatial Association Rules with ARES (2005)
3. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: KDD, pp. 429–435 (2002)
4. Chiu, D.-Y., Wu, Y.-H., Chen, A.L.P.: An efficient algorithm for mining frequent sequences by a new strategy without support counting. In: ICDE, pp. 375–386 (2004)
5. Cohen, J., Eshleman, J., Hagenbuch, B., Kent, J., Pedrotti, C., Sherry, G., Waas, F.: Online expansion of largescale data warehouses. In: PVLDB, vol. 4(12), pp. 1249–1259 (2011)
6. Egho, E., Jay, N., Raïssi, C., Napoli, A.: A FCA-based analysis of sequential care trajectories. In: Napoli, A., Vychodil, V. (eds.) The Eighth International Conference on Concept Lattices and their Applications - CLA 2011, Nancy, France. INRIA Nancy Grand Est - LORIA (October 2011)
7. Han, J., Fu, Y.: Mining multiple-level association rules in large databases. *IEEE Transactions on Knowledge and Data Engineering* 11(5), 798–805 (1999)
8. Massegli, F., Cathala, F., Poncelet, P.: The PSP approach for mining sequential patterns. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 176–184. Springer, Heidelberg (1998)
9. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224 (2001)
10. Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q., Dayal, U.: Multi-dimensional sequential pattern mining. In: CIKM, pp. 81–88 (2001)
11. Plantevit, M., Laurent, A., Laurent, D., Teisseire, M., Choong, Y.W.: Mining multidimensional and multilevel sequential patterns. *TKDD* 4(1), 1–37 (2010)
12. Salvemini, E., Fumarola, F., Malerba, D., Han, J.: FAST sequence mining based on sparse id-lists. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 316–325. Springer, Heidelberg (2011)

13. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
14. Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large datasets. In: SDM, pp. 166–177 (2003)
15. Yang, Z., Kitsuregawa, M., Wang, Y.: Paid: Mining sequential patterns by passed item deduction in large databases. In: IDEAS, pp. 113–120 (2006)
16. Yu, C.-C., Chen, Y.-L.: Mining sequential patterns from multidimensional sequence data. *IEEE Trans. Knowl. Data Eng.* 17(1), 136–140 (2005)
17. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. *Mach. Learn.* 42(1-2), 31–60 (2001)
18. Zhang, C., Hu, K., Chen, Z., Chen, L., Dong, Y.: Approxmgmsp: A scalable method of mining approximate multidimensional sequential patterns on distributed system. In: FSKD (2), pp. 730–734 (2007)

Graph-Based Approaches to Clustering Network-Constrained Trajectory Data

Mohamed Khalil El Mahrsi^{1,2} and Fabrice Rossi²

¹ Télécom ParisTech, Département INFRES
46, rue Barrault 75634 Paris CEDEX 13, France
`khalil.mahrsi@telecom-paristech.fr`

² Équipe SAMM EA 4543, Université Paris I Panthéon-Sorbonne
90, rue de Tolbiac 75634 Paris CEDEX 13, France
`fabrice.rossi@univ-paris1.fr`

Abstract. Clustering trajectory data attracted considerable attention in the last few years. Most of prior work assumed that moving objects can move freely in an euclidean space and did not consider the eventual presence of an underlying road network and its influence on evaluating the similarity between trajectories. In this paper, we present an approach to clustering such network-constrained trajectory data. More precisely we aim at discovering groups of road segments that are often travelled by the same trajectories. To achieve this end, we model the interactions between segments w.r.t. their similarity as a weighted graph to which we apply a community detection algorithm to discover meaningful clusters. We showcase our proposition through experimental results obtained on synthetic datasets.

Keywords: similarity, clustering, moving objects, trajectories, road network, graph.

1 Introduction

Traffic congestion has become a major problem that affects many human activities on a daily basis, resulting in both serious transportation delays and environmental damages. Monitoring the state of the road network is commonly conducted by using dedicated sensors that register the number of vehicles passing by the section where they are installed. The prohibitive cost of deploying and maintaining such sensors limits their deployment to the highways and the road network's main arteries. Subsequently, the collected data portray a partial and incomplete state of the road network, thus complicating data mining tasks that aim at extracting useful knowledge about flow dynamics and the behavior of drivers moving along the network.

An alternative (or complementary) approach to addressing these shortcomings may consist in analyzing GPS logs collected using location-aware devices (e.g. classic GPS, smartphones, PDAs, etc.). These logs can be acquired through probing vehicles, dedicated data acquisition campaigns (using buses, taxis or an

enterprise’s fleet of vehicles) or even by means of a crowdsourced approach where different individuals willingly contribute by uploading their different commute logs. Therefore, it is perfectly feasible to collect large amounts of trajectory data that can be stored in dedicated databases (known as Moving Object Databases [1]). These data offer a better coverage of the road network and can be, later on, explored using data mining and statistical learning techniques.

Clustering is one of such techniques. Prior work on trajectory data clustering focused mainly on the case where moving objects move freely in a euclidean space [2–5]. By doing so, these approaches did not account for the presence, in the case of car trajectories as well as in other cases, of an underlying network that constrains the movement. The network’s constraints, however, do play a paramount role in determining the similarity between the trajectories to be clustered. Moreover, the majority of these approaches relied on the use of density-based clustering which makes them vulnerable to the way the parameters of the clustering algorithm are selected.

In [6], we presented a framework for clustering network-constrained trajectories using a graph-based approach. This framework was directed towards discovering groups of similar trajectories that moved along the same parts of the road network. The hierarchical, non-parametric algorithm that we used in the clustering step made our framework flexible and suitable for exploring the discovered groups of trajectories at various levels of detail: the user can start with a limited number of high-level, coarse clusters and delve (by means of consecutive zooming) in the refinement of the clusters he deems interesting.

The work presented in this paper builds upon the one undertaken in [6]. We extend our framework to the case of road segments as we try to discover relevant groups of segments that are commonly used and explored together by a considerable number of trajectories. Our contributions can be summarized as follows:

- We define a similarity measure that evaluates the resemblance between pairs of road segments based on the trajectories that travelled along both of them;
- We use a graph representation to model interactions between the different road segments. The resulting similarity graph is partitioned using modularity-based community detection in order to discover a hierarchy of nested clusters of road segments;
- We test our proposition on synthetic datasets and showcase how it can be used, in association with the technique we presented in [6], for understanding and characterizing the traffic in the road network.

The rest of this paper is organized as follows. In Section 2, we present the network-constrained trajectories data model and we formalize our segment clustering problem. Our segment clustering approach is described in detail in Section 3. Section 4 discusses the computational complexity of our proposition as well as how the discovered clusters can be interpreted and used, complementarily with the trajectory clusters presented in [6], in order to discover useful knowledge about the flow dynamics and traffic in the road network. Experimental results

are presented in Section 5. Related work is discussed in Section 6. Finally, Section 7 concludes the paper.

2 Data Representation and Problem Statement

We opt for the symbolic data representation which is the model of choice adopted for representing network-constrained trajectories in most of prior work [7–10]. In this model, the road network is modeled as a graph, defined as follows.

Definition 1 (Road Network). *The road network is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{S})$. The set of vertices \mathcal{V} represents intersections and terminal points of roads whereas the set of directed edges \mathcal{S} represents the road segments interconnecting them. A directed edge $s = (v_i, v_j)$ indicates that a road segment links the two nodes v_i and v_j and that it can be traveled from v_i in the direction of v_j but not the other way around (unless another edge states otherwise).*

Given this graph representation, moving objects (i.e. vehicles) moving along the road network produce trajectories that can be modeled conformably to the following definition.

Definition 2 (Constrained Trajectory). *A constrained trajectory T that travels along the road network \mathcal{G} can be modeled as a sequence of visited segments:*

$$T = \langle id, \{s_1, s_2, \dots, s_l\} \rangle$$

id being the identifier of the trajectory, l its length (i.e. number of segments) and $\forall 1 \leq i < l, s_i$ and s_{i+1} are connected segments belonging to \mathcal{S} .

In a real-case scenario, trajectories are collected as GPS logs (sequences of latitude and longitude points) on which a map matching technique (e.g. [7, 9]) is applied in order to produce the sequence of traveled segments. The map matching step is out of the scope of this paper. Hence, we suppose that the trajectories are already and correctly map matched to the corresponding road segments.

Finally, we formalize the road segment clustering problem that we study in this paper as follows.

Definition 3 (Road Segment Clustering Problem). *Given a road network represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ and a set of trajectories $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ that traveled along it, road segment clustering aims to partition the set of road segments \mathcal{S} into a set of disjoint clusters $\mathcal{C}_{\mathcal{S}} = \{C_1, C_2, \dots, C_K\}$ in such a fashion that:*

- *Segments grouped in the same cluster C_i are visited by a considerable amount of common trajectories (i.e. a trajectory T that visits a segment $s \in C_i$ also visits a fair amount of segments in this same cluster);*
- *Segments belonging to two different clusters C_i and C_j are visited by as few common trajectories as possible (i.e. they are unlikely to be part of a same trajectory).*

3 A Graph-Based Approach to Road Segment Clustering

We now present our solution to the road segment clustering problem introduced in the previous section. First, we define a similarity measure between road segments based on the comparison of the common trajectories that visited them (Section 3.1). Based on this measure, we build a graph depicting the relationships between different road segments (Section 3.2). The graph is then partitioned using a modularity-based community detection algorithm in order to discover a hierarchy of nested segment clusters (Section 3.3).

3.1 Road Segment Similarity

Similarly to the bag-of-words model (where a text is considered as an unordered collection of words), we consider each road segment as a bag-of-trajectories that visited it (i.e. $\forall s \in \mathcal{S}, s \equiv \{T \in \mathcal{T} : s \in T\}$).

In order to compare two road segments s_i and s_j , one can simply observe how often they co-appear in trajectories (i.e. calculate $|\{T \in \mathcal{T} : s_i \in T \wedge s_j \in T\}|$). The larger the number of concomitant appearances of both segments is, the more they are considered similar. However, different trajectories do not hold the same discriminative power when it comes to characterizing the similarity between road segments they visit: a lengthy trajectory that travels along a considerable number of road segments is not very informative when judging the similarity between two segments in particular and, vice versa, short trajectories are highly relevant to the formation of the cluster that contains the segments they visit.

We account for this observation by devising a tfidf-like weighting strategy where the contribution of each trajectory is proportional to its length. The weight $\omega_{T,s}$ assigned to trajectory T while inspecting a road segment s is expressed in formula (1):

$$\omega_{T,s} = \frac{n_{s,T}}{\sum_{T' \in \mathcal{T}} n_{s,T'}} \cdot \log \frac{|\mathcal{S}|}{|s \in \mathcal{S} : s \in T|} \quad (1)$$

The first part in this weight calculates the contribution of T to the segment s by calculating the ratio between the number of appearances $n_{s,T}$ of s in T and the total number of appearances of s in the whole dataset of trajectories \mathcal{T} . Since multiple visits of a same road segment are very rare, this part is often equal to $\frac{1}{|\{T \in \mathcal{T} : s \in T\}|}$. The second part evaluates the importance of the trajectory across the whole set of road segments: the more segments a trajectory visits the less important it becomes and vice versa.

We use a cosine similarity to measure the similarity between two road segments e_i and e_j as expressed in formula (2):

$$\text{Similarity}(s_i, s_j) = \frac{\sum_{T \in \mathcal{T}} \omega_{T,s_i} \cdot \omega_{T,s_j}}{\sqrt{\sum_{T \in \mathcal{T}} \omega_{T,s_i}^2} \cdot \sqrt{\sum_{T \in \mathcal{T}} \omega_{T,s_j}^2}} \quad (2)$$

3.2 Road Segment Similarity Graph

We model the similarity relationships between road segments using an undirected, weighted graph $\mathcal{SG}_{\mathcal{S}} = (\mathcal{S}, \mathcal{E}, \mathcal{W})$. Each road segment in \mathcal{S} is mapped to a vertex in $\mathcal{SG}_{\mathcal{S}}$. An edge between a pair of segments s_i and s_j exists if and only if $\text{Similarity}(s_i, s_j) > 0$ (i.e. if there is at least one common trajectory that crossed both segments). In which case the similarity is assigned as a weight to that edge. This concept of similarity graph is depicted in Fig. 1.

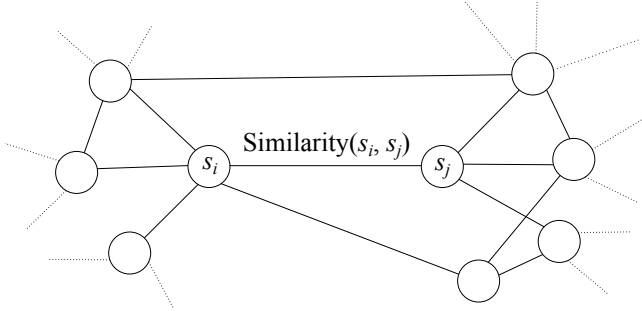


Fig. 1. Excerpt from a segment similarity graph. Vertices represent the studied road segments while weighted edges indicate the presence and strength of the similarity between pairs of segments.

The main advantage of using this graph representation, besides being natural and easy to understand, is that it does not invent an "artificial" similarity between totally incompatible road segments. On the contrary, it emphasizes on the fact that road segments that do not share common trajectories are independent and should, therefore, not be "immediately" grouped in the same cluster since there is no similarity edge linking them.

3.3 Clustering the Similarity Graph

Road networks are complex and contain a considerable amount of segments, resulting, therefore, in a large similarity graph. Moreover, since one common trajectory is sufficient for a similarity edge to exist between a pair of segments, the vertices of the similarity graph tend to have high degrees (although, from our observations, this degree distribution does not follow a proper power law). Modularity-based community-detection algorithms are a popular and widely adopted choice to clustering such graphs [11].

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with vertices $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$, weighted edges \mathcal{E} such as $\omega_{ij} \geq 0$ and $\omega_{ij} = \omega_{ji}$, and given a partition of the vertices into K clusters (or communities) C_1, \dots, C_K , the modularity of the partition is expressed according to formula (3):

$$Q = \frac{1}{2m} \sum_{k=1}^K \sum_{i,j \in C_k} \left(\omega_{ij} - \frac{d_i d_j}{2m} \right) \quad (3)$$

$d_i = \sum_{j \neq i} \omega_{ij}$ and $m = \frac{1}{2} \sum_i d_i$. The modularity measures the quality of the clustering by inspecting the arrangement of the edges within the communities of vertices. A high modularity is an indicator that the edges within the communities outnumber (or have higher weights than) those in a similar randomly generated graph (i.e. one that does not present a community structure). Communities discovered using modularity optimization have a structure that is similar to the structure of cliques. In our context of segment clustering, this means that segments grouped together are heavily connected (which is the intended result) and are travelled by a considerable number of shared trajectories.

To cluster the segment similarity graph, we use the implementation of hierarchical modularity-based clustering described in [12]. The pseudo-code is given in Algorithm 1. First, the algorithm retrieves a partition of the vertices with optimal modularity (line 1): the *Partition* procedure start by considering the trivial partition where each vertex is in its own community and merges communities in a greedy fashion (i.e. each time, it merges the two communities that produce the maximum increase of modularity). The merging operation stops when no possible merge can be done without a degradation of the modularity. In which case the *Partition* procedure proceeds to a refinement step where members of different communities are interchanged in an attempt to further improve the modularity of the partition.

Algorithm 1. Hierarchical modularity-based clustering.

Input: an undirected, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$

Output: hierarchy of nested clusters of vertices

```

1:  $C_1^{(1)}, \dots, C_K^{(1)} \leftarrow \text{Partition}(\mathcal{G})$  ▷ initial partition
2:  $K_T \leftarrow K$  ▷ clusters counter
3:  $l \leftarrow 1$  ▷ hierarchy level
4: repeat
5:    $l \leftarrow l + 1$ 
6:   for all cluster  $C \in C_1^{(l-1)}, \dots, C_{K_T}^{(l-1)}$  do
7:     extract the sub-graph  $\mathcal{G}_C$  of vertices belonging to  $C$ 
8:      $C_1^C, \dots, C_k^C \leftarrow \text{Partition}(\mathcal{G}_C)$ 
9:     if  $\text{TestSig}(C_1^C, \dots, C_k^C)$  then
10:       $C_{K_T+1}^{(l)}, \dots, C_{K_T+k}^{(l)} \leftarrow C_1^C, \dots, C_k^C$ 
11:       $K_T \leftarrow K_T + k$ 
12:     end if
13:   end for
14: until no significant subdivision of level  $l$  can be found

```

Once the initial partition is retrieved, the algorithm proceeds iteratively to construct the hierarchy of communities (lines 4 through 14). For each community at a given level, the sub-graph containing only the vertices of the community and the edges connecting them is isolated (line 7). This subgraph is partitioned separately (by invoking *Partition* as shown in line 8). The *TestSig* evaluates the significance of the found partition (by comparing its modularity to the modularity of partitions obtained on similar randomly generated graphs). If the partition

is significant indeed, its communities are considered for partitioning in the next iteration (lines 9-12), otherwise it's rejected and the original community is retained. The iterations stop when none of the communities at level l yield a significant partition (line 14).

Modularity-based graph clustering approaches are very popular and achieve good results in practice [11]. Nevertheless, we do not exclude the use of other graph clustering alternatives (e.g. spectral clustering [13]) if such techniques can yield better results.

4 Discussion

First, we focus on how the produced clusters can be explored and analyzed in order to deduce useful knowledge about the flow dynamics and the drivers' behavior in the road network (Section 4.1). Then, we address the algorithmic complexity of our approach (Section 4.2).

4.1 Cluster Exploration

We illustrate how the trajectory clusters [6] and segment clusters can be explored and used conjointly. For illustration purposes, we use a synthetic dataset containing 85 trajectories that moved along the Oldenburg road network (cf. Section 5 for more details about this network) and visited a total of 485 distinct road segments. We manually partitioned the trajectories into five clusters (depicted in Figure 2) that we consider hereafter as the ground-truth clusters.

Applying the trajectory clustering [6] results in a hierarchy of clusters where the optimal level w.r.t. modularity (i.e. the very first level) contains only three trajectory clusters: the ground-truth clusters 2 and 3 are considered as part of a same cluster (the same occurs with clusters 4 and 5). Nevertheless, all the ground-truth clusters are retrieved correctly (some of them are even refined) in the following levels. The cluster hierarchy is especially suitable for exploring large datasets where a flat clustering can still produce a high number of clusters: the analyst can start with the few, coarse clusters contained in the first hierarchical levels in order to gain a quick grasp of the general tendencies and movement patterns in the road network. He, then, can choose clusters of interest that he can explore, by means of successive zooms, in higher detail. This idea is depicted in Figure 3 which shows a coarse trajectory cluster and its three, more refined subclusters.

Segment clusters are not as easy to grasp and understand as trajectory clusters. Even though it is feasible to try and explore these clusters as stand-alone clusters, we recommend involving the trajectory clusters in the process. Cross-comparing both types of clusters can reveal interesting information about flow dynamics and yield a better interpretation of the clusters. For example, a segment cluster can be interpreted based solely on the trajectory groups that interacted with it, thus revealing potential hubs, etc. Fig. 4 shows the crossed matrix of the second level trajectory clusters (reported on the rows) and the second

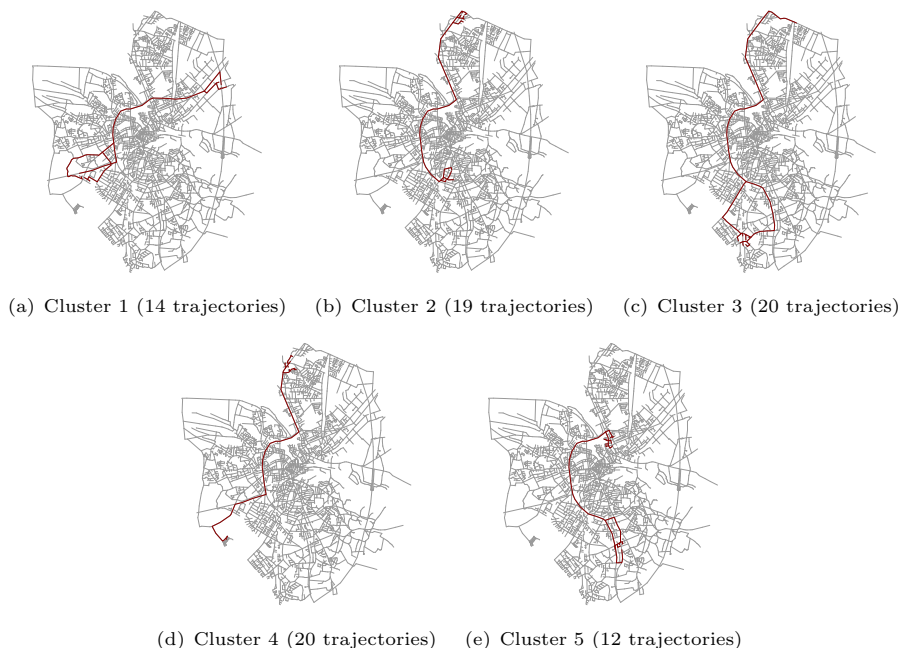


Fig. 2. Ground-truth clusters in the dataset

level road segment clusters (on the columns) and gives an idea about the sizes of the clusters and how clusters of one type interact with those of the other type.

The crossed matrix does indeed reveal some interesting patterns and interactions. For instance, the fourth segments clusters is explored exclusively by two trajectory clusters. Visualizing both this segment cluster and its visiting trajectory clusters (Fig. 5) shows that the segment cluster plays the role of a hub for these two groups of trajectories that converge to it from two different areas in order to travel to two different destinations.

Crossing trajectory clusters and segment clusters is flexible and can be done at various levels of the hierarchies of both cluster types. However, it is totally up to the user to decide the relevance of the crossed clusters. The case of the eleventh segment cluster (cf. Fig. 4) illustrates this point: this segment cluster is very interesting since it interacts with six trajectory clusters. However, it is evident that the segment cluster contains a lot of "noise" segments which is expressed by the considerable amount of white space in the six first rows (representing the trajectory clusters) in the column representing the cluster in the crossed matrix. Consequently, drawn conclusions about the interactions between the clusters won't be very reliable. A wiser alternative would be to study the interactions between the, more refined, subclusters of this segment cluster with the six trajectory cluster it interacts with.



Fig. 3. A coarse cluster containing 39 trajectories (a) and its more detailed subclusters (b-d)



Fig. 4. Crossed matrix of the trajectory clusters (rows) and road segment clusters (columns). Each cell gives an idea about the interaction between the corresponding trajectory and segment clusters: the more black the cell contains the more trajectories in the trajectory cluster cross segments belonging to the segment cluster.

4.2 Algorithmic Complexity

Let n be the number of trajectories in \mathcal{T} and m the number of road segments in \mathcal{S} . Road segments can be represented as a matrix M containing m rows (each representing a road segment) and n columns (each corresponding to a trajectory). $m_{i,j}$ corresponds to the weight of the trajectory represented by column j while inspecting the segment represented by row i . Using this vector model representation, comparing two road segments can then be done in $O(n)$ time

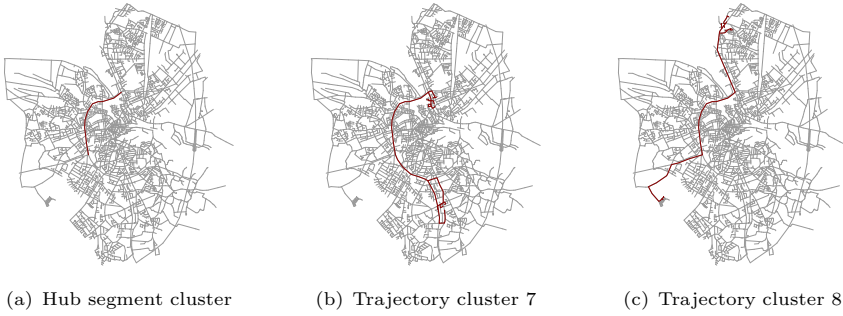


Fig. 5. A segment cluster (a) playing the role of a hub for two different trajectory clusters ((b) and (c)) that borrow it to travel to two separate destinations

complexity. Constructing the similarity graph requires $\frac{m(m-1)}{2}$ similarity calculations. Therefore, the cost of constructing the graph is $O(nm^2)$.

The similarity graph contains m vertices (representing the m segments of \mathcal{S}) and, at most, $\frac{m(m-1)}{2}$ edges. Therefore, the theoretical (maximal) complexity of the community detection algorithm used in our clustering phase is $O(m^3)$ [11]. However, this complexity is rarely observed in practice where the complexity is somewhere near $O(m^2)$.

The complexity of the approach we presented in [6] can be deduced using the same reasoning: the trajectory similarity graph is constructed in $O(mn^2)$ and is clustered in $O(n^3)$ in theory ($O(n^2)$ in practice).

5 Experimental Results

In this section, we validate the effectiveness of our approach by comparing it to two alternative graph clustering techniques. First, we describe our experimental setting, including the used datasets and the evaluated algorithms in Section 5.1. Then cluster quality results are presented in Section 5.2.

5.1 Experimental Setting

In order to validate our choice of modularity-based clustering, we compare it to two other graph clustering techniques: i. spectral clustering; and ii. label propagation clustering. In spectral clustering [13], eigenvectors are extracted from the graph’s Laplacian and are used to conduct a k -means clustering in order to partition the graph’s vertices. Label propagation, on the other hand, works by labeling the vertices with unique labels and then updating the labels by majority voting in the neighborhood of the vertex [14].

We compare the performances of the three algorithms on five synthetic datasets (cf. Table 1) produced with the Brinkhoff generator [15] using the Oldenburg road network. The latter is composed of 6105 vertices and about 14070 road segments. Each dataset contains 100 trajectories visiting a various amount of road segments.

Table 1. Characteristics of the five synthetic datasets

Dataset	Number of segments	Number of edges in the similarity graph
1	2562	79811
2	2394	100270
3	2587	110095
4	2477	87023
5	2348	80659

The performance of each algorithm is evaluated by measuring the quality of the segment partition \mathcal{C}_S it produces according to formula (4):

$$\mathcal{Q}(\mathcal{C}_S) = \sum_{C \in \mathcal{C}} \frac{1}{|C|} \sum_{s_i, s_j \in C} \frac{|\{T \in \mathcal{T} : s_i \in T \wedge s_j \in T\}|}{|\{T \in \mathcal{T} : s_i \in T \vee s_j \in T\}|} \quad (4)$$

$|C|$ is the number of segments in clusters C , $|\{T \in \mathcal{T} : s_i \in T \wedge s_j \in T\}|$ is the number of trajectories both road segments s_i and s_j while $|\{T \in \mathcal{T} : s_i \in T \vee s_j \in T\}|$ is the number of trajectories that travelled along at least one of them.

5.2 Results

Contrary to the spectral clustering algorithm, the modularity-based and label propagation algorithms do not give the user the possibility to configure the number of resulting clusters: the label propagation algorithm produces just one flat partition while the modularity-based algorithm produces a partial hierarchy (i.e. a hierarchy that does not retain all the merging operations).

First, we compare modularity-based clustering and spectral clustering based on the former’s optimal number of clusters (i.e. the number of clusters at the hierarchy’s first level). The results are depicted in Table 2.

In order to compare the three algorithms at once (cf. Table 3), we proceed as follows. Since label propagation clustering produces only on partition, we configure the spectral clustering to produce the same number of clusters as this partition. As for modularity-based clustering, we choose the hierarchical level that produces the closest number of clusters to those discovered by label propagation.

From both Table 2 and Table 3 we can verify that, as expected, the clustering quality increases as the number of clusters increases. Results also show the superiority of modularity-based clustering over label propagation and spectral clustering and suggest that applying the former results in better and more compact clusters of road segments.

We also notice that label propagation results in a large number of clusters. This supports the observation we made in Section 4.1 where we claimed that flat clustering is not suitable for exploring large datasets.

Table 2. Comparison between spectral clustering and modularity-based clustering

Dataset	Clustering quality (clusters)	
	Spectral	Modularity
1	306.33 (23)	657.20 (23)
2	254.97 (21)	524.46 (21)
3	245.64(20)	561.08 (20)
4	249.89 (22)	594.75 (22)
5	284.74 (26)	666.23 (26)

Table 3. Cluster qualities achieved by the three algorithms on the five datasets

Dataset	Clustering quality (clusters)		
	Label prop.	Spectral	Modularity
1	684.19 (68)	678.81 (68)	1614.40 (67)
2	550.66 (59)	549.70 (59)	1276.63 (57)
3	606.45 (66)	567.57 (66)	1516.45 (61)
4	634.63 (68)	637.62 (68)	1406.38 (57)
5	604.97 (64)	539.27 (64)	1418.67 (65)

6 Related Work

Approaches to trajectory clustering are mainly adaptations of existing algorithms to the case of trajectories. Existing problem formulations and propositions include flock patterns [3], convoy patterns [5], the TRACCLUS partition-and-group framework [4] and the T-OPTICS and TF-OPTICS algorithms [2]. The aforementioned algorithms use euclidean-based similarities and distances and can, therefore, be used only in the case of unconstrained trajectories. Furthermore, the majority of these approaches use density-based algorithms which suffer from two major drawbacks: i. their results are very sensitive to the parameter values; and ii. they assume that trajectories in the same cluster have a rather homogeneous density, which is rarely the case (as discussed in [10]).

Roh et Hwang [10] present a network-aware approach to clustering trajectories where the distance between trajectories in the road network is measured using shortest path calculations. A baseline algorithm, using agglomerative hierarchical clustering, as well as a more efficient algorithm, called NNCluster, are presented for the purpose of regrouping the network constrained trajectories. In [8], the authors describe an approach to discovering "dense paths" or sequences of frequently traveled segments in a road network. This approach resembles our segment-based clustering although they diverge on many key aspects. For instance, the approach in [8] produces flat clusters using a density-based approach (which requires fine tuning) whereas ours produces a hierarchy of nested clusters and does not require parametrization. In [6], we presented our graph-based framework to clustering network-constrained trajectories. The work described in

the present paper build upon this framework as it extends it to the case of road segment clustering.

A wide variety of graph clustering algorithms was proposed in the literature, including spectral clustering [13], clustering using label propagation [14], etc. (complete surveys on graph clustering can be found in [16, 11]). Among these propositions, modularity-based community detection algorithms stand out for the good results they yield in practice. We use the hierarchical modularity-based clustering implementation described in [12] (which follows the recommendations in [17]) in our clustering step of our framework in order to detect the presence of clusters among road segments.

7 Conclusion

In this paper, we presented a framework for clustering road segments based on the moving object trajectories that travelled along them. The main novelty of the framework is the use of a graph representation to structure the similarity relationships and interactions between road segments. This framework presents many advantages: i. it does not require parameters, contrary to the majority of existing approaches that are very sensitive to their threshold values; and ii. it also produces a hierarchy of nested clusters promoting exploration at various levels of granularity and detail in situations where a flat clustering approach would have produced a unique level containing a very large number of clusters. Moreover, we showed how segment clusters can be used in conjunction with the trajectory clusters we defined in [6] in order to better understand flow dynamics in the road network.

The framework, however, is not flawless. The community detection algorithm used in the clustering step can be sensitive in presence of noise (i.e. marginal road segments that do not forcefully belong to any cluster) which can degrade the quality of the discovered clusters. Also, the computational cost of the approach and the fact that it requires predisposing of all the data beforehand prohibits it from being used in a streaming context.

In future work, we will focus on alternative graph representations for trajectory data. Mainly, the use of a bipartite graph to represent interactions between trajectories and segments. Such graphs can be partitioned using bi-clustering algorithms in order to simultaneously discover clusters of trajectories and road segments (this is done separately in the present framework) which has the main advantage of automatically crossing both types of clusters based on how they interact, thus relieving the user from this delicate task.

References

1. Giannotti, F., Pedreschi, D. (eds.): *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer (2008)
2. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.* 27(3), 267–289 (2006)

3. Benkert, M., Gudmundsson, J., Hübner, F., Wolle, T.: Reporting flock patterns. In: Azar, Y., Erlebach, T. (eds.) *ESA 2006*. LNCS, vol. 4168, pp. 660–671. Springer, Heidelberg (2006)
4. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: a partition-and-group framework. In: *SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pp. 593–604. ACM, New York (2007)
5. Jeung, H., Shen, H.T., Zhou, X.: Convoy queries in spatio-temporal databases. In: *ICDE 2008: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pp. 1457–1459. IEEE Computer Society, Washington, DC (2008)
6. El Mahrsi, M.K., Rossi, F.: Modularity-Based Clustering for Network-Constrained Trajectories. In: *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2012, Bruges, Belgique*, pp. 471–476 (April 2012)
7. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*, pp. 853–864. VLDB Endowment (2005)
8. Kharrat, A., Popa, I.S., Zeitouni, K., Faiz, S.: Clustering algorithm for network constraint trajectories. In: *SDH. Lecture Notes in Geoinformation and Cartography*, pp. 631–647. Springer (2008)
9. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate gps trajectories. In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2009*, pp. 352–361. ACM, New York (2009)
10. Roh, G.-P., Hwang, S.-W.: NNCluster: An efficient clustering algorithm for road network trajectories. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) *DASFAA 2010, Part II*. LNCS, vol. 5982, pp. 47–61. Springer, Heidelberg (2010)
11. Fortunato, S.: Community detection in graphs. *Physics Reports* 486(3-5), 75–174 (2010)
12. Rossi, F., Villa-Vialaneix, N.: Représentation d’un grand réseau à partir d’une classification hiérarchique de ses sommets. *Journal de la Société Française de Statistique* 152(3), 34–65 (2011)
13. Luxburg, U.: A tutorial on spectral clustering. *Statistics and Computing* 17(4), 395–416 (2007)
14. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76(3) (September 2007)
15. Brinkhoff, T.: A framework for generating network-based moving objects. *Geoinformatica* 6, 153–180 (2002)
16. Schaeffer, S.E.: Graph clustering. *Computer Science Review* 1(1), 27–64 (2007)
17. Noack, A., Rotta, R.: Multi-level algorithms for modularity clustering. In: Vahrenhold, J. (ed.) *SEA 2009*. LNCS, vol. 5526, pp. 257–268. Springer, Heidelberg (2009)

Finding the Most Descriptive Substructures in Graphs with Discrete and Numeric Labels

Michael Davis, Weiru Liu, and Paul Miller

Centre for Secure Information Technologies (CSIT),
School of Electronics, Electrical Engineering and Computer Science,
Queen's University, Belfast, United Kingdom
{mdavis05,w.liu}@qub.ac.uk, p.miller@ecit.qub.ac.uk

Abstract. Many graph datasets are labelled with discrete and numeric attributes. Frequent substructure discovery algorithms usually ignore numeric attributes; in this paper we show that they can be used to improve discrimination and search performance. Our thesis is that the most descriptive substructures are those which are normative both in terms of their structure and in terms of their numeric values. We explore the relationship between graph structure and the distribution of attribute values and propose an outlier-detection step, which is used as a constraint during substructure discovery. By pruning anomalous vertices and edges, more weight is given to the most descriptive substructures. Our experiments on a real-world access control database returns similar substructures to unconstrained search with 30% fewer graph isomorphism tests.

Keywords: graph mining, frequent substructure discovery, numeric attributes, outlier detection.

1 Introduction

A common task in graph mining is to discover frequently-occurring substructures for concept learning, clustering or anomaly detection. Frequent substructures are defined as those which pass some minimum support threshold [10,13,18] or in information-theoretic terms, as the patterns which can be used to maximally compress the input graph [5]. In this paper, we consider how numeric attributes can be combined with structural data, to constrain the search for the most descriptive substructures.

To count the frequency of each pattern, discovery algorithms must compare subgraphs for identity, or Graph Isomorphism (GI). GI is computationally complex to decide for the general case [8], but in practice the complexity is highly dependent on the features of the graphs under consideration. Common special cases can be solved in polynomial time, using techniques such as sorting candidate substructures by their canonical labels [10,13], organising discovered subgraphs into spanning trees [2,18] or by performing a heuristic search [5]. However, it is always possible to come up with a set of input graphs where even the best algorithms perform poorly.

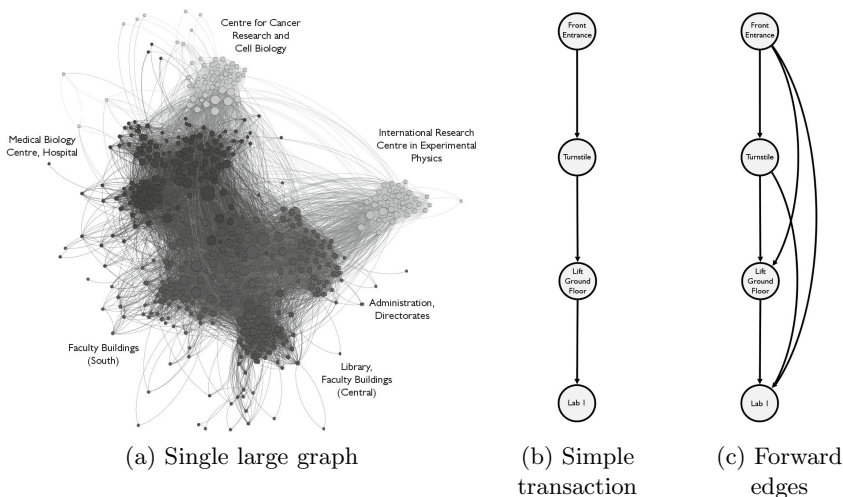


Fig. 1. Graph of Access Control Transactions on a University Campus

Substructure discovery algorithms typically operate on graphs where the labels represent discrete attributes of vertices or edges. Many graph datasets also contain numeric labels or weights, representing attributes such as size, distance, time, frequency or amount. We propose that the best substructures are not only frequent, but also have the most normative numeric attributes. In this paper, we analyse how the dependencies between graph structure and attributes affect the complexity of substructure discovery. Our contribution is to show how numeric attributes can be used to constrain the search in a way that preserves anti-monotonicity. By pruning unlikely candidates early, we focus computational resources on the most descriptive patterns. In our experiments, we were able to discover the most descriptive substructures with a 30% reduction in the number of GI tests required, compared to an unconstrained search.

Applications of substructure discovery include: discovering molecular structures from chemical compounds, *e.g.* for predictive toxicology; learning communication patterns in an e-mail network; and video scene analysis. Our motivating application is to detect “suspicious” behaviour patterns in secure buildings, such as airports, hospitals and power stations. Our experimental data is from the building access control system for a university campus, represented by the graph in Fig. 1a. Vertices represent door sensors and directed edges represent movements between pairs of sensors. The density of transactions is higher in areas with greater security requirements, *viz.* laboratories for laser, radiation and medical research. As we are interested in paths taken by individuals through the network, we reorganised the graph as a transaction database, where each subgraph represents the movement of an individual within a given day (Fig. 1b–1c). We score each subgraph based on its structural elements and its numeric

timing values [6]. Our method shows greater discrimination than scoring based on frequent substructures without numeric attributes.

This paper is organised as follows: Sect. 2 is a brief survey of substructure discovery in graphs and constraint-based graph mining. In Sect. 3, we explore how the distribution of labels and attributes affects the complexity of substructure discovery. In Sect. 4, we outline a method of using numeric outlier detection to constrain the search for normative substructures. Sect. 5 presents our experimental method and datasets (including a method for generating random graph attributes), results and a discussion of complexity; and conclusions are in Sect. 6.

2 Related Work

Frequent Substructure Discovery. Frequent substructure discovery algorithms attempt to find the subgraphs which occur most frequently in a graph database. Early approaches were based on Apriori-style itemset mining: AGM [10] and FSG [13] generate candidate substructures by growing them one vertex or one edge at a time, respectively. Frequent substructures are those which exceed a specified minimum support threshold. The main weakness is that candidate generation is expensive, as canonical labels must be calculated for a large number of redundant candidates.

gSpan [18] avoids candidate generation. Canonical labels are determined by the minimum representation of vertex orderings as discovered by a Depth-First Search (DFS). These labels are organised into a hierarchical spanning tree. Frequent structures are discovered by traversing this tree, checking for substructures which exceed minimum support. CloseGraph [19] and SPIN [9] improve on gSpan by mining only “closed” or “maximal” frequent subgraphs, *i.e.* frequent substructures which are not part of any larger frequent substructure. [2] generalises the canonical form found in gSpan and demonstrates that canonical labels based on Breadth-First Search (BFS) are equally valid.

Subdue [5] represents another class of substructure discovery algorithm, based on information theory. Rather than searching for substructures with minimum support, Subdue looks for the substructures which can be used to best compress the input graph based on the Minimum Description Length (MDL) principle. Complexity is managed with a heuristic: candidate substructures are discovered using a greedy beam search (a limited-length queue of the best few patterns found so far). This allows Subdue to search in single large graphs, which is not generally possible with AGM, FSG and gSpan. The disadvantage of the greedy search strategy is that some interesting patterns could be missed.

Constraint-Based Graph Mining. One of the main problems with pattern-mining algorithms is the large numbers of patterns produced. One possible solution is to introduce *constraints*, which are used to prune away uninteresting patterns and focus on the most meaningful subgraphs. [15] defines a “cohesive pattern constraint” on a connected subgraph, where the vertex attributes are the same within some subspace and some density constraint is met. The cohesive

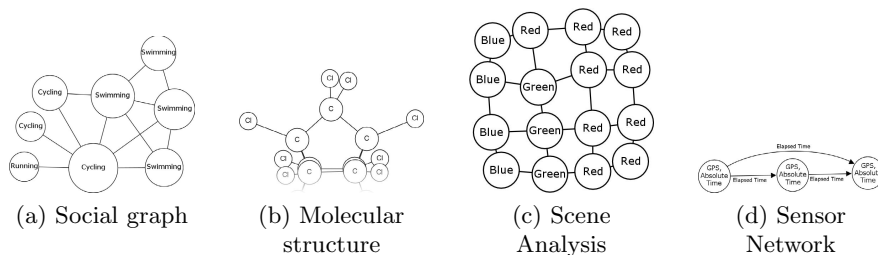


Fig. 2. Examples of the distribution of attributes on various kinds of graph

pattern constraint is defined for graphs with discrete attributes; in this paper, we define a constraint on graphs with numeric attributes.

In [11], gSpan is extended by including edge weights into the support calculation and pruning low-weighted substructures from the search. Anti-monotonicity is an important property of pattern growth-based algorithms: if a substructure fails to achieve minimum support, all of its supergraphs will also fail. Two of the proposed weighting schemes preserve anti-monotonicity, by thresholding the weight measure in addition to thresholding for minimum support. A third weighting scheme uses a heuristic and does not rely on preserving anti-monotonicity. All three weighting schemes assume that higher weights are more significant (which may not necessarily be the case).

Constraints on weighted graphs are considered within a more general framework in [7]. Attribute-based constraints (which are not guaranteed to be anti-monotonic) are used to prune substructures by running a measure function on the edge (or vertex) weights and comparing the output to a threshold. If the definition of the measure function is extended to take multi-dimensional numeric attributes as its input, then the outlier detection step that we propose in Sect. 4 could be considered a measure function within this theoretical framework.

3 Graph Mining with Attribute-Based Constraints

Previous work on attribute-based constraints has assumed independence between the structure of a graph and its attributes. If this is the case, then attribute-based constraints are not anti-monotonic. However, in most real-world graphs, the structure and attributes are not independent. In this section, we discuss a graph/attribute model where the attributes of vertices and edges are conditionally dependent on the attributes of similar or neighbouring vertices and edges in the graph. In Sect. 4, we will present a method to constrain substructure discovery based on the values of numeric attributes.

Fig. 2 shows four examples to illustrate the dependence of graph structure and attributes. The social graph in Fig. 2a is labelled with the favourite sport of the actors: people tend to form friendship bonds with people of similar interests. In molecules (Fig. 2b), the vertex label (atom name) is conditionally

dependent on the molecular structure; the degree of each vertex is dependent on the number of free electrons of each atom; and the length of the edges (bonds) is dependent on the atomic weights of the vertices. Fig. 2c is from video scene analysis, showing a green object moving across a red and blue background. Each vertex represents a superpixel in the frame: the colour attribute is conditionally dependent on the colour of adjacent vertices. The velocities of adjacent vertices are also conditionally dependent, as the superpixels in the object will move together and those of the background will move together. Fig. 2d continues the example shown in Fig. 1: the time taken to travel between a pair of sensors is conditionally dependent on the name and GPS coordinates of the sensor.

All of the above examples are labelled graphs, which have an arbitrary number of discrete and numeric labels on their vertices and edges. Formally:

Definition 1. A labelled graph G is a tuple $\langle V, E, L, \mathcal{L}_V, \mathcal{L}_E \rangle$. V is a set of vertices and E is a set of edges: $E \subseteq \{\langle v, w \rangle : v, w \in V \times V\}$. If the tuple $\langle v, w \rangle$ is ordered, the edge is directed, otherwise it is undirected. L is a set of graph labels; \mathcal{L}_V and \mathcal{L}_E are label-to-value mapping functions.

Definition 2. The set of graph labels L is the union of the sets of vertex labels L_V and edge labels L_E . L is partitioned into discrete labels L^D and numeric labels L^N , $L^D \cap L^N = \emptyset$. Thus $L = L_V \cup L_E = L^D \cup L^N$. Let A^D be the set of discrete attribute values and $A^N \subset \mathbb{R}$ be the set of numeric attribute values.

Definition 3. The label-to-value mapping function for vertices is denoted as:

$$\begin{aligned} \mathcal{L}_V : V \times (L_V \cap L^D) &\rightarrow A^D \\ V \times (L_V \cap L^N) &\rightarrow A^N \end{aligned}$$

For a vertex-weighted graph, the weight function $\mathcal{W}(v)$ is treated as a special case of its numeric attributes: $\forall v \in V : \mathcal{W}(v) = \mathcal{L}_V(v, \text{“weight”})$. (The label-to-value mapping function for edges \mathcal{L}_E can be denoted in a similar manner.)

During substructure discovery, one important method of reducing the complexity of the GI test is vertex partitioning [8,10,13,18]. Vertices can be partitioned into similar disjoint sets or equivalence classes. We extend this notion to also define edge partitions:

Definition 4. The vertex partition set and edge partition set are defined as:

$$V = \bigcup_i V_i \quad E = \bigcup_i E_i$$

where all vertices in the same partition share the same discrete attribute values:

$$\forall v \in V_i \quad \forall w \in V_i \quad \forall l \in (L_V \cap L^D) : \quad \mathcal{L}_V(v, l) = \mathcal{L}_V(w, l)$$

Similarly, all edges in the same partition share the same discrete attribute values, with the additional constraint that their source and target vertices are from the same partitions:

$$\forall \langle v, w \rangle \in E_i \quad \forall \langle x, y \rangle \in E_i : \quad v \in V_j \wedge x \in V_j \wedge w \in V_k \wedge y \in V_k$$

In the case of an undirected graph, $\langle v, w \rangle \Leftrightarrow \langle w, v \rangle$

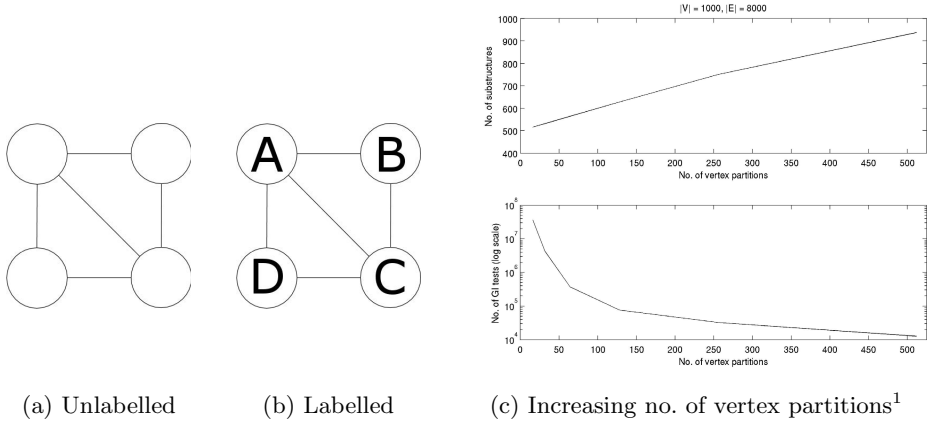


Fig. 3. Relationship between number of vertex partitions and complexity

There is an important relationship between the number of partitions and the complexity of GI and substructure discovery. To illustrate this, consider the unlabelled partial clique in Fig. 3a. This graph contains eight distinct subgraphs with two or more vertices; 27 subgraph instances in all. As multiple subgraph instances share the same vertex partition set, it requires 20 GI tests to determine which instances are isomorphic. Compare this to the graph in Fig. 3b, which has the same structure, but each vertex is uniquely labelled. Once again there are 27 substructure instances, but each has a distinct vertex and edge partition set. Subgraphs with different partition sets cannot be isomorphic, so we can determine that there are 27 distinct subgraphs without needing to do any isomorphism tests.

In practice, real-world graphs will lie somewhere between these extremes. Fig. 3c shows the relationship between the number of vertex partitions and the complexity of substructure discovery: the number of distinct subgraphs rises with the number of partitions, but the number of instances of each subgraph falls. This leads to an exponential reduction in the number of GI tests and thus the complexity of substructure discovery.

In addition to reducing the number of GI tests required, increasing the number of partitions constrains the search by reducing the support for each substructure. Formally, we define a constraint as follows:

Definition 5. A constraint c is a Boolean predicate which any subgraph $g \in G$ must fulfil. c is said to be anti-monotone if it satisfies $\forall g' \subset g : c(g) \implies c(g')$.

¹ Fig. 3c shows the results for R-MAT random graphs, with 0, 1, ..., 9 binary labels, i.e. 0–512 vertex partitions. The experiment was repeated 10 times (and across multiple sizes of graph) and the results averaged. The attribute values in Fig. 3c were assigned independently from a uniform distribution. Our experiments on synthetic and real datasets (Sect. 5) verify that the complexity of substructure discovery increases with the homogeneity of vertices and edges, and that this holds when the independence assumption is removed.

An example of an anti-monotone constraint is minimum support: a graph g can reach minimum support only if all of its subgraphs g' reach minimum support. In previous work on constraint-based graph mining [7,11], it has been assumed that the structure of the graph and its attributes are independent, and therefore attribute-based constraints are not anti-monotonic. However, in real-world graphs, the independence assumption does not hold (Fig. 2).

Our approach is to consider the dependencies between attributes as a Random Field [1]: the discrete attribute values on each vertex (or edge) are dependent on its adjacent vertices, but are conditionally independent of the rest of the graph. We use the conditional independence (CI) assumption as the basis of a generator for graph attributes (see Sect. 5). In the next section, we use the CI assumption to define a constraint on numeric attribute values: as attributes depend on graph structure, the constraint is used to prune instances, or reduce support for specific substructures, so the property of anti-monotonicity is preserved.

4 Frequent Substructure Discovery with Numeric Attribute Constraints

In the previous section, we discussed the relationship between the number of discrete partitions and the complexity of substructure discovery. In this section, we discuss how to use numeric attributes as a constraint during substructure discovery. We have argued that graph attributes are dependent on graph structure. Therefore, we can define the most descriptive substructures as those which are normative both in terms of their structure and in terms of their numeric attributes. The corollary is that vertices or edges containing numeric outliers are abnormal and can therefore be pruned early in the discovery process. We determine whether numeric attributes are “normal” or anomalous by means of a numeric outlier detection function:

Definition 6. *We define a numeric outlier function \mathcal{O} on a dataset D as:*

$$\mathcal{O} : D \rightarrow \mathbb{R} \quad \forall d \in D : \mathcal{O}(d) = \begin{cases} q_0 & \text{if } d \text{ is “normal” w.r.t. } D \\ q & \text{otherwise} \end{cases}$$

where q_0 is some constant value and $q \neq q_0$ is a value measuring the degree of outlierness.

The value of q_0 and the range of \mathcal{O} will depend on the specific choice of outlier detection function. For our experiments in Sect. 5, we chose Local Outlier Factors (LOF) [3]. LOF is a density-based measure: the LOF score of a sample p is a measure of its outlierness with respect to its local neighbourhood, computed as:

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

where N is the number of samples in the dataset, $MinPts$ is the minimum number of points to consider as the local neighbourhood and lrd is a function

which computes the local reachability density of the neighbourhood (the inverse of the average reachability distance in the neighbourhood). Intuitively, the LOF score is based on the distance of a sample from its local neighbourhood and the relative density of the neighbourhood. A sample d belonging to a dense cluster or deep within a sparse cluster has $\text{LOF}(d) \cong 1$. Outliers have LOF values several times larger. Thus LOF satisfies the property given in Def. 6: $\text{LOF}(d) \cong 1$ for normal values of d and $\text{LOF}(d) \gg 1$ for anomalous values. LOF is well-suited to unsupervised learning, as it makes no assumptions about the underlying distribution of the data and can cope with clusters of different sizes and densities. For more discussion on the choice of outlier function, see the comments in Sects. 5 and 6 and our previous work [6].

To calculate $\mathcal{O}(d_v)$ for a vertex v , we define d_v as a multi-dimensional feature vector across all the numeric attributes of v : $d_v = \mathcal{L}_V(v, L^N)$. The outlier factor for each d_v is calculated relative to the dataset defined by its vertex partition, $\forall d_v \in D_i : v \in V_i$.

To use \mathcal{O} as a constraint on substructure discovery, structural elements are classified as normal or anomalous by Def. 7:

Definition 7. *A vertex $v \in V_i$ is normal if $\mathcal{O}(\mathcal{L}_V(v, L^N)) \cong q_0$, anomalous otherwise. An edge $e \in E_i$ is normal if $\mathcal{O}(\mathcal{L}_E(e, L^N)) \cong q_0$, anomalous otherwise.*

During substructure discovery, anomalous vertices and edges are pruned from the graph. This can be done as a pre-processing step before generating all frequent 1- and 2-vertex subgraphs. As we only consider elements with “normal” numeric values to be part of normative substructures, this pruning dramatically reduces the number of GI tests required, without significantly affecting which substructures are discovered. We validate this experimentally in the next section.

5 Experiments

The purpose of our experiments is to analyse the effect of pruning anomalous vertices and edges from graphs on the performance and accuracy of substructure discovery. Specifically, we analysed the number of graph isomorphism tests required; the computation time for substructure discovery; and the accuracy and meaningfulness of the discovered substructures. We show that our constraint-based approach can more efficiently find frequent subgraphs than an unconstrained approach. In many cases where the input graph is intractable with an unconstrained approach due to the computational or memory overheads, our approach allows the graph to be processed.

Synthetic Datasets. Our experimental setup included Erdős-Rényi random graphs and R-MAT random graphs with up to 10,000 vertices. For the R-MAT graphs, we added edges with mean degree 2, 4, 6 and 8 and probabilities that an edge is placed in one of the four quadrants of the graph as $\langle 0.57, 0.19, 0.19, 0.05 \rangle$, to ensure that the random graphs exhibited clustering/community properties. Attributes were added in three ways: random selection from a uniform distribution (“white noise”); according to pre-defined prior probabilities; and according

to the generative algorithm described below. The purpose of the experiments on synthetic data was to compare the effect of increasing size, structure, density and dependencies between graph structure and attributes on the complexity of substructure discovery, and to evaluate the effect of our constraint-based approach on the performance of substructure discovery.

Random Generation of Graph Attributes. The R-MAT graph generator [4] creates random graphs with properties similar to many real-world graphs (small-world, power-law degree distribution, *etc.*). As R-MAT creates unlabelled graphs, we had to devise a way to assign labels and attribute values. A naïve approach is to randomly allocate attribute values according to some prior distribution, but this assumes that the graph structure and attributes are independent. Here we present AttributeGen, an algorithm to generate random attribute values on an unlabelled graph. Alg. 1 shows the version for discrete attributes.

The Random Field model assumes that the attribute values on each vertex depend on its neighbours. For AttributeGen, we used a simpler assumption, that the values on each vertex depend only on its higher-degree neighbour; *i.e.*, the dependencies between attributes will be propagated from hubs to leaves. Vertices with no higher-degree neighbour are assigned values from a prior distribution over A (line 6: we define one distribution per vertex label). Edges and the other vertices are assigned values from a posterior distribution over A . Edge attributes are conditional on the vertex partition of the source vertex (line 8). Vertex attributes on a target vertex are conditional on the vertex partition of the source vertex and the edge partition of the connecting edge (line 10).

Random numeric values are generated by an analogous method. From our analysis of numeric attributes on real-world datasets (see below), we see that numeric attributes are generated by multiple processes; the specific mixture of processes depends on the vertex (or edge) partition. We use this observation to extend Alg. 1 to generate numeric as well as discrete attribute values. Each numeric attribute $\mathcal{L}_V(v, l)$ on a given vertex partition $V_i : v \in V_i$ is modelled as a mixture of Gaussian processes:

Algorithm 1. Discrete AttributeGen

Require: Unlabelled Random Graph $G = \langle V, E \rangle$, Labels L_V, L_E , Attribute Values A ,

Prior distributions P_V , Posterior distributions Q_V, Q_E

- 1: Define vertex labels in G from L_V and edge labels from L_E
 - 2: Sort vertices V by degree in descending order
 - 3: **for all** $v \in V$ in order **do**
 - 4: **for all** $l \in L_V$ **do**
 - 5: **if** $\mathcal{L}_V(v, l)$ is unassigned **then**
 - 6: Assign $\mathcal{L}_V(v, l) \leftarrow a \in A : a$ is randomly selected from $p_l \in P_V$
 - 7: **for all** $e \in E : e$ is adjacent to $v, l \in L_E$ **do**
 - 8: Assign $\mathcal{L}_E(e, l) \leftarrow a \in A : a$ is randomly selected from $q_{v,l} \in Q_E$
 - 9: Let w be the adjacent vertex: $\exists w \in V : e = \langle v, w \rangle$
 - 10: Assign $\mathcal{L}_V(w, l) \leftarrow a \in A : a$ is randomly selected from $q_{v,e,l} \in Q_V$
-

$$P_{V_i,t} = \sum_j \omega_j \cdot \eta_j(\mu_j, \sigma_j)$$

similar to those illustrated in Fig. 5. ω_j is the weight of each component in the mixture; $\sum_j \omega_j = 1$ forms a probability distribution Ω over all the components. Numeric attributes are assigned by randomly selecting component j of the mixture from Ω , then choosing a random numeric value from the Gaussian distribution $\eta_j(\mu_j, \sigma_j)$. Thus the distributions of numeric values are dependent on the vertex partition and conditionally independent of the rest of the graph.

Ideally, we should learn the prior and posterior distributions of the attribute values from data, to create attribute generators for specific types of graph (social graphs, molecular structures, *etc.*). As that was outside the scope of this work, the distributions for the synthetic data in the experiments were created manually.

Access Control System Dataset. Our real-world dataset is from the access control system logs of a large university campus. The ≈ 1 million log entries are graphically represented in Fig. 1a, showing the movements of approximately 6,500 students and staff. The ≈ 800 vertices represent door sensors; directed edges represent movements between pairs of sensors. We are interested in finding patterns representing “suspicious” behaviour, particularly in high-security areas such as laboratories for laser, radiation and medical research [6]. For the purpose of our experiments, we reorganised the graph as a transaction database, where each graph transaction represents the movement of an individual within a given 24-hour period (Fig. 1b). If a user fails to swipe in at a particular sensor (*e.g.* if someone holds open a door for them), this creates missing edges in the graph. We compensated for this effect by including forward edges from each sensor to all subsequent sensors visited by the user (Fig. 1c).

Numeric attributes were calculated from the log entries and added to the edges in the graph as shown in Fig. 4. Absolute time is the time of day (seconds since midnight) when the user presented their ID card to a door sensor at the end of the path segment. Elapsed time is the difference in seconds between the absolute time at the current sensor and the absolute time at the previous sensor. Day of Week (DoW) is strictly an ordinal attribute, but it was convenient to represent it numerically: LOF combined it with the other attributes in multi-dimensional space, effectively clustering different patterns of behaviour on different days. Weekend patterns are quite different from weekday patterns, perhaps representing the movements of security staff or cleaners as well as weekend workers.

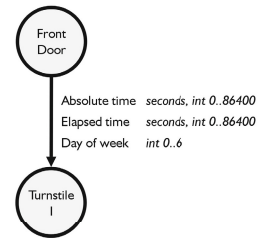


Fig. 4. Numeric edge labels

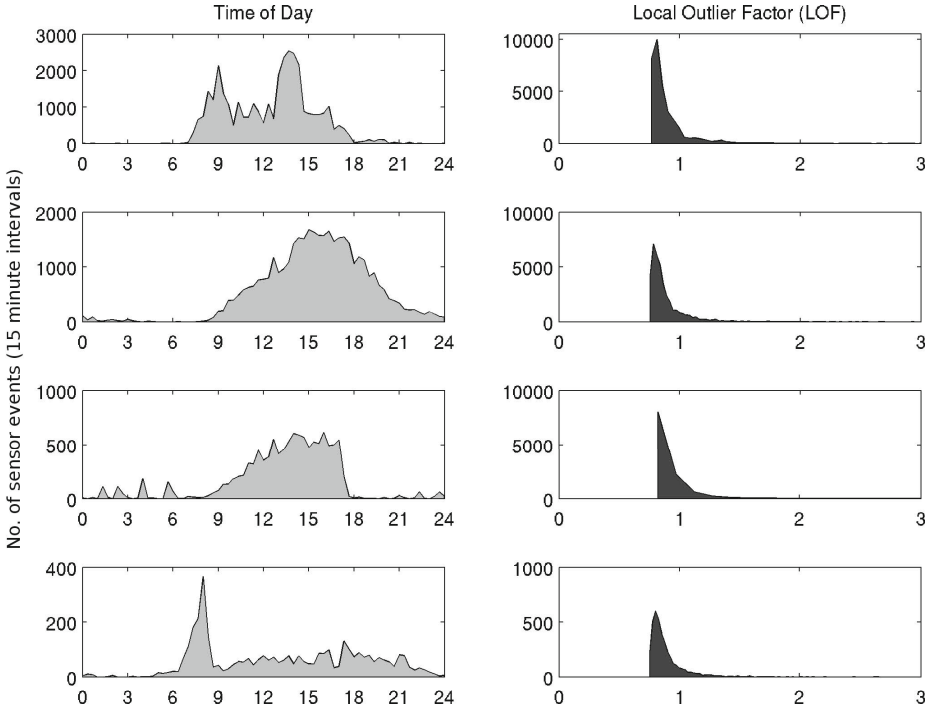


Fig. 5. Distribution of Empirical Data and Local Outlier Factors

We cannot assume that the numeric attributes within real-world graphs are a simple Gaussian; nor can we assume that the probability distribution is the same across all partitions. The left-hand side of Fig. 5 shows four examples of the distribution of the Time of Day numeric attribute across the edges of four of the edge partitions. For each partition, there is a mixture of underlying processes: in a lab, some people work there and stay for many hours; others go in simply to speak to a colleague for a few minutes; security staff may show up periodically for short intervals in the middle of the night. The mixture of processes between partitions is also different: the behaviours in a lab are very different from the behaviours in a lift.

This analysis supports our decision to use a density-based approach to calculate numeric outliers. The distribution of LOF scores for each of the empirical distributions is shown to the right of Fig. 5. Although the data distributions are very different, the distributions of LOF scores are very similar, with normal values clustered around 1 and anomalous values stretching out in a long tail to the right. This verifies that by using LOF, we do not have to make any assumptions about the underlying distribution of the data.

The experimental results in the next section verify that LOF is an effective measure for pruning anomalous vertices and edges.

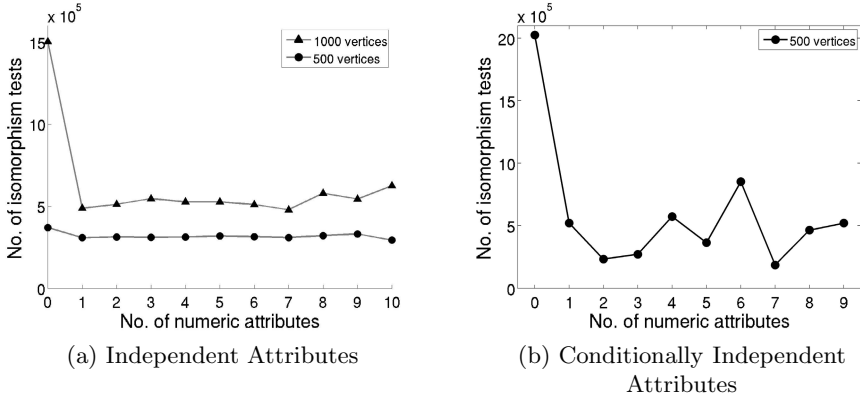


Fig. 6. Frequent substructure discovery on R-MAT random graphs

5.1 Results

Synthetic Datasets. The experiments on random single large graphs were repeated 10 times for each combination of no. of vertices, no. of vertex partitions and no. of numeric attributes, and the results averaged. In all cases, run time was directly proportional to the number of GI tests. The time taken to calculate numeric outliers was trivial compared to the time to discover substructures: ≈ 0.2 seconds for a graph with 10,000 vertices and 100 numeric attributes. LOF’s $O(n^2m)$ complexity is acceptable on graphs where each partition has up to a few thousand vertices or edges. For graphs with larger partitions, some alternative measures are suggested in Sect. 6.

Fig. 6a shows the results for graphs where the attributes were assigned independently from a uniform distribution. On the 1,000 vertex graphs, we measured an average 66% reduction in the number of GI tests required when using numeric outliers to prune the graph, compared to an unconstrained search. (As the attributes are independent, we cannot make any claims about the meaningfulness of the discovered substructures in this case; *cf.* our comparison with random substructure removal, Fig. 8).

Fig. 6b shows the results for graphs where the attributes are conditionally dependent on the graph structure (Alg. 1), so the anti-monotone condition holds. These experiments show an average 80% reduction in the number of GI tests, demonstrating that our constraint-based approach is most effective when the graph exhibits conditional independence between structure and attributes.

We found that graphs with 10,000 vertices and a small number of partitions were not tractable without using numeric attributes, as there were millions of instances of each pattern, requiring more memory to process than was available in our experimental setup. However, we were able to process the 10,000 vertex graphs with the numeric constraint, as there were an order of magnitude fewer instances to compare for isomorphism. This suggests that our method could be useful when processing Very Large Graphs.

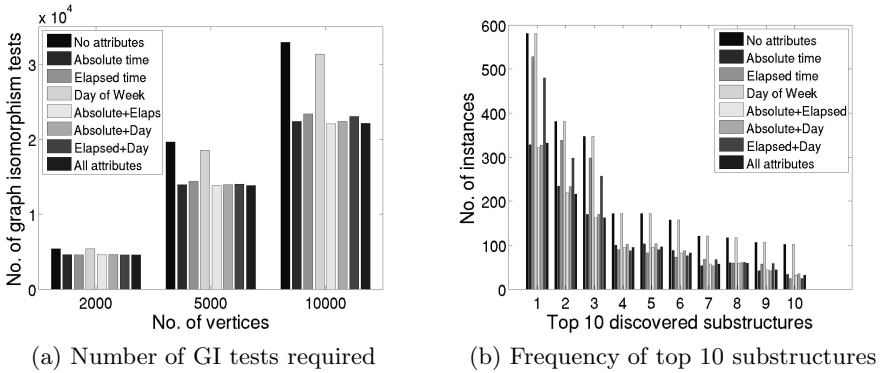


Fig. 7. Effect of pruning numeric anomalies on frequent substructure discovery

Access Control System Dataset. The results on the real-world graph transaction database are shown in Fig. 7. Fig. 7a shows the number of GI tests for datasets of 2,000–10,000 vertices. In the real-world data, numeric attributes vary in their ability to discriminate between normal and anomalous patterns, but combining attributes gives the best performance overall. Absolute and elapsed time are both good discriminators, but day of week is very poor. However, if we combine day of week with elapsed time, we get a slightly better result than using elapsed time on its own; and the best results were achieved by combining all three attributes. The benefit of our approach increased with increasing size of database, as we were able to prune more anomalous substructures. In the dataset with 10,000 vertices, we reduced the number of GI tests by around 30%, which equated to a speed-up of 1.45.

Next, we wanted to validate that the discovered substructures are meaningful. Fig. 7b compares the ten best substructures discovered by Subdue (with no numeric attributes) to the substructures discovered when we added attributes. Our approach discovered the same substructures as Subdue, but fewer instances of each. The relative order of the top ten substructures was changed slightly. Where there is a large difference in relative frequency (*e.g.* 1st–3rd substructures), the ordering was unchanged: these substructures are robust against the removal of anomalous edges. In cases where the relative frequencies were very similar, the order was sometimes transposed (*e.g.* 4th and 5th substructures exchanged places). This is because greater weight is given to substructures with normal numeric values.

To investigate this effect further, we compared our method of pruning anomalous edges to random removal of edges from the graph. We conducted experiments where we randomly deleted 10%–90% of the edges in the graph before searching for frequent substructures. The results are shown in Fig. 8.

Fig. 8a shows the effect on performance. It is necessary to remove around 45% of the graph in order to reduce the number of GI tests by a similar amount as our approach.

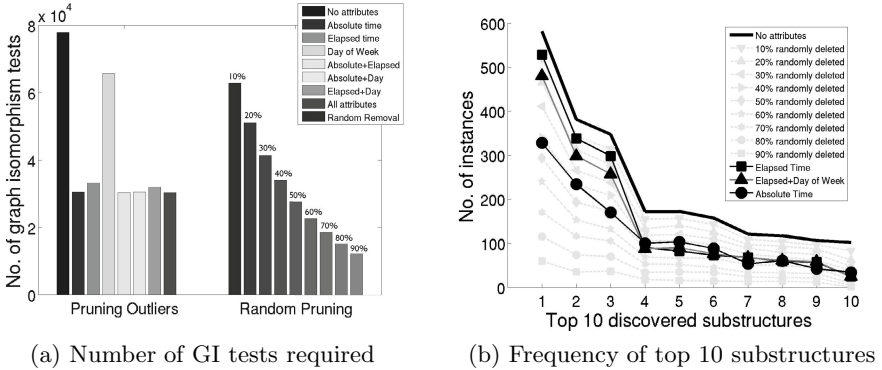


Fig. 8. Effect of randomly removing graph edges on substructure discovery

Fig. 8b shows the frequency counts of the top 10 substructures. Substructures with edges randomly removed are shown in grey; substructures discovered by our approach are superimposed in black. Randomly removing edges increases the entropy of the graph: the shape of the curve becomes flatter as more of the graph is removed. Substructure 1 is quite distinct; even with 90% edge removal, it remains the most descriptive pattern. Substructures 2–3 are also quite robust; discrimination is lost at around 70% edge removal. However, discrimination between substructures 4–10 diminishes after 20% edge removal and by 50% edge removal, the order is random. These results show that randomly deleting graph edges does not preserve the meaningfulness of the output.

5.2 Analysis of Complexity

Here we briefly discuss the benefits (and costs) of using our approach with respect to the complexity of substructure discovery. Substructure discovery algorithms have essentially two parts: finding instances in a graph (or graph database); and grouping instances together into common substructures for evaluation.

Finding Instances. The graph space is searched starting from each vertex instance. If $n = |V|$ and $m = |E|$, an exhaustive search of all possible substructure instances in a single graph has complexity $O(n^2 + nm)$. For a graph database with K transactions, $\mathcal{D} = \{G_1, \dots, G_K\}$, the complexity of instance discovery is $O(n_1^2 + n_1m_1 + \dots + n_K^2 + n_Km_K)$. In practice, algorithms do not perform an exhaustive search, as the cost is prohibitive in all but the most trivial graphs. Subdue constrains the search with its parameters; gSpan uses minimum support. gSpan’s approach does not scale to single large graphs; graph transactions with more than a few hundred vertices and edges are intractable [11].

The benefit of our approach is to prune away the parts of the search space which contain numeric anomalies. If $0 < p \leq 1$ is the proportion of “normal” vertices and $0 < q \leq 1$ is the proportion of “normal” edges, the complexity of an exhaustive search for substructure instances reduces to $O((pn)^2 + pqnm)$.

Grouping Instances into Substructures. Instances are grouped together using isomorphism tests (or canonical labels, which involves finding a set of automorphisms and determining which is “least”). The complexity of GI is one of the most famous unsolved problems in complexity theory, so a formal analysis is well outside the scope of this paper. The interested reader is referred to [8] for an introduction and to [14] for the state-of-the-art. The best proven worst-case complexity for the general case is $e^{O(\sqrt{n \log n})}$, though most real-world graphs exhibit polynomial complexity. Suffice to say that the GI test is the most computationally expensive part of the discovery process: any reduction to the number of GI tests will have a dramatic effect on the computational cost.

The upper bound on the number of GI tests required is $O(|I| \times |S|)$, where I is the set of instances and S is the set of discovered substructures. In practice, we only approach this upper bound where the input graph exhibits low entropy: see the discussion following Def. 4. Our approach reduces $|I|$ as discussed above. There is little effect on the frequent substructures in S (see Fig. 7b, 8b), but infrequent substructures will lose support and will be pruned earlier, so we expect some reduction in $|S|$.

Cost of Calculating Numeric Anomalies. LOF has complexity $O(n^2d)$, where d is the number of numeric attributes. For a graph with N vertex partitions (Def. 4), the complexity of calculating LOF for all vertices is $O(n_1^2d + \dots + n_N^2d)$. Thus, the cost of calculating LOF is typically orders of magnitude smaller than the cost of substructure discovery, but rises significantly when the input graph exhibits low entropy (as there are fewer partitions with more vertices in each).

Where each n_i is small (not more than a few thousand), LOF’s $O(n^2d)$ complexity is acceptable. For very large or very regular graphs, the complexity can be reduced by replacing LOF with an approximation algorithm such as aLOCI [16] or PINN [17], which has sub-quadratic complexity, $O(dn \log n)$.

In summary, our approach reduces the complexity of substructure discovery at both the instance mining and substructure grouping phases, and the cost of calculating the numeric anomaly scores is typically orders of magnitude less than the cost savings. The resulting performance improvement means that substructure discovery remains tractable for larger graphs than is possible with the standard algorithms.

6 Conclusions

In this paper, we presented a method of using numeric outliers as a constraint on the search for frequent substructures in graphs. Our thesis is that the “best” substructures are those which are not only the most frequent, but which are also normative in terms of their numeric attributes.

Previous work on attribute-based constraints has assumed independence between graph structure and attributes, but this assumption does not hold for real-world graphs. Our outlier-based constraint and algorithm for generating random attributes on graphs assume conditional independence between graph structure and attributes.

Our experiments on random graphs demonstrate that in many cases where the input graph is intractable with an unconstrained approach, our approach allows the graph to be processed. In experiments on real-world data, we find similar substructures to an unconstrained search, with around 30% fewer graph isomorphism tests. Where discovered substructures are of similar frequency, we are better able to discriminate between them, because we give greater weight to substructures with normal numeric attributes.

Future Work. The algorithm for generating random attributes (Sect. 5) must be provided with prior and posterior distributions of attribute values. We plan to analyse real-world datasets representing different kinds of graph to learn these distributions. This will allow us to generate different types of random graph which share the characteristics of real-world graphs.

Instead of hard-pruning anomalies, the measure \mathcal{O} could be used in the calculation of how much support each subgraph instance contributes. Instances with normal numeric values would contribute a support of 1, whereas instances with anomalous numeric values would contribute a support of less than 1. We plan to conduct further experiments to compare this alternative support measure with the pruning approach.

We used LOF as the measure function \mathcal{O} . As discussed in Sect. 5.2, where the vertex or edge partitions are very large, LOF's complexity may be unacceptable. This could be addressed by replacing LOF with aLOCI [16] or PINN [17].

We tested our approach on graphs with a moderate number of numeric attributes (up to 10). If there are very many attributes, the numeric feature vectors become very sparsely distributed in high-dimensional space. In this case, LOF's ability to discriminate between normal and anomalous values is diminished. However, not all attributes are of equal importance to all clusters. For high-dimensional data, we could amend our approach to detect numeric anomalies in subspaces rather than in full space by choosing only locally-relevant attributes on which to calculate the outlier score [12].

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
2. Borgelt, C.: Canonical forms for frequent graph mining. In: 30th Annual Conf. German Classification Society, Gfkl 2006, pp. 337–349. Springer (2006)
3. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. SIGMOD Rec. 29(2), 93–104 (2000)
4. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: A recursive model for graph mining. In: SDM 2004. SIAM (2004)
5. Cook, D.J., Holder, L.B.: Graph-based data mining. IEEE Intelligent Systems 15, 32–41 (2000)
6. Davis, M., Liu, W., Miller, P., Redpath, G.: Detecting anomalies in graphs with numeric labels. In: CIKM 2011, pp. 1197–1202. ACM (2011)
7. Eichinger, F., Huber, M., Böhm, K.: On the usefulness of weight-based constraints in frequent subgraph mining. In: ICAI 2010, pp. 65–78. BCS SGAI (December 2010)

8. Fortin, S.: The graph isomorphism problem. Tech. rep., Univ. of Alberta (1996)
9. Huan, J., Wang, W., Prins, J., Yang, J.: SPIN: Mining maximal frequent subgraphs from graph databases. In: KDD 2004, pp. 581–586. ACM (2004)
10. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
11. Jiang, C., Coenen, F., Zito, M.: Frequent sub-graph mining on edge weighted graphs. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) DAWAK 2010. LNCS, vol. 6263, pp. 77–88. Springer, Heidelberg (2010)
12. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery in Data* 3(1), 1:1–1:58 (2009)
13. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001, pp. 313–320. IEEE (2001)
14. McKay, B.D., Piperno, A.: Practical graph isomorphism, II (January 2013), <http://arxiv.org/abs/1301.1493v1>
15. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM, pp. 593–604 (2009)
16. Papadimitriou, S., Kitagawa, H., Gibbons, P., Faloutsos, C.: Loci: fast outlier detection using the local correlation integral. In: Proceedings of the 19th International Conference on Data Engineering 2003, pp. 315–326 (March 2003)
17. de Vries, T., Chawla, S., Houle, M.: Finding local anomalies in very high dimensional space. In: ICDM 2010, pp. 128–137. IEEE (2010)
18. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM 2002, pp. 721–724. IEEE (2002)
19. Yan, X., Han, J.: CloseGraph: Mining closed frequent graph patterns. In: KDD 2003, pp. 286–295. ACM (2003)

Learning in Probabilistic Graphs Exploiting Language-Constrained Patterns

Claudio Taranto, Nicola Di Mauro, and Floriana Esposito

Department of Computer Science, University of Bari "Aldo Moro"
via E. Orabona, 4 - 70125, Bari, Italy
{claudio.taranto,nicola.dimauro,floriana.esposito}@uniba.it

Abstract. The probabilistic graphs framework models the uncertainty inherent in real-world domains by means of probabilistic edges whose value quantifies the likelihood of the edge existence or the strength of the link it represents. The goal of this paper is to provide a learning method to compute the most likely relationship between two nodes in a framework based on probabilistic graphs. In particular, given a probabilistic graph we adopted the language-constrained reachability method to compute the probability of possible interconnections that may exist between two nodes. Each of these connections may be viewed as feature, or factor, between the two nodes and the corresponding probability as its weight. Each observed link is considered as a positive instance for its corresponding link label. Given the training set of observed links a L2-regularized Logistic Regression has been adopted to learn a model able to predict unobserved link labels.

1 Introduction

Over the last few years the extension of graph structures with uncertainty has become an important research topic [13,18,12], leading to *probabilistic graph* model. Probabilistic graphs model uncertainty by means of probabilistic edges whose value quantifies the likelihood of the edge existence or the strength of the link it represents. One of the main issues in probabilistic graphs is how to compute the connectivity of the network. The network reliability problem [3] is a generalization of the pairwise reachability, in which the goal is to determine the probability that all pairs of nodes are reachable from one another. Unlike a deterministic graph in which the reachability function is a binary value function indicating whether or not there is a path connecting two nodes, in the case of probabilistic graphs the function assumes probabilistic values.

The concept of *reachability* in probabilistic graphs is used, along with its specialization, as a tool to compute how two nodes in the graph are likely to be connected. Reachability plays an important role in a wide range of applications, such as in peer-to-peer networks, for probabilistic-routing problem, in road network, and in trust analysis in social networks. Reachability is quite similar to the general concept of *link prediction* [5], whose task may be formalized as follows. Given a networked structure (V, E) made up of a set of data instances V and a

set of observed links E among some nodes in V , the task corresponds to predict how likely should exist an unobserved link between two nodes. The extension to probabilistic graphs adds an important ingredient that should be adequately exploited. The key difference with respect to classical link prediction is that here the observed connections between two nodes cannot be considered always true, and hence methods exploiting probabilistic links are needed.

The goal of this paper is to provide a learning method to compute the most likely relationship between two nodes in probabilistic graphs. In particular, given a probabilistic graph we adopted the reachability tool to compute the probability of some possible interconnections that may exists between two nodes. Each of these connections may be viewed as a feature, or a pattern, between the two nodes and the corresponding probability as its weight. Each observed labeled link is considered as a positive instance for its corresponding link label. The link label corresponds to the value of the output variable y_i , and the features between the two nodes, computed with the reachability tool, correspond to the components of the corresponding vector \mathbf{x}_i . Given the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, obtained from n observed links, a L2-regularized Logistic Regression has been adopted to learn a model to be used to predict unobserved link labels. The proposed approach is quite similar to that of *propositionalization* proposed in the field of Statistical Relational Learning [6], where the relational data are flattened to a propositional representation using relational features in order to have efficient learning results. Here the further problem that we have to handle is that the relational representation is uncertain.

The application domains we chosen correspond to the problem of recommender systems [4] and to the protein interactions task [11]. In the first domain the aim is to predict the unknown rating between an user and an item, while in the second one the goal is to predict the presence or absence of an interaction between two proteins. Experiments proved that the proposed approach achieves significant results when compared to a Singular Value Decomposition (SVD) approach [14], representing one of the best recent methods for the recommendation task [9].

The rest of this paper is organized as follows. The next section introduces the probabilistic graph model. Then, Section 3 describes how the link classification problem is solved combing a linear classifier and a set of relational probabilistic features. Section 4 shows the results of the proposed approach on some real world problems. Lastly, Section 5 concludes the paper.

2 Probabilistic Graphs

Let $G = (V, E)$, be a graph where V is a collection of nodes and $E \in V \times V$ is the set of edges, or relationships, between the nodes.

Definition 1 (Probabilistic graph). A probabilistic graph is a system $G = (V, E, \Sigma, l_V, l_E, s, t, p_e)$, where (V, E) is an directed graph, V is the set of nodes, E is the set of ordered pairs of nodes where $e=(s,t)$, Σ is a set of labels, $l_V : V \rightarrow \Sigma$

is a function assigning labels to nodes, $l_E : E \rightarrow \Sigma$ is a function assigning labels to the edges, $s : E \rightarrow V$ is the source node of an edge, $t : E \rightarrow V$ is the target node of an edge, $p_e : E \rightarrow [0, 1]$ is a function assigning existence probability values to the edges.

The existence probability $p_e(a)$ of an edge $a = (u, v) \in E$ is the probability that the edge a , between u and v , can exist in the graph. A particular case of probabilistic graph is the *discrete graph*¹, where binary edges between nodes represent the presence or absence of a relationship between them, i.e., the existence probability value on all observed edges is 1. The *possible world semantics*, specifying a probability distribution on discrete graphs and formalized in the *distribution semantics* of Sato [15] for the first order logic, is usually used for probabilistic graphs. We can imagine a probabilistic graph G as a sampler of worlds, where each world is an instance of G . A discrete graph G' is sampled from G according to the probability distribution P_e , denoted as $G' \sqsubseteq G$, when each edge $a \in E$ is selected to be an edge of G' with probability $p_e(a)$. Edges labelled with probabilities are treated as mutually independent random variables indicating whether or not the corresponding edge belongs to a discrete graph.

Assuming independence among edges, the probability distribution over discrete graphs $G' = (V, E') \sqsubseteq G = (V, E)$ is given by

$$P(G'|G) = \prod_{a \in E'} p_e(a) \prod_{a \in E \setminus E'} (1 - p_e(a)). \quad (1)$$

Definition 2 (Simple path). *Given an uncertain graph G , a simple path of a length k from u to v in G is an acyclic path denoted as a sequence of edges $p_{u,v} = \langle e_1, e_2, \dots, e_k \rangle$, such that $e_1 = (u, v_1)$, $e_k = (v_{k-1}, v)$, and $e_i = (v_{i-1}, v_i)$ for $1 < i < k - 1$.*

Given an uncertain graph G , and $p_{u,v}$ a path in G from the node u to the node v , $\ell(p_{u,v}) = l_E(e_1)l(e_2) \cdots l(e_k)$ denotes the concatenation of the labels of all the edges in $p_{u,v}$. We adopt a *regular expression* R to denote what is the exact sequence of the labels that the path must contain.

Definition 3 (Language-constrained simple path). *Given a probabilistic graph G and a regular expression R , a language constrained simple path is a simple path p such that $\ell(p) \in L(R)$, where $L(R)$ is the language described by R .*

2.1 Inference

Given a probabilistic graph G , a main task corresponds to compute the probability that there exists a simple path between two nodes u and v , that is, querying for the probability that a randomly sampled discrete graph contains a simple path between u and v . More formally, the *existence probability* $P_e(q|G)$ of a simple path q in a probabilistic graph G corresponds to the marginal $P((q, G')|G)$ with respect to q :

¹ Sometimes called *certain graph*.

$$P_e(q|G) = \sum_{G' \sqsubseteq G} P(q|G') \cdot P(G'|G), \quad (2)$$

where $P(q|G') = 1$ if there exists the simple path q in G' , and $P(q|G') = 0$ otherwise. In other words, the existence probability of the simple path q is the probability that the simple path q exists in a randomly sampled discrete graph.

Definition 4 (Language-constrained simple path probability). *Given a probabilistic graph G and a regular expression R , the language-constrained simple path probability of $L(R)$ is*

$$P_e(q|L(R), G) = \sum_{G' \sqsubseteq G} P(q|G', L(R)) \cdot P(G'|G), \quad (3)$$

where $P(q|G', L(R)) = 1$ if there exists a simple path q in G' such that $\ell(q) \in L(R)$, and $P(q|G', L(R)) = 0$ otherwise.

The previous definition give us the possibility to compute the probability of a set of simple path queries, or patterns, fulfilling the structure imposed by a regular expression. In this way we are interested in discrete graphs that contain at least one simple path belonging to the language denoted by the regular expression.

Computing the existence probability directly using (2) or (3) is intensive and intractable for large graphs since the number of discrete graphs to be checked is exponential in the number of probabilistic edges. It involves computing the existence of the simple path in every discrete graph and accumulating their probability.

A natural way to overcome the intractability of computing the existence probability of a simple path is to approximate it using a Monte Carlo sampling approach [8]:

1. we sample n possible discrete graphs, G_1, G_2, \dots, G_n from G by sampling edges uniformly at random according to their edge probabilities; and
2. we check if the simple path exists in each sampled graph G_i .

This process provides the following basic sampling estimator for $P_e(q|G)$:

$$P_e(q|G) \approx \widehat{P_e(q|G)} = \frac{\sum_{i=1}^n P(q|G_i)}{n}. \quad (4)$$

Note that is not necessary to sample all the edges to check whether the graph contains the path. For instance, assuming to use an iterative depth first search (DFS) procedure to check the path existence. When a node is just visited, we will sample all its adjacent edges and pushing them into the stack used by the iterative procedure. We will stop the procedure either when the target node is reached or when the stack is empty (non existence).

3 Link Classification

After having defined the probabilistic graph, we can adopt language-constrained simple paths in order to extract probabilistic features (patterns) to describe the link between two nodes in the graph.

Given a probabilistic graph G , with the set V of nodes and the set E of edges, and $Y \subseteq \Sigma$ a set of edge labels, we have a set of edges $D \subseteq E$ such that for each element $e \in D$: $l_E(e) \in Y$. In particular D represents the set of observed links whose label belongs to the set Y .

Given the set of training links D and the set of labels Y we want to learn a model able to correctly classify unobserved links. A way to solve the classification task can be that of using a language based classification approach. Given an unobserved edge $e_i = (u_i, v_i)$, in order to predict its class $\hat{y}_i \in Y$ we can solve the following maximization problem:

$$\hat{y}_i = \arg \max_j P(q_j(u_i, v_i)|G), \quad (5)$$

where $q_j(u_i, v_i)$ is the unknown link with label $q_j \in Y$ between the nodes u_i and v_i . In particular, the maximization problem corresponds to compute the link prediction for each $q_j \in Y$ and then choosing that label with maximum likelihood.

The previous link prediction task is based on querying the probability of some language-constrained simple path. In particular, predicting the probability of the label q_j as $P(q_j(u_i, v_i)|G)$ in (5) corresponds to compute the probability $P(q|G)$ for a query path in a language L_j , i.e., computing $P(L_j|G)$ as in (3):

$$\hat{y}_j = \arg \max_j P(q_j(u_i, v_i)|G) \approx \arg \max_j P(q|L_j, G). \quad (6)$$

The previous query based approach consider the languages used to compute the (6) as independent form each other without considering any correlation between them. A more interesting approach that we want investigate in this paper is to learn from the probabilistic graph a linear model of classification combining the prediction of each language constrained simple path. In particular, given an edge e and a set of k languages $\mathcal{L} = \{L_1, \dots, L_k\}$, we can generate k real valued features x_i where $x_i = P(q|L_i, G)$, $1 \leq i \leq k$. The original training set of observed links D can hence be transformed into the set of instances $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1, \dots, n}$, where \mathbf{x}_i is a k -component vector of features $x_{ij} \in [0, 1]$, and y_i is the class label of the corresponding example \mathbf{x}_i .

Linear classification represents one of the most promising learning technique for problems with a huge number of instances and features aiming at learning a weight vector \mathbf{w} as a model. L2-regularized Logistic Regression belongs to the class of linear classifier and solves the following unconstrained optimization problem:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \left(\frac{\mathbf{w}^T \mathbf{w}}{2} + C \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) \right), \quad (7)$$

where $\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) = \xi(\mathbf{w}; \mathbf{x}_i, y_i)$ denotes the specific loss function, $\frac{1}{2} \mathbf{w}^T \mathbf{w}$ is the regularized term, and $C > 0$ is a penalty parameter. The decision function corresponds to $\text{sgn}(\mathbf{w}^T \mathbf{x}_i)$. In case of binary classification $y_i \in \{-1, +1\}$, while for multi class problems the one vs the rest strategy can be used.

Among many methods for training logistic regression models, such as iterative scaling, nonlinear conjugate gradient, quasi Newton, a new efficient and robust truncated Newton, called trust region Newton method, has been proposed [10]. In order to find the parameters \mathbf{w} minimizing $f(\mathbf{w})$ it is necessary to set the derivative of $f(\mathbf{w})$ to zero. Denoting with $\sigma(y_i \mathbf{w}^T \mathbf{x}_i) = (1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))^{-1}$, we have:

$$\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} + C \sum_{i=1}^n (\sigma(y_i \mathbf{w}^T \mathbf{x}_i) - 1) y_i \mathbf{x}_i = 0.$$

To solve the previous score equation, the Newton method requires the Hessian matrix:

$$\frac{\partial^2 f(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = \mathbf{I} + C \mathbf{X}^T \mathbf{D} \mathbf{X},$$

where \mathbf{X} is the matrix of the \mathbf{x}_i values, \mathbf{D} is a diagonal matrix of weights with i th diagonal element $\sigma(y_i \mathbf{w}^T \mathbf{x}_i)(1 - \sigma(y_i \mathbf{w}^T \mathbf{x}_i))$, and \mathbf{I} is the identity matrix.

The Newton step is $\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} + \mathbf{s}^{\text{old}}$, where \mathbf{s}^{old} is the solution of the following linear system:

$$\frac{\partial^2 f(\mathbf{w}^{\text{old}})}{\partial \mathbf{w} \partial \mathbf{w}^T} \mathbf{s}^{\text{old}} = -\frac{\partial f(\mathbf{w}^{\text{old}})}{\partial \mathbf{w}}.$$

Instead of using this update rule, [10] propose a robust and efficient trust region Newton method, using new rules for updating the trust region, whose corresponding algorithm has been implemented in the LIBLINEAR² system.

4 Experimental Evaluation

The application domains we chosen to validate the proposed approach are that of recommender systems and interactions between proteins.

In order to validate the proposed approach in recommender systems the first dataset we used is the *MovieLens* dataset³, made available by the GroupLens research group at University of Minnesota for the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems. We used the MovieLens 100K version consisting of 100000 ratings (ranging from 1 to 5) regarding 943 users and 1682 movies, divided into five folds. Each user has rated at least 20 movies and there are simple demographic info for the users (such as age, gender, occupation, and zip code). In this paper we used the ratings only without considering the demographic information.

The *Hetrec2011-lastfm* [2] dataset, related to recommender systems domain (music recommendation), is the second dataset we used to validate the proposed

² <http://www.csie.ntu.edu.tw/~cjlin/liblinear>.

³ <http://ir.ii.uam.es/hetrec2011/datasets.html>

method. This dataset contains social networking, tagging, and music artist listening information from a set of 2K users from Last.fm online music system. In this dataset we have 1892 users, 17632 artists, 12717 bi-directional user friend relations and 92834 user-artist relations. We have discretized the user-listened artist relations into three (equal bins) classes *play1*, *play2* and *play3* indicating the frequency with which a user has listened to a specific artist, where $play1 < play2 < play3$. Hetrec2011-lastfm dataset has been divided into 4 fold made up of ~ 70000 training ratings and ~ 20000 testing ratings.

For these two datasets the goal is to predict the user's interest with respect to an unknown object. In Movielens dataset, we want to predict the user's interest with respect to a new film, while in the hetrec2011-lastfm dataset the goal is to predict the frequency with which a user may listen to a new artist.

The last dataset we used, *ppi*, describes interactions among proteins [11]. The dataset is composed by 3888 probabilistic interactions among 918 proteins, and it has been divided into 4 fold consisting of 5832 training interactions and 1944 testing interactions, where further 3888 negative interactions between unlinked proteins have been added. Here the goal is to predict the presence or the absence of an interaction between two proteins.

Hence, in some domains both data and probabilistic relationships between them are observable (like in *ppi*), while in other domains (as in Movielens and hetrec2011-lastfm) it is necessary to elicit the uncertain relationships among the given evidence.

4.1 Probabilistic Graph Creation in Recommender System Domain

When we work with a set of data, in which the probabilistic relationships between data are hidden, a common approach to elicit these connections is based on using similarity measures. To model the data with a graph we can adopt different similarity measures for each type of node involved in the relationships.

In a recommender system domain we have two types of entities: the users and the items, and the only observed relationship corresponds to the ratings that a user has assigned to a set of items. The goal is to predict the rating a user could assign to an object that he never rated in the past. In the collaborative filtering approach there are two methods to predict unknown rating exploiting users or items similarity. User-oriented methods estimate unknown ratings based on previous ratings of similar users, while in item-oriented approaches ratings are estimated using previous ratings given by the same user on similar items.

Let U be a set of n users and I a set of m items. A rating r_{ui} indicates the preference degree the user u expressed for the item i , where high values mean stronger preference. Let S_u be the set of items rated from user u . A user-based approach predicts an unobserved rating \widehat{r}_{ui} as follows:

$$\widehat{r}_{ui} = \overline{r}_u + \frac{\sum_{v \in U | i \in S_u} \sigma_u(u, v) \cdot (r_{vi} - \overline{r}_v)}{\sum_{v \in U | i \in S_u} |\sigma_u(u, v)|}, \quad (8)$$

where $\overline{r_u}$ represents the mean rating of user u , and $\sigma_u(u, v)$ stands for the similarity between users u and v , computed, for instance, using the Pearson correlation:

$$\sigma_u(u, v) = \frac{\sum_{a \in S_u \cap S_v} (r_{ua} - \overline{r_u}) \cdot (r_{va} - \overline{r_v})}{\sqrt{\sum_{a \in S_u \cap S_v} (r_{ua} - \overline{r_u})^2 \sum_{a \in S_u \cap S_v} (r_{va} - \overline{r_v})^2}}.$$

On the other side, item-based approaches predict the rating of a given item using the following formula:

$$\widehat{r_{ui}} = \frac{\sum_{j \in S_u | j \neq i} \sigma_i(i, j) \cdot r_{uj}}{\sum_{j \in S_u | j \neq i} |\sigma_i(i, j)|}, \quad (9)$$

where $\sigma_i(i, j)$ is the similarity between the item i and j .

These neighbourhood approaches see each user connected to other users or consider each item related to other items as in a network structure. In particular they rely on the direct connections among the entities involved in the domain. However, as recently proved, techniques able to consider complex relationships among the entities, leveraging the information already present in the network, involves an improvement in the processes of querying and mining [17,16].

Given the set of observed ratings $\mathcal{K} = \{(u, i, r_{ui}) | r_{ui} \text{ is known}\}$, we add a node with label **user** for each user in \mathcal{K} , and a node with label **item** for each item in \mathcal{K} . The next step is to add the edges among the nodes. Each edge is characterized by a label and a probability value, which should indicate the degree of similarity between the two nodes. Two kind of connections between nodes are added. For each user u , we added an edge, labeled as **simU**, between u and the k most similar users to u . The similarity between two users u and v is computed adopting a weighted Pearson correlation between the items rated by both u and v .

In particular, the probability of the edge **simU** connecting two users u and v is computed as: $P(\mathbf{simU}(u, v)) = \sigma_u(u, v) \cdot w_u(u, v)$, where $\sigma_u(u, v)$ is the Pearson correlation between the vectors of ratings corresponding to the set of items rated by both user u and user v , and $w_u(u, v) = |S_u \cap S_v| / |S_u \cup S_v|$. For each item i , we added an edge, with label **simI**, between i and the most k similar items to i . In particular, the probability of the edge **simI** connecting the item i to the item j has been computed as: $P(\mathbf{simI}(i, j)) = \sigma_i(i, j) \cdot w_i(i, j)$, where $\sigma_i(i, j)$ is the Pearson correlation between the vectors corresponding to the histogram of the set of ratings for the item i and the item j , and $w_i(i, j) = |\overline{S}_i \cap \overline{S}_j| / |\overline{S}_i \cup \overline{S}_j|$, where \overline{S}_i is the set of users rating the item i . Finally, edges with probability equal to 1, and with label r_k between the user u and the item i , denoting the user u has rated the item i with a score equal to k , are added for each element (u, i, r_k) belonging to \mathcal{K} .

4.2 Feature Construction

After having constructed the probabilistic graph, the next step corresponds to the features construction that will serve as input to the classification model.

Adopting a recommender system dataset we can assume that the values of r_{ui} are discrete and belonging to a set R . Given the recommender probabilistic graph G , the query based classification approach try to solve the problem $\widehat{r}_{ui} = \arg \max_j P(\mathbf{r}_j(u, i)|G)$, where $\mathbf{r}_j(u, i)$ is the unknown link with label \mathbf{r}_j between the user u and the item i . This link prediction task is based on querying the probability of some language constrained simple path. For instance, a user-based collaborative filtering approach may be obtained by querying the probability of the edges, starting from a user node and ending to an item node, denoted by the regular expression $L_i = \{\mathbf{simU}^1\mathbf{r}_i^1\}$. In particular, predicting the probability of the rating j as $P(\mathbf{r}_j(u, i))$ corresponds to compute the probability $P(q|G)$ for a query path in L_j , i.e., $\widehat{r}_{ui} = \arg \max_j P(\mathbf{r}_j(u, i)|G) \approx \arg \max_j P(L_j|G)$. In the same way, item-based approach could be obtained by computing the probability of the paths constrained by the language $L_i = \{\mathbf{r}_i^1\mathbf{simI}^1\}$.

In the case of interactions among proteins we can assume that we have one label r_{ui} for all the edges. Given the proteins interactions probabilistic graph G , the query based classification approach try to solve the problem of querying the probability of some language constrained simple path made up of a series of homogeneous edges.

However, the power of the proposed framework is most evident when the labels of the edges are heterogeneous (as for the recommender system case). In fact, in such a situation our approach gives us the possibility to construct more complex queries such as that constrained by the language $L_i = \{\mathbf{r}_i\mathbf{simI}^n : 1 \leq n \leq 2\}$, that gives us the possibility to explore the graph by considering not only direct connections. Hybrid queries, such as those constrained by the language $L_i = \{\mathbf{r}_i\mathbf{simI}^n : 1 \leq n \leq 2\} \cup \{\mathbf{simU}^m\mathbf{r}_i^1 : 1 \leq m \leq 2\}$, give us the possibility to combine the user information with item information.

In order to use the feature based classification approach proposed in this paper we can define a set of regular expression \mathcal{L} and then computing for each language $L_i \in \mathcal{L}$ the probability $P(L_i|G)$ between two nodes in the graph. In particular in recommender system case, the set of observed ratings $\mathcal{K} = \{(u, i, r_{ui})|r_{ui} \text{ is known}\}$ is mapped to the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1, \dots, n}$, where x_{ij} is the probability $P(L_j|G)$ between the nodes u and i , and y_i is equal to r_{ui} . The proposed link classification method has been implemented in the **Eagle** system⁴ that provides a set of tools to deal with probabilistic graphs.

4.3 Validation

For each dataset, given the training/testing set, the validation procedure followed the steps:

1. creating the probabilistic graph from the training ratings data set as reported in Section 4.1;
2. defining a set \mathcal{L} of regular expressions to be used to construct a specific set of features as described in Section 4.2;

⁴ <http://www.di.uniba.it/~claudiotaranto/eagle.html>

3. learning the L2-regularized Logistic Regression model; and,
4. testing the links reported in the testing data set \mathcal{T} by computing, for each pair $(u, i) \in \mathcal{T}$ the predicted value adopting the learned classification model and comparing the result with the true prediction reported in \mathcal{T} .

For the ppi dataset, the first step is not necessary because the connections are observable. For MovieLens graph construction, edges are added using the procedure presented in Section 4.1, where we set the parameter $n = 30$, indicating that an user or a film is connected, respectively, to 30 most similar users (resp., films). The value of each feature have been obtained with the Monte Carlo inference procedure by sampling M discrete graphs. In order to construct the set of features, we proposed to query the paths belonging to the set of languages $\mathcal{L}ml_k$ reported in Table 1. The first language-constrained simple paths Lml_1 corresponds to adopt a user-based approach, while the second language Lml_2 gives us the possibility to simulate an item-based approach. Then, we propose to extend the basic languages Lml_1 and Lml_2 in order to construct features that consider a neighbourhood with many nested levels. Finally, we constructed hybrid features by combining both the user-based and item-based methods and the large neighbourhood explored with paths whose length is greater than one (Lml_5 , Lml_8 and Lml_9). We defined two sets of features $\mathcal{F}ml_1 = \{Lml_1, Lml_2, Lml_3, Lml_4, Lml_5\}$, based on simple languages, and $\mathcal{F}ml_2 = \{Lml_3, Lml_4, Lml_5, Lml_6, Lml_7, Lml_8, Lml_9\}$, exploiting more complex queries.

Table 1. Language constrained simple paths used for the MovieLens dataset

$Lml_1 = \{\text{sim}U^1 r_k^1\}$
$Lml_2 = \{r_k^1 \text{sim}F^1\}$
$Lml_3 = \{r_k^1 \text{sim}F^n : 1 \leq n \leq 2\}$
$Lml_4 = \{\text{sim}U^n r_k^1 : 1 \leq n \leq 2\}$
$Lml_5 = \{\text{sim}U^n r_k^1 : 1 \leq n \leq 2\} \cup \{r_k^1 \text{sim}F^n : 1 \leq n \leq 2\}$
$Lml_6 = \{r_k^1 \text{sim}F^n : 1 \leq n \leq 3\}$
$Lml_7 = \{\text{sim}U^n r_k^1 : 1 \leq n \leq 3\}$
$Lml_8 = \{\text{sim}U^n r_k^1 : 1 \leq n \leq 3\} \cup \{r_k^1 \text{sim}F^n : 1 \leq n \leq 3\}$
$Lml_9 = \{\text{sim}U^n r_k^1 : 1 \leq n \leq 4\} \cup \{r_k^1 \text{sim}F^n : 1 \leq n \leq 4\}$

For hetrec2011-lastfm graph construction, edges are added using the procedure presented in Section 4.1, where we set the parameter $n = 1500$, indicating that an user or an artist is connected, respectively, to 1500 most similar users, resp. artists. We defined two sets of features, as reported in Table 2: $\mathcal{F}_{wsr} = \{Llfm_1, Llfm_2, Llfm_3, Llfm_4, Llfm_5\}$ based on simple languages without considering the social relationships among the elements in the network, and $\mathcal{F}_{psr} = \{Llfm_1, Llfm_2, Llfm_3, Llfm_4, Llfm_5, Llfm_6, Llfm_7, Llfm_8\}$ in which social connections are considered.

In the ppi dataset we used the set of features \mathcal{F}_{ppi} reported in Table 3 based on simple and complex queries.

Table 2. Language constrained simple paths used for the hetrec2011-lastfm dataset

$Llfm_1 = \{\text{simUser}^1 r_k^1\}$
$Llfm_2 = \{r_k^1 \text{simArtist}^1\}$
$Llfm_3 = \{\text{simUser}^n r_k^1 : 1 \leq n \leq 2\}$
$Llfm_4 = \{r_k^1 \text{simArtist}^n : 1 \leq n \leq 2\}$
$Llfm_5 = \{\text{simUser}^1 r_k^1 \text{simArtist}^1\}$
$Llfm_6 = \{\text{friend}^1 r_k^1\}$
$Llfm_7 = \{\text{simUser}^1 \text{friend}^1 r_k^1\}$
$Llfm_8 = \{\text{friend}^1 r_k^1 \text{simArtist}^1\}$

Table 3. Language constrained simple paths used for the ppi dataset

$Lppi_1 = \{\text{interact}^1 \text{interact}^1\}$
$Lppi_2 = \{\text{interact}^1 \text{interact}^1 \text{interact}^1\}$
$Lppi_3 = \{\text{interact}^1 \text{interact}^1 \text{interact}^1 \text{interact}^1\}$
$Lppi_4 = \{\text{interact}^1 \text{interact}^1 \text{interact}^1 \text{interact}^1\} \cup \{\text{interact}^1 \text{interact}^1\}$
$Lppi_5 = \{\text{interact}^n : 1 \leq n \leq 3\}$
$Lppi_6 = \{\text{interact}^n : 1 \leq n \leq 4\}$

In order to learn the classification model as reported in Section 3, we used the L2-regularized Logistic Regression implementation included in the LIBLINEAR system [10]. Given a set \mathcal{T} of testing instances, the accuracy of the proposed framework has been evaluated according to the *macroaveraging mean absolute error* [1], for the recommender case,

$$MAE^M(\widehat{r}_{ui}, \mathcal{T}) = \frac{1}{k} \sum_{j=1}^k \frac{1}{|T_j|} \sum_{x_i \in T_j} |\widehat{r}_{ui} - r_{ui}|,$$

where $T_j \subset \mathcal{T}$ denotes the set of test rating whose true class is j , or with the conditional log likelihood, area under the Precision-Recall (AUC-PR) and Receiver Operating Characteristic (AUC-ROC) curves for the case of ppi dataset.

4.4 Results

Table 4 shows the results on MovieLens dataset obtained adopting the proposed approach implemented in the **Eagle** system when compared to those obtained with the RecSys SVD approach based implementation⁵. The first row reports the mean value of the MAE^M averaged on the five folds obtained with an SVD approach and with the proposed method. As we can see the error achieved by our method is lower than that obtained by the SVD method. The results improve when we use the set $\mathcal{F}ml_2$ of features. The difference of the results obtained with the two methods is statistically significant, with a p-value for the t-test equal to 0.0000004 when using the set $\mathcal{F}ml_1$ of features, and equal to 0.0000002 for the other set of features. The last two columns report the results of two baseline

⁵ <https://github.com/ocelma/python-recsys>

methods. The second last column reports the results obtained with a system that predicts a rating adopting a uniform distribution, while the last column reports the results of a system that uses a categorical distribution that predicts the value k of a rating with probability $p_k = |D_k|/N$, where D_k is the number of ratings belonging to the dataset having value k , and N is the total number of ratings.

Table 4. MAE^M values obtained with **Eagle** and SVD on MovieLens dataset

Fold	SVD	Eagle@ $\mathcal{F}ml_1$	Eagle@ $\mathcal{F}ml_2$	U	C
1	0.9021	0.8372	0.8044		
2	0.9034	0.8323	0.8055		
3	0.9111	0.8429	0.8256		
4	0.9081	0.8494	0.8231		
5	0.9159	0.8507	0.8270		
Mean	0.908±0.006	0.842±0.007	0.817±0.011	1.6	1.51
p-value		0.0000004	0.0000002		

In Table 5 we can see the errors committed by each method for each rating. The rows for the methods **U** and **C** report the mean of the MAE^M value for each fold using a system adopting a uniform or a categorical distribution. The dataset is not balanced and both the **SVD** and the proposed method adhere more to the categorical distribution proving that they are able to recognize the unbalanced distribution of the dataset.

Table 5. MAE^M values for each class obtained with **Eagle** and SVD on MovieLens dataset

Method	r1	r2	r3	r4	r5
U	2.0	1.4	1.2	1.4	2.0
C	2.53	1.65	1.00	0.89	1.47
SVD	1.62	1.03	0.55	0.44	0.88
Eagle@$\mathcal{F}ml_1$	1.14	0.80	0.65	0.65	0.93
Eagle@$\mathcal{F}ml_2$	1.03	0.73	0.66	0.66	0.96

Hetrec2011-lastfm dataset is composed by two types of edges: similarity edges (simUser and simArtist) and social relationship edges (friend). In this paper, we want to evaluate whether adopting the social connections improves the classification performances [7]. Table 6 shows the hetrec2011-lastfm results for each class comparing **Eagle@ \mathcal{F}_{wsr}** and **Eagle@ \mathcal{F}_{psr}** , we can see that **Eagle@ \mathcal{F}_{psr}** that adopt social relationship edges achieves better results than **Eagle@ \mathcal{F}_{wsr}** that does not use these connections.

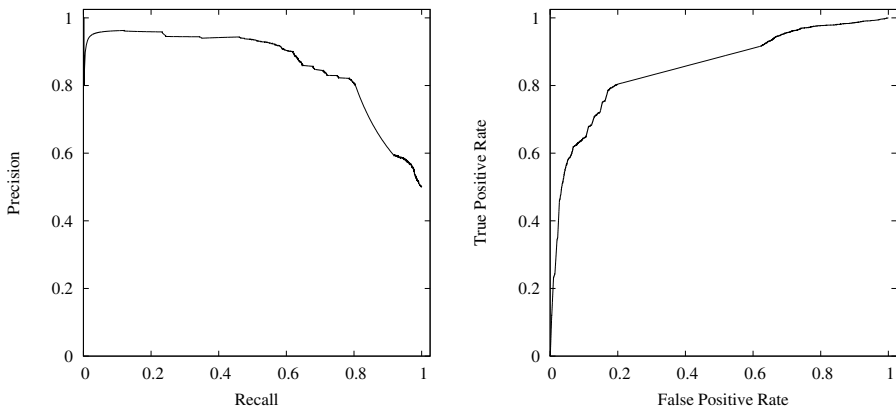
Table 7 shows the results on ppi dataset obtained adopting the proposed approach implemented in the **Eagle** system when compared to those obtained

Table 6. MAE^M values for each class obtained with **Eagle** on hetrec2011-lastfm dataset

Fold	Method	play1	play2	play3	All
1	Eagle @ \mathcal{F}_{wst}	0.6315	0.5686	0.4216	0.5405
	Eagle @ \mathcal{F}_{pst}	0.6047	0.2524	0.5946	0.4839
2	Eagle @ \mathcal{F}_{wst}	0.6090	0.5975	0.4460	0.5508
	Eagle @ \mathcal{F}_{pst}	0.5794	0.2326	0.6268	0.4796
3	Eagle @ \mathcal{F}_{wst}	0.6194	0.5875	0.4542	0.5537
	Eagle @ \mathcal{F}_{pst}	0.6062	0.1963	0.6796	0.4940
4	Eagle @ \mathcal{F}_{wst}	0.6295	0.6077	0.4181	0.5517
	Eagle @ \mathcal{F}_{pst}	0.5976	0.2432	0.5840	0.4749
Average Eagle @ \mathcal{F}_{wst}		0.6223	0.5903	0.4349	0.5492
Eagle @ \mathcal{F}_{pst}		0.5969	0.2311	0.6212	0.4831

Table 7. MAE^M, CLL, PR and ROC on ppi dataset

Fold	Train Set	Test Set	Random	Eagle
1	5829	1943	0.500	0.190
2	5829	1943	0.500	0.184
3	5829	1943	0.500	0.203
4	5829	1943	0.500	0.189
Mean			0.500	0.191
CLL			Mean	-0.439
			StdDev	0.144
PR			Mean	0.861
			StdDev	0.011
ROC			Mean	0.854
			StdDev	0.012


Fig. 1. On the left side AUC-PR and on the right side AUC-ROC on the ppi dataset

with a random approach. Furthermore we show for each CLL, PR and ROC the mean and standard deviation values. Figure 1 shows on the left side the PR curve and on the right side the ROC curve on the ppi dataset.

5 Conclusions

In this paper we adopt the probabilistic graphs framework to deal with uncertain problems exploiting both edges probabilistic values and edges labels denoting the type of relationships between two nodes. We proposed a learning method to compute the most likely relationship between two nodes in probabilistic graphs. Given the training set of observed links a L2-regularized Logistic Regression has been adopted to learn a model able to predict the label of unobserved links. The experimental evaluation proved that the proposed approach achieves better results when compared to that obtained with models induced by Singular Value Decomposition.

References

1. Baccianella, S., Esuli, A., Sebastiani, F.: Evaluation measures for ordinal regression. In: Proceedings of the 2009 9th International Conference on Intelligent Systems Design and Applications, ISDA 2009, pp. 283–287. IEEE Computer Society (2009)
2. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: Proceedings of the 5th ACM Conference on Recommender Systems, RecSys 2011. ACM, New York (2011)
3. Colbourn, C.J.: The Combinatorics of Network Reliability. Oxford University Press (1987)
4. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 107–144. Springer (2011)
5. Getoor, L., Diehl, C.P.: Link mining: a survey. SIGKDD Explorations 7(2), 3–12 (2005)
6. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
7. He, J., Chu, W.W.: A social network-based recommender system (snrs). In: Memon, N., Xu, J.J., Hicks, D.L., Chen, H. (eds.) Data Mining for Social Network Data. Annals of Information Systems, vol. 12, pp. 47–74. Springer (2010)
8. Jin, R., Liu, L., Ding, B., Wang, H.: Distance-constraint reachability computation in uncertain graphs. Proc. VLDB Endow. 4, 551–562 (2011)
9. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM (2008)
10. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region newton method for logistic regression. Journal of Machine Learning Research 9, 627–650 (2008)
11. Peregrin-Alvarez, J.M., Xiong, X., Su, C., Parkinson, J.: The modular organization of protein interactions in *escherichia coli*. PLoS Computational Biology 5(10) (2009)

12. Pfeiffer III, J.J., Neville, J.: Methods to determine node centrality and clustering in graphs with uncertain structure. In: Proceedings of the Fifth International Conference on Weblogs and Social Media. The AAAI Press (2011)
13. Potamias, M., Bonchi, F., Gionis, A., Kollios, G.: k-nearest neighbors in uncertain graphs. *Proc. VLDB Endow.* 3, 997–1008 (2010)
14. Pryor, M.H.: The effects of singular value decomposition on collaborative filtering. Tech. Rep. PCS-TR98-338, Dartmouth College, Computer Science, Hanover, NH (1998)
15. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming, ICLP 1995, pp. 715–729. MIT Press (1995)
16. Taranto, C., Di Mauro, N., Esposito, F.: Probabilistic inference over image networks. In: Agosti, M., Esposito, F., Meghini, C., Orio, N. (eds.) *IRCDL 2011. CCIS*, vol. 249, pp. 1–13. Springer, Heidelberg (2011)
17. Witsenburg, T., Blockeel, H.: Improving the accuracy of similarity measures by using link information. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) *ISMIS 2011. LNCS*, vol. 6804, pp. 501–512. Springer, Heidelberg (2011)
18. Zou, Z., Gao, H., Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 633–642. ACM (2010)

Improving Robustness and Flexibility of Concept Taxonomy Learning from Text

Fabio Leuzzi¹, Stefano Ferilli^{1,2}, and Fulvio Rotella¹

¹ Dipartimento di Informatica – Università di Bari

{fabio.leuzzi, stefano.ferilli, fulvio.rotella}@uniba.it

² Centro Interdipartimentale per la Logica e sue Applicazioni – Università di Bari

Abstract. The spread and abundance of electronic documents requires automatic techniques for extracting useful information from the text they contain. The availability of conceptual taxonomies can be of great help, but manually building them is a complex and costly task. Building on previous work, we propose a technique to automatically extract conceptual graphs from text and reason with them. Since automated learning of taxonomies needs to be robust with respect to missing or partial knowledge and flexible with respect to noise, this work proposes a way to deal with these problems. The case of poor data/sparse concepts is tackled by finding generalizations among disjoint pieces of knowledge. Noise is handled by introducing soft relationships among concepts rather than hard ones, and applying a probabilistic inferential setting. In particular, we propose to reason on the extracted graph using different kinds of relationships among concepts, where each arc/relationship is associated to a weight that represents its likelihood among all *possible worlds*, and to face the problem of sparse knowledge by using generalizations among distant concepts as bridges between disjoint portions of knowledge.

1 Introduction

The spread and abundance of electronic documents requires automatic techniques for extracting useful information from the text they contain. The availability of conceptual taxonomies can be of great help, but manually building them is a complex and costly task. Obtaining automatically *Full Text Understanding* is not trivial, due to the intrinsic ambiguity of natural language and to the huge amount of common sense and linguistic/conceptual background knowledge needed to switch from a purely syntactic representation to the underlying semantics. Nevertheless, even small portions of such knowledge may significantly improve understanding performance, at least in limited domains. Although standard tools, techniques and representation formalisms are still missing, lexical and/or conceptual taxonomies can provide a useful support to many NLP tasks, allowing automatic systems to exploit different kinds of relationships that are implicit in the text but required to correctly understand it. Building on previous work, we propose a technique to automatically extract conceptual graphs from text and reason with them. Since automated learning of taxonomies needs to be

robust with respect to missing or partial knowledge and flexible with respect to noise and to the need of shifting the representation, this work proposes a way to deal with these problems. Data poorness, concept sparsity and representation shift are tackled by finding generalizations among disjoint pieces of knowledge in order to build a bridge between them. Noise is handled by introducing soft relationships among concepts rather than hard ones, and applying a probabilistic inferential setting to reason on the extracted graph using different kinds of relationships, where each arc/relationship is associated to its likelihood of being true in all *possible worlds*.

This work is organized as follows: the next section describes related works; Section 3 outlines the proposed approach; then we present an evaluation of our solution; lastly we conclude with some considerations and future works.

2 Related Work

Many approaches have been attempted to build taxonomies and ontologies by mining large amounts of text. They can be arranged into two main groups: the former is based only on what is contained in the text, while the latter additionally exploits some external resources to fill the gap between the purely syntactic level and the semantic one. The former approaches are particularly indicated when there exists no kind of structured and *Machine-readable* external knowledge whatever. Following this approach, [1] builds concept hierarchies using Formal Concept Analysis by grouping objects with their attributes, which are determined from text by linking terms with verbs. Conversely, we are interested in building conceptual graphs by relying on the whole net of concepts and relationships rather than only on shared attributes. The other approach, proposed in [12], defines a language to build formal ontologies by deductive discovery as in logic programming. In particular, the author defines both a specific language for manipulating Web pages and a logic program to discover the concept lattice. Differently from [12], we do not limit the nature/kind of relationships to a predefined set, since new relationships are created as soon as a new (i.e., never encountered before) verbal relationship between concepts is found. On the other hand, approaches in the second group were focused on building taxonomies and/or ontologies. For instance, [11, 10] build ontologies by labeling taxonomic relations only, while we label also non-taxonomic ones with actions (verbs); [13] builds a taxonomy considering only concepts that are present in a domain but do not appear in others, while we are interested in all recognized concepts independently of their being generic or domain-specific.

As regards our proposal, for the syntactic analysis of the input text we exploit the *Stanford Parser* and *Stanford Dependencies* [9, 2], that can identify the most likely syntactic structure of sentences (including active/passive and positive/negative forms), and specifically ‘subject’ or ‘(direct/indirect) object’ components. They also normalize the words in the input text using lemmatization instead of stemming, which allows to distinguish their grammatical role and is more comfortable to read by humans.

We also use ProbLog [14] to apply probabilistic reasoning on the extracted knowledge. It is essentially Prolog where all clauses are labeled with the probability that they are true, that in turn can be extracted from large databases by various techniques. A ProbLog program $T = \{p1 : c1, \dots, pn : cn\}$ specifies a probability distribution over all its possible non-probabilistic subprograms according to the theoretical basis in [15]. The semantics of ProbLog is then defined by the success probability of a query, which corresponds to the probability that the query succeeds in a randomly sampled program. Indeed, the program can be split into a set of labeled facts $p_i :: f_i$, meaning that f_i is a fact with probability of occurrence p_i , and a Prolog program using those facts, which encodes the background knowledge (BK). Probabilistic facts correspond to mutually independent random variables (RVs), which together define a probability distribution over all ground logic programs $L \subseteq L_T$ (where L_T is the set of all f_i 's):

$$P(L|T) = \prod_{f_i \in L} p_i \prod_{f_i \in L_T \setminus L} (1 - p_i)$$

In this setting we will use the term *possible world* to denote the least Herbrand model of a subprogram L together with BK , and we will denote by L both the set of sampled facts and the corresponding world.

Lastly, we need in some steps of our technique to assess the similarity among concepts in a given conceptual taxonomy. A classical, general measure, is the *Hamming distance* [6], that works on pairs of equal-length vectorial descriptions and counts the number of changes required to turn one into the other. Other measures, specific for conceptual taxonomies, are sf_{Fa} [4] (that adopts a global approach based on the whole set of super-concepts) and sf_{WP} [16] (that focuses on a particular path between the nodes to be compared).

3 Proposed Approach

This proposal relies on a previous work [5], in which we assume that each noun in the text corresponds to an underlying *concept* (phrases can be preliminarily extracted using suitable techniques, and handled as single terms). A concept is described by a set of characterizing attributes and/or by the concepts that interact with it in the world described by the corpus. The outcome is a graph, where nodes are the concepts/nouns recognized in the text, and edges represent the relationships among these nodes, expressed by verbs in the text (the direction of edges denotes the role of the associated nodes in the relationship). In this work, we introduce two novelties: the former handles noise by weighting the relationships among concepts, where each arc/relationship is associated to a weight that represents its likelihood among all *possible worlds*; the latter faces the problem of sparse knowledge by using generalizations among distant concepts as bridges between disjoint portions of knowledge.

3.1 Graph Construction

Natural language texts are processed by the Stanford Parser in order to extract triples $\langle \textit{subject}, \textit{verb}, \textit{complement} \rangle$ that will represent the concepts (the *subjects* and *complements*) and attributes (*verbs*) for the graph. We have adopted some representational tricks: indirect complements are treated as direct ones by embedding the corresponding preposition into the verb; sentences involving verb ‘to be’ or nouns with adjectives contributed in building the sub-class structure of the taxonomy (e.g., “the penguin is a bird” yields *is_a(penguin, bird)*). For the sake of clarity, we must specify that all edges are verbal (labeled with the verb linking the two concepts) and among all we exploit the ones labeled with ‘is_a’ in order to build the taxonomy. In order to enrich the representation formalism previously defined, we analyzed the syntactic tree to seize the sentence positive or negative form based on the absence or presence (respectively) of a *negation modifier* for the verb. Moreover we decided to take into account separately the frequency of each arc between the concepts in positive and negative sentences.

This setting allowed us to give robustness to our solution through a statistical approach. In fact, the obtained taxonomy could be inspected and used by filtering out all portions that do not pass a given level of reliability. This could be useful for the identification of relevant concepts, as shown in [5], or for other applications that will be explained in the next two subsections.

3.2 Reasoning ‘by Association’

Reasoning ‘by association’ means finding a path of pairwise related concepts that establishes an indirect interaction between two concepts c' and c'' in the semantic network. In this work we propose two reasoning strategies: the former works in breadth and aims at obtaining the minimal path between concepts together with all involved relations, the latter works in depth and exploits ProbLog in order to allow probabilistic queries on the conceptual graph.

3.2.1 Non-probabilistic Reasoning

In this strategy, we propose to look for a minimal path using a *Breadth-First Search* (BFS) technique, applied to both concepts under consideration. Figure 1 shows an example of this kind of reasoning. The expansion steps of the two processes are interleaved, checking at each step whether the new set of concepts just introduced has a non-empty intersection with the set of concepts of the other process. When this happens, all the concepts in such an intersection identify one or more shortest paths connecting c' and c'' , that can be retrieved by tracing back the parent nodes at each level in both directions up to the roots c' and c'' . Since this path is made up of concepts only, to obtain a more sensible ‘reasoning’ it must be filled with the specific kind of interaction represented by the labels of edges (verbs) that connect adjacent concepts in the chain. In this work we provide also the number of positive/negative instances, and the corresponding ratios over the total, in order to help understanding different gradations (such as permitted, prohibited, typical, rare, etc.) of actions between two objects.

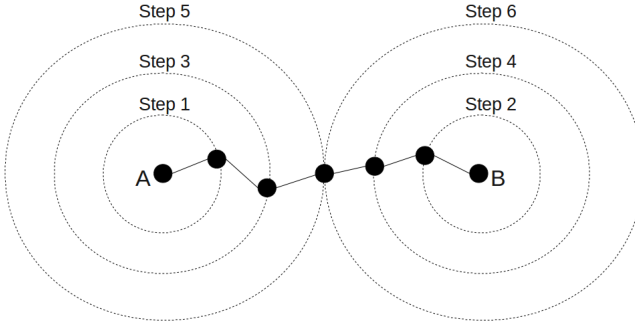


Fig. 1. Example of non-probabilistic reasoning

While this value does not affect the reasoning strategy, it allows to distinguish which reasoning path is more suitable for a given task.

3.2.2 Probabilistic Reasoning

Since real world data are typically noisy and uncertain, there is a need for strategies that soften the classical rigid logical reasoning. In particular, one might run into the above problems when knowledge is learned from text, as in our case. This requires an inference engine that allows to perform several kinds of probabilistic queries, like choosing the best (i.e., the most likely) path, or computing the exact probability of all possible paths between two concepts. We exploited ProbLog for this purpose, whose descriptions are based on the formalism $p_i :: f_i$ where f_i is a ground literal having probability p_i . In our case, f_i is of the form $link(subject, verb, complement)$ and p_i is the ratio between the sum of all examples for which f_i holds and the sum of all possible links between $subject$ and $complement$.

Figure 2 shows an example where many links are present between *farmer* and *plant*, expressing different kinds of interactions between these two concepts on which probabilistic reasoning can be applied. For example, if we ask for a path between *farmer* and *plant* and wonder what might be the most likely explanation thereof, we have to compute all different proofs or *explanations* (if any) of the query $?-path(farmer, plant)$ exploiting SLD-resolution. Each successful proof in the SLD-tree uses a set of facts $\{p_{i1} :: c_{i1}, \dots, p_{in} :: c_{in}\} \subseteq T$. Exploiting those facts we can compute the maximum probability as [8]:

$$P_x(q|T) = \max_{e \in E(q)} P(e|T) = \max_{e \in E(q)} \prod_{c_i \in e} p_i$$

where $E(q)$ is the set of all explanations for a query q . However one of the problems of these approaches is the tight connection between the quality of the reasoning results and that of the network, in turn depending on the processed

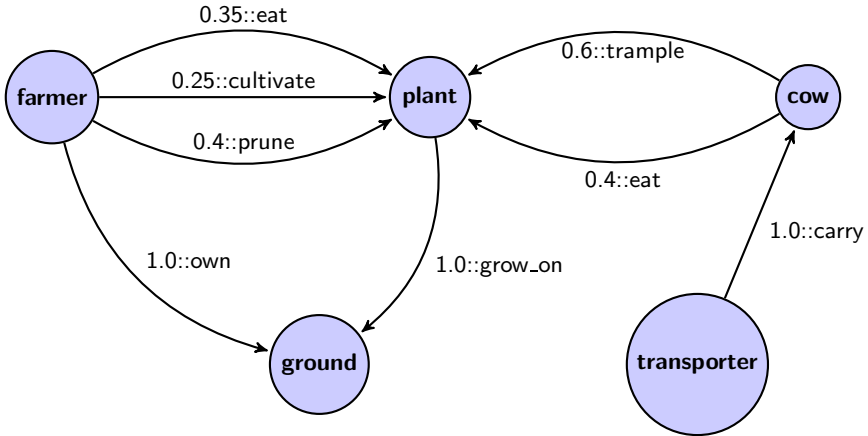


Fig. 2. Example of ProbLog Network

texts. Indeed, if two nodes belong to disjoint regions of the graph, reasoning cannot succeed. We tackle this problem by defining a generalization operator as follows.

3.3 Generalization Operator

Let us first provide a more precise account of our generalization.

Definition 1 (Generalization). *Given two concepts G and C , G generalizes C if anything that can be labeled as C can be labeled as G as well, but not vice-versa.*

The use of generalizations provides many opportunities of enrichment and/or manipulations on the graph:

1. building taxonomic structures by (repeatedly) applying this operator, also after the addition of new text (possibly causing the presence of new nodes in the graph);
2. shifting the representation, by removing the generalized nodes from the graph and leaving just their generalization (that inherits all their relationships to other concepts);
3. extending the amount of relationships between concepts belonging to the same connected component of the graph, or building bridges between disjoint components that open new (previously impossible) reasoning paths.

In particular, the third case can be viewed as a tool that aids reasoning ‘by association’ to reach its objectives, and hence as a multi-strategy setting in which induction and deduction cooperate.

More formally, we provide the following definition.

Algorithm 1. Pair-wise clustering of all concepts in the network.

Input: matrix $C \times C+A$ (where C is the set of objects/concepts, and A is the set of positive and negative verbs), that for each cell has the value 1 if at least one link exist, 0 otherwise; *THRESHOLD* for Hamming distance.

Output: set of clusters.

```

pairs ← empty
averages ← empty
for all  $O_i \mid O_i \neq \text{zero\_vector} \wedge i \in C$  do
  newCluster ←  $O_i$ 
  clusters.add(newCluster)
end for
for all  $\text{pair}(C_k, C_z) \mid C \in \text{clusters} \wedge k, z \in [0, \text{clusters.size}]$  do
  if completeLink( $C_k, C_z$ ) then
    pairs.add( $C_k, C_z$ )
    averages.add(getScoreAverage( $C_k, C_z$ ))
  end if
end for
pair ← getPairWithMaxMin(pairs, averages)
merge(pair)

```

completeLink → check the complete link assumption for the passed clusters.

getPairWithMaxMin → get the pair with the maximum or minimum average depending on task.

Definition 2 (Bridge). Given a piece of knowledge K represented as a concept graph made up of many disjoint regions, a bridge in K is a modification of the graph that connects some of such regions, allowing to reach any node of either from any node of the other. A bridge that connects n isolated sub-graphs is named n -way bridge.

Regardless of the task, three steps can be outlined to describe the general procedure:

1. *Concept Grouping*, in which all concepts are grossly partitioned to obtain subsets of concepts: we group similar concepts if the aim is to enrich the relationships, or dissimilar ones in the bridging perspective (Algorithm 1);
2. *Word Sense Disambiguation*, that associates a single meaning to each term by solving possible ambiguities using the domain of discourse (Algorithm 2);
3. *Computation of taxonomic similarity*, in which WordNet [3] is exploited in order to further filter with an external source the groups found in step 1, and to choose an appropriate subsumer (Algorithm 3).

To generalize two or more concepts, we use as their description their direct neighbor concepts plus the verbs (in positive or negative form) used to connect them. Thus, considering the set *Attributes* of all verbs in their positive and negative forms, we build a matrix \mathcal{C}

$$\text{Concepts} \times (\text{Concepts} \cup \text{Attributes})$$

where:

- $\mathcal{C}_{i,j} = 1$ if j denotes a concept column and there is at least a relationship between concepts i and j , $\mathcal{C}_{i,j} = 0$ otherwise;

Algorithm 2. Find “best synset” for a word.

Input: word t , list of domains with weights.

Output: best synset for word t .

```

best_synset ← empty
best_domain ← empty
for all synset( $s_t$ ) do
  max_weight ←  $-\infty$ 
  optimal_domain ← empty
  for all domains( $d_s$ ) do
    if weight( $d_s$ ) > max_weight then
      max_weight ← weight( $d_s$ )
      optimal_domain ←  $d_s$ 
    end if
  end for
  if max_weight > weight(best_domain) then
    best_synset ←  $s_t$ 
    best_domain ← optimal_domain
  end if
end for

```

- $C_{i,j} = 1$ if j denotes an attribute column and there is at least a relationship between concept i and verb j , $C_{i,j} = 0$ otherwise.

Each row is a feature vector describing the concept, and hence two vectors can be compared according to the *Hamming distance*. Pairwise clustering under the complete link assumption is applied to these descriptions: initially, each non-null row becomes a singleton cluster; then, clusters are merged while a merging condition is fulfilled. In its standard view, complete link states that *the distance of the farthest items of the involved clusters must be less than a given threshold*.

As stated earlier, generalizations can be carried out for different objectives, affecting the way complete link is applied. In particular, if the objective is the enrichment of relationships within connected components, it is applied in the standard way, otherwise, if the objective is to build bridges, *the distance of the closer items of the involved clusters must be greater than a given threshold*. When the condition is satisfied, the average score between all pairs of items in the two clusters is saved, and only the pair of clusters corresponding to the smallest (respectively, greatest) average is merged. We define more formally the clustering of dissimilar objects as follows.

Definition 3 (Inverse clustering). *Given a set of objects and a distance measure, inverse clustering is obtained by iteratively grouping most dissimilar objects while their distance is above a given distance threshold.*

Now, the clusters contain similar (resp., dissimilar) concepts that can be generalized in order to create new relationships (resp., to merge nodes) for enrichment (resp., bridging) purposes. However, this procedure alone might be unreliable, both because terms that occur seldom in the corpus have few connections (which would affect their cluster assignment due to underspecification), and because the expressive power of this formalism is too low to represent complex contexts

Algorithm 3. Effective generalization research.

Input: the set of C clusters returned by pair-wise clustering; T similarity threshold; max the max number of generalizations to try to extract from a cluster.

Output: set of candidate generalizations.

```

generalizations ← empty set
for all  $c \in C$  do
  good_pairs ← empty set
  for all pair( $O_i, O_j$ ) |  $i, j \in c$  do
    if similarity_score(pair( $O_i, O_j$ )) >  $T$  then
      good_pairs.add(pair( $O_i, O_j$ ), wordnet_hypernym(pair( $O_i, O_j$ )))
    end if
  end for
  for all  $i \in [0, max]$  do
    if good_pairs ≠ empty set then
      new_set ← {good_pairs.getBestPair, good_pairs.getStar}
      generalizations.add(new_set)
      good_pairs.remove(new_set)
    end if
  end for
end for

```

good_pairs → contains a list of pairs that satisfy T , with their relative subsumer.

good_pairs.getBestPair → get the pair that has the best similarity score.

good_pairs.getStar → get the Star of the pair

good_pairs.remove → remove all pairs in the passed set.

wordnet_hypernym → get the subsumer discovered in WordNet for the passed pair.

(which would affect even more important concepts). Note that the bridging setting is less affected by the underspecification problem, because it tends to group dissimilar concepts. Since underspecification corresponds to almost zero vectors, taking dissimilar vectors we tend to group the most specified distant vectors. Indeed, since there cannot be any 1 in the same positions in both descriptions (because this would mean that there exists a path between them), the more 1's overall in the two descriptions, the larger their distance, which means that the bridge is merging two hub (i.e., highly connected) nodes. This clearly improves the quality of the bridge. This solution allows to limit the shortest average length among all possible paths built between the sub-graphs that the bridge connects. However, the support of an external resource might be desirable. We consider WordNet as a sensible candidate for this, and try to map each concept in the network to the corresponding synset (a non trivial problem due to the typical polysemy of many words) using the *one domain per discourse* assumption as a simple criterion for Word Sense Disambiguation, whose algorithm is described in [5]. Thus, WordNet allows to check and confirm/reject the similarity of concepts belonging to the same cluster, by considering all possible pairs of concepts whose similarity is above a given threshold. Similarity is determined by blending two measures, that mutually smooth each other, as follows:

$$sf(A, B) = sf_{Fa}(A, B) \cdot sf_{WP}(A, B)$$

The former adopts a global perspective rather than depending on the choice of a single path between the two concepts [4]:

$$sf_{Fa}(i', i'') = sf(n, l, m) = \alpha \frac{l + 1}{l + n + 2} + (1 - \alpha) \frac{l + 1}{l + m + 2}$$

where:

- n is the number of ancestors of i' but not of i'' ;
- l is the number of common ancestors between i' and i'' ;
- m is the number of ancestors of i'' but not of i' ;
- α is a weight that determines the importance of i' with respect to i'' (0.5 means equal importance).

The latter considers the actual generalizing path [16]:

$$sf_{WP}(A, B) = \frac{2 * depth(lcs(\{A, B\}))}{depth(A) + depth(B)}$$

where:

- $depth(c)$ is the depth of concept c in the taxonomy;
- $lcs(C)$ is the *Least Common Subsumer* (LCS) in the taxonomy of concepts in C .

At this point, a set of similar pairs is selected, then the *star* of the best pairs is computed and used to obtain the *generalization set*. Let us state this more formally.

Definition 4 (Star). *Given a set S of unordered pairs of items and an unordered pair $P = \{a, b\}$ s.t. $P \in S$, we define the function $star(P)$ that returns a set of pairs S' s.t. $\forall P' \in S', (a \in P' \vee b \in P') \wedge P' \neq P$ (i.e., the set of pairs that contain a or b).*

Now we need again function $lcs(C)$, that returns the first common node found exploring the hypernyms of concepts in C in WordNet.

Definition 5 (Generalization set). *Given a set of concepts C , and a pair*

$$P = \arg \max_{\substack{P' \in C \times C \\ sf(P') > T}} sf(P')$$

the generalization set of P in C with threshold T is

$$gs(P, C, T) = \{i \mid i \in P \wedge P \in S\}$$

where T is a similarity threshold and $S = \{P'' \in star(P) \mid lcs(P) = lcs(P'')\}$.

After obtaining the generalization set, we can finally generalize it using the recognised least common subsumer. Obviously, we described how to obtain a generalization over a single star per cluster. However, removing the subset of pairs already used for the last generalization, we can reiterate this procedure more times obtaining many generalizations per cluster (if any).

Table 1. Examples of smooth reasoning ‘by association’ through BFS (start and target nodes in emphasis)

#	Subject	Verb	Complement
1	<i>young</i>	look [Pos: 3/3]	television
	television	talk_about [Pos: 3/3], critic [Pos: 1/1]	facebook
	facebook	help [Pos: 1/4, Neg: 3/4], distract [Pos: 1/1]	<i>schoolwork</i>
2	<i>people</i>	be_in [Pos: 1/1]	group
	group	invite [Pos: 2/2]	facebook
	facebook	help [Pos: 1/1]	<i>individual</i>
3	everyone	want [Pos: 5/5]	<i>occupation</i>
	occupation	maintain [Pos: 1/1]	lifestyle
	lifestyle	see_in [Pos: 1/1], change_in [Pos: 1/1],	<i>media</i>

4 Evaluation

The proposed approach was evaluated using *ad-hoc* tests, with the objective of obtaining qualitative outcomes that may indicate its strengths and weaknesses. Our aim is to keep massive experiments for further studies, in order to evaluate quantitative results only if qualitative ones are consistent. Although preliminary, these results seem enough to suggest that the approach is promising. We exploited a dataset made up of documents concerning *social networks* on socio-political and economic topic, including 669 objects (subjects and/or complements) and 727 verbs. The size of the dataset was deliberately kept small in order to increase the probability of having problems such as noise and poor knowledge. The settings of each specific operator are reported in the following subsections.

4.1 Reasoning ‘by Association’

The first experiment concerned reasoning ‘by association’, both deterministic and probabilistic. The former setting, a sample of whose outcomes is reported in Table 1, aimed at investigating the minimum path (if any) between two nodes in the graph. In order to show the strengths of this kind of reasoning, each verb is labeled with the frequency with which it occurs in the paths *subject*, *verb*, *complement*. Focusing on case 1, we wanted to explore the relationship between *young* and *schoolwork*. The association chain includes verb *look*, that occurs only in positive sentences with probability 1.0; this means that the available knowledge consistently indicates that *young* “always” *look television*. The same case involves a relationship between *facebook* and *schoolwork*, in which the verb *help* appears with probability 0.25 in positive sentences and 0.75 in negative ones. This can be interpreted as *facebook* “may” *help schoolwork*, or with the specific associated probability. It should be pointed out that this reasoning strategy shows all possible verbs between each pair of adjacent nodes in the path.

The latter experiment, reported in Table 2, shows a sample of probabilistic queries executed on the same paths. We have computed the exact success

Table 2. Examples of probabilistic reasoning ‘by association’ through ProbLog

#	Query	Probability
1	problog_exact(path(<i>young</i> - <i>schoolwork</i>))	0.530
	problog_max(path(<i>young</i> - <i>schoolwork</i>))	0.375
	problog_approx(path(<i>young</i> - <i>schoolwork</i>))	0.555
2	problog_exact(path(<i>people</i> - <i>individual</i>))	0.167
	problog_max(path(<i>people</i> - <i>individual</i>))	0.167
	problog_approx(path(<i>people</i> - <i>individual</i>))	0.162
3	problog_exact(path(<i>occupation</i> - <i>media</i>))	0.750
	problog_max(path(<i>occupation</i> - <i>media</i>))	0.500
	problog_approx(path(<i>occupation</i> - <i>media</i>))	0.744

Table 3. Generalizations for pairwise clustering of similar concepts, and corresponding conceptual similarity scores (bottom)

#	Bridge	Subsumer	Subs. Domain	Concepts	Conc. Domain
1	No	variable [105857459]	mathematics	variable [105857459] factor [105858317]	mathematics mathematics
2	No	person [100007846]	biology, person	type [109909060] collegian [109937056]	person factotum
3	No	person [100007846]	biology, person	type [109909060] model [110324851]	person person
4	No	integer [113728499]	mathematics	nineteen [113747989] forty [113749527]	number number
5	No	person [100007846]	biology, person	scholar [110251779] job [110222949] name [110344443]	school person person

#	Pairs	<i>Fa</i> score	<i>WP</i> score	Score
1	variable, factor	0.7	0.857	0.6
2	type, collegian	0.659	0.737	0.486
3	type, model	0.692	0.778	0.538
4	nineteen, forty	0.75	0.75	0.562
5	scholar, job	0.711	0.823	0.585
	scholar, name	0.678	0.778	0.528

probability, the most likely explanation probability and an approximate probability according to the MonteCarlo method implemented in [7] keeping a 95% of confidence interval. The last type of query has been provided because exact inference can be intractable even for small networks, and so sometimes it can be reasonable to make approximate inference. The probability of each sentence was computed according to the criteria described in Section 3.2.2. For instance, in case 1 with most likely explanation, the proof entails the set $\{(young, look, television), (television, talk_about, facebook), (facebook, not_help, schoolwork)\}$, respectively with probability $\{1.0, 0.75, 0.5\}$, whose product is 0.375.

Table 4. Generalizations for pairwise *Inverse clustering*, and corresponding conceptual similarity scores (bottom)

#	Bridge	Subsumer	Subs. Domain	Concepts	Conc. Domain
1	Yes	teaching [100887081]	pedagogy	talk [100893243] lecture [100892861]	pedagogy pedagogy
2	Yes	person [100007846]	biology, person	scientist [110560637] youth [110804406]	person person
3	Yes	music [107020895]	music	introduction [106396930] end [106398401]	literature, music literature, music
4	Yes	figure [113741022]	number	two [113743269] pair [113743605]	number number
5	Yes	person [100007846]	biology, person	viewer [110633450] model [110324851] volunteer [110759151]	person person person
6	No	theory [105888929]	factotum	basis [105793554] theory [105888929]	factotum factotum
7	No	municipality [108626283]	administration, geography, town planning	hometown [108671644] city [108524735]	geography literature, administration, geography, town planning
8	No	territorial dominion [108552138]	administration, town planning	state [108544813] department [108548733]	geography geography
9	No	structure [104341686]	factotum	wall [104546855] level [103365991]	buildings buildings
10	No	representation [104076846]	factotum	scene [106614729] photo [103925226]	photography, racing, sociology, telecommunication photography

#	Pairs	Fa score	WP score	Score
1	talk , lecture	0.739	0.869	0.643
2	scientist, youth	0.724	0.727	0.526
3	introduction, end	0.75	0.714	0.536
4	two, pair	0.72	0.823	0.593
5	viewer, model	0.71	0.666	0.474
	viewer, volunteer	0.71	0.666	0.474
6	basis, theory	0.694	0.842	0.584
7	hometown, city	0.738	0.842	0.621
8	state, department	0.75	0.75	0.562
9	wall, level	0.733	0.8	0.586
10	scene, photo	0.735	0.823	0.605

4.2 Generalization Operator

Two toy experiments are reported for concept generalization, the former aimed at the enrichment of relationships, the latter with the bridging perspective. The maximum threshold for the *Hamming distance* was set to 0.001, while the minimum threshold for *taxonomic similarity* was fixed at 0.45 in both.

Table 3 shows that, consistently with the enrichment-only perspective, no bridges are built. Conversely, applying *Inverse clustering* yields, as expected, also bridges among two or more disjoint graph regions. Analyzing the two conceptual similarity measures in both experimental settings, they both return very high values for almost all pairs, leading to final scores that neatly pass the 0.45 threshold. Another very interesting outcome is that sf_{WP} is always greater than sf_{Fa} for ‘enrichment’ generalizations. Since the former is more related to a specific path, and hence to the goodness of the chosen subsumer, this confirms the

previous outcomes (suggesting that the chosen subsumer is close to the generalized concepts). This regularity is not present in Table 4, where *Inverse clustering* was used to build bridges, which supports the motivations for using sf_{Fa} : the identification of similar concepts when this is not so evident based only on the single subsuming path. Observing the outcome, three aspects can be emphasized: the effectiveness of the search for bridges in case 5, in which a *three-way bridge* was built; the overall quality of the generalizations obtained; and the opportunity to perform not only representation shifts, but also an alignment as in case 1 of Table 3. Note that, after enriching the network, one is able to reason ‘by association’ also exploiting the *new* paths opened by generalization. We have decided to assign probability 1.0 to each generalization because an oracle, in our case WordNet, has suggested the common subsumer. For example, if we search for a path between *serenity* and *food*, we have to pass through verbs $\{\textit{find} [\textit{Pos}: 2/2], \textit{want} [\textit{Pos}: 3/3]\}$ for reaching *type* and then follow the new arc to reach *model*; then, through the verbs $\{\textit{eat} [\textit{Pos}: 1/4 \textit{Neg}: 3/4], \textit{buy} [\textit{Pos}: 2/2]\}$ we finally reach *food*.

5 Conclusions

This work proposes a technique to automatically extract conceptual graphs from text and reason with them. In particular, it presented a way to deal with missing (or partial) knowledge, noisy data, and discussed the need of shifting the representation, facing these problems through reasoning ‘by association’ and a generalization operator. Preliminary experiments confirmed the results of previous works and show that the approach can be viable, and suggested possible directions for future work. Further extensions and refinements include the definition of an Anaphora Resolution strategy that allows also handling of Named Entities, a study on how to set automatically suitable thresholds for searching generalizations, and the design of techniques that allow to find appropriate subsumers without the need for existing resources, such as WordNet.

References

- [1] Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Int. Res.* 24(1), 305–339 (2005)
- [2] de Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure trees. In: *LREC* (2006)
- [3] Fellbaum, C. (ed.): *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge (1998)
- [4] Ferilli, S., Biba, M., Di Mauro, N., Basile, T.M.A., Esposito, F.: Plugging taxonomic similarity in first-order logic horn clauses comparison. In: Serra, R., Cucchiara, R. (eds.) *AI*IA 2009*. LNCS, vol. 5883, pp. 131–140. Springer, Heidelberg (2009)
- [5] Ferilli, S., Leuzzi, F., Rotella, F.: Cooperating techniques for extracting conceptual taxonomies from text. In: *Proceedings of The Workshop on Mining Complex Patterns at AI*IA XIIth Conference* (2011)

- [6] Hamming, R.W.: Error detecting and error correcting codes. *Bell System Technical Journal* 29(2), 147–160 (1950)
- [7] Kimmig, A., Santos Costa, V., Rocha, R., Demoen, B., De Raedt, L.: On the efficient execution of probLog programs. In: Garcia de la Banda, M., Pontelli, E. (eds.) *ICLP 2008*. LNCS, vol. 5366, pp. 175–189. Springer, Heidelberg (2008)
- [8] Kimmig, A., De Raedt, L., Toivonen, H.: Probabilistic explanation based learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) *ECML 2007*. LNCS (LNAI), vol. 4701, pp. 176–187. Springer, Heidelberg (2007)
- [9] Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: *Advances in Neural Information Processing Systems*, vol. 15, MIT Press (2003)
- [10] Maedche, A., Staab, S.: Mining ontologies from text. In: Dieng, R., Corby, O. (eds.) *EKAW 2000*. LNCS (LNAI), vol. 1937, pp. 189–202. Springer, Heidelberg (2000)
- [11] Maedche, A., Staab, S.: The text-to-onto ontology learning environment. In: *ICCS-2000 - Eight International Conference on Conceptual Structures, Software Demonstration* (2000)
- [12] Ogata, N.: A formal ontology discovery from web documents. In: Zhong, N., Yao, Y., Ohsuga, S., Liu, J. (eds.) *WI 2001*. LNCS (LNAI), vol. 2198, pp. 514–519. Springer, Heidelberg (2001)
- [13] Cucchiarelli, A., Velardi, P., Navigli, R., Neri, F.: Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In: *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press (2006)
- [14] De Raedt, L., Kimmig, A., Toivonen, H.: Problog: a probabilistic prolog and its application in link discovery. In: *Proceedings of 20th IJCAI*, pp. 2468–2473. AAAI Press (2007)
- [15] Sato, T.: A statistical learning method for logic programs with distribution semantics. In: *Proceedings of the 12th ICLP 1995*, pp. 715–729. MIT Press (1995)
- [16] Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA, pp. 133–138. Association for Computational Linguistics (1994)

Discovering Evolution Chains in Dynamic Networks

Corrado Loglisci, Michelangelo Ceci, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro"
via Orabona, 4 - 70126 Bari - Italy
{corrado.loglisci,michelangelo.ceci,donato.malerba}@uniba.it

Abstract. Most of the works on learning from networked data assume that the network is static. In this paper we consider a different scenario, where the network is dynamic, i.e. nodes/relationships can be added or removed and relationships can change in their type over time. We assume that the “core” of the network is more stable than the “marginal” part of the network, nevertheless it can change with time. These changes are of interest for this work, since they reflect a crucial step in the network evolution. Indeed, we tackle the problem of discovering evolution chains, which express the temporal evolution of the “core” of the network. To describe the “core” of the network, we follow a frequent pattern-mining approach, with the critical difference that the frequency of a pattern is computed along a time-period and not on a static dataset. The proposed method proceeds in two steps: 1) identification of changes through the discovery of emerging patterns; 2) composition of evolution chains by joining emerging patterns. We test the effectiveness of the method on both real and synthetic data.

1 Introduction

In recent years, there has been a constantly growing interest in learning from networked data [5]. This is due to the fact that in many application domains, data naturally come in the form of a network, such as in protein interaction networks, social networks, linked web documents, and co-author networks, just to mention some of the most prominent examples. Any dataset represented as a set of relations and foreign key constraints in a relational database can be naturally represented as a network, which makes learning algorithms developed for networked data naturally applicable to any relational database. For the same reason, this class of algorithms is applicable to spatial data which are characterized by spatial relationships (e.g., topological, directional and distance-based relationships), although in this case the additional challenge comes from the fact that the (many) spatial relationships are *implicit* in the data [7].

Most of the algorithms developed to learn or analyze networked data assume that the network is static and unchangeable, i.e., the structure and the properties of a network do not vary over time. This assumption seems to be too restrictive in real scenarios where networks can be dynamic and exhibit changes especially

when modeling phenomena which evolve over time. In particular, nodes and edges of the networks may appear and disappear over time and relationships can change in their nature.

The importance of knowledge discovery from dynamic networks has been recognized only recently; hence the body of methods and techniques for the analysis of dynamic networks is much less developed than for static networks. Research on learning from dynamic networks follows three main lines: *i*) detection of communities over time, *ii*) characterization of the evolution of the networks, and *iii*) prediction of nodes/edges of the networks. Sun et al. [8] propose a technique to discover communities and detect changes in dynamic graphs represented in the form of contingency matrices with encoding schemes. A different approach principled on frequent graph-based patterns is reported in [2], where the representation of the time-evolving graphs as a sequence of cumulative graphs enables the discovery of rules which characterize the evolution of the network in terms of topological changes. Algorithms developed for predictive tasks are quite recent. Most of them try to make inferences at a specific time point, typically the time point next to the last observed. For instance, in [10] a hybrid framework combines the temporal information with topological patterns and a probabilistic relational model to infer the existence of links in social networks.

In this paper, we tackle a different task whose goal is to discover *evolution chains*, which express the temporal evolution of patterns, here intended as sets of labelled edges of the dynamic network. Indeed, in some applications where the network can be observed at different time-points, it is important to discover what is the “core” portion of the network which changes with time. We assume that the “core” of the network is more stable than the “marginal” part of the network, nevertheless it can change with time. These changes are of interest for this work, since they reflect a crucial step in the network evolution. To identify patterns concerning the “core” of the network, we follow a frequent pattern-mining approach, with the critical difference that the frequency of a pattern is computed along a time-period and not on a static dataset. In other terms, we assume that “frequent” patterns (along a time-period) do capture the more stable structure of the “core” of the network, while the many “infrequent” patterns do represent marginal aspects of the dynamic network, which are much more unstable and, thus, less interesting.

An example of evolution chain which can be extracted in the context of social network analysis is the following:

$$\begin{aligned} & \{(user_a, user_b, friendship), (user_b, user_c, participation_to_same_event)\}^{\tau_1} \\ & \{(user_a, user_b, friendship), (user_b, user_c, membership_to_a_group)\}^{\tau_2} \\ & \{(user_b, user_c, membership_to_a_group), (user_c, user_d, publish_on_the_wall_of)\}^{\tau_3} \end{aligned}$$

It includes three frequent patterns discovered in as many consecutive time-periods (τ_1 , τ_2 and τ_3). These patterns express topological regularities of the network. Nevertheless, not all frequent patterns are taken, but only those which meet two conditions:

- i) their frequency significantly changes between the considered time-period and the previous one;
- ii) the “similarity” between a pattern and the pattern associated with the previous time-period is maximized.

Thus, as a time-period is observed, we extract *emerging* patterns from that period and we incrementally join them with the sequence of patterns generated in the previous time-periods in order to generate complete evolution chains. In order to consider conservative periods, in the joining process we can join frequent patterns extracted from non-consecutive time-periods if, in the intermediate periods, they do not show any changes in frequency.

The paper is organized as follows. In Section 2 we formally define the problem of discovering evolution chains. The proposed computational solution is reported in Section 3. In Section 4 we report and discuss experimental results on both real and synthetic data. Finally, conclusions are drawn.

2 Problem Formulation

Before formally defining the problem we intend to solve, some definitions are necessary. Let $D = \langle D_1, D_2, \dots, D_n \rangle$ be a sequence of time-ordered observations of the network, obtained at regular time points. At each time-point t_i , the network is described by the set $D_i = (N_i \times N_i, E_i)$, $N_i \subseteq \mathcal{N}$ and $E_i \subseteq \mathcal{E}$, where \mathcal{N} and \mathcal{E} denote the sets of the nodes and types of edges observed in $\{t_1 \dots t_n\}$, respectively. An edge is modelled as a triple (n_1, n_2, e_{12}) , where n_1 and n_2 are the connected nodes while e_{12} is a label denoting the edge type.¹ We say that an edge (n_1, n_2, e_{12}) *occurs* at a time-point t_i if the observation $D_i = (N_i \times N_i, E_i)$ includes it, i.e., $(n_1, n_2, e_{12}) \in D_i$.

In this work, a *pattern* P is a set of edges. We say that P *occurs* at a time-point t_i if all edges in P occur at the same time-point t_i . To give the definition of frequent pattern in a temporal interval, we first introduce the concept of time-period.

A *time-period* (or simply *period*) τ in $\{t_1 \dots t_n\}$ is a sequence of consecutive time-points $\{t_i, \dots, t_j\}$ ($t_1 \leq t_i, t_j \leq t_n$). The width w of τ is the number of time-points in τ , i.e. $w = |\{t_i, \dots, t_j\}|$. Here we assume that all the periods have the same width w . Two periods $\tau = \{t_i, \dots, t_{i+w}\}$ and $\tau' = \{t_{i+w+1} \dots t_{i+2w}\}$ are said *consecutive*. As we consider consecutive time-periods, we can enumerate them and use the notation τ_{h+1} to indicate a period consecutive to τ_h .

Definition 1. *Given a time-period τ_h and a pattern P_{τ_h} we say that P_{τ_h} is frequent in τ_h if it occurs in at least minSupp time points of τ_h .*

We consider the relative frequency of P_{τ_h} computed as the number of time points in τ_h in which P_{τ_h} occurs, divided by the width of τ_h (i.e. $j - i + 1$). On the basis of the concept of frequent pattern, we can give the following definition:

¹ We assume that two nodes can be connected by multiple edges of different types and that edges are not symmetric.

Definition 2. Given a node $X \in \mathcal{N}$, an evolution chain L_X is a sequence of frequent patterns $\langle P_{\tau_h}, P_{\tau_{h+1}}, \dots, P_{\tau_{h+v}} \rangle$ where the node X belongs to some triple in P_{τ_h} and for each $i = 0, \dots, v-1$, $P_{\tau_{h+i+1}}$ differs from $P_{\tau_{h+i}}$ in only one triple. The sequence $\tau_{h+1}, \dots, \tau_{h+v}$ is called supporting period for the evolution chain.

Intuitively, the patterns $P_{\tau_{h+i}}$ and $P_{\tau_{h+i+1}}$ represent a relevant state of the network in the two consecutive periods τ_{h+i} and τ_{h+i+1} . The fact that $P_{\tau_{h+i+1}}$ differs from $P_{\tau_{h+i}}$ in only one triple guarantees that an evolution chains catches only slight differences in the structure of the patterns. This is coherent with condition *ii*), according to which the “similarity” between a pattern and the pattern associated with the previous time-period is maximized.

The problem we intend to solve can be formalized as follows:

Given: the set of time-stamped observations of the network $D = \langle D_1, D_2, \dots, D_n \rangle$, a set of consecutive time-periods τ_1, \dots, τ_m ($n \gg m$), and a node $X \in \mathcal{N}$,
Find: the set of evolution chains $\mathcal{L}_X = \{L_X\}$ whose supporting periods are included in τ_1, \dots, τ_m .

A computational solution to this problem is described in the following.

3 The Method

The proposed solution is structured in two-steps. The first step aims to discover emerging patterns, while the second step incrementally joins the extracted patterns in order to compose, through the periods, evolution chains. The two steps are detailed in the following.

3.1 Emerging Patterns to Represent Dynamic Networks

Emerging patterns (EPs) [6] are a particular kind of frequent patterns (FPs) used to characterize a partition of the data with respect to other partitions. The main property is that their support (relative frequency) significantly changes from one partition to another one. The greater the change of the support of a pattern, the more interesting the pattern. Changes in the support are quantitatively estimated in terms of growth rate (GR), which is a frequency ratio computed as the ratio $GR(P) = \text{supp}_{\text{partition}_i}(P) / \text{supp}_{\text{partition}_j}(P)$, where $\text{supp}_{\text{partition}_i}(P)$ is the support of the pattern P in the partition i and $\text{supp}_{\text{partition}_j}(P)$ is the support of P in the partition j . Examples of the application of emerging patterns in the spatio-temporal context can be found in [4],[3].

In our context, EPs are used to characterize the changes that the network may exhibit in a time-period with respect to the previous time-period both in the co-occurrences of the edges and in the presence of types of edges. In particular, EPs are discovered by evaluating the FPs generated in the period τ_i (FP_i) against those generated in the previous period τ_{i-1} (FP_{i-1}). Each pattern P of FP_i becomes emerging if it satisfies the following conditions:

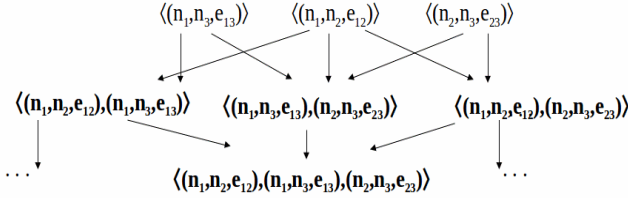


Fig. 1. The lattice generated during the process of frequent pattern mining

- it differs, for only one triple, from at least one of the patterns of FP_{i-1} ;
- there exists a pattern $P' \in FP_{i-1}$, $P' \neq P$ such that

$$GR(P, P') = \text{supp}_{\tau_i}(P) / \text{supp}_{\tau_{i-1}}(P') \geq \text{minGR},$$

where minGR is a user-defined threshold.

Note that the above definition of EP differs from the classical definition which captures differences in the support on the same pattern, and not on “slightly” different patterns. However, this divergence is necessary to catch evolutions which, otherwise, would not be detected.

Algorithmically, FPs are discovered in each period by exploiting the well-known *Apriori* algorithm [1]. In addition to frequent patterns, the Apriori algorithm returns also a graph-based structure (lattice) whose nodes correspond to possibly generated patterns while the edges denote a subset relationship among the connected patterns. More precisely, each edge connects a pattern P of length k to k patterns Q_1, \dots, Q_k of length $k - 1$ which are subsets of P . In Figure 1, the pattern $\langle\langle n_1, n_2, e_{12} \rangle, \langle n_1, n_3, e_{13} \rangle, \langle n_2, n_3, e_{23} \rangle\rangle$ of length 3 is connected with the three patterns $\langle\langle n_1, n_2, e_{12} \rangle, \langle n_1, n_3, e_{13} \rangle\rangle$, $\langle\langle n_1, n_3, e_{13} \rangle, \langle n_2, n_3, e_{23} \rangle\rangle$, $\langle\langle n_1, n_2, e_{12} \rangle, \langle n_2, n_3, e_{23} \rangle\rangle$. In the process of frequent pattern mining, it holds the *anti-monotonicity* of the support according to which if P is frequent then Q_1, \dots, Q_k are also frequent, while if one among Q_1, \dots, Q_k is infrequent, then P is infrequent too. This property is exploited in order to: *i*) generate k -length patterns from frequent $(k - 1)$ -length patterns, and *ii*) avoid to generate k -length patterns from infrequent $(k - 1)$ -length patterns.

In our approach, we exploit the anti-monotonicity property in the process of extracting emerging patterns. In particular, for each frequent k -length pattern P in FP_i , we consider its $(k-1)$ -length patterns Q_1, \dots, Q_k and we retrieve them from the frequent $(k-1)$ -length patterns in FP_{i-1} . From the retrieved set of frequent patterns in FP_{i-1} , we identify their corresponding (k) -length patterns in FP_{i-1} . The set of patterns obtained in this way (denoted as $\mathcal{P}_P, \tau_{i-1}$), contains patterns which have the same length of P , share $k - 1$ triples with P , and are frequent in τ_{i-1} . Then, P is considered to be an emerging pattern if $\exists P' \in \mathcal{P}_P, \tau_{i-1}$, $P' \neq P$, s.t. $GR(P, P') = \text{supp}_{\tau_i}(P) / \text{supp}_{\tau_{i-1}}(P') \geq \text{minGR}$.

A concrete example is reported in Figure 2. Let the pattern $P = \langle\langle n_1, n_2, e_{12} \rangle, \langle n_1, n_3, e_{13} \rangle, \langle n_2, n_3, e_{23} \rangle\rangle$ ($k=3$) be frequent in the period τ_i . The patterns of

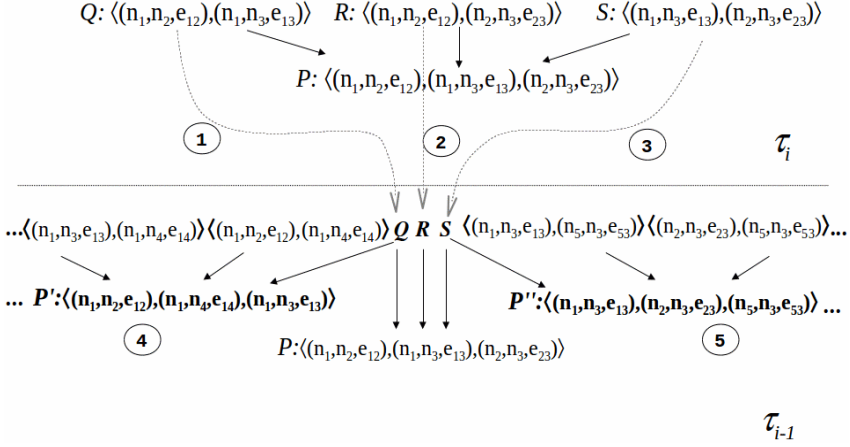


Fig. 2. Emerging patterns are selected among the frequent patterns which differ in one triple only

length $k=2$ connected to P are $Q : \langle (n_1, n_2, e_{12}), (n_1, n_3, e_{13}) \rangle$, $R : \langle (n_1, n_2, e_{12}), (n_2, n_3, e_{23}) \rangle$ and $S : \langle (n_1, n_3, e_{13}), (n_2, n_3, e_{23}) \rangle$.

Then, they are searched (circles 1,2,3) in the lattice of the period τ_{i-1} and, once found, are used to retrieve the set $\mathcal{P}_P, \tau_{i-1}$ of frequent k -length patterns which may have been derived by joining Q, R, S with other $(k - 1)$ -length patterns. In the example², k -length patterns are $P : \langle (n_1, n_2, e_{12}), (n_1, n_3, e_{13}), (n_2, n_3, e_{23}) \rangle$, $P' : \langle (n_1, n_2, e_{12}), (n_1, n_4, e_{14}), (n_1, n_3, e_{13}) \rangle$ and $P'' : \langle (n_1, n_3, e_{13}), (n_2, n_3, e_{23}), (n_5, n_3, e_{53}) \rangle$. Finally, we determine the growth-rate $supp_{\tau_i}(P)/supp_{\tau_{i-1}}(P')$ and $supp_{\tau_i}(P)/supp_{\tau_{i-1}}(P'')$: if at least one of these values exceeds the threshold $minGR$, we consider P as emerging.

3.2 Discovering Evolution Chains as Incremental Join of EPs

In order to formally define the problem of discovering evolution chains, some preliminary definitions have to be introduced. Let $\mathcal{S}_{\mathcal{N}} : \mathcal{N} \times \mathcal{N} \rightarrow [0, 1]$ and $\mathcal{S}_{\mathcal{E}} : \mathcal{E} \times \mathcal{E} \rightarrow [0, 1]$ be two similarity functions between nodes and types of edges, respectively. These two functions return real values and are here considered as background knowledge for the investigated problem. They can naturally model similarity among types of edges or similarity between types of nodes in heterogeneous networks (in the social networks, the similarity between the edges corresponding to “friendship” and “membership to the same group” can be 0.9).

Their availability is a quite reasonable assumption since in real-world networks we can easily define notions of similarity on nodes and types of edges.

Considering the notions introduced so far, the problem of discovering evolution chains in a dynamic network can be so formulated:

² For the sake of simplicity, in Figure 2 we do not report all the patterns which are derived by joining Q, R, S with other $(k - 1)$ -length patterns.

Given: the set of sets of EPs \mathcal{P} mined in the periods τ_1, \dots, τ_m ; $\mathcal{S}_{\mathcal{N}}$, $\mathcal{S}_{\mathcal{E}}$ two similarity measures on \mathcal{N} and \mathcal{E} respectively; $\sigma_{\mathcal{N}}$ and $\sigma_{\mathcal{E}}$ two minimum similarity thresholds for $\mathcal{S}_{\mathcal{N}}$ and $\mathcal{S}_{\mathcal{E}}$ respectively; a node $X \in \mathcal{N}$

Find: the set of evolution chains \mathcal{L}_X .

The intuition behind the solution here proposed is that dynamic networks actually may exhibit topological changes in some parts (nodes and relationships can be added/removed and relationships can change in their type over time) while keeping others unchanged, especially between adjacent periods. We use this intuition by considering as valid those chains which connect both unchanged and changed parts of the network. This is coherent with our assumption that the "core" of the network is more stable than the marginal part which makes our approach particularly adequate for networks that exhibit concept drift rather than concept shift [9].

The proposed solution joins EPs in adjacent periods (τ_{i-1}, τ_i) only if they have the same length (v edges) and differ for one edge: $v - 1$ edges would represent the unchanged part of the network while the two different edges (one for each FP used in the construction of the EP) would denote the changed part. It is noteworthy that this does not inhibit our approach from considering multiple changes in the same network, since multiple EPs can be extracted. When several patterns are candidates to be used for the join operation, we exploit the notion of similarity for nodes and types of edges by joining the candidate for which the new edge is "enough" similar to the removed one. Similarity is the average pairwise similarity between the nodes and the types of edges (computed according to $\mathcal{S}_{\mathcal{N}}$ and $\mathcal{S}_{\mathcal{E}}$). Indeed, in real-world dynamic networks, we do not expect drastic changes in adjacent periods but rather mild changes which could be originated from slight variations on the topological aspects and on the occurrences of the edges. The integration of the similarity measures $\mathcal{S}_{\mathcal{N}}$, $\mathcal{S}_{\mathcal{E}}$ allows us also to prevent the generation of meaningless and noise evolution chains.

In order to build chains and, at the same time, guarantee the completeness of the results, the approach adopts two mechanisms of space search:

- *backtracking*, which, starting from chains discovered until the previous time-period, explores backward the EPs of the previous periods in order to identify alternative chains;
- *skipping*, which, considering the possibility that EPs of adjacent periods could be not joined, analyzes forward the remaining periods in order to find EPs suitable for joining.

Indeed, this inability to join EPs in adjacent periods could be due to different factors: i) the nodes and the edges of the EPs might exhibit low similarity which does not exceed the minimum threshold, ii) EPs might present completely different edges or, conversely, identical edges. Indeed, when the FPs (in adjacent time-periods) are completely different, we cannot identify the unchanged parts of the network (as described above). On the contrary, when the FPs are the same and no EP can be joined, we cannot identify the changed parts of the network.

Algorithm 1. Discovering Evolution Chains

```

1: input:  $\mathcal{P}, \mathcal{S}_{\mathcal{N}}, \mathcal{S}_{\mathcal{E}}, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \text{minGR}, X \in \mathcal{N}$ 
   output:  $\mathcal{L}_X$ 
2:  $\text{found} := \text{false}; h := 1; \text{candidates} := \emptyset;$ 
3: while not found do
4:   for all  $EP \in \text{getEPs}(\mathcal{P}, h)$  do
5:     if  $\text{contains}(X, EP)$  then
6:        $\text{candidates} := EP$ 
7:        $\text{found} := \text{true}$ 
8:     end if
9:   end for
10:  if  $\text{candidates} = \emptyset$  then
11:     $h := h + 1$ 
12:  else
13:     $\text{selected} := \underset{EP \in \text{candidates}}{\text{arg min}} \text{length}(EP)$ 
14:     $\text{selected} := \underset{EP \in \text{selected}}{\text{arg max}} \text{Growth\_Rate}(EP)$ 
15:  end if
16: end while
17:  $i := h + 1$ 
18:  $\text{push}(\tau_{\text{stack}}, i)$ 
19:  $\text{mark}(\text{selected})$ 
20:  $\text{push}(EP_{\text{stack}}, \text{selected})$ 
21: while  $EP_{\text{stack}} \ll \emptyset$  do
22:   $\mathcal{L}_X \leftarrow \text{FWjoin}(\tau_{\text{stack}}, EP_{\text{stack}}, \mathcal{P}, \mathcal{S}_{\mathcal{N}}, \mathcal{S}_{\mathcal{E}}, \sigma_{\mathcal{N}}, \sigma_{\mathcal{E}}, \text{minGR}, \mathcal{L}_X)$ 
23:   $\text{pop}(EP_{\text{stack}})$ 
24:   $\text{pop}(\tau_{\text{stack}})$ 
25: end while

```

The algorithmic description is reported in Algorithms 1 and 2. In order to clarify how they work, we report an explanatory example in Figure 3. Consider τ_1, τ_2, τ_3 as time-periods, the input node X as n_1 and the thresholds $\sigma_{\mathcal{N}} = \sigma_{\mathcal{E}} = \sigma = 0.25$. The similarity measures $\mathcal{S}_{\mathcal{N}}, \mathcal{S}_{\mathcal{E}}$ return values reported in Figure 3a). The reflexive similarity of nodes and edges (e.g., (n_1, n_1)) is 1. As to the Algorithm 1, the first operation (lines 3-16) aims at finding the first period where X occurs. In the example, the search starts from τ_1 and finds n_1 in the EPs of τ_1 . If n_1 had not been found there, the search would have proceeded through the next periods. The presence of n_1 in a set of EPs (*candidates*) leads to select only one EP with minimum length and maximum growth rate (lines 13-14), that is, FP_1 in Figure 3b). The use of a length-based selection criterion is justified by the fact that the anti-monotonicity property of the support guarantees that the shorter the pattern, the higher the frequency, and the better the pattern represents the network. Moreover, the selection by growth rate allows us to consider EPs which better represent the changes in the network between the previous period and the current one. However, only when X is found in the first period of the network τ_1 , as in the current example, the selection is performed on the set of FPs and considers the length and support of the patterns (circles A,B in Figure 3b). This because we cannot discover EP in the first period τ_1 since EPs are determined by evaluating FPs in τ_i against the FPs in τ_{i-1} .

The EPs selected at the lines 13-14 and the associated period are stored in two stack structures (lines 18,20) which will be used to explore (in forward and backward mode) the next periods and the EPs there discovered.

Algorithm 2. Incremental Forward Join of EPs (*FWjoin*)

```

1: input:  $\tau\_stack, EP\_stack, \mathcal{P}, \mathcal{S}_N, \mathcal{S}_E, \sigma_N, \sigma_E, minGR$ 
   output:  $\mathcal{L}_X$ 
2:  $i := pop(\tau\_stack)$ 
3:  $m := |\mathcal{P}|$  {Number of time-periods}
4:  $selected\_EP := pop(EP\_stack)$ 
5: while  $i \leq m$  do
6:    $candidates \leftarrow getEPs(\mathcal{P}, i)$ 
7:    $candidates \leftarrow select\_by\_length(candidates, selected\_EP)$ 
8:    $candidates \leftarrow select\_by\_triples(candidates, selected\_EP)$ 
9:    $candidates \leftarrow remove\_marked(candidates)$ 
10:   $candidates \leftarrow select\_by\_similarity(candidates, selected\_EP, \mathcal{S}_N, \mathcal{S}_E, \sigma_N, \sigma_E)$ 
11:  for all  $EP \in candidates$  do
12:    if  $i < m$  then
13:       $candidate := \underset{EP \in candidates}{\arg \max} Growth\_Rate(EP)$ 
14:       $push(\tau\_stack, i)$ 
15:       $mark(candidate)$ 
16:       $push(EP\_stack, candidate)$ 
17:       $selected\_EP := candidate$ 
18:       $\mathcal{L}_X \leftarrow \mathcal{L}_X \cup concatenate(\mathcal{L}_X, selected\_EP, \{candidate\})$ 
19:    else
20:       $candidates \leftarrow select\_by\_minGR(candidates, minGR)$ 
21:       $\mathcal{L}_X \leftarrow \mathcal{L}_X \cup concatenate(\mathcal{L}_X, selected\_EP, candidates)$ 
22:    end if
23:  end for
24:   $i = i + 1$ 
25: end while

```

Once the latest selected EP is stored in the stack, it is considered for the possible join with EPs of the next periods (forward mode - lines 2,4 of the algorithm *FWjoin*). The operation is performed by first selecting, among the FPs identified as emerging (see Section 3.1), those which have the same length of *selected_EP* and differ from it in only one edge (lines 6-8). In the case no EP is found, the process *skips* the current period τ_i and continues the search in the next period (line 24). In the example, we have EP_3 and EP_4 (discovered in τ_2) which are identified as candidates to be joined with FP_1 (*selected_EP*), since they differ for only one triple from FP_1 (in Figure 3b, differences are represented by means of the circles C,D and E,F, respectively). Only one pattern among EP_3 and EP_4 , identified with the similarity measures and the growth rate values, will be used for the join with FP_1 (lines 10,13 and 20).

The measures $\mathcal{S}_N, \mathcal{S}_E$ are used to determine the similarity of the pairs of patterns (FP_1, EP_3) and (FP_1, EP_4) by considering the similarity among nodes and among types of edges in the different edges of (FP_1, EP_3) , and (FP_1, EP_4) . In particular, the similarity value between two patterns is obtained as the mean of three similarities obtained from the two different edges, two values obtained from the pairs of nodes and one obtained from the pair of edges: in the example, we have 0.45 for (FP_1, EP_3) and 0.4 for (FP_1, EP_4) (Figure 3a). Among the patterns for which the similarity and GR thresholds are exceeded, the chosen pattern is that which shows the highest growth rate value (lines 13, 20). In Figure 3a, EP_3 and EP_4 exceed the thresholds, but EP_3 is preferred for its highest value of similarity with FP_1 (0.45). The pattern EP_3 is stored (lines 14-16) and considered for subsequent join operations. In the subsequent iteration,

4 Experiments

In order to prove the viability of the proposed approach, we performed experiments on real world and synthetic datasets. The first one is a social political dynamic network derived from the news reports concerning the relationships among nations and world-wide organizations: social and political relationships correspond to the types of edges of the network while nations and world-wide organizations are the nodes. The second one has been specifically built in order to test the computational properties of the approach. Periods are determined according to an equal-width discretization technique which partitions the observations $D : \langle D_1, D_2 \dots D_i \dots D_n \rangle$ in a sequence $\langle \tau_1, \tau_2, \dots, \tau_m \rangle$ of consecutive time-periods with identical width.

4.1 Real-World Dynamic Network

The network is collected under the study KEDS (Kansas Event Data System)³. In this dataset, our approach aims at building an explanatory model able to identify particular connections established among nations over time as well as track the change of social and political relations.

Dataset Description. The dataset includes 123,821 edges collected between April 1979 and December 2009 (D) and the time-points are in the format year/month/day. The number of nodes \mathcal{N} is 228 while the types of edges are 20, i.e. 20 names of social and political relations reported in natural language. In this domain, understanding the evolution of the network in terms of type of edges (social and political relations) can be more interesting than considering the evolution on the nodes (nations). Coherently, we fix $\mathcal{S}_{\mathcal{N}}$ to return the middle of the range of the similarity, namely 0.5, while the measure $\mathcal{S}_{\mathcal{E}}$ is defined as the semantic similarity on the types of edges. In particular, we exploited the “Measures of Semantic Relatedness tools” (<http://cwl-projects.cogsci.rpi.edu/msr/>).

Experimental Setup. Experiments are performed to test the influence of the input parameters on the final evolution chains. Moreover, we define a quantitative measure in order to conduct an objective evaluation of the discovered chains. Such a measure estimates the *rarity* of the information expressed in each chain. More formally, let $L: EP_1, EP_2, \dots, EP_q$ be a chain discovered in the periods $\tau_1, \tau_2, \dots, \tau_m$ and let $[i_1, s_1), [i_2, s_2), \dots, [i_k, s_k]$ be a pre-defined equal-width discretization⁴ on the values of the growth-rate, the *rarity* is computed as:

$$rarity_{GR}(L) = 1 - \frac{1}{q} \left[\sum_{j=1,2,\dots,q} rarity_{GR}(EP_j) \right] \tag{1}$$

$$rarity_{GR}(EP_j) = \frac{\#EP_s^{(i,s)}}{\#EP_s^{\tau_j}} \tag{2}$$

where $\#EP_s^{(i,s)}$ is the number of EPs whose growth-rate is included in the same bin and $\#EP_s^{\tau_j}$ is the number of EPs generated in the period when EP_j

³ <http://web.ku.edu/keds/data.html>

⁴ Note that GR cannot be equal to infinity, since constructed from frequent patterns.

is generated. Therefore, the *rarity* of a chain ranges in $[0, 1]$ where the higher values the rarer the chain is. The same measure can be also defined for the similarity. Intuitively, the *rarity* is high for evolution chains whose EPs have less concurrent EPs in the same GR bin. In this case, the considered chain represents evolutions possibly not caught by other chains. According to its definition, the higher the *rarity*, the better the chain.

Results. Results are collected by varying the minimum thresholds σ_N , σ_E and $minGR$ with two different width of time-periods δ_τ . The thresholds σ_N, σ_E are set to the same value of σ . The first node X is set as "usa" (United States of America) and $minSupp=1.5\%$. Results are shown in Table 1 and Table 2 where we report the number of discovered chains, average length of the chains, average number of FPs and EPs generated in the periods involved in the chains and *rarity*. Each row in Table 1 presents the values averaged on $minGR=64, 8, 4, 2$, while the values of the rows in Table 2 are averaged on $\sigma=0.4, 0.25, 0.15, 0.1$.

A first consideration can be drawn from the number and length of the chains in Table 1: decreasing the minimum similarity leads to have a greater set of chains with higher length. Indeed, low values of σ lead to select EPs with low similarities in the join operations (besides those with high similarities) with the result of *i*) avoiding skipping and *ii*) continuing to apply the join operation for the chains currently processed. The same motivation applies also to the sets of FPs and EPs: the number of FPs and EPs tends to grow because we have to consider EPs with low similarities, due to the decrease of σ .

As expected, the threshold σ influences $rarity_{GR}$: the higher the value of similarity threshold the higher the average rarity. This allows us to point out a peculiarity of the approach: patterns of edges, which are dissimilar each other, can participate to a chain, but the resulting chains have relative low uniqueness in terms of frequency ($rarity_{GR}$), so chains which relate two nodes belonging to different time-periods with dissimilar intermediate edges can be very rare.

The different settings of δ_τ identify two different widths of the periods $\tau_1, \tau_{h+1}, \dots, \tau_m$: when δ_τ is 240 we have a smaller set of periods which explains smaller *avg length* and higher number of chains. By considering results in Table 1, we notice that $\delta_\tau = 120$ leads to better values of $rarity_{GR}$. Indeed, with a larger duration of a time-period ($\delta_\tau=240$) we collect a greater set of edges, namely observations of the network, which can lead to the generation of new patterns, which, in their turn, motivate the lower values of $rarity_{GR}$ with respect to $\delta_\tau=120$.

In Table 2 we can observe the correlation between the threshold $minGR$ and the $rarity_{similarity}$ and the influence of $minGR$ on the final chains. Indeed, low values of growth-rate (obtained by decreasing $minGR$) lead to consider a larger set of EPs (for the join operation) which can increase the probability that an higher number of EPs can fall into the bins of the discretization of $rarity_{similarity}$, with the final result of generating less rare chains. It is noteworthy that, in this case, δ_τ does not influence $rarity_{similarity}$. This means that chains are more uniformly distributed in terms of the similarity of involved EPs.

In the following we report the (unique) evolution chain obtained with $\sigma=0.4$, $minGR=64$, $X = "usa"$.

Table 1. Results with different values of σ and δ_τ

δ_τ	σ	# chains	avg length	avg FPs	avg EPs	rarity _{GR}
120	0,4	1,75	5,00	168,06	166,66	0,2875
	0,25	1,75	5,00	170,35	168,88	0,2875
	0,15	25,75	12,16	186,53	185,59	0,09028
	0,1	52,31	12,29	179,82	179,82	0,07775
240	0,4	23,75	2,96	380,73	351,70	0,09475
	0,25	24,25	3,45	977,71	945,55	0,08875
	0,15	30,5	4,10	937,74	905,88	0,06180
	0,1	32	4,39	931,98	900,31	0,06750

Table 2. Results with different values of $minGR$ and δ_τ

δ_τ	minGR	# chains	avg length	avg FPs	avg EPs	rarity _{similarity}
120	64	4,25	6,325	172,87	172,2075	0,87387
	8	16,75	9,09	177,63	176,58	0,4875
	4	14,75	9,66	177,67575	176,63	0,6204
	2	45,8125	9,3725	176,5825	175,53	0,846
240	64	24,25	3,7725	813,75	780,825	0,92625
	8	23,75	3,995	704,9425	673,7375	0,91275
	4	23,5	3,965	712,09675	681,5525	0,87325
	2	39	3,165	997,365	967,325	0,74833

$\{(usa, isr, consult), (igo, pse, consult)\}^{(1979-12-13/1980-04-11)}$,
 $\{(usa, isr, consult), (syr, usa, consult)\}^{(1981-04-10/1981-08-08)}$,
 $\{(usa, isr, consult), (isr, usa, appeal)\}^{(1981-08-09/1981-12-07)}$,
 $\{(usa, isr, consult), (isr, usa, consult)\}^{(1984-08-02/1984-11-30)}$,
 $\{(igo, isr, appeal), (isr, usa, consult)\}^{(1998-07-02/1998-10-30)}$

It describes the chain developed from the period 1981 – 04 – 10/1981 – 08 – 08 to the period 1998 – 07 – 02/1998 – 10 – 30 and has "usa" as first node and "igo" (Intergovernmental organizations) as last node. It depicts the evolution on the edges of the network which involve also the nodes "pse", "syr", "isr" (Palestinian Occupied Territories, Syria, Israel). This chain has $rarity_{GR}=0.22$ and $rarity_{similarity}=0.706$.

With $\sigma=0.1$ and $minGR=64$ we obtain the following evolution chain:

$\{(usa, lbn, consult), (lbn, usa, consult)\}^{(1979-12-13 \ 1980-04-11)}$
 $\{(usa, lbn, express_intent_to_cooperate), (lbn, usa, consult)\}^{(1983-04-06 \ 1983-08-04)}$
 $\{(usa, lbn, express_intent_to_cooperate), (lbn, usa, fight)\}^{(1983-08-05 \ 1983-12-03)}$
 $\{(usa, lbn, fight), (lbn, usa, fight)\}^{(1983-12-04 \ 1984-04-02)}$

In this chain ($rarity_{GR}=0.14$, $rarity_{similarity}=0.4$), the relation between the nodes "usa" and "lbn" (Lebanon) changes from "consult" to "fight" through "express_intent_to_cooperate". Note that in the time-period 1983-08-05 / 1983-12-03 there is an asymmetric relationship among the nodes "usa" and "lbn".

4.2 Synthetic Dynamic Network

Synthetic datasets are generated by varying δ_τ , cardinality of \mathcal{E} and \mathcal{N} as well as the number of edges per time-point. In Table 3, we report a summary of the

Table 3. Artificial dataset characteristics

δ_τ	$ \mathcal{N} $	$ \mathcal{E} $	#edge types
500	(5,10,15,20)	(5,10,15,20)	(5,10,15,20)
1000	(5,10,15,20)	(5,10,15,20)	(5,10,15,20)
2000	(5,10,15,20)	(5,10,15,20)	(5,10,15,20)
3000	(5,10,15,20)	(5,10,15,20)	(5,10,15,20)

Table 4. Experiments on synthetic datasets

δ_τ		(# nodes, # edges, #edge types)			
		(5,5,5)	(10,10,10)	(15,15,15)	(20,20,20)
500	avg length	2	2.14	2.02	2
	time(mins)	68.1	147.8	162.9	97.67
	# chains	2000	7891	6937	19
1000	avg length	2	2.47	2.11	-
	time(mins)	50.4	113.1	27.96	20.3
	# chains	1360	48894	165	0
2000	avg length	2	2.15	2.24	-
	time(mins)	66.4	117.6	60.7	38.4
	# chains	2000	3001	1074	0
3000	avg length	2	2.23	2.18	-
	time(mins)	80	145.2	71.7	41.8
	# chains	1960	3225	3082	0

artificial datasets: for instance, when $\delta_\tau=500$, we have four datasets where the cardinalities of \mathcal{N} , \mathcal{E} and the number of types of edge per time-point are equal to (5,5,5), (10,10,10), (15,15,15), (20,20,20) for the first, second, third and fourth dataset, respectively. The edges of a period τ_i are generated independently from those of other periods in order to evaluate our approach in the (worst) case in which (possible) chains are randomly generated. This choice avoids possible biases introduced by the criterion used in the generation of chains, but, on the other hand, only allows a quantitative evaluation of generated chains and not a qualitative evaluation. Coherently with this choice, we set the threshold *minGR* to 1.0. The values of similarities among the nodes and among the types of edges are identical and set to 0.1 ($\sigma=0.1$).

A first observation we can draw (see Table 4) is that the computational cost of the approach is related to the produced results, namely the number of discovered chains and their length: the time performances grows up when # chains and *avg length* increase, especially in the settings (10,10,10), (15,15,15). Indeed, when the network becomes more complex (e.g. (20,20,20)), the running times are shorter due to the small number of discovered chains. This behavior can be motivated by the fact that the increase of the size of \mathcal{N} and \mathcal{E} does not imply an increase in the frequency of the patterns and, subsequently, of the emerging patterns and chains, with shorter running time for their (incremental) evaluation.

5 Conclusions

In this paper we investigated the task of discovering evolution chains in dynamic networks. The proposed solution is based on the extraction of emerging patterns

which are subsequently joined in order to generate evolution chains expressed as time-period stamped patterns. Experiments prove the applicability of the approach in real-world challenges. In particular, obtained results qualitatively prove the soundness and the usefulness of extracted chains in capturing changes in the “core” of a social and political network. Moreover, experiments on artificially generated data show that the algorithm well scales on large datasets, depending on the data distribution. For future work, we plan two directions: *i*) automatic determination of time-period widths on the basis of the underlying distribution of the data, *ii*) discovering chains in streaming environments where the networks typically exhibit gradual and sudden concept drift.

Acknowledgments. This work is in partial fulfillment of the PRIN 2009 Project “Learning Techniques in Relational Domains and Their Applications” funded by the Italian Ministry of University and Research (MIUR).

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) VLDB, pp. 487–499. Morgan Kaufmann (1994)
2. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining graph evolution rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
3. Ceci, M., Appice, A., Loglisci, C., Caruso, C., Fumarola, F., Malerba, D.: Novelty detection from evolving complex data streams with time windows. In: Rauch, J., Raś, Z.W., Berka, P., Elomaa, T. (eds.) ISMIS 2009. LNCS, vol. 5722, pp. 563–572. Springer, Heidelberg (2009)
4. Ceci, M., Appice, A., Malerba, D.: Discovering emerging patterns in spatial databases: A multi-relational approach. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenić, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 390–397. Springer, Heidelberg (2007)
5. Di Mauro, N., Malerba, D.: Mining networked data. In: Chawla, N., King, I., Sperduti, A. (eds.) Symposium on Computational Intelligence and Data Mining, IEEE-CIDM 2011, p. xx. IEEE (2011)
6. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: KDD, pp. 43–52 (1999)
7. Malerba, D.: A relational perspective on spatial data mining. IJDMMM 1(1), 103–118 (2008)
8. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 687–696. ACM, New York (2007)
9. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
10. Zhu, J., Xie, Q., Chin, E.J.: A hybrid time-series link prediction framework for large social network. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part II. LNCS, vol. 7447, pp. 345–359. Springer, Heidelberg (2012)

Supporting Information Spread in a Social Internetworking Scenario

Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino

DIMET, University “Mediterranea” of Reggio Calabria, Via Graziella,
Località Feo di Vito, 89122 Reggio Calabria, Italy
{bucca,lax,a.nocera,ursino}@unirc.it

Abstract. The problem of quickly, capillary and effectively spreading information over social networks has become extremely important in many areas of our society. This problem has been widely studied in the recent literature and is still open, but it becomes even more challenging, due to the new issues to deal with, in a multi-social-network context, where the possibility that information can cross different social networks has a fundamental role. As a matter of fact, this is the scenario towards which social networks are evolving with a rapid increase of the mutual interaction among them. In this new scenario, called *Social Internetworking Scenario* (SIS, for short), we propose an approach devoted to favor information spreading, by identifying two stereotypes, specific for SISs, which are expected to be good spreaders: the *starter* and the *bridge*.

Keywords: Social Networks, Social Internetworking Scenarios, Information Spread, Starters, Bridges.

1 Introduction

The problem of spreading information over large communities as much quickly, capillary and effectively as possible, has a crucial importance in many areas, like economy, government, culture, society, etc. The problem is much more evident if we consider new generation communication systems, such as social networks, which are radically changing the communication model with respect to the past. Information spreading has been first considered in the context of economy, above all in marketing [21,41]. Here, the communication happens in a broadcast fashion, starting from a given subject (i.e., a company) and aiming at directly reaching a large number of other subjects (i.e., customers). The networks considered in this context are the provider-consumer ones, generally very different from the current social networks. For instance, in the past, a direct relationship between providers and consumers who were linked to each other by means of advertising systems operating in broadcasting and/or by means of large-scale distributors did not often exist. With the appearance of social networks, this frame of mind drastically changed, since at their core there is the direct relationship between two partners. This fact influenced also economy and marketing since, now, thanks to

e-commerce, the relationship between providers and consumers is often direct. This new form of communication may give relevant benefits in terms of effectiveness. Think, for instance, of a suggestion to buy a product provided by a friend with respect to the same suggestion provided by a TV advertising. Thus, the role of recipients becomes very different, moving from the passive one of the past to an active participation to the communication process. This radical change led to the necessity of designing new approaches to improving information spreading, which strictly take both the new communication architecture and the role of involved actors into account. The recent literature includes a lot of efforts in this direction, like [22,27,13,7], and the problem is still open. However, an even newer phenomenon should be taken into account. Social networks are proliferating, and users are going towards contemporary memberships to different social networks, also addressing multiple aspects of their personality. Each community is not an isolated element, but a component of a overall scenario where all the communities interact with each other to communicate, to share resources, to acquire opinions, and so forth. This is what is called Social Internetworking Scenario (SIS, for short). Here, information spreading is even more challenging and includes some new issues to deal with, mainly related to the possibility that information can cross different social networks. We expect that if we apply to a SIS one of the techniques conceived in the literature for facilitating information spreading in a (single) social network, we will be far from the best possible result, since those techniques do not take into account the problem of *cross-spreading*, which is instead crucial in this context. Unfortunately, due to the currentness of the social internetworking field, in the literature no technique exists (to the best of our knowledge) to attack the above problem in the context of SISs. This work is aimed at giving a first proposal in this setting, by defining a suitable model and a method which enhance single-social-network approaches by taking the features of the new context into account. In particular, the stereotypes we look for in order to identify good information spreaders are of two kinds. The former, which we call *starter*, is a sort of leader, who is often (but not necessarily) a power user with high activity on the network, heterogeneous interests (in such a way that she can be followed by many people with different interests and needs [9,46]) and with good attractiveness (in the sense that the information posted by her in the past attracted the interest of many other users). The latter is specific of the SIS context. It is called *bridge*. The identikit of a bridge requires that she should join more social networks and, above all, that she and her friends should be characterized by a high centrality degree, and these last ones should be interested in the topics characterizing the information to spread. It is worth noting that the above stereotypes depend on the topic, in the sense that it can happen that a user is a good starter (bridge, resp.) for a certain information topic and a bad starter (bridge, resp.) for another one. In our model a user is characterized by a *starter degree* and a *bridge degree*. The best users to spread information are certainly those having both a high starter degree and a high bridge degree. Among the other ones, if a SIS consists of few social networks, starter degree must be preferred to bridge degree. By contrast, if a SIS consists

of many social networks, bridge degree must be preferred to starter degree. On the basis of these degrees, we may also evaluate the maximum spread that can be obtained for a given information entity starting from a certain user considered as a starter. As a final remark, we can observe that even though our approach is based on the usage of only data marked by users as public, a problem of privacy anyway arises. Indeed, we use such public data for classifying people (although our classification does not include any negative connotation), in order to discover knowledge about them which is not voluntarily and explicitly included by users. However, this fact is inherently related to the social network phenomenon, where users are in general aware (or at least they should be aware) that exposing a part of their life leads to the concrete possibility that someone can collect, analyze, elaborate such an information and can exploit it also for business. This is in fact what currently happens in real life, so that we feel that the results of our research may have a meaningful applicative impact.

The plan of this paper is as follows. In the next section, we present the adopted SIS model. How starter and bridge degrees can be computed for a given user is illustrated in Section 3. In Section 4, we show that our approach is not inherently limited by the chosen stereotypes, but can be extended to other possible ones leading to a sort of general approach. The related literature is analyzed in Section 5. Finally, in Section 6, we draw our conclusions.

2 The Underlying SIS Model

In this section, we illustrate the *SIS* model underlying our approach. It represents a set of concepts and relationships currently very common in social networks. It considers the following sets:

- the set *users* of the users of the *SIS*;
- the set *social_networks* of the social networks of the *SIS*;
- the set *resources* of the resources posted in the *SIS* by its users;
- the set *opinions* of the opinions posted in the *SIS* by its users;
- the set *tags* of the tags used by at least one user to label at least one resource or one opinion;
- the set *comments* of the comments posted by users and referring to a resource or an opinion;

In addition to these sets, which represent concepts intrinsic in social networks, our model considers a further set which plays a key role. In particular, it considers the set of tags representing the information entity to spread. We will use the term *context* (of that information entity) to represent this set of tags. The relationships considered in our model often represent actions performed by users. Currently, our model considers the following relationships:

- *membership* $_{u_i, s_k}$; it indicates that the user u_i joined the social network s_k .
- *resource_posting* $_{u_i, r_j, s_k}$ (resp., *opinion_posting* $_{u_i, o_j, s_k}$); it denotes that u_i posted the resource r_j (resp., the opinion o_j) in s_k .

- $resource_tagging_{u_i, r_j, t_h, s_k}$ (resp., $opinion_tagging_{u_i, o_j, t_h, s_k}$); it indicates that u_i specified the tag t_h as one of the tags in the label of r_j (resp., o_j) in s_k .
- $resource_accessing_{u_i, r_j, s_k}$ (resp., $opinion_accessing_{u_i, o_j, s_k}$); it indicates that u_i accessed r_j (resp., o_j) in s_k .
- $resource_commenting_{u_i, r_j, c_x, s_k}$ (resp., $opinion_commenting_{u_i, o_j, c_x, s_k}$); it denotes that u_i submitted the comment c_x for r_j (resp., o_j) in s_k .
- $friendship_{u_i, u_l, s_k}$; it indicates that u_i and u_l declared their friendship in s_k .

Observe that opinions and resources could be considered of the same nature (for instance, an opinion could be considered as a textual resource expressing some ideas of the user posting it). Moreover, they are characterized by exactly the same relationships. For this reason, as pointed out in the introduction, we will use the term *information_entity* to represent both of them. Starting from the sets and the relationships introduced above, our model defines the following *derived sets*, which will be exploited in the detection of starters and bridges.

- pr_{u_i} ; it represents the profile of the user u_i ; it consists of the set of the tags mostly used by her in her past activities.
- pr_{e_j} ; it represents the profile of the information entity e_j ; it consists of a set of tags indicating the content of e_j (in case e_j is a resource) or the subjects of e_j (in case e_j is an opinion).
- pr_{cnt_z} ; it represents the profile of a context cnt_z , i.e. the set of tags representing the corresponding information to spread.
- pr_{s_k} ; it represents the profile of the social network s_k ; this profile consists of the set of the tags mostly used therein.
- $users_{s_k}$; it represents the set of the users of s_k .
- $max_friends_{s_k}$; it represents the maximum number of friends of a user in s_k .
- $social_networks_{u_i}$; it represents the set of the social networks joined by u_i .
- $posted_entities_{u_i}$; it represents the set of information entities posted by u_i .
- $friends_{u_i}$; it represents the set of the users who declared their friendship with u_i in one or more social networks.

3 Starter and Bridge Detection

In this section, we show how the starter and bridge degrees can be computed for a given user.

3.1 Starter Detection

In our application scenario a starter can be defined as an individual who, over a significant period of time, generates information entities (i.e., resources and/or opinions) that other individuals access and comment. Preliminarily, we recall from the introduction that, given a user, her attitude to act as a starter depends on the *context* of the information entity to spread; we have seen that the *context* of an information entity consists of the set of its topics. The following considerations help us to find an identikit of a starter:

1. A starter should have been recently active, i.e., she should have recently posted many information entities.
2. Information entities posted by a starter should have been frequently accessed and commented.
3. Information entities posted by a starter should have attracted the interest of many other users.
4. Users who accessed the information entities posted by a starter should have heterogeneous interests and needs. This condition does not exclude the possibility that a starter is monothematic. However, in this case, she could attract the interest of only a portion of users, i.e. the ones interested in the corresponding topic. By contrast, if a starter posts information entities about different topics, then she could attract the interest of a larger portion of users and, therefore, she could have a larger number of followers.

In order to “quantify” these considerations we must introduce the following support sets; these are derived from the basic ones defined in Section 2.

- $posted_entities_{u_i, cnt_z}$ represents the set of information entities of $posted_entities_{u_i}$ which refer to the context cnt_z . This set can be defined as follows:

$$posted_entities_{u_i, cnt_z} = \{e_j \in posted_entities_{u_i} | J(pr_{e_j}, pr_{cnt_z}) > th_c\}$$

Here, $J(A, B)$ represents the Jaccard coefficient of the sets A and B , whereas th_c is a suitable threshold. We recall that the Jaccard Coefficient $J(A, B)$ between two sets A and B is defined as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

- $posted_entities_{u_i, cnt_z, T}$ represents the set of information entities of $posted_entities_{u_i, cnt_z}$ posted in the last T days.
- $accesses_{u_i, cnt_z, T}$ represents the set of accesses, performed by the *SIS* users, to the information entities of $posted_entities_{u_i, cnt_z, T}$.
- $comments_{u_i, cnt_z, T}$ represents the set of comments, performed by the *SIS* users, to the information entities of $posted_entities_{u_i, cnt_z, T}$.
- $posted_sn_{u_i, cnt_z, T}$ represents the set of social networks where u_i submitted at least one information entity of $posted_entities_{u_i, cnt_z, T}$.
- $accessing_users_{u_i, cnt_z, T}$ represents the set of users who accessed at least one information entity of $posted_entities_{u_i, cnt_z, T}$.
- $commenting_users_{u_i, cnt_z, T}$ represents the set of users who submitted a comment for at least one information entity of $posted_entities_{u_i, cnt_z, T}$.

Starting from these sets we can introduce the following derived metrics, each expressing a contribution in the measuring of the starter degree of u_i :

$$1. \text{ entities_std}_{u_i, cnt_z} = \alpha \frac{|posted_entities_{u_i, cnt_z, T}|}{|posted_entities_{u_i, cnt_z, \beta T}|} + (1 - \alpha) \frac{|posted_entities_{u_i, cnt_z, T}|}{\max_{u_l \in \{u_i\} \cup friends_{u_i}} (|posted_entities_{u_l, cnt_z, T}|)}$$

This parameter is an indicator of the tendency of u_i to post information entities stimulating discussions. The first factor is an indicator of the trend of the posting frequency of u_i ; the second term is an indicator of the posting

activity of u_i w.r.t. the one of the other users in contact with her. The parameter α , belonging to the real interval $[0, 1]$ allows the tuning of the contribution of the two factors. The parameter β is a positive integer higher than or equal to 2 allowing the tuning of the time period to be considered in the computation of the trend of the posting frequency of u_i .

2. $accesses_std_{u_i, cnt_z}$ and $comments_std_{u_i, cnt_z}$ are computed in a way analogous to $entities_std_{u_i, cnt_z}$ except that, in the formula, the sets $posted_entities$ are substituted by the sets $accesses$ and $comments$.

3. $acc_users_std_{u_i, cnt_z} = \frac{|accessing_users_{u_i, cnt_z, T}|}{|\bigcup_{s_n \in posted_sn_{u_i, cnt_z, T}} users_{s_n}|}$ and

$$comm_users_std_{u_i, cnt_z} = \frac{|commenting_users_{u_i, cnt_z, T}|}{|\bigcup_{s_n \in posted_sn_{u_i, cnt_z, T}} users_{s_n}|}.$$

These parameters represent an indicator of the fraction of the users reached by the information entities about a context cnt_z posted by u_i who really accessed and commented these last information entities.

4. $het_acc_users_std_{u_i, cnt_z} =$

$$1 - \frac{\sum_{(u_l, u_m) \in accessing_users_{u_i, cnt_z, T} \wedge (u_m \neq u_l)} J(pr_l, pr_m)}{|\accessing_users_{u_i, cnt_z, T}| \cdot (|\accessing_users_{u_i, cnt_z, T}| - 1)}$$

$$and$$

$$het_comm_users_std_{u_i, cnt_z} = 1 - \frac{\sum_{(u_l, u_m) \in commenting_users_{u_i, cnt_z, T} \wedge (u_m \neq u_l)} J(pr_l, pr_m)}{|\commenting_users_{u_i, cnt_z, T}| \cdot (|\commenting_users_{u_i, cnt_z, T}| - 1)}.$$

These parameters are indicators of the heterogeneity of the users who accessed the information entities posted by u_i . The former (resp., the latter) is defined as an average on the Jaccard coefficients computed on all the possible pairs of $accessing_users_{u_i, cnt_z, T}$ (resp., $commenting_users_{u_i, cnt_z, T}$).

Observe that the values of all the metrics defined above can range in the real interval $[0, 1]$. We are now able to illustrate how the “starter degree” std_{u_i, cnt_z} of a user u_i in a context cnt_z can be computed. Specifically: $std_{u_i, cnt_z} = agg(entities_std_{u_i, cnt_z}, accesses_std_{u_i, cnt_z}, comments_std_{u_i, cnt_z}, acc_users_std_{u_i, cnt_z}, het_acc_users_std_{u_i, cnt_z}, comm_users_std_{u_i, cnt_z}, het_comm_users_std_{u_i, cnt_z})$. Here, agg is a suitable aggregation operator which returns a value in the real interval $[0, 1]$. For instance, it could represent a weighted mean of the involved parameters.

3.2 Bridge Detection

In our application scenario, a bridge can be defined as an individual who has accounts in more social networks and stimulates the exchange of information entities among these last ones. Analogously to the starter case, the attitude of a user to act a bridge depends on the context of the information entity to spread. Starting from this definition it is possible to find an identikit of a bridge:

- Given a user, the higher the number of social networks she joins and the higher her potential bridge degree.
- Given a user, the more central her friends in the corresponding social networks, the higher her possibility to disseminate information entities and, ultimately, the higher her potential bridge degree.

In order to “quantify” these considerations we must introduce the following support sets and measures derived from the basic ones illustrated in Section 2:

- *context_experts_{s_k,cnt_z}* represents the set of the users of a social network s_k expert in the context cnt_z associated with an information entity. This set can be defined as follows:

$$context_experts_{s_k, cnt_z} = \{u_l \in users_{s_k} \mid J(pr_{u_l}, pr_{cnt_z}) > th'_c\}$$

Here th'_c is a suitable threshold.

- *expert_friends_{u_i,cnt_z,s_k}* represents the set of the users of s_k expert in cnt_z who declared their friendship with u_i .
- *centrality_degree_{u_i,cnt_z,s_k}* represents the centrality degree of u_i in s_k as far as cnt_z is concerned. In this computation we consider the relationships between u_i and her friends who are expert in cnt_z . This degree is computed by means of a modified version of the PageRank algorithm [6]. Its formula is the following:

$$\delta + (1 + \delta) \cdot \left(\sum_{u_l \in expert_friends_{u_i, cnt_z, s_k}} \frac{centrality_degree_{u_l, cnt_z, s_k}}{|expert_friends_{u_l, cnt_z, s_k}|} \right)$$

Here δ is the so called “dumping factor” introduced in the definition of PageRank. It is often set to 0.75. Observe that *centrality_degree_{u_i,cnt_z,s_k}* ranges in the real interval $[0, +\infty)$. In order to perform its normalization we can divide it by the maximum centrality degree of a user expert of cnt_z in s_k . The formula is the following:

$$norm_centrality_degree_{u_i, cnt_z, s_k} = \frac{centrality_degree_{u_i, cnt_z, s_k}}{\max_{u_l \in users_{s_k}} (centrality_degree_{u_l, cnt_z, s_k})}$$

We are now able to illustrate how the “bridge degree” bd_{u_i, cnt_z} of u_i in cnt_z can be computed. Specifically:

$$bd_{u_i, cnt_z} = \frac{\sum_{s_k \in social_networks_{u_i}} |users_{s_k}| \cdot norm_centrality_degree_{u_i, cnt_z, s_k}}{\sum_{s_k \in social_networks} |users_{s_k}|}$$

The rationale underlying this formula is the following: the higher the number and the dimension of social networks joined by u_i , the higher her bridge degree.

4 Extending Our Approach to Other Stereotypes

As pointed out in Section 5, starters and bridges can be considered as special cases of stereotypes in a SIS. Stereotypes allow the categorization of people to groups; a “general idea” or a “label” can be associated with each group. Other possible stereotypes in a SIS could be the “power user”, the “spammer”, and so on. By generalizing the idea presented in this paper, it would be possible to define a set of stereotypes of interest. Analogously to starters and bridges, also these new stereotypes should be related to the reference context. By considering

all the stereotypes it could be possible to construct a stereotypical map for each user. Specifically, given a user u_i , her stereotypical map $st_map_{u_i}$ consists of a matrix having a row for each context and a column for each stereotype considered in the SIS. The generic element $st_map_{u_i}[z, w]$ is a number in the real interval $[0, 1]$ and indicates how much the personal traits of u_i are compliant with the features of the stereotype st_w as far as the topics represented by the context cnt_z are concerned. The knowledge of these maps could be beneficial for several applications; among them we cite:

- *Enrichment of user profiles based on user behaviors.* Generally, the classic profile of a user stores her interests and needs (usually represented by means of some tags); often, it can also register the actions performed by her, as well as her relationships. A user stereotypical map could enrich a classical user profile by adding new information, generally not considered in it. This information appears extremely useful to foresee the possible behavior of the corresponding user. However, the exploitation of tags in classical user profiles presents some problems. One of them is the possible presence of semantic anomalies (e.g., synonymies, homonymies, polysemies) among tags [16]. A second problem is the power law distribution of tags [12,17]; this last problem could affect not only tags but also user actions and relationships (think, for instance, to a power law distribution of friends). Owing to these problems, it is very difficult to classify or characterize a set of users in such a way as to construct homogeneous groups of them. The exploitation of user stereotypical maps is a solution to this problem. In fact, the map entries are identical for all users because they correspond to the stereotypes considered in the SIS. As a consequence, users can be compared and categorized on the basis of the same features.
- *Computation of trust and reputation of users.* Trust and reputation measure the reliability of a user in a community. Trust is a subjective measure; indeed, the trust of a user u_i in a user u_l indicates how much u_l is considered reliable by u_i . Reputation is an objective measure since it indicates how much a user u_i is considered reliable by a community. Stereotypes represent a powerful support in the computation of trust and reputation. In fact, a coefficient in the real interval $[0, 1]$ could be associated with each stereotype. It expresses the “goodness” of this last one; the higher its value, the higher the “goodness” of the corresponding stereotype. Once “goodness coefficients” have been associated with all stereotypes of the SIS, the trust and the reputation of a user can be easily computed by taking both her stereotypical degrees (specified in her stereotypical map) and “goodness coefficients” into consideration.
- *Team Building.* It is well known that a team wholly composed by people having the same psychological and behavioral features is often characterized by negative dynamics. For instance, in a company, a team consisting of only technically talented experts is often characterized by a negative form of competition because its components tend to assert themselves by nature. This fact creates stress among them, lowers the quality of their interactions

and, ultimately, deters them to achieve the goals for which the team was built [28]. On the contrary, if a team is harmonious, composed by people having “orthogonal” psychological and behavioral traits, it is possible to expect that it will better accomplish assigned goals. Clearly, the knowledge of the stereotypical map of a user gives important information about her psychological and behavioral traits and, consequently, can become a precious support when a team must be built.

As a further development of the idea expressed in this paper, it could be possible, given a context cnt_z , to construct a stereotypical map $sis_st_map_{cnt_z}$ of a SIS. This could consist of a set of graphs: $sis_st_map_{cnt_z} = \{sis_reg_{cnt_z}^1, \dots, sis_reg_{cnt_z}^y, \dots, sis_reg_{cnt_z}^m\}$. Each graph represents a region of the map and is associated with one or more reference stereotypes. Clearly, graphs could partially overlap. The knowledge of the stereotypical map of a SIS could be useful for several applications. Among them we cite:

- *“Cold Start” Problem.* This is one of the main problems in social sites. It concerns the difficulty to involve a new user in the activities performed by the other ones in a site. For this purpose, generally, a social site provides a new user with a set of suggestions of users, resources, etc. Clearly, these suggestions could be as much as possible of interest to her. For this reason, many sites require a new user to fill a questionnaire when she joins them. Generally, these questionnaires have been conceived to know the user’s main interests and needs, whereas a little emphasis is given to her behavior also because, generally, there is no way to fruitfully exploit this information. However, we argue that this knowledge could be very useful if there exists a way to exploit it, and, in our opinion, stereotypes (which just concern the behavioral traits of a user) can help in this last process. In this case the questionnaire should also have a part devoted to know the main stereotypical traits of the user. These last ones could complement the knowledge about her interests and needs and, consequently, could improve the suggestions of users and resources. For instance, given a context, if the stereotypical traits of a user are similar to the ones of a given region of the SIS, the other members of the region could be suggested to her. Analogously, it could be possible to suggest those resources exploited by the users whose stereotypical traits are the most similar to the ones of the user.
- *Information search support.* Assume that a user is searching for information of her interest or that she wants to submit a query referring to a given context. If she does not preliminarily select the potential targets she could waste a lot of time and could be overwhelmed by useless information and/or answers. A SIS stereotypical map could supply her a useful support to face this issue. As a matter of fact, having this tool at disposal, she could search for information or submit queries in those regions of the SIS characterized by positive stereotypes (such as power users) and could avoid regions characterized by negative stereotypes (such as spammers) for the contexts of her interest.

5 Related Literature

The idea of labeling users as starters or bridges is a particular case of the most general activity of stereotyping user behavior. The concept of stereotype, in its modern psychology meaning, was originally proposed in sociology and psychology [31,24]. In the context of Computer Science, stereotypes have been used in the definition of models to represent groups of users [18,45], in e-commerce [2], in the development of techniques to hasten the learning process of robots [38,34], and in several other application fields [37,8,26]. However, the use of stereotypes in online social networks, and, more in general, in online social communities, received little attention in the past [30]. In [38], an approach for the creation and the exploitation of stereotyped partner models to speed up the process of learning about a robot's interactive partner is investigated. In [18], the authors analyze some possible improvements in agent modeling using *re-evaluative stereotyping with switching*. The use of stereotypes has been proposed to create models of individual users [34] and to reduce the latency problem in collaborative filtering recommender systems [37]. A formal evaluation method to test the accuracy and/or the homogeneity of stereotypes derived from the explicit characteristics of users is proposed in [45].

Starter and bridge detection could be used to tackle the problem of information propagation. This problem has been first addressed in the telecommunication networks research area [15,25,32]. It was also shown that the dynamics of the information flow in a network strongly resembles the epidemic spread in a population [3,4]. However, the kind of investigation on information spread dealt with in our scenario is very different from the one considered in telecommunication networks and epidemic spread. As a matter of fact, telecommunication researchers investigate issues like the role of mobile devices, the physical features of exploited networks, and so forth; furthermore, typically, their goals are energy saving or network design. Instead, we focus on the problem of information propagation in social networks. In this context, several approaches have been proposed. In particular, in [23], a mechanism which uses gossip algorithms [5] for information dissemination on social networks is presented. In order to manage traffic and to solve possible bottleneck problems, this mechanism uses two strategies: the former spreads rumors inside the social network and finds the network of interests; the latter collects messages in the network of interests with consideration of a threshold of independent paths. Information propagation problem has been studied for both single-piece and multi-piece information spreading [36]. Interesting results regarding how information is propagated over real-life environments centered on social networks have been presented in [22,27,13,7]. Differently from the above proposals, in order to support information spread, our paper aims at identifying starters and bridges.

The problem of finding starters is faced in [1], where the authors propose three heuristic algorithms for finding such users among the communities of a social network. An approach devoted to identify the influence and the roles of nodes of a social network on the basis of the structure of the corresponding communities is presented in [46]. Differently from our proposal, these two approaches are solely

based on the link structure of the social network and do not take the important parameter of user activity into consideration. In [9] the authors propose a model to mine the top-k influential bloggers on the basis of their interest domains and their links. The proposed mining technique leverages on two main contributions, namely Accumulated Posts and General Links. The former measures the influence of a blogger's posts, the latter measures the reachability of a blogger's page inside the Web on the basis of its external links. In [39] the authors propose an approach to the mining of the top-k influential nodes in a mobile social network (i.e., a social network inferred by mobile call logs). This approach first detects communities in a mobile social network on the basis of information diffusion and, then, uses a dynamic programming algorithm to select the best communities to consider for inferring influential nodes. An approach for the evaluation of the influential strength of a blogger and for the identification of the most influential bloggers in the blogosphere is proposed in [29]. In this paper the authors propose an MIV (Marketing Influential Value) based model which provides tools to measure the capability of a blog to influence marketing. As for a comparison between our approach and the ones described in [9,39,29], we observe that these last three approaches are not specific for social networks and, thus, they handle and exploit information typically not provided in our context.

Most of the approaches for bridge identification [33] proposed in the past aim at finding bridges among different communities of a single social network. In particular, methodologies for the detection of node categories (such as highly connected nodes or nodes belonging to densely connected subgroups) have been originally proposed in Social Network Analysis [10]. They can be exploited to assess if a node can assume a specific role (e.g., bridge, starter, power user, etc.) in a single social network on the basis of its position and of the network structure. All these approaches are strictly based on centrality measures (such as degree, closeness, eigenvector, and betweenness [19]) which are derived from the examination of the structural connections characterizing the network. All of them can be adapted to behave as bridge detection approaches, if the considered role is bridge. As a matter of fact, some works perform this adaptation in several application fields (e.g., biology, communication theory, information science, marketing, epidemiology, telecommunications) [44,20,11,42] as a *side effect* during the solution of specific problems. In these approaches bridge detection is performed only on the basis of the network structure.

Only few approaches capable of explicitly performing bridge detection have been proposed. In particular, the authors of [43] present SCAN, a Structural Clustering Algorithm for Networks. SCAN aims at clustering the nodes of a network on the basis of the analysis of their neighborhoods. Once clusters have been defined, it determines which nodes belong to a cluster, which ones can be regarded as bridges and which ones are outliers. In [14], the authors focus on several kinds of user nodes in a network which play a key role in the interactions among the network components. Their approach considers the network topology as well as the information registered on the links between user nodes. It strongly relies on the concept of network transitivity [40], which represents a common

property in most networks. In [35] the authors define several roles which can be assumed by a user node in a network. They show how these roles can support existing link mining techniques and propose a technique for their detection, based on the network topology and on the knowledge of the number of communities each node is related to. We observe that the techniques proposed in [43] and [35] do not take the activity of users into account, thus missing an important parameter that, instead, has a great weight in the detection of bridges. All the proposals for bridge detection described above have not been conceived for a Social Internetworking Scenario. On the other hand, this scenario cannot be considered as a trivial union of several social networks, as widely remarked in the introduction. In other words, these techniques are not oriented to facilitate information cross-spreading, which is crucial in the context of social internetworking. From this point of view, our work includes a strong element of originality, since it poses cross-spreading as the key concept which drives the proposed approach. We feel that both this originality and the evident difficulty of testing our technique in real-life social networks in reasonable time, allow the validation of our approach to be limited to the above comparison with the literature as well as to all the reasonings included in the work and showing that it can be expected that our stereotypes are really able to improve information spread. In other words, being this work a first idea, whose significance is, from our point of view, well supported by both a number of solid argumentations, a net originality, and a convincing theoretical model, we postpone the hard task of involving real-life social networks for the experimental validation to our future work.

6 Conclusion

In this paper we present an approach devoted to support information spread in a SIS. First we have seen that the solutions to this problem presented in the past for single social networks cannot be directly applied to a SIS, due to the peculiarities of this scenario. Then, we have proposed a new solution centered on the presence of two stereotypes, namely starters and bridges, which are specific in the new context of SISs. Finally, we have seen that the knowledge of these and other stereotypes can be beneficial in many application contexts. As for future work, a possible development of our research efforts in this field could be the definition of approaches that, given an information entity, are capable of identifying the users of the SIS who are the most interested in it. After this, they could find the best paths in the SIS to reach these users. Furthermore, it could be challenging to investigate the derivation of other stereotypes (such as power users and spammers) in a SIS. In particular, it could be interesting to verify if and how techniques already proposed for solving this problem in a single social network can be extended to SISs, as well as to define new techniques specific for this scenario which highly benefit from its peculiarities.

Acknowledgement. The Authors thank the anonymous Referees whose suggestions allowed them to highly improve the quality of this paper.

References

1. Anjerani, M., Moeini, A.: Selecting influential nodes for detected communities in real-world social networks. In: Proc. of the Iranian Conference on Electrical Engineering, ICEE 2011, Tehran, Iran, pp. 1–6 (2011)
2. Ardissono, L., Goy, A., Petrone, G., Segnan, M., Console, L., Lesmo, L., Simone, C., Torasso, P.: Agent technologies for the development of adaptive web stores. In: Sierra, C., Dignum, F.P.M. (eds.) AgentLink 2000. LNCS (LNAI), vol. 1991, pp. 194–213. Springer, Heidelberg (2001)
3. Barthelemy, M., Barrat, A., Pastor-Satorras, R., Vespignani, A.: Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Physical Review Letters* 92, 178701–178704 (2004)
4. Boguna, M., Pastor-Satorras, R.: Epidemic spreading in correlated complex networks. *Physical Review E* 66, 047104–047107 (2002)
5. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Gossip algorithms: Design, analysis and applications. In: Proc. of the International Joint Conference of the IEEE Computer and Communications Societies, INFOCOMM 2005, Miami, FL, USA, vol. 3, pp. 1653–1664 (2005)
6. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* 30(1-7), 107–117 (1998)
7. Buccafurri, F., Lax, G.: Improving Similarity-Based Methods for Information Propagation on Social Networks. *Networked Digital Technologies* 87(3), 391–401 (2010)
8. Burnett, C., Norman, T.J., Sycara, K.: Bootstrapping trust evaluations through stereotypes. In: Proc. of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, Toronto, Ontario, Canada, pp. 241–248. International Foundation for Autonomous Agents and Multiagent Systems (2010)
9. Caiv, Y., Chen, Y.: Mining influential bloggers: From general to domain specific. In: Velásquez, J.D., Ríos, S.A., Howlett, R.J., Jain, L.C. (eds.) KES 2009, Part II. LNCS, vol. 5712, pp. 447–454. Springer, Heidelberg (2009)
10. Carrington, P., Scott, J., Wasserman, S.: Models and Methods in Social Network Analysis. Cambridge University Press (2005)
11. Catanzaro, M., Caldarelli, G., Pietronero, L.: Assortative model for social networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 70(3), 037101–037104 (2004)
12. Cattuto, C., Schmitz, C., Baldassarri, A., Servedio, V.D.P., Loreto, V., Hotho, A., Grahl, M., Stumme, G.: Network properties of folksonomies. *Artificial Intelligence Communications* 20(4), 245–262 (2007)
13. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the flickr social network. In: Proc. of the International Conference on World Wide Web, WWW 2009, Madrid, Spain, pp. 721–730. ACM (2009)
14. Chou, B.-H., Suzuki, E.: Discovering Community-Oriented Roles of Nodes in a Social Network. In: Bach Pedersen, T., Mohania, M.K., Tjoa, A.M. (eds.) DAWAK 2010. LNCS, vol. 6263, pp. 52–64. Springer, Heidelberg (2010)
15. Datta, A., Quarteroni, S., Aberer, K.: Autonomous Gossiping: A Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Wireless Mobile Ad-Hoc Networks. In: Bouzeghoub, M., Goble, C.A., Kashyap, V., Spaccapietra, S. (eds.) ICSNW 2004. LNCS, vol. 3226, pp. 126–143. Springer, Heidelberg (2004)
16. De Meo, P., Nocera, A., Terracina, G., Ursino, D.: Recommendation of similar users, resources and social networks in a Social Internetworking Scenario. *Information Sciences* 181(7), 1285–1305 (2011)

17. De Meo, P., Quattrone, G., Ursino, D.: Exploitation of semantic relationships and hierarchical data structures to support a user in his annotation and browsing activities in folksonomies. *Information Systems* 34(6), 511–535 (2009)
18. Denzinger, J., Hamdan, J.: Improving Modeling of Other Agents using Tentative Stereotypes and Compactification of Observations. In: *Proc. of the International Conference on Intelligent Agent Technology, IAT 2004, Beijing, China*, pp. 106–112. IEEE Computer Society (2004)
19. Freeman, L.C.: A set of measuring centrality based on betweenness. *Sociometry* 40(1), 35–41 (1977)
20. Goldenberg, J., Han, S., Lehmann, D.R., Hong, J.W.: The Role of Hubs in the Adoption Process. *Journal of Marketing* 73, 1–13 (2009)
21. Goldenberg, J., Libai, E., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters* 12(3), 211–223 (2001)
22. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. In: *Proc. of the International Conference on World Wide Web, WWW 2004*, pp. 491–501. ACM, New York (2004)
23. Hamed, A., Arman, M., Ehsan, A.: Towards an Efficient Method for Spreading Information in Social Network. In: *Proc. of the Asia International Conference on Modelling & Simulation, Bandung, Bali, Indonesia*, pp. 152–157. IEEE Computer Society (2009)
24. Hamilton, D.L., Troler, T.K.: Stereotypes and stereotyping: An overview of the cognitive approach. In: Dovidio, J.F., Gaertner, S.L. (eds.) *Prejudice, Discrimination, and Racism*, pp. 127–163. Academic Press, US (1986)
25. Khelil, A., Becker, C., Tian, J., Rothermel, K.: An epidemic model for information diffusion in MANETs. In: *Proc. of the International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM 2002, Atlanta, GA, USA*, pp. 54–60. ACM (2002)
26. Kuflik, T., Shapira, B., Shoval, P.: Stereotype-based versus personal-based filtering rules in information filtering systems. *Journal of the American Society for Information Science and Technology* 54(3), 243–250 (2003)
27. Kumar, R., Novak, J., Raghavan, P., Tomkins, A.: On the bursty evolution of blogspace. *World Wide Web* 8(2), 159–178 (2005)
28. LaFasto, F.M.J., Larson, C.: *When Teams Work Best*. Sage Publications, Inc. (2001)
29. Li, Y.M., Lai, C.Y., Chen, C.W.: Discovering influencers for marketing in the blogosphere. *Information Sciences* 181(23), 5143–5157 (2011)
30. Lin, F., Chen, C., Tsai, K.: Discovering Group Interaction Patterns in a Teachers Professional Community. In: *Proc. of the Annual Hawaii International Conference on System Sciences, HICSS 2003, Big Island, Hawaii, USA*, p. 116. IEEE Computer Society (2003)
31. Lippmann, W.: *Public Opinion*. Macmillan (1922)
32. Monclar, R., Tecla, A., Oliveira, J., de Souza, J.M.: MEK: Using spatial-temporal information to improve social networks and knowledge dissemination. *Information Sciences* 179(15), 2524–2537 (2009)
33. Nocera, A., Ursino, D.: PHIS: a system for scouting potential hubs and for favoring their “growth” in a Social Internetworking Scenario. *Knowledge-Based Systems* (forthcoming)
34. Rich, E.: User modeling via stereotypes. In: Maybury, M.T., Wahlster, W. (eds.) *Readings in Intelligent User Interfaces*, pp. 329–342. Morgan Kaufmann Publishers Inc., San Francisco (1998)

35. Scripps, J., Tan, P.N., Esfahanian, A.H.: Node Roles and Community Structure in Networks. In: Proc. of the International Workshop on Knowledge Discovery on the Web and on Social Network Analysis, WebKDD/SNA-KDD 2007, San Jose, CA, USA, pp. 26–35. ACM (2007)
36. Shah, D.: Gossip Algorithms. *Foundations and Trends® in Networking* 3(1), 1–125 (2008)
37. Sollenborn, M., Funk, P.: Category-Based Filtering and User Stereotype Cases to Reduce the Latency Problem in Recommender Systems. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 395–420. Springer, Heidelberg (2002)
38. Wagner, A.R.: Using stereotypes to understand one’s interactive partner. In: Proc. of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, Toronto, Ontario, Canada, pp. 1445–1446. International Foundation for Autonomous Agents and Multiagent Systems (2010)
39. Wang, Y., Cong, G., Song, G., Xie, K.: Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, pp. 1039–1048. ACM (2010)
40. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* 393(6684), 440–442 (1998)
41. Wong, Y.H., Chan, R.Y.K., Leung, T.K.P.: Managing information diffusion in Internet marketing. *European Journal of Marketing* 39(7/8), 926–946 (2005)
42. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: Proc. of the International Conference on Hypertext and Hypermedia, Hypertext 2006, Odense, Denmark, pp. 111–114. ACM (2006)
43. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: SCAN: a structural clustering algorithm for networks. In: Proc. of the International Conference on Knowledge Discovery and Data Mining, KDD 2007, San Jose, CA, USA, pp. 824–833. ACM (2007)
44. Yoneki, E., Hui, P., Crowcroft, J.: Distinct Types of Hubs in Human Dynamic Networks. In: Proc. of the International Workshop on Social Network Systems, SNS 2008, Glasgow, Scotland, UK, pp. 7–12. ACM (2008)
45. Zhang, X., Han, H.: An empirical testing of user stereotypes of information retrieval systems. *Journal of Information Processing and Management* 41(3), 651–664 (2005)
46. Zhu, T., Wu, B., Wang, B.: Social Influence and Role Analysis Based on Community Structure in Social Network. In: Huang, R., Yang, Q., Pei, J., Gama, J., Meng, X., Li, X. (eds.) ADMA 2009. LNCS, vol. 5678, pp. 788–795. Springer, Heidelberg (2009)

Context-Aware Predictions on Business Processes: An Ensemble-Based Solution

Francesco Folino, Massimo Guarascio, and Luigi Pontieri

Institute for High Performance Computing and Networking (ICAR)
National Research Council of Italy (CNR)
Via Pietro Bucci 41C, 87036 Rende (CS), Italy
{ffolino,guarascio,pontieri}@icar.cnr.it

Abstract. The discovery of predictive models for process performances is an emerging topic, which poses a series of difficulties when considering complex and flexible processes, whose behaviour tend to change over time depending on context factors. We try to face such a situation by proposing a predictive-clustering approach, where different context-related execution scenarios are equipped with separate prediction models. Recent methods for the discovery of both Predictive Clustering Trees and state-aware process performance predictors can be reused in the approach, provided that the input log is preliminary converted into a suitable propositional form, based on the identification of an optimal subset of features for log traces. In order to make the approach more robust and parameter free, we also introduce an ensemble-based clustering method, where multiple PCTs are learnt (using different, randomly selected, subsets of features), and integrated into an overall model. Several tests on real-life logs confirmed the validity of the approach.

Keywords: Process Mining, Clustering, Prediction, Ensemble Learning.

1 Introduction

Process mining techniques [2] are a valuable tool for the analysis of business processes, which can extract useful information out of historical process logs and provide the analyst with a high-level process model. While traditional approaches focused on the discovery of control-flow models describing how process activities were executed in the past, increasing attention is being gained by the discovery of predictive process models, capable to offer operational support at run-time. In particular, an emerging research stream [11,3] concerns the induction of models for forecasting performances metrics on new process instances. In particular, in [3], an annotated finite-state model is induced from a given log, where the states correspond to abstract representations of process traces. Conversely, a non-parametric regression model is used in [11] to build the prediction for a new (possibly partial) trace upon its similarity to historical ones, where the similarity between two traces bases on comparing their respective abstract views.

However, it is not easy to make accurate forecasts for fine-grain measures (like, e.g., processing times), especially when the analyzed process exhibits complex and flexible dynamics, and its execution schemes and performances change over time, depending on the context. In general, the precision of induced workflow models can be increased by exploiting ad-hoc clustering methods [16], while regarding each resulting cluster as evidence for a peculiar execution scenario. However, few efforts have been spent so far to improve the accuracy of performance predictors through trace clustering [14].

In this paper, we describe a general predictive-clustering computation scheme, meant to detect different context-related execution scenarios (or *process variants*), and to equip each of them with a process performance model. The approach exploits and integrates different kinds of techniques (concerning data transformation and selection, predictive clustering of propositional data, process-performance model induction), yet being parametric w.r.t. each of them, and yields a clustering-based performance prediction model, which can effectively support forecasts over unfinished process instances. A first possible implementation of this approach, proposed in [14], consists in exploiting a PCT learning algorithm [6] for the discovery of a predictive clustering model, and the method in [3] for deriving a performance prediction model out of each discovered cluster. Owing to both scalability and effectiveness reasons, prior to applying the former learning method, the input log is converted into a propositional form, where the performance values associated with each trace are encoded as numeric target features with the help of state-abstraction techniques features. Since selecting a good subset of such features is crucial to obtaining an accurate clustering model, we also propose to solve the clustering task via an alternative ensemble-based method, where multiple PCTs are learnt, by using different randomly-selected subsets of these features, and eventually combined into a single clustering model.

After introducing some preliminary concepts, in Section 2, we describe the general approach, in Section 3, and the two concrete instantiations of it, in Section 4. Some major experimental findings are discussed in Section 5, before providing the reader with a few concluding remarks.

2 Preliminaries and Formal Framework

Log data and Performance Measures. As usual, we assume that for each process instance (a.k.a. “case”) a *trace* is recorded, encoding the sequence of *events* happened during its enactment. Let \mathcal{T} be a reference universe of all (possibly partial) traces that may appear in a log. For any *trace* $\tau \in \mathcal{T}$, $len(\tau)$ is the number of events recorded in τ ; moreover, for $i = 1 \dots len(\tau)$, $\tau[i]$ denotes the i -th event of τ , while $task(\tau[i])$ and $time(\tau[i])$ are the task and timestamp associated with $\tau[i]$, respectively. Two tuples are also defined for τ , characterizing its execution context: (i) (“intrinsic”) *data properties*, denoted by $data(\tau)$, and (ii) (“extrinsic”) *environmental features*, denoted by $env(\tau)$, and capturing the state of the BPM system when τ started. For short, $context(\tau)$ denotes the juxtaposition of $data(\tau)$ and $env(\tau)$. Moreover, for $i = 0 \dots len(\tau)$, $\tau[i]$ is a *prefix*

sub-trace of τ , which contains the first i events of τ and the same context data. (i.e., $context(\tau[i]) = context(\tau)$).

A log L is a finite subset of \mathcal{T} , while the *prefix set* of L , denoted by $\mathcal{P}(L)$, is the set of all the prefixes of L 's traces, i.e., $\mathcal{P}(L) = \{\tau[i] \mid \tau \in L \text{ and } 0 \leq i \leq len(\tau)\}$.

Let $\hat{\mu} : \mathcal{T} \rightarrow \mathbb{R}$ be an (unknown) function assigning a performance value to any (possibly unfinished) process trace. For the sake of concreteness, we will hereinafter focus on the special case where the target performance value associated with each trace is the remaining process time (measured, e.g., in days, hours, or in finer grain units), i.e., the time needed to finish the corresponding process enactment. Moreover, we will assume that such a performance value is known for any prefix trace in $\mathcal{P}(L)$, for any given log L . Indeed, for any log trace τ , the (actual) remaining-time value of $\tau[i]$ is $\hat{\mu}(\tau[i]) = time(\tau[len(\tau)]) - time(\tau[i])$.

Abstraction-based Prediction. A (predictive) **Process Performance Model (PPM)** is a model that can predict the performance value (i.e., remaining time) of any process enactment, based on its associated (possibly incomplete) sequence of events. Such a model can be viewed as a function $\mu : \mathcal{T} \rightarrow \mathbb{R}$ estimating $\hat{\mu}$ all over the trace universe. Learning a PPM is then a special induction problem, where the training set is a log L , and the value $\hat{\mu}(\tau)$ of the target measure is known for each (sub-)trace $\tau \in \mathcal{P}(L)$. Recent approaches to this problem (e.g., [3,11]) share the idea of regarding traces at a higher level of abstraction – indeed, performances hardly depends on the particular sequence of events occurred, but rather on certain properties of them. Three families of such trace abstraction functions are defined next.

Definition 1 (Trace Abstraction Function). Let $h \in \mathbb{N}^+ \cup \{\infty\}$ be a threshold on past history. A *trace abstraction function* $abs_h^{mode} : \mathcal{T} \rightarrow \mathcal{R}$ is a function mapping each trace $\tau \in \mathcal{T}$ to an element $abs_h^{mode}(\tau)$ in a space \mathcal{R} of abstract representations.¹ For any $\tau \in \mathcal{T}$, let us denote $n = len(\tau)$ and $j = n - h + 1$ if $n > h$, and $j = 1$ otherwise. Then we define: **(i)** $abs_h^{list}(\tau) = \langle task(\tau[j]), \dots, task(\tau[n]) \rangle$, **(ii)** $abs_h^{bag}(\tau) = [(t, p) \mid t \in abs_h^{set}(\tau) \text{ and } p = |\{\tau[k] \mid j \leq k \leq n, task(\tau[k]) = t\}|]$, and **(iii)** $abs_h^{set}(\tau) = \{task(\tau[j]), \dots, task(\tau[n])\}$. \square

Example 1. We next introduce a running example, inspired to a real-life case study (also used for validating our approach) that concerns a transshipment process. Basically, for each container c passing through the harbor, a distinct log trace τ_c is stored, registering all the tasks applied to c , which may include the following ones: moving c by means of either a straddle-carrier (*MOV*) or a shore crane (*OUT*), and swapping c with another container (*SHF*). Several data attributes are available for τ_c , including physical properties (e.g., size, weight) of container c , its previous and next calls (namely *PrevHarbor* and *NextHarbor*), the navigation lines delivering/loading c (namely *NavLine_IN* and *NavLine_OUT*). A few additional features are computed for τ_c to characterize the state of the transshipment system at the time, say t_c , when c arrived at the harbor: the

¹ Each $\alpha \in \mathcal{R}$ is a high level representation of some traces, meant to capture some performance-relevant state of the process analyzed.

hour (*ArrivalHour*), day of the week (*ArrivalDay*) and month (*ArrivalMonth*) extracted from t_c , and a rough *Workload* indicator (counting how many containers were in the harbor at time t_c). Let τ be a log trace associated with a sequence $\langle e_1, e_2, e_3 \rangle$ of three events such that $task(e_1) = task(e_2) = MOV$ and $task(e_3) = OUT$. With regard to the abstraction functions in Def. 1, it is easy to see that for the prefix $\tau(2)$ (i.e., the partial enactment consisting of the first two events) it is: $abs_{\infty}^{list}(\tau(2)) = \langle MOV, MOV \rangle$, $abs_{\infty}^{bag}(\tau(2)) = [MOV^2]$, $abs_{\infty}^{set}(\tau(2)) = \{MOV\}$. Using instead the shortest possible threshold $h = 1$ on history horizon, all the abstractions $abs_1^{list}(\tau(2))$, $abs_1^{bag}(\tau(2))$ and $abs_1^{bag}(\tau(2))$ just consist of the sole element MOV , which is indeed the last task in $\tau(2)$. \triangleleft

Predictive Clustering. The core idea of *Predictive Clustering* approaches [6] is that, based on a suitable clustering model, predictions for new instances can be based on the cluster where they are estimated to belong. Two kinds of features are considered for any element z in the given instance space $Z = X \times Y$: *descriptive* features and *target* features (to be predicted), denoted by $descr(z) \in X$ and $targ(z) \in Y$, respectively. Then, a **Predictive Clustering Model (PCM)**, for a given training set $L \subseteq Z$, is a function $q : X \rightarrow Y$ of the form $q(x) = p(c(x), x)$, where $c : X \rightarrow \mathbb{N}$ is a partitioning function and $p : \mathbb{N} \times X \rightarrow Y$ is a (possibly multi-target) prediction function. Clearly, whenever there are more than one target features, q clearly encodes a multi-regression model. Several PCM learning methods have been proposed in the literature, which can work with general relational data [6], or with propositional data only (e.g., system CLUS [1]).

A novel specific sub-class of such models can be defined for log traces, annotated with both context and performance data. In fact, as we believe that process performances may depend on context factors, when trying to predict the performances of any (partial) trace τ , we regard its associated context data $context(\tau)$ as descriptive attributes.

Definition 2 (Context-Aware Performance Prediction Model (CA-PPM)).

Let L be a log (over \mathcal{T}), with context features $context(\mathcal{T})$, and $\hat{\mu} : \mathcal{T} \rightarrow \mathbb{R}$, be a performance measure, known for all $\tau \in \mathcal{P}(L)$. Then a *context-aware performance prediction model (CA-PPM)* for L is a pair $M = \langle c, \langle \mu_1, \dots, \mu_k \rangle \rangle$, with k denoting the number of different clusters found for L . Model M encodes the unknown performance function $\hat{\mu}$ in terms of a predictive clustering model μ^M , such that: (i) $c : context(\mathcal{T}) \rightarrow \{1, \dots, k\}$, (ii) $\mu_i : \mathcal{T} \rightarrow \mathbb{R}$, for $i \in \{1, \dots, k\}$, and (iii) $\mu^M(\tau) = \mu_j(\tau)$ with $j = c(context(\tau))$. \square

Our ultimate goal is to find a CA-PPM M such that, for any (possibly partial) trace τ , the associated performance value $\hat{\mu}(\tau)$ is well approximated by $\mu^M(\tau)$. Performance predictions will hence rely on a partitioning function c , assigning (possibly novel) process instances to trace clusters (based on their context data), and on multiple process performance predictors μ_i (one for each cluster). Such a model is a special kind of PPM model, relying on a predictive clustering one. As such, it can be built by combining a predictive clustering model and multiple simpler PPMs (as building blocks for implementing c and each μ_i , respectively), as discussed in the next section.

3 Solution Approach: Meta-algorithm CA-PPM Discovery

Seeking an explicit encoding for the hidden performance measure $\hat{\mu}$, based on a given log L , can be stated as the search for a CA-PPM (cf. Def. 2) minimizing some loss measure, like those in [6], possibly evaluated on an different sample $L' \subseteq \mathcal{T}$ than the one used for the training. However, in order to prevent long computations, a heuristics approach can be undertaken (like in [14]), where a CA-PPM is discovered by solving two subproblems: (P1) find a function c (locally) minimizing the loss on a propositional view of the input log, summarizing the correlation between execution patterns and performance values; and (P2) learn the predictors μ_i out of the clusters generated by c .

The main motivation for using a propositional view of the log is the belief that a direct application of classic predictive clustering methods to process logs is likely to yield poor scalability and accuracy results. In particular, our two-phase approach reduces the search space, as it avoids considering all possible log partitions, with all of their associated prediction functions.

In order to build such a log view, a set of target features must be defined for each trace τ — which is associated, indeed, with a sequence of time values (namely, $\hat{\mu}(\tau(1)), \hat{\mu}(\tau(2)), \dots, \hat{\mu}(\tau)$). Heuristically, each trace is mapped into a vector space, whose dimensions coincide with relevant states of the (hidden) process performance model. Such target features are defined by way of the trace abstraction functions in Def. 1, which try to transform, indeed, each trace into an abstract representation of its enactment state, based on its past history.

Specifically, given an abstraction function $abs : \mathcal{T} \rightarrow \mathcal{R}$, a “candidate” target feature corresponds to each abstract (state) representation $\alpha \in \mathcal{R}$, and the value $val(\tau, \alpha)$ of this feature for any trace τ is $val(\tau, \alpha) = agg(\langle \hat{\mu}(\tau(i_1)), \dots, \hat{\mu}(\tau(i_s)) \rangle)$, where $\{i_1, \dots, i_s\} = \{j \in \mathbb{Z} \mid 0 \leq j \leq len(\tau) \text{ and } abs(\tau(i_j)) = \alpha\}$, and $i_j < i_k$ for any $0 \leq j < k \leq s$, while agg is a function aggregating a sequence of measure values into a single one (e.g., the average, median, first, last in the sequence). In our tests, the last element of a sequence was always chosen as such an aggregate. As a special case, we assume that function agg returns a “null” value when provided with an empty list – i.e., $agg(\langle \rangle) = \text{NULL}$.

Definition 3 (Pivot Abstractions and Log Sketch). Let L be a log, $abs : \mathcal{T} \rightarrow \mathcal{R}$ be a trace abstraction function. Let $abs(L) \subseteq \mathcal{R}$ be the set of all the state abstractions that are generated by applying abs to L , and let $\Phi : \mathcal{R} \rightarrow \{\text{true}, \text{false}\}$ be a function encoding some given criterion for selecting state abstractions — a concrete instantiation of such a function (denoted by $\Phi[\phi, \sigma]$) is described in the next section. Then, any state abstraction $\alpha \in abs(L)$ is a *Pivot (state) Abstraction* for L , w.r.t. abs and Φ , if $\Phi(\alpha) = \text{true}$, i.e. if α is selected by Φ . $PA_{abs}^\Phi(L)$ will indicate the set of all pivot state abstractions for L w.r.t. abs and Φ . Moreover, given a set $\mathcal{A} = \{\alpha_{j1}, \dots, \alpha_{ju}\}$ of such pivot state abstractions, the *Performance Sketch* $PS_{\mathcal{A}}(L)$ of L w.r.t. \mathcal{A} is a (propositional) view of L such that: (i) each trace τ in L corresponds to a distinct data instance z_τ in $PS_{\mathcal{A}}(L)$, (ii) $context(\tau)$ are the descriptive features of z_τ and (iii) $val(\tau, \alpha_{j1}), \dots, val(\tau, \alpha_{ju})$ are the target features of z_τ . \square

The reason for defining some selection criterion over state abstractions is that their number may be very high, so that the clustering algorithm may well get confused when searching over a high-dimensional and sparse space (while taking long computation times).

The whole solution approach is encoded below in a general algorithmic form.

Definition 4 (Meta-algorithm CA-PPM Discovery). Given a log L , compute a CA-PPM model M for L as follows:

1. Derive $context(\tau)$ for each $\tau \in L$, by suitably computing $env(\tau)$;
2. Compute a reference set R of state abstractions for L via a given trace abstraction function abs ; // i.e., $R = \{ abs(\tau) \mid \tau \in L \}$
3. $T := PCM_mine(L, R)$; // build a PCM by using some performance sketch of L w.r.t. a subset of R , chosen according to a suitable selection criterion (cf. Def. 3)
4. Let c and p be the partitioning and prediction functions of T , resp., and $L[1], \dots, L[k]$ be the clusters found;
5. **for** $i = 1..k$ **do**
6. $\mu_i := PPM_mine(L[i])$; // induce the PPM model μ_i out of $L[i]$
7. **end**
8. **return** $\langle c, \langle \mu_1, \dots, \mu_k \rangle \rangle$; □

4 Instantiations of CA-PPM Discovery

The general computation scheme introduced in the previous section is parametric w.r.t. three major kinds of tasks: (i) selecting a subset of the target features (i.e., state abstractions) in the a propositional view produced for the log, (ii) inducing a multi-target predictive clustering model from a such a log view (Step 3), and (iii) inducing a single process-performance model from each trace cluster (Step 6). Various instantiations of this scheme can be devised by suitably implementing these tasks. Two concrete algorithms implementing it are illustrated in the rest of this section, named CA-TP and Ens-CA-TP, respectively.

4.1 Algorithm CA-TP

As this algorithm has the same structure as meta-algorithm CA-PPM Discovery, we will only describe next how the three major points of parametricity mentioned right above are actually implemented in it.

Selection of candidate state abstractions A simple greedy strategy (proposed in [14]) for the selection of state abstractions, relies on the usage of an ad-hoc scoring function $\phi : \mathcal{R} \times 2^{\mathcal{T}} \rightarrow [0, 1]$ to give each state abstraction $\alpha \in \mathcal{R}$ a score $\phi(\alpha, L)$. This score is meant to quantify the confidence in the fact that α is a good target feature for finding an effective predictive clustering model for L . More specifically:

$$\phi(\alpha, L) = \sqrt[3]{\phi_{var}(\alpha, L) \times \phi_{corr}(\alpha, L) \times \phi_{supp}(\alpha, L)} \quad (1)$$

where $\phi_{var}(\alpha, L)$, $\phi_{corr}(\alpha, L)$ and $\phi_{supp}(\alpha, L)$ are all functions ranging on $[0, 1]$. Basically, $\phi_{var}(\alpha, L)$ gives preference to higher-variability abstractions – the more the variability of trace measures the higher the score. Conversely, $\phi_{corr}(\alpha, L)$ measures the correlation between the values taken by α on a trace and the associated descriptive (context). Finally, $\phi_{supp}(\alpha, L)$ simply is $2 \times \min(0.5, |\{\tau \in L \mid val(\tau, \alpha) > 0\}|)$. In this way, the selection is biased towards the creation of target features that will ensure a good trade-off between support, correlation with descriptive features (i.e., those guiding the partitioning of log traces) and performance values’ variability (as to find clusters showing quite different performance models).

Based on the scoring function above and on a suitable threshold $\sigma \in [0, 1]$, a subset of pivot abstractions is selected by simply keeping any abstraction with a score higher than σ . This corresponds to considering a specific instantiation of the general selection criterion Φ , denoted by $\Phi[\phi, \sigma]$ hereinafter, such that, for any $\alpha \in abs(L)$, $\Phi[\phi, \sigma](\alpha) = \text{true}$ iff $\phi(\alpha) \geq \sigma$. Once these pivot abstractions have been chosen, the corresponding performance sketch can be derived for the given log (cf. Def 3), and exploited to induce a preliminary (propositional) predictive clustering model — with the target features being an approximated representation of how remaining times vary along process enactments — prior to refining the prediction function by inducing a more precise process performance model for each cluster (Steps 5-7 in Def. 4).

Inducing a predictive clustering model (Function `PCM_mine`). In algorithm CA-TP, predictive clustering models take the form of *Predictive Clustering Trees (PCTs)* [6] (more precisely, multi-regression PCTs), which showed good accuracy and scalability in many real-life applications. Such a model, where the cluster assignment function is encoded by a *decision tree*, is learnt with system CLUS [1], via a recursive partitioning of the training set. Roughly speaking, at each step, a split test is greedily chosen, over one descriptive feature, which (locally) minimizes a loss function, according to (prototype-based) intra-class variances. In particular, distances are measured by applying the classical Euclidean distance to target features only, while the prototype of each cluster C_i (also used as the local, constant, predictor for C_i) is the projection of C_i ’s centroid (i.e., the cluster mean, under the above distance) onto the target subspace.

Inducing a state-based performance model (Function `PPM_mine`). In order to implement Step 6, the method proposed in [3] is used to discover, for each cluster, a *PPM* model, in the form of an *Annotated Finite State Machine (A-FSM)*. Basically, in [3], an FSM is first built, where a one-to-one mapping exists between its nodes and the representations yielded by a given abstraction function abs , while each transition is labelled with an event property (namely, a task label in our setting). Assuming, e.g., that function abs_{∞}^{list} is used and that a, b, c are three process tasks, a transition labelled with c from state $\langle a, b \rangle$ to state $\langle a, b, c \rangle$ will appear in the resulting FSM model if there is some trace τ in the input log such that $abs_{\infty}^{list}(\tau(i)) = \langle a, b \rangle$ and $abs_{\infty}^{list}(\tau(i + 1)) = \langle a, b, c \rangle$. Such a model is eventually turned into an A-FSM, by equipping each node s with some statistics (e.g., the average), computed from the values that $\hat{\mu}$ takes on all trace prefix

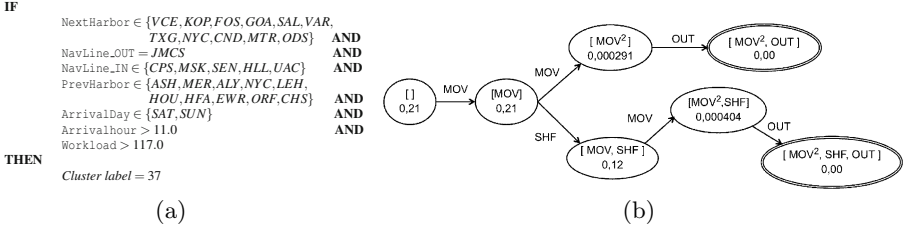


Fig. 1. Excerpt of a CA-PPM: (a) decision rule and (b) A-FMS model for one cluster

$\tau \in \mathcal{P}(L)$ such that $abs(\tau)$ coincides with the abstraction of s . The A-FSM model of cluster $L[i]$ is used to implement function μ_i , estimating the remaining time of any (sub-) trace $\tau(j)$ falling in $L[i]$. In fact, reasonably assuming that valid context data are available for τ (i.e., $context(\tau)$ does not contain missing/null values), the partitioning function c can always assign τ to one of the discovered clusters (denoted by $c(\tau)$). In case $\tau(j)$ is mapped (by iterated applications of abs) to an unseen sequence of states of the model, the forecast is simply derived from the latest valid estimate computed for the same enactment. Precisely, denoting by $\tau(j')$ the longest prefix of $\tau(j)$ reaching a valid state in the A-FSM model, we estimate $\mu(\tau(j)) = \max\{0, \mu_i(\tau(j')) - time(\tau[j]) + time(\tau[j'])\}$.

Example 2. Fig. 1 shows an excerpt of a CA-PPM which was mined from log data like those described in Example 1, by using algorithm CA-TP with the abstraction function abs_4^{bag} . The left side of Fig. 1 reports the decision rule associated with cluster 37 — corresponding to a leaf of the PCT model found by CLUS [1]. Clearly, cluster membership depends on both case properties and environmental data. Despite its simplicity, the rule helps recognize quite a specific, and yet relatively frequent, scenario for handling containers — the cluster gathers, indeed, 43 of the 5336 traces. The A-FSM encoding the prediction function (μ_{37}) of the same cluster is depicted on the right of the figure. Notice that notation MOV^2 , appearing in certain nodes, means task MOV occurring twice. The forecast made by this A-FSM, for a novel container case, clearly depends on the latter 4 tasks it has undergone. In particular, two consecutive MOV tasks are estimated to yield shorter processing times (than when the first MOV is followed by a SHF). Let us assume that the following context data are associated with the trace τ of Example 1: $NavLine_IN = CPS$, $NavLine_OUT = JMCS$, $NextHarbor = KOP$, $PrevHarbor = ASH$, $Workload=200$, $ArrivalDay = SUN$ and $ArrivalHour = 16$. Clearly all τ 's prefixes are assigned to cluster 37, and the respective forecasts (made by way of the A-FSM in Fig. 1.b) are: $\mu(\tau(0)) = \mu(\tau(1)) = 0.21$, $\mu(\tau(2)) = 0.000291$, and $\mu(\tau(3)) = 0.0$. \triangleleft

4.2 An Ensemble-Based Solution: Algorithm Ens-CA-TP

A critical point in the computation of a CA-PPM (as it is specified in Def. 4) concerns the clustering phase (Step 3 in Def. 4), and, in particular, the careful

selection of a subset of (pivot) state abstractions, to be used as target variables in a propositional encoding of the log. In algorithm **CA-TP**, this sub-problem is simply faced by requiring the user to set a lower threshold on the scores (computed via the function in Eq. 1) of the candidate state abstractions. However, it is not easy, in general, to determine an optimal value for this threshold, which can ensure that the selected features are an adequate choice for eventually finding a good **CA-PPM** model. In order to overcome this limitation, and to also reduce the amount of human intervention required, we next introduce an ensemble-based approach to log clustering, as an alternative implementation of function **PCM_mine** where no relevance threshold must be stated at all. This gives rise to a second instantiation, named hereinafter **Ens-CA-TP**, of the general computation scheme of Def. 4. This algorithm coincides with algorithm **CA-TP** (shown in the previous subsection) in the implementation of all of the steps but the third one (where the abstract function **PCM_mine** is invoked). Hence, for the sake of conciseness, the rest of this subsection only illustrates how this specific task is accomplished in algorithm **Ens-CA-TP** through an ensemble clustering approach, while omitting the description of the other steps.

In general, the core idea of ensemble approaches is to combine a set of models, all fulfilling the same mining task, in order to obtain a better composite global model. In particular, in the case of predictive models, for any new instance a prediction can be made by combining the prediction functions of all the models in the ensemble. The specification of a typical ensemble method consists of three building blocks: (i) a *base induction algorithm* (a.k.a. base learner), (ii) an *ensemble generation strategy*, where different base models are produced by applying instantiations of the base learner to sets of instances derived from the original training set, and (iii) a *combination strategy*, determining how the different models in the ensemble are eventually integrated together.

In the rest of this section, we explain how these three components are specified in algorithm **Ens-CA-TP**, relatively to the implementation of function **PCM_mine**.

Base learner and ensemble generation. **Ens-CA-TP** reuses, as base learner, the same method [1] as algorithm **CA-TP**, so that an ensemble of PCTs is eventually built. A necessary condition for having an effective ensemble is that its base models are both accurate enough and quite different from each other. Typically, diversity is obtained by manipulating either the training set or the learning algorithm itself. For example, in [17], where the discovery of ensembles of multi-objective decision trees has been studied, two ensemble generation methods are considered: bagging and random forests. Incidentally, in a bagging [8] scheme, each model is trained on a sample of instances (taken with replacement from the training set), with each sample having the same size as the original training set; conversely, a Random Forest ensemble [9] consists of many individual, unpruned decision trees, each of which is induced quickly by iteratively choosing the best split among a random sample of the input attributes. Both approaches were shown capable to help improve the performances and robustness of PCT learners, but are not suitable for our setting, where the main concern is to reduce the dimensionality of the output space.

We thus propose to build an ensemble of PCT models, by applying the same learning algorithm [1] to m different performance sketches of the input process log, each of which provides a different view of the input traces, based on a different set of pivot state abstractions. In order to generate each PCT model T_i (with $i = 1, \dots, m$), we randomly choose a value $\sigma_i \in \{0, 0.8\}$, and compute the set \mathcal{A}^i of all the state abstractions that get a score higher than σ_i by the function in Eq. 1 — i.e., $\mathcal{A}^i = PA^{\Phi[\phi, \sigma_i]}$ (cf. section 4.2). In more details, we set $\sigma_i = \max\{0.4 \times \sigma', 0.8\}$, by taking σ' out of a lognormal distribution with mean 0 and standard deviation 0.25. Based on the selected pivot abstraction \mathcal{A}^i , a propositional encoding $PS_{\mathcal{A}^i}$ of the log can be obtained, and given as input to the base learner, in order to build the i -th model (i.e., T_i) of the ensemble. In order to make the PCT induction faster, and to possibly increase the level of diversity among base models, we preliminary manipulate $PS_{\mathcal{A}^i}$, by applying a random-projection procedure [5] mapping the target features of $PS_{\mathcal{A}^i}^i$ into a k -dimensional space, where k itself is chosen randomly. More specifically, we set $k = \max\{40, k'\}$, where k' is a random variable following a Poisson distribution with mean 25. Notice that the particular probability distributions used for setting variables σ_i and k were chosen pragmatically, based on the results of a series of experiments, conducted against different data distributions.

Combination. In general, there are two main approaches to combining clustering models: (i) pure consensus methods and (ii) meta-learning. The first family of methods includes, e.g., those using a majority voting scheme [13], performance-guided weighting schemes [19], hypergraph partitioning techniques [21], or linear programming [7], just to cite a few. Meta-learning methods, which generally try to learn from the output of base models, include, in particular, stacking techniques [12,15,22]. In the case of clustering, these latter kind of techniques basically build a meta-dataset (encoding how training data have been clustered by the different models in the ensemble), which is eventually used as the input for a further (meta-) clustering step — where whatever clustering algorithm could be employed, in principle.

In algorithm **Ens-CA-TP**, the second kind of combination strategy (i.e. meta-learning) is adopted. In more details, each trace τ is turned into a tuple of the form $\langle c_1(\tau), \dots, c_m(\tau) \rangle$, where $c_i(\cdot)$ denotes the partitioning function of the PCT model T_i , for $i = 1, \dots, m$. Notice that the prediction function of the models are disregarded at all, as we are only interested here in exploiting these models in order to find a consensus clustering over the input log. In this way, a higher-level training set is produced, from which a meta-clustering model can be mined out.

Two classical algorithms can be alternatively used in **Ens-CA-TP** to compute such a partitioning: k -means and EM (Expectation Maximization). In general, the number of clusters is a critical parameter which can strongly impact on the quality of the final consensus clustering model. When applying either of the meta-clustering methods above, two options are available in algorithm **Ens-CA-TP** for automatically setting the number of meta-clusters: (i) **MaxCl#**, the maximum number of clusters found by the various models in the input ensemble; and (ii) **AvgCl#**, a weighted average of the cardinalities of all the clustering models in the

ensemble, where the weight of each model coincides with the accuracy score of the model itself. More precisely, the weight of each model T_i in the ensemble (with $i = 1, \dots, m$) is computed as follows: $weight(T_i) = \frac{\max_{j=1, \dots, m} \{Err(T_j)\} - Err(T_i)}{\max_{j=1, \dots, m} \{Err(T_j)\}}$, where $Err(T_i)$ is the average of the three error metrics (namely, *rmse*, *mae*, *mape*) computed for T_i (via cross-validation), and rescaled all onto $[0,1]$ — i.e., $Err(T_i) = \frac{1}{3} \times \left[\frac{rmse(T_i)}{\max_{j=1, \dots, m} \{rmse(T_j)\}} + \frac{mae(T_i)}{\max_{j=1, \dots, m} \{mae(T_j)\}} + mape(T_i) \right]$. These simple heuristics have been chosen mainly for scalability reasons, in the place of more refined methods available in the literature, such as, e.g., [18,20,10].

Notice, moreover, that the distance function $d(\cdot, \cdot)$, used in the k-means procedure, essentially corresponds to a weighted mismatch score, computed as follows. Let x and y be two distinct traces, and $\langle c_1^x, \dots, c_m^x \rangle$ and $\langle c_1^y, \dots, c_m^y \rangle$ be their respective representations in the meta-dataset encoding the clusterings of the base models — here c_i^x (resp., c_i^y) denotes the cluster label assigned to x (resp., y) by the i -th model. Then, $d(x, y) = |\{i \in 1, \dots, m \mid c_i^x \neq c_i^y\}| \times \frac{1}{m}$.

In this way, considering all the PCTs in the ensemble, the input log can be partitioned into a set of clusters, as required in algorithm **CA-PPM Discovery** (Step 4). However, function **PCM_{mine}** is also expected to return a *PCM* model, which includes, in particular, a partitioning function $c(\cdot)$ of the form $c : context(\mathcal{T}) \rightarrow \mathbb{N}$, assigning any trace $\tau \in \mathcal{T}$ to a cluster, based only on its context features $context(\tau)$. This overall partitioning function is built in algorithm **Ens-CA-TP** by combining the partitioning functions $c_1(\cdot), \dots, c_m(\cdot)$ of the base PCTs in the ensemble as follows: any new (possibly partial) trace τ is first converted into the meta-tuple $\langle c_1(context(\tau)), \dots, c_m(context(\tau)) \rangle$, which is then assigned to its closest (meta-)cluster — by considering either the distances between this tuple and all clusters' centroids, in the case of a k-means meta-clustering, or all clustering membership probabilities, in the case of an EM-based meta-clustering.

5 Experiments

A series of tests are described in this section, which were performed on the logs of the transshipment scenario mentioned in Example 1, using both algorithms **CA-TP** and **Ens-CA-TP** to discover a **CA-PPM** for the prediction of remaining processing times. Specifically, we report results obtained (using different kinds of abstraction functions) on a sample of 5336 traces, corresponding to all the containers that passed through the system in the first third of year 2006.

Prediction effectiveness was measured, according to a 10-fold cross validation scheme, by way of three classic error metrics: *root mean squared error (rmse)*, *mean absolute error (mae)*, and *mean absolute percentage error (mape)*. All the error results shown in the following have been averaged over 10 trials.

Table 1 reports the average errors, and the associated variances, made by algorithm **Ens-CA-TP** with a fixed size ($m = 50$) of the underlying ensemble, while varying both the meta-clustering algorithm (either *k*-means or *EM*) for combining them, and the heuristics for automatically setting the number of meta-clusters (either **MaxCl#**, or **AvgCl#**). These results are compared with those

Table 1. Time prediction errors (average \pm std_dev) obtained, in combination with abstraction function abs_2^{LIST} , by the algorithms: (i) CA-TP, provided with a randomly selected value of σ , and (ii) Ens-CA-TP, used with different meta-clustering options. Each parametrization of Ens-CA-TP is identified by a pair of the form (*meta-clustering method, cluster#-selection mode*), with *KM* denoting *k*-means.

Metric	CA-TP	Ens-CA-TP			
		(KM. AvgCl#)	(KM. MaxCl#)	(EM. AvgCl#)	(EM. MaxCl#)
rmse	0.327 \pm 0.070	0.193 \pm 0.008	0.174 \pm 0.010	0.303 \pm 0.087	0.236 \pm 0.101
mae	0.107 \pm 0.023	0.066 \pm 0.001	0.056 \pm 0.002	0.073 \pm 0.005	0.063 \pm 0.005
mape	0.259 \pm 0.061	0.147 \pm 0.036	0.164 \pm 0.012	0.229 \pm 0.031	0.182 \pm 0.062

Table 2. Error reductions (%) achieved by CA-TP and by Ens-CA-TP w.r.t. the baseline performance-prediction method (*FSM* [3])

Parameters (abs_h^{mode})		rmse ($\Delta\%$)		mae ($\Delta\%$)		mape ($\Delta\%$)	
mode	h	CA-TP	Ens-CA-TP	CA-TP	Ens-CA-TP	CA-TP	Ens-CA-TP
LIST	1	-1.2%	-1.1%	-1.6%	-1.4%	-5.8%	-6.0%
	2	-28.1%	-25.5%	-55.2%	-53.5%	-31.3%	-29.2%
	4	-65.6%	-62.7%	-71.4%	-72.6%	-73.9%	-60.8%
	8	-64.1%	61.2%	-71.4%	-70.3%	-74.9%	-73.4%
	16	-64.1%	-59.3%	-71.4%	-69.1%	-74.9%	-73.1%
	Total		-44.6%	-42.0%	-54.2%	-53.4%	-52.1%
BAG	1	-1.2%	-1.1%	-1.6%	-1.4%	-5.8%	-6.0%
	2	-27.7%	-23.8%	-53.3%	-47.1%	-33.0%	-28.9%
	4	-65.6%	-62.3%	-72.4%	-73.8%	-73.4%	-75.3%
	8	-65.6%	-66.4%	-72.4%	-72.5%	-75.5%	-76.6%
	16	-65.6%	-64.2%	-72.4%	-71.9%	-75.5%	-75.9%
	Total		-45.1%	-43.6%	-54.4%	-53.3%	-52.6%
Grand Total		-44.9%	-42.8%	-54.2%	-53.4%	-52.4%	-50.5%

obtained by using algorithm CA-TP with a randomly chosen value of the relevance threshold σ — taken from the same probability distribution as that used by Ens-CA-TP to generate an ensemble of PCT models. The goal of such an analysis is to study the improvement that can be obtained by our ensemble-based clustering, with respect to the case where a (possibly inexperienced) user has to decide how target features are to be selected, by choosing a particular value of σ . The figures in Table 1 demonstrate, as expected, that the ensemble-based approach is more accurate and stabler than an unbiased application of algorithm CA-TP. In fact, irrespective of the consensus strategy adopted, the average errors and their associated variances are always lower than those obtained with a single PCT model. Similar results were obtained as well when using other trace abstraction functions abs_h^{mode} than the one reported in the table — details are omitted for space reasons. Notably, the application of the Wilcoxon test [23] to the errors obtained by the two algorithms CA-TP and Ens-CA-TP proved that the improvement obtained by the latter is statistically significant (at a 0.05 level), whatever values are chosen for its parameters. However, due to lack of space, we next report only some detailed results obtained by Ens-CA-TP, using either *k*-means or *EM* together with the MaxCl# option.

Table 2 summarizes the percentage of error reduction ($\Delta\%$), in the prediction of remaining times, obtained by CA-TP and Ens-CA-TP w.r.t. the performance prediction method proposed in [3], here denoted by *FSM*. The tests were performed using different trace abstraction functions abs_h (set-based abstractions are omitted for lack of space). Differently from the experiments above, algorithm CA-TP is provided with the optimal value of the relevance threshold (namely, $\sigma = 0.4$), pragmatically found throughout numerous trials on the same dataset. For the sake of clarity, the table only focuses on the case where the ensemble created by Ens-CA-TP (still consisting of 50 PCTs) is combined by using *k*-means along with the option **MaxCl#** — i.e., the maximum number of clusters in the ensemble is chosen. Notice that, for all metrics, variance values (not reported here for lack of space) were always lower than 5% of the respective average. The goal of this empirical analysis is two-fold: (i) showing the superiority of the general context-aware approach encoded by meta-algorithm CA-PPM **Discovery** w.r.t. previous performance prediction methods, and (ii) assessing the capability of algorithm Ens-CA-TP to achieve good results, without any external guidance on the filtering of the target features. In fact, the above results clearly show that both algorithms outperforms neatly the baseline method (*FSM*), as confirmed by a Wilcoxon test on detailed results. On the other hand, the achievements of the ensemble-based algorithm are always very close to those of the “biased” version of CA-TP (instantiated with an optimal value of σ). As far as concerns the impact of trace abstraction functions, prediction accuracies seems to depend mainly on the history horizon *h*. Indeed, appreciable benefits are obtained as soon as $h > 1$, with the best performances achieved with $h = 8$ (when each error shrinks at least of 61% w.r.t. the baseline), while no substantial further improvement is obtained with $h > 8$. The effect of the abstraction mode looks less marked, since very similar (good) results are found in both cases.

Table 3. Computation times (sec) for algorithm CA-TP. the baseline performance-prediction method (*FSM* [3]). and algorithm Ens-CA-TP. used with two different choices (namely. *k*-means and *EM*) of the underlying meta-clustering method.

Parameters (abs_h^{mode})		Ens-CA-TP		CA-TP	FSM[3]
mode	<i>h</i>	k-means	EM		
LIST	1	59.2	32.5	16.8	3.9
	2	76.5	241.1	20	5.6
	4	75.3	305.7	19.6	10.7
	8	86.9	373.4	20.2	16.0
	16	168.9	397.4	92.3	89.8
	Total		93.4	270.0	33.8
BAG	1	61.3	42.4	17.0	4.0
	2	74.8	235.8	19.7	5.5
	4	72.5	315.0	18.7	8.4
	8	85.3	304.6	19.8	10.6
	16	145.7	328.2	79.0	32.3
	Total		87.92	245.2	30.9
Grand Total		90.67	257.6	32.4	18.7

Table 3 shows the average computation times spent by the baseline method [3], CA-TP and Ens-CA-TP (using both k -means and EM for consensus clustering), on a dedicated machine with an dual-core Intel processor, 2GB of RAM and Windows XP Pro. As expected, the latter two methods (and, in particular, Ens-CA-TP) always take longer times than the baseline, yet getting a good trade-off between effectiveness and efficiency. Moreover, preliminary experiments with a parallelized version of Ens-CA-TP make us confident about the possibility of achieving substantial scalability gains.

All the experiments discussed so far were performed by always using the same ensemble size (namely, $m=50$), which seemed to us capable to ensure a good trade-off between effectiveness, robustness and scalability. In fact, in a preliminary series of tests (where m was made moving from 2 to 512), we noticed that a wide range of ensemble sizes allows to obtain almost the same averages and variances for all the error metrics — apart from the case of little ensembles (namely, those consisting of less than 8 models), which tend to yield higher (and more variable) errors. Detailed results from this analysis are not reported here, due to space reasons.

6 Conclusions

We have described a predictive clustering method for discovering performance-oriented process models, where a number of homogeneous execution groups are recognized, and provided with separate performance-prediction model. In particular, we have proposed the adoption of an ensemble learning approach for the delicate task of finding preliminary clustering out a propositional encoding of the input log. The approach has been implemented and validated on a real case study, where it showed promising results.

As to future work, we plan to investigate on using novel methods for selecting relevant space abstractions, and on adopting more powerful process models to capture concurrent behaviors effectively. We will also explore the possibility to convert the discovered PCTs all into a set of decision rules, each associated each with an A-FSM predictor, and to directly combine these rule-based predictive models according according to a consensus regression approach (similarly to [4]).

Acknowledgements. This work was partially funded by the Italian Ministry MIUR, under the research project “*FRAME*” (PON 2007-2013).

References

1. CLUS: A predictive clustering system, <http://dtai.cs.kuleuven.be/clus/>
2. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering* 47(2), 237–267 (2003)
3. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. *Information Systems* 36(2), 450–475 (2011)
4. Aho, T., Zenko, B., Dzeroski, S.: Rule ensembles for multi-target regression. In: *Proc. of 9th Int. Conf. on Data Mining, ICDM 2009*, pp. 21–30 (2009)

5. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proc. of 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD 2001, pp. 245–250 (2001)
6. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101(1-2), 285–297 (1998)
7. Boulicaut, C., Ostendorf, M.: Combining multiple clustering systems. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 63–74. Springer, Heidelberg (2004)
8. Breiman, L.: Bagging predictors. *Machine Learning* 24(2), 123–140 (1996)
9. Breiman, L.: Random forests. *Machine Learning* 45(1), 5–32 (2001)
10. Casillas, A., de Lena, M.T.G.d., Martínez, R.: Document clustering into an unknown number of clusters using a genetic algorithm. In: Matoušek, V., Mautner, P. (eds.) TSD 2003. LNCS (LNAI), vol. 2807, pp. 43–49. Springer, Heidelberg (2003)
11. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: Proc. of 16th Int. Conf. on Cooperative Information Systems, CoopIS 2008, pp. 319–336 (2008)
12. Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: Proc. of 20th Int. Conf. on Machine Learning, ICML 2003, pp. 186–193 (2003)
13. Filkov, V., Skiena, S.S.: Heterogeneous data integration with the consensus clustering formalism. In: Rahm, E. (ed.) DILS 2004. LNCS (LNBI), vol. 2994, pp. 110–123. Springer, Heidelberg (2004)
14. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Proc. of 20th Int. Conf. on Cooperative Information Systems, CoopIS 2012, pp. 287–304 (2012)
15. Goder, A., Filkov, V.: Consensus clustering algorithms: Comparison and refinement. In: Proc. of Workshop on Algorithm Engineering and Experiments, ALENEX 2008, pp. 109–117 (2008)
16. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. *IEEE Transaction on Knowledge and Data Engineering* 18(8), 1010–1027 (2006)
17. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Ensembles of multi-objective decision trees. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 624–631. Springer, Heidelberg (2007)
18. Mufti, G.B., Bertrand, P., El Moubarki, L.: Determining the number of groups from measures of cluster stability. In: Proc. of Int. Symp. on Applied Stochastic Models and Data Analysis, ASMDA 2005, pp. 404–412 (2005)
19. Opitz, D.W., Shavlik, J.W.: Generating accurate and diverse members of a neural-network ensemble. In: Proc. of Advances in Neural Information Processing Systems 8, NIPS 1995, pp. 535–541 (1995)
20. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proc. of 17th Int. Conf. on Machine Learning, ICML 2000, pp. 727–734 (2000)
21. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research* 3, 583–617 (2002)
22. Topchy, A.P., Jain, A.K., Punch, W.F.: A mixture model for clustering ensembles. In: Proc. of 4th SIAM Int. Conf. on Data Mining, SDM 2004 (2004)
23. Wilcoxon, F.: Individual comparisons by ranking methods. *Biometrics Bulletin* 1(6), 80–83 (1945)

Erratum: Machine Learning as an Objective Approach to Understanding Music

Claire Q¹ and Ross D. King²

¹ Aberystwyth University, UK
ceq08@aber.ac.uk

² University of Manchester, UK
ross.king@manchester.ac.uk

A. Appice et al. (Eds.): NFMCP 2012 Workshop, LNAI 7765, pp. 64–78, 2013.
© Springer-Verlag Berlin Heidelberg 2013

DOI 10.1007/978-3-642-37382-4_16

The paper starting on page 64 of this publication has been withdrawn because Figure 1 is incorrect and it is unclear if the paper's results can be reproduced.

The original online version for this chapter can be found at
http://dx.doi.org/10.1007/978-3-642-37382-4_5

Author Index

- Atzmueller, Martin 33
Buccafurri, Francesco 200
Ceci, Michelangelo 185
Costa, Gianni 94
Davis, Michael 138
Di Mauro, Nicola 155
Egho, Elias 109
El Mahrsi, Mohamed Khalil 124
Esposito, Floriana 155
Ferilli, Stefano 170
Ferri, Cèsar 1
Folino, Francesco 215
Guarascio, Massimo 215
Gubrynowicz, Ryszard 79
Guo, Hongyu 49
Hajja, Ayman 79
Hernández-Orallo, José 1
Ienco, Dino 109
Jay, Nicolas 109
King, Ross D. 64
Köhler, Stefan 33
Kuželka, Ondřej 17
Lax, Gianluca 200
Leuzzi, Fabio 170
Liu, Weiru 138
Loglisci, Corrado 185
Malerba, Donato 185
Manco, Giuseppe 94
Martínez-Plumed, Fernando 1
Masciari, Elio 94
Miller, Paul 138
Napoli, Amedeo 109
Neubeck, Philipp 33
Nocera, Antonino 200
Paquet, Eric 49
Poncelet, Pascal 109
Pontieri, Luigi 215
Q, Claire 64
Quantin, Catherine 109
Raïssi, Chedy 109
Ramírez-Quintana, María José 1
Ras, Zbigniew W. 79
Rossi, Fabrice 124
Rotella, Fulvio 170
Seipel, Dietmar 33
Szabóová, Andrea 17
Taranto, Claudio 155
Teisseire, Maguelonne 109
Ursino, Domenico 200
Viktor, Herna Lydia 49
Wieczorkowska, Alicja A. 79
Železný, Filip 17