Karell Bertet
Daniel Borchmann
Peggy Cellier
Sébastien Ferré (Eds.)

14th International Conference on Formal Concept Analysis

Rennes, France, June 13-16, 2017
Supplementary proceedings

Volume Editors

Karell Bertet
Université de La Rochelle, L3i
Avenue Michel Crépeau, 17042 La Rochelle, France
E-mail: kbertet@univ-lr.fr

Daniel Borchmann
Technische Universitat Dresden, Fakultat Informatik, Institut fur Theoretische Informatik
Dresden, Germany
E-mail: daniel.borchmann@tu-dresden.de

Peggy Cellier
INSA Rennes, IRISA
Campus Beaulieu, 35042 Rennes cedex, France
E-mail: peggy.cellier@irisa.fr

Sébastien Ferré
University of Rennes 1, IRISA
Campus Beaulieu, 35042 Rennes cedex, France
E-mail: sebastien.ferre@irisa.fr

ISBN 978-2-9527630-5-9

Preface

This volume is the supplementary volume of the 14th International Conference on Formal Concept Analysis (ICFCA 2017), held from June 13th to 16th 2017, at IRISA, Rennes. The ICFCA conference series is one of the major venues for researches from the field of Formal Concept Analysis and related areas to present and discuss their recent work with colleagues from all over the world. Since it has been started in 2003 in Darmstadt, the ICFCA conference series had been held in Europe, Australia, America, and Africa.

The field of Formal Concept Analysis (FCA) originated in the 1980s in Darmstadt as a subfield of mathematical order theory, with prior developments in other research groups. Its original motivation was to consider complete lattices as lattices of concepts, drawing motivation from philosophy and mathematics alike. FCA has since then developed into a wide research area with applications much beyond its original motivation, for example in logic, data mining, learning, and psychology.

The FCA community is mourning the passing of Rudolf Wille on January 22nd 2017 in Bickenbach, Germany. As one of the leading researchers throughout the history of FCA, he was responsible for inventing and shaping many of the fundamental notions of this area. Indeed, the publication of his article "Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts" is seen by many as the starting point of Formal Concept Analysis as an independent direction of research. He was head of the FCA research group in Darmstadt from 1983 until his retirement in 2003, and remained an active researcher and contributor thereafter. In 2003, he was among the founding members of the ICFCA conference series.

For this supplementary volume, 13 papers were chosen to be published: four papers judged mature enough to be discussed at the conference and nine papers presented in the demonstration and poster session.

This proceedings volume would not have been possible without the valuable work by the authors, the members of the Program Committee, and the members of the Editorial Board. We also want to express our gratitude to the team of local organizers, who made sure that the conference ran smoothly and was a pleasant experience for all its participants.

June 2017

Karell Bertet
Daniel Borchmann
Peggy Cellier
Sébastien Ferré

Table of Contents

Preface . . . . .

Papers

Indexes to Evaluate Reduced Concept Lattices 1
Sérgio Dias, Luis Zárate, Mark Song and Newton Vieira

Words Grow on Lattices: Analysing Sequences through FCA 17
Krzysztof Nowak, Daniel Salmond and Raymee Chau

Mining Triclusters of Similar Values in Triadic Real-Valued Contexts 31
Dmitry Egurnov, Dmitry Ignatov and Engelbert Mephu Nguifo

Classification of demographic sequences based on pattern structures and emerging patterns 49
Danil Gizdatullin, Dmitry Ignatov, Ekaterina Mitrofanova and Anna Muratova

Demonstration and Poster Session

Direct and Ordered Direct Bases Updates for One-set Extensions of a Closure System 67
Kira Adaricheva and Taylor Ninesling

Local Generation of AOC-posets: Reducing the Complexity of Conceptual Navigation for SPLE Product Selection 71
Alexandre Bazin, Jessie Carbonnel and Giacomo Kahn

Logical Information Systems for Linguistic Data: TermLis 75
Annie Foret

Mining Relevant Interval Rules 79
Thomas Guyet, René Quiniou, and Véronique Masson

Navigation and Exploration Tool for Polyadic FCA 83
Levente Lorand Kis, Christian Săcărea and Diana Troancă

Program for Conceptual Navigation in Relational Databases 87
Jens Kötters

Lattice Miner 2.0: A Formal Concept Analysis Tool 91
Rokia Missaoui and Kevin Emamirad

FACT - A tool for Temporal Formal Concept Analysis 95
Beniamin Movileanu, Christian Săcărea, Diana-Florina Şotropa

Exploration of Textual Sequential Patterns 99
Hedi-Théo Sahraoui, Pierre Holat, Peggy Cellier, Thierry Charnois and Sébastien Ferré

Author Index . . . . .

Indexes to Evaluate Reduced Concept Lattices

Sérgio M. Dias[1,2,3] (✉), Luis E. Zárate[2], Mark A. J. Song[2], and Newton J. Vieira[3]

[1] Federal Service of Data Processing (SERPRO)
Av. José Cândido da Silveira, 1.200
Cidade Nova, Belo Horizonte, 31.035-536, Minas Gerais, Brazil
sergio.dias@serpro.gov.br
[2] Department of Computer Science
Pontifical Catholic University of Minas Gerais (PUC Minas), Minas Gerais, Brazil
{zarate,song}@pucminas.br
[3] Department of Computer Science
Federal University of Minas Gerais (UFMG), Minas Gerais, Brazil
nvieira@dcc.ufmg.br

Abstract. In formal concept analysis (FCA), the problem of obtaining a concept lattice of appropriate size and structure, that exposes the truly relevant aspects, is one of the most important problems. Even when the number of formal concepts is not very large, the essential aspects, those effectively needed, can be immersed in a maze of irrelevant details. In order to deal with the complexity of structure obtained, there are many techniques, with different characteristics, for concept lattice reduction. Some works apply objective indexes to evaluate their techniques. However, those objective measures and criteria are generally used only in the selection of formal concepts. In this work we propose the use of three indexes to evaluate reduced concept lattice based on proper implications that represent the original and reduced lattice structure. The *information content*, *fidelity* and *representativeness*. The indexes are applied in three reduction techniques using a small example. The results show the characteristics of each lattice after the reduction.

Keywords: Formal concept analysis, Lattice reduction, Indexes, Proper implications.

1 Introduction

Formal concept analysis (FCA) is currently considered an important formalism for knowledge extraction, representation and analysis from data [35,29] with applications in several areas [30,44,18,33,22]. The problem of obtaining a concept lattice of appropriate size and structure, that exposes the really relevant aspects, is one of the most important problems when using FCA. In fact, FCA induces a potentially high combinatorial complexity and the structures obtained, even from a small dataset, may become difficult to manipulate [25]. Moreover, besides being computationally expensive to compute all formal concepts, their number and interrelationships can impair an effective analysis [36].

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

In order to deal with the complexity of structure obtained, many techniques, with different characteristics, for *concept lattice reduction* have been given in the literature [15,26,40,32,17,16,31,9,23,25,36,24]. However, at present there is a high subjectivity when assessing the quality of a reduction technique. Some techniques are accompanied by interest measures [26,45,40,28] or criteria [6,39] which are intended to guarantee the lattice quality. In contrast, specific measures and criteria for selection of formal concepts have also been widely used [27,28].

Existing objective measures and criteria are generally used only in the selection of formal concepts. In other words, measures evaluate only one formal concept. Examples are the indexes available in [28]. However, pointing the relevance of formal concepts is of little importance to determine the quality of a concept lattice or a structure formed by subset of formal concepts. It is necessary indexes to evaluate all lattice structure (a reduced concept lattice or a selected subset of formal concepts that not necessary form a concept lattice).

In this work we propose the use of three indexes to evaluate reduced concept lattices: *information content*, *fidelity* and *representativeness*. The indexes make use of a set of proper implications [41,7] of the original $\mathcal{I}$ and reduced formal contexts (or concept lattices) $\mathcal{I}_r$. The sets $\mathcal{I}$ and $\mathcal{I}_r$ represent the knowledge of original and reduced concept lattice, respectively. Using those sets, the proposed indexes, with different characteristics and purpose, evaluate all the lattice structure. The indexes are applied in three reduction techniques using a small example. Among all reduction techniques analyzed in [15,13], we selected *minimal set of attributes selection grouping* [46], using the technique JBOS [14] and *constraints learned from data* [6]. The results show the characteristics of each lattice after the reduction.

The remaining sections of this paper are organized as follows. Section 2 introduce some core concepts of FCA, clarifies what is a technique for concept lattice reduction and discuss the use of proper implications. Section 3 presents the proposed indexes. Section 4 applies the tree indexes in a small example. Finally, Section 5 draws some conclusions.

2 Formal concept analysis: short review

This short review presents the notions and terminology which are important for the understanding of our work. The notions and terminology are based in [19].

In FCA the initial data are presented as a *formal context*, a triplet (G, M, I), where G is a set of elements called *objects*, M is a set of elements called *attributes* and $I \subseteq G \times M$ is called an *incidence relation*. If $(g, m) \in I$, one says that "*the object g has the attribute m*".

Given a set of objects $A \subseteq G$ from a formal context (G, M, I), the set of attributes that is common to all those objects is termed A'. Similarly, for a set $B \subseteq M$, B' is the set of objects that have all the attributes from B. That is to say $A' = \{m \in M | \forall g \in A \colon (g, m) \in I\}$ and $B' = \{g \in G | \forall m \in B \colon (g, m) \in I\}$[4].

[4] The notation x' will be used as abbreviation $\{x\}'$, whether x is an object or an attribute.

By using such *derivation operators*, the notion of *formal concept* is defined as a pair $(A, B) \in \mathcal{P}(G) \times \mathcal{P}(M)$ such that $A' = B$ and $B' = A$, where A is called the *extent* and B the *intent* of the concept.

Given a formal context with attributes M, the set of *generators* of a formal concept (A, B) is $ger(A, B) = \{D \subseteq M | D' = A\}$; and the set of *minimum generators* of (A, B) is $mger(A, B) = \{D \in ger(A, B) | \nexists m \in D(D \setminus \{m\})' = A\}$.

The set of formal concepts is ordered by the partial order $\preceq$ such that for any two formal concepts (A_1, B_1) and (A_2, B_2), $(A_1, B_1) \preceq (A_2, B_2)$ iff $A_1 \subseteq A_2$ (equivalently, $B_2 \subseteq B_1$). The set of concepts ordered by $\preceq$ constitutes a complete lattice [12], so it's called *concept lattice*. The concept lattice obtained from a formal context (G, M, I) is denoted $\mathcal{B}(G, M, I)$.

The first part of the *basic theorem* on concept lattices [42] says that a concept lattice $\mathcal{B}(G, M, I)$ is a complete lattice in which for any arbitrary set $C \subseteq \mathcal{B}(G, M, I)$ the *infimum* and *supremum* are given by $\bigwedge C = (\bigcap X, (\bigcup Y)'')$ and $\bigvee C = ((\bigcup X)'', \bigcap Y)$, where $X = \{A \,|\, (A, B) \in C\}$ and $Y = \{B \,|\, (A, B) \in C\}$.

2.1 Concept lattice reduction

As the complexity of concept lattices is considered a limitation for an effective use of FCA in many situations, several techniques have been proposed [31,11,43,3,40,26,36,45,4]. Such techniques are called *concept lattice reduction* [15,13].

Definition 1 *A technique for* ***concept lattice reduction*** *is one that aims to reduce the complexity of a concept lattice, both in terms of magnitude and of inter-relationships, while maintaining relevant information.* □

Here, note that, "relevant information" is subjective and dependent on the user's interpretation [5].

In [15,13], we identified three classes of techniques for concept lattice reduction when we consider classical FCA:

- The techniques of *redundant information removal* modify the formal context, but the resulting concept lattice is isomorphic to the original one [46,34,19,31].
- A *simplification* technique is one that, from a formal context or a concept lattice, abstracts some non-essential differences (in accordance with some criteria) between concepts, objects or attributes [2,20,14,10].
- Finally, a *selection* technique is one that, from a formal context or concept lattice, selects a subset of formal concepts, objects or attributes that satisfy a set of constraints [26,36,5,3,45].

In our work, we considered only those techniques that presuppose a formal context already built. Moreover, only works of classical FCA were taken into account - extensions of FCA were not considered.

2.2 Implication rules

Given a formal context (G, M, I) or a concept lattice $\mathcal{B}(G, M, I)$, from them can be extracted exact implication rules (from now on named as *implications*). The definition of an implication is given as follows[19]:

Definition 1. *Being a formal context whose attributes set is M. An implication is an expression $P \to Q$, which $P, Q \subseteq M$.* □

An implication $P \to Q$, extracted from a formal context, or respective concept lattice, have to be such that $P' \subseteq Q'$. In other words: every object wich has the attributes of P, it also have the attributes of Q.

Note that, if X is a set of attributes, then X *respects* an implication $P \to Q$ iff $P \not\subseteq X$ or $Q \subseteq X$. An implication $P \to Q$ *holds* in a set $\{X_1, \ldots, X_n\} \subseteq M$ iff each X_i respects $P \to Q$; and $P \to Q$ *is an implication of the context* (G, M, I) iff it holds in its set of object intents (an object intent is the set of its attributes). An implication $P \to Q$ *follows from* a set of implications $\mathcal{I}$, iff for every set of attributes X if X respects $\mathcal{I}$, then it respects $P \to Q$. A set of implications $\mathcal{I}$ is said to be *complete* in (G, M, I) iff every implication of (G, M, I) follows from $\mathcal{I}$. A set of implications $\mathcal{I}$ is said to be *redundant* iff it contains an implication $P \to Q$ that follows from $\mathcal{I} \backslash \{P \to Q\}$. Finally, an implication $P \to Q$ is considered *superfluous* iff $P \cap Q \neq \emptyset$.

Here will be convenient that each implication represents a *minimal condition*; i.e., the smallest set of attributes that results in one single attribute. For this, we will require that the complete set of implications $\mathcal{I}$ of a formal context (G, M, I) have the following characteristics:

- the right hand side (rhs) of each implication is unitary: if $P \to m \in \mathcal{I}$, then $m \in M$;
- superfluous implications are not allowed: if $P \to m \in \mathcal{I}$, then $m \notin P$;
- specializations are not allowed, i.e. left hand sides (lhs) are minimal: if $P \to m \in \mathcal{I}$, then there is not any $Q \to m \in \mathcal{I}$ such that $Q \subset P$.

A complete set of implications in (G, M, I) with such properties is denoted set of *proper implications* [41] or *unary implication system* (UIS)[7].

Definition 2. *Let $\mathcal{J}$ be the complete closed set of implications of a formal context (G, M, I). Then the set of* proper implications *$\mathcal{I}$ for (G, M, I) is defined as: $\{P \to m \in \mathcal{J} \mid P \subseteq M$ and $m \in M \setminus P$ and $\forall Z \subset P$: $Z \to m \notin \mathcal{J}\}$.* □

The proposed indexes to evaluate reduced concept lattice (discussed later) are based on proper implications, which can be obtained from a formal context or concept lattice [41,8,7,38].

However, in some cases we extract such implications from a subset of formal concepts, which forms a given section of a concept lattice, and that section might not form a complete concept lattice [37][5]. That is the case for techniques of the

[5] The sets of objects or attributes of the considered concepts are not closure systems.

selection class which does not have access to the whole original concept lattice [13]. Thus it might not be possible to obtain a complete (with respect to some formal context) set of proper implications. Here we discuss how to extract *proper implications* with support greater than zero ($P \rightarrow m \in \mathcal{J} | P \neq \oslash$) from a subset of formal concepts.

Considering a single formal concept, the set of implications to be obtained will share the following properties with the original set of proper implications [41]:

Proposition 1 *It will not have specializations of its implications.* □

Proposition 2 *The rhs of its implications will be unitary.* □

Proposition 3 *It will be consistent with the original derivation operators when applied to only the elements of the concept.* □

Proposition 4 *It will have only implications with support greater than zero.* □

Proposition 5 *The set of implications must reflect the lattice formed by the actual set of chosen concepts.* □

Propositions 1, 2, 3 and 4 are guaranteed by the proper use of minimum generators [7] as lhs, and property 5 is guaranteed by ensuring those properties globally (topological ordering given by the cover relation of the lattice, smallest intentions first). The order in which the set of concepts is explored is important to ensure that the lhs of an implication be as small as possible.

Note that, despite the computational cost, minimal generators have gained prominence recently in FCA. their importance is due, mainly, to the fact that they favor the principle of *minimum description length* (MDL), i.e. the best hypothesis for a given set of data is the one that leads to the best compression of the data [21].

3 Indexes based on proper implications

Three indexes with distinct characteristics and objectives are discussed here: *information content*, *fidelity* and *representativeness*.

The *information content* is an index that aims to measure the information associated with a logical expression based on the ratio between the number of truth assignments that make the logical expression false (or true) and the total number of possible truth assignments [1]; in general, it is assumed that the assignments are equally likely. This index should say for example that a set of implications $\mathcal{I}$ plus an implication that does not follow from $\mathcal{I}$, is more informative than $\mathcal{I}$; in the same way, a generalization of a given implication i should be considered more informative than i.

The *fidelity* is an index that aims to measure the success rate when the rules derived from a reduced formal context (or lattice) are applied to the original

formal context objects [14]. Note that, when the implications of a reduced formal context are checked for original objects they may eventually fail, as is to be expected that such a smaller context contains some inaccuracies. In other words, modifications in objects, attributes or incidences made by reduction techniques can produce an implication rules set with inaccuracies when analyzed by means of original object set.

Finally, the *representativeness* is an index that aims to measure the success rate of implications considering only implications with satisfied left hand sides [15].

3.1 Information content index

Considering implications as logical sentences and attributes in M as sentential symbols, the set of *counter-models* of an implication $P \to m$, $cm(P \to m)$, consists of every $X \subseteq M$ such that $P \subseteq X$ and $m \notin X$, i.e., $cm(P \to m) = \{X \mid P \subseteq X \subseteq M \backslash \{m\}\}$. Intuitively, it consists of all (the $2^{|M|-(|P|+1)}$) possibilities of truth values for the attributes in M that makes $P \to m$ false. Considering each of the $2^{|M|}$ possibilities equally likely, the information content of $P \to m$, $cont(P \to m)$, is $cont(P \to m) = 2^{|M|-(|P|+1)}/2^{|M|} = 1/2^{|P|+1}$.

A set of attributes does not respect a set $\mathcal{I}$ of implications iff it does not respect *some* implication in $\mathcal{I}$. Consequently, the set of counter-models of $\mathcal{I} = \{P_1 \to m_1, \ldots, P_n \to m_n\}$ is $cm(\mathcal{I}) = \bigcup_{i=1}^{n} cm(P_i \to m_i)$. To determine the number of counter-models of $\mathcal{I}$ one have to apply the inclusion-exclusion principle (IEP). Defining $S_1 = \sum_{i=1}^{n} |cm(P_i \to m_i)|$ and, for $2 \leq k \leq n$, $S_k = \sum_{1 \leq i_1 < \cdots < i_k \leq n} |cm(P_{i_1} \to m_{i_1}) \cap \ldots \cap cm(P_{i_k} \to m_{i_k})|$, it follows, by IEP, that the number of counter-models of $\mathcal{I}$ is $|cm(\mathcal{I})| = S_1 - S_2 + \cdots + (-1)^{n+1} S_n$.

Example 1 *If $M = \{a, b, c\}$, there are $2^3 = 8$ possibilities of truth values for the 3 attributes. Then $cont(a \to b) = cont(\bar{a} \cup b) = 2^{3-2}/2^3 = 2/8$, as $a \to b$ is false only when a is true and b is false; and c can be true or false. The implication $a \to c$ also has two possibilities of being false, but one is the same as that of $a \to b$ being false: when a is true and both b and c are false; thus, $cont(\{a \to b, a \to c\}) = (2 + 2 - 1)/2^3 = 3/8$.* □

A concise way to represent the set of all counter-models is to list only the attributes that necessarily belong to each counter-model (the attributes of the lhs) and the attributes that do not belong to any counter-model (the attribute of the rhs). Next, this representation will be made by using a set of barred and/or non barred attributes; a non barred attribute means that it occurs in all counter-models, and a barred attribute means that it does not occur in any counter-model. For example, the set $\{a, b, \overline{e}\}$ represents all counter-models C such that $a, b \in C$ and $e \notin C$; in that case, if $M = \{a, b, c, d, e\}$, then $\{a, b, \overline{e}\}$ represents a total of $2^{5-3} = 4$ counter-models (c and d might be present or not in each counter-model of the represented set). In this way, the counter-models of $\mathcal{I} = \{P_1 \to m_1, \ldots, P_n \to m_n\}$ are represented by the set $\mathcal{C}(\mathcal{I}) = \{P_1 \cup \{\overline{m_1}\}, \ldots, P_n \cup \{\overline{m_n}\}\}$. As C_1 and C_2 belonging to $\mathcal{C}(\mathcal{I})$ are sets of attributes,

each one representing a set of counter-models, then if one of them contains a non barred attribute and the other contains the same attribute barred, they are *contradictory* (do not have counter-models in common). If C_1 and C_2 are not contradictory, then the *intersection of their counter-models* is represented by $C_1 \cup C_2$. For example, if $C_1 = \{a, b, \overline{c}\}$ and $C_2 = \{a, \overline{e}\}$, then $C_1 \cup C_2 = \{a, b, \overline{c}, \overline{e}\}$ represents the intersection of the counter-models represented by C_1 and C_2 (which is $\{\{a, b\}, \{a, b, d\}\}$). Thus, in preparation to apply IEP, we define:

$$C_1 \sqcap C_2 = \begin{cases} \emptyset, & \text{if } C_1 \text{ and } C_2 \text{ are contradictory;} \\ C_1 \cup C_2, & \text{otherwise.} \end{cases}$$

Now, making $C_1 = P_1 \cup \{\overline{m_1}\}, \ldots, C_n = P_n \cup \{\overline{m_n}\}$, the terms of IEP are: $S_1 = \sum_{i=1}^{n} N(C_i)$ and $S_k = \sum_{1 \leq i_1 < \cdots < i_k \leq n} N(C_{i_1} \sqcap \ldots \sqcap C_{i_k})$, where if C represents a set of counter-models, then $N(C) = 2^{|M|-|C|}$. And the number of counter-models of $\mathcal{I}$, represented by $\mathcal{C}(\mathcal{I})$, is $\sum_{i=1}^{n} (-1)^{n+1} S_n$.

Obviously, it is expected that a method of reduction entails a decrease in the information content. The greater the decrease, the higher the actual reduction with respect to the knowledge expressed by the set of implications. Only in very special situations such content could increase as, for example, in situations where there are few eliminations and a lot of generalizations.

In the worst case the calculation of the information content has cost $O(2^{|X|})$, where $X \subseteq M$ is the set of attributes referred to in the set of implications, and thus has limited applicability.

3.2 Fidelity and representativeness index

Modifications made by reduction techniques can produce inaccuracies in the set of implications that are observed when considering objects in an actual application or even objects of the original formal context. In other words, when considering an object $g \in G$, where G is the set of objects of the original formal context, an implication $P \rightarrow m \in \mathcal{I}_r$ can *fail*, i.e. it can be the case that $g \in P'$ and $g \notin m'$ (the derivation operator here is from the reduced formal context). An index that measures the success rate of the objects to implications in general is call *fidelity* (F), which can be defined as follows [14]:

$$F = \frac{\sum_{g \in G} |\overline{Ant^{\mathcal{I}}(g)} \cup Cons^{\mathcal{I}}(g)|}{|\mathcal{I}||G|}$$

where $\mathcal{I}$ is the actual set of implications (possibly after reduction), $Ant^{\mathcal{I}}(g) = \{P \rightarrow m \in \mathcal{I} | g \in P'\}$ and $Cons^{\mathcal{I}}(g) = \{P \rightarrow m \in \mathcal{I} | g \in m'\}$. Fidelity is nothing more than the success rate of implications in a set of implications $\mathcal{I}$ with respect to the objects in G, considering that an implication $P \rightarrow Q$ is successful for an object g if $g \notin P'$ or $g \in m'$.

Since only simplification techniques can create new knowledge (mainly generalizations), only such techniques tend to decrease the fidelity index, because the implications that express new knowledge can fail.

In some applications, the strength of a reduction in preserving what is relevant can be measured by the success rate of the implications considering only the objects that satisfy the lhs of the implications of the reduced formal context ($Ant^{\mathcal{I}}$). Thus, objects g satisfying an implication $P \to m$ just because $g \notin P'$ are not taken into account when calculating the *representativeness* index [15]:

$$\mathrm{R} = \frac{\sum_{g \in G} \frac{|Ant^{\mathcal{I}}(g) \cap Cons^{\mathcal{I}}(g)|}{|Ant^{\mathcal{I}}(g)|}}{|G|}$$

where in the case that $Ant^{\mathcal{I}}(g) = \emptyset$, $|Ant^{\mathcal{I}}(g) \cap Cons^{\mathcal{I}}(g)|/|Ant^{\mathcal{I}}(g)|$ must take value 0. Note that $|Ant^{\mathcal{I}}(g) \cap Cons^{\mathcal{I}}(g)|/|Ant^{\mathcal{I}}(g)|$, in the case where the formal context is effectively representative of the universe into consideration, can be seen as the probability that, for the object g, implications $P \to m$ has their rhs satisfied (that is, $g \in m'$), given that its lhs is satisfied (i.e., $g \in P'$). And thus representativeness is the average of these probabilities.

Example 2 *Let g be an object such that $g' = \{a, b, c\}$. Consider the implications $i = a \to c$, $j = a \to d$, $k = e \to c$ and $l = d \to e$. The implications i, k and l are successful for object g, and implication j fails for g. Thus for the object g three implications, i, k and l, contributes to increasing the index of fidelity, in the universe of all four implications; but only one implication, i, contributes to increasing the index of representativeness, in the universe of two implications, i and j.* □

The failure of an implication for a given object causes a decrease in both the *fidelity* and *representativeness* indexes. A further decrease in the representativeness index occurs when an object fails to give support to any implication. As a result representativeness is usually smaller than fidelity. Particularly, if an implication ceases to have support representativeness decreases, but not fidelity.

Unlike the calculation of the information content, the fidelity and representativeness indexes can be computed efficiently. Suppose k is the average size of the implications in $\mathcal{I}$. For each object, we need to go through all the implications. Therefore the fidelity and representativeness complexities are $O(k|\mathcal{I}||G|)$.

4 Examples of application

Among all reduction technique analyzed in [15,13], we selected *minimal set of attributes selection* [46], *grouping using the technique JBOS* [14] and *constraints learned from data* [6]. The selected techniques are representative of redundant information removal, simplification and selection classes, respectively [15,13]. Note that we selected techniques to exemplify some scenarios. Obviously, other techniques may be evaluated.

Consider the formal context in Table 1. The concept lattice has 24 formal concepts and it is depicted in Figure 1. The complete set of proper implications for the formal context, $\mathcal{I}_o$, has 126 proper implications. To simplify the visual analysis, only those implications with support greater than zero ($P \to m \in$

$\mathcal{J}|P \neq \oslash$ – total of 22), $sup(\mathcal{I}_o)$, are presented; they are shown in Table 2. As noted in the table, the information content of the complete set of proper implications is 0.976, slightly larger than the information content of the subset with support greater than zero, which is 0.947. Both sets $\mathcal{I}_o$ and $sup(\mathcal{I}_o)$ have fidelity (F) and representativeness (R) equal to 1 when considering the original formal context (Table 1). In the following sections $\mathcal{I}_o$ and $\mathcal{I}_r$ mean actually $sup(\mathcal{I}_o)$ and $sup(\mathcal{I}_r)$ respectively.

	Attributes									
Objects	a	b	c	d	e	f	g	h	i	j
1	x			x		x				x
2	x			x						x
3	x			x			x			x
4	x			x		x		x		x
5			x		x					
6		x			x			x	x	
7		x			x				x	
8		x			x			x	x	
9	x				x			x	x	
10			x		x			x	x	
11			x		x				x	
12		x		x		x		x		x

Table 1: Formal context.

Proper implications				
$\lvert\mathcal{I}_o\rvert = 126$				
$cont(\mathcal{I}_o) = 0.976$				
Proper implications with support - $\{P \to m \in \mathcal{J}\lvert P \neq \oslash\}$				
$g \to a$	$c \to e$	$a, e \to h$	$e, h \to i$	$g \to j$
$j \to d$	$d, h \to f$	$b, j \to h$	$b, e \to i$	$d \to j$
$g \to d$	$h, j \to f$	$b, d \to h$	$a, e \to i$	
$f \to d$	$b, j \to f$	$b, f \to h$	$c, h \to i$	
$i \to e$	$b, d \to f$	$a, i \to h$	$f \to j$	
$\lvert sup(\mathcal{I}_o)\rvert = 22$				
$cont(sup(\mathcal{I}_o)) = 0.947$				
$F = 1$				
$R = 1$				

Table 2: Proper implications.

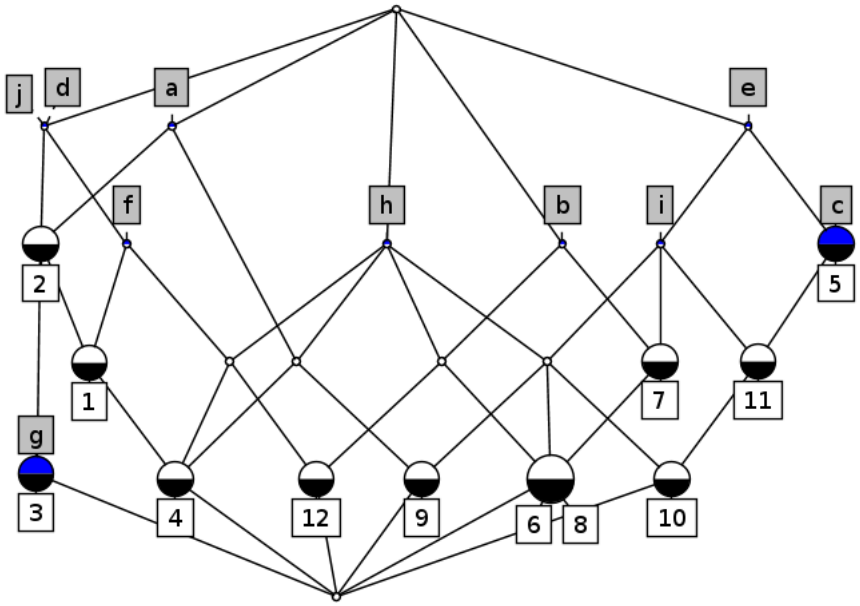


Fig. 1: Concept lattice.

4.1 Reduction technique: minimal set of attributes selection

Minimal sets can be obtained from identifying reducible attributes [19], using a discernibility matrix [46]. For example, consider a formal context (G, M, I) and concepts $(A, B), (C, D) \in \mathcal{B}(G, M, I)$. The *discernibility* between concepts (A, B) and (C, D) is given by the symmetric difference between B and D, i.e. $Dis_{(A,B),(C,D)} = (B \cup D) \backslash (B \cap D)$. Given the discernibility matrix Dis, a minimal set of attributes $X \subseteq M$ can be determined that results in a lattice isomorphic to the original (i.e., such that $\mathcal{B}(G, X, I')$ is isomorphic to $\mathcal{B}(G, M, I)$, where $I' = I \cap (G \times X)$) by making X meet the restriction: X is one of those sets Y of lowest cardinality such that $\forall Z \in Dis : Y \cap Z \neq \emptyset$ [46].

The concept lattice of Figure 1 has 2 minimal sets of attributes: $\{a, b, c, d, e, f, g, h, i\}$ and $\{a, b, c, e, f, g, h, i, j\}$. Note that, a, b, c, e, f, g, h and i are attributes *absolutely necessary*. On the other hand, d and j are attributes *relatively necessary* [46]. Note that d and j appear in same formal concept in Figure 1. Their corresponding lattices are isomorphic to the original lattice and the set of proper implications of each one has 97 implications. The fact that minimal sets lead to sets of proper implications with the same cardinality is justifiable because each minimal set forms the smallest closure system for the set of attributes. From the perspective of formal context, a minimal set is the smallest number of attributes able to characterize and differentiate the objects. In the concept lattice level, each formal concept will have the smallest number of attributes that defines it. So it has the fewest possible implications. When using minimal set selection, one

has a minimum knowledge representation model, which might be represented by a closure system, formal context, concept lattice or set of implications.

As the reduced lattice is isomorphic to the original, Table 4a shows that only the number of attributes was reduced in 10%. Table 2 shows that the information content of the 22 implications with support greater than zero is 0.947 while Table 4a shows that the information content of the 15 implications retained after reduction is 0.894. As expected the information content decreased after reduction, even the reduced lattice being isomorphic to the original concept lattice. This decrease is a consequence of cutting all the implications referring to the deleted attribute. The fidelity index has value $F = 1$ for both the original and reduced set of implications, because the implications of the latter are a subset of the former. On the other hand, the representativeness decreases from $R = 1$ to $R = 0.91$ due to the fact that no implication in $\mathcal{I}_r$ is supported by the object 2. The implications $d \rightarrow j$ and $j \rightarrow d$ which were supported by that object are no longer in the reduced set. There is no implication failure, but now there is an object that does not support any implication.

4.2 Reduction technique: grouping using the technique JBOS

JBOS (*junction based on objects similarity*) seeks to replace groups of objects considered similar by objects that represent them, based on a qualitative assessment of its attributes [14]. Objects considered similar to others with a threshold ϵ form groups with a maximum of α objects. The similarity between g and h ($g, h \in G$) is given by the weighted sum of the weights of attributes in which both objects agree with each other (both having them or both not having them). Two objects g and h are considered similar by the algorithm, and therefore are likely to stay in the same group if and only if $sim(g, h) \geq \epsilon$.

Assuming that the weight of an attribute is proportional to the frequency of its occurrence in objects, $\epsilon = 0.6$ and $\alpha = 3$[6], the set of objects of the formal context of Table 1 forms five groups: $\{1, 2, 3\}$, $\{4, 12\}$, $\{5, 11, 10\}$, $\{6, 7, 8\}$, $\{9\}$. The set of attributes of each object representative of a group H is $\bigcap\{g' \mid g \in H\}$ [14]. Objects of the same group are now indistinguishable in the new lattice, in the sense that they have exactly the same attributes.

Table 4b shows that when compared to the original lattice the number of objects of the latter was reduced by 58% and the number of formal concepts has been cut in half. Despite the substantial reduction of objects and concepts, the number of attributes remained the same. The information content decreased (from 0.947 to 0.923) indicating that there was a greater reduction due to elimination of implications. The decrease in fidelity from 1 to 0.94 and in representativeness from 1 to 0.83 are due to the implications of $\mathcal{I}_r$ which fails for some objects. As an example, the implication $a, h \rightarrow i \in \mathcal{I}_r$ fails for the object 4 (the object 4 has attributes a and h, but does not have attribute i). The difference in the indexes reflects the way in which both indexes are computed.

[6] Since the formal context of Table 1 is relatively small, the choices of ϵ and α were made in order to achieve around 40% of reduction on the set of formal concepts.

4.3 Reduction technique: constraint learned from data

A kind of selection technique that does not have concomitant access to all knowledge is the selection of formal concepts based on constraints enforced during the construction of the concept lattice [6,40]. In addition to restrictions, it is often also used some background knowledge to guide the selection process. In [6] a method of imposing constraints in extracting formal concepts was discussed. The authors presents various kinds of restrictions, among them *constraint learned from data*. The data are supplied by means of a formal context (G, M_o, I), where M_o are the attributes of the original context (G_o, M_o, I_o). That formal context can be a sample of the original data or a set artificially constructed by an expert. The intended application is interested only in formal concepts from (G_o, M_o, I_o) whose intentions are also intentions of (G, M_o, I). Table 3 shows an example of a formal context with four new objects (13, 14, 15 and 16) to be used in conjunction with the formal context of Table 1.

	a	b	c	d	e	f	g	h	i	j
13	x			x		x				x
14	x			x		x		x		x
15		x			x			x	x	
16	x				x			x	x	

Table 3: Background knowledge (G, M_o, I).

Using the background knowledge (G, M_o, I) from Table 3, the constraint learned from data technique select only the formal concepts from (G_o, M_o, I_o) whose intentions are also intentions of (G, M_o, I). The following formal concepts are selected:

- $(\{3, 4, 6, 8, 9, 10\}, \{h\})$
- $(\{1, 2, 3, 4, 9\}, \{a\})$
- $(\{4, 9\}, \{a, h\})$
- $(\{6, 8, 9, 10\}, \{e, h, i\})$
- $(\{4\}, \{a, d, f, h, j\})$
- $(\{9\}, \{a, e, h, i\})$
- $(\{6, 8\}, \{b, e, h, i\})$
- $(\{1, 4\}, \{a, d, f, j\})$

Table 4c shows that there is a reduction of 63% in the number of formal concepts, 25% in the number of objects and 20% in the number of attributes. There is a high elimination of implications and the information content drops to 0.554. Fidelity remains equal to 1. However, there is a high drop to 0.5 in representativeness. This is a consequence of the selected formal concepts not having some objects in any of its extensions: objects $2, 3, 5, 7, 11$ and 12 are not present in any extension of the selected concepts.

Indexes	Reduction
$cont(\mathcal{I}_r) = 0.894$	Objects: 0%
$F = 1$	Attributes: 10%
$R = 0.91$	Concepts: 0%

(a) Minimal set of attributes.

Indexes	Reduction
$cont(\mathcal{I}_r) = 0.923$	Objects: 58%
$F = 0.94$	Attributes: 0%
$R = 0.83$	Concepts: 50%

(b) Object grouping.

Indexes	Reduction
$cont(\mathcal{I}_r) = 0.554$	Objects: 25%
$F = 1$	Attributes: 20%
$R = 0.5$	Concepts: 63%

(c) Selection of concepts.

Table 4: Indexes and reduction.

5 Conclusions and future research

As the complexity of concept lattices is considered a key problem for an effective use of FCA in many situations, several techniques, with different characteristics, have been proposed for concept lattice reduction. Note that in real life, one deals with much unnecessary data and that reduction is vital in order to disregard it. Moreover, besides being computationally expensive to compute all formal concepts, their number and interrelationships can impair an effective analysis in a reasonable amount of time.

Some works apply specific indexes to evaluate their techniques for concept lattice reduction. However, those objective measures and criteria are generally used only in the selection of formal concepts.

In this work we discuss three indexes to evaluate reduced concept lattice, they are: information content, fidelity and representativeness. The fist one, information content index, measure the information associated with a logical expression based on the ratio between the number of truth assignments that make the logical expression false (or true) and the total number of possible truth assignments. The second one, fidelity index, measure the quality of a reduction would be the success rate when the rules derived from a reduced formal context (or lattice) are applied to the original formal context objects. Finally, the representativeness index, takes the success rate of implications considering only implications with satisfied left hand sides. The three indexes are complementary and using proper implications extracted from the original and reduced concept lattice evaluate all lattice structure. Besides, proposed indexes are independent of the application one have in mind.

The indexes are applied in three reduction techniques by means of small example. Every example leads to a decrease in the information content index; only the JBOS leads to a loss in the fidelity index, as a consequence of the production of some failing implications; and all examples lead to loss in the representativeness index. The results shown the characteristics of each lattice after be reduced.

Existing measures and criteria, proposed here and found in literature, do not identify what aspects of knowledge get preserved, eliminated, inserted or transformed by a reduction technique. As future works we intend to propose a way to identify these aspects. Finally, as the information content has limited applicability due the complexity, we would like to improve the index and present a new algorithm to calculate the number of counter-models.

Acknowledgments

The first author would like to thank the partial support of the Federal Service of Data Processing - `www.serpro.gov.br`. We would also acknowledge the financial support received from the Foundation for Research Support of Minas Gerais state, FAPEMIG; the National Council for Scientific and Technological Development, CNPq; Coordination for the Improvement of Higher Education Personnel, CAPES.

References

1. Bar-Hillel, Y., Carnap, R.: Semantic information. The British Journal for the Philosophy of Science 4(14), 147–157 (1953)
2. Belohlavek, R., Baets, B., Konecny, J.: Zoom-In/Zoom-Out Algorithms for FCA with Attribute Granularity. In: Computer and Information Sciences II, pp. 549–555. Springer London (2012)
3. Belohlavek, R., Macko, J.: Selecting important concepts using weights. In: Proceedings of international conference on Formal concept analysis - ICFCA. pp. 65–80. Springer-Verlag, Berlin, Heidelberg (2011)
4. Belohlavek, R., Outrata, J., Trnecka, M.: Impact of boolean factorization as preprocessing methods for classification of boolean data. Annals of Mathematics and Artificial Intelligence 72(1-2), 3–22 (2014)
5. Belohlavek, R., Vychodil, V.: Formal concept analysis with background knowledge: attribute priorities. Trans. Sys. Man Cyber Part C 39(4), 399–409 (2009)
6. Belohlavek, R., Vychodil, V.: Closure-based constraints in formal concept analysis. Discrete Applied Mathematics 161(13–14), 1894 – 1911 (2013)
7. Bertet, K., Monjardet, B.: The multiple facets of the canonical direct unit implicational basis. Theoretical Computer Science 411(22–24), 2155 – 2166 (2010)
8. Bertet, K.: The dependence Graph of a Lattice. In: International Conference on Concept Lattices and their Applications - CLA. vol. 972, pp. 223–232. CEUR-WS.org (2012)
9. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Scalable Estimates of Concept Stability. In: Proceedings of international conference on Formal concept analysis - ICFCA. pp. 157–172 (2014)
10. Ch., A.K., Dias, S.M., Vieira, N.J.: Knowledge reduction in formal contexts using non-negative matrix factorization. Mathematics and Computers in Simulation 109(0), 46–63 (2015)
11. Chen, J., Li, J., Lin, Y., Lin, G., Ma, Z.: Relations of reduction between covering generalized rough sets and concept lattices. Information Sciences 304, 16–27 (2015)

12. Davey, B., Priestley, H.: Introduction to lattices and order. Cambridge University Press, Cambridge, England (1990)
13. Dias, S.M.: Redução de Reticulados Conceituais (Concept Lattice Reduction). Ph.D. thesis, Department of Computer Science, Federal University of Minas Gerais (UFMG), Belo Horizonte, Minas Gerais, Brazil (5 2016), in Portuguese
14. Dias, S.M., Vieira, N.J.: Reducing the Size of Concept Lattices: The JBOS Approach. In: Proceeding of the International Conference on Concept Lattices and their Applications - CLA. vol. 672, pp. 80–91. Seville, Spain (2010)
15. Dias, S.M., Vieira, N.J.: Concept lattices reduction: definition, analysis and classification. Expert Systems with Applications 42(20), 7084–7097 (2015)
16. Düntsch, I., Gediga, G.: Simplifying contextual structures. In: Pattern Recognition and Machine Intelligence, pp. 23–32. Springer (2015)
17. Elloumi, S., Ferjani, F., Jaoua, A.: Using minimal generators for composite isolated point extraction and conceptual binary relation coverage: Application for extracting relevant textual features. Information Sciences 336, 129 – 144 (2016)
18. Formica, A.: Concept similarity in Formal Concept Analysis: An information content approach. Knowledge-Based Systems 21(1), 80 – 87 (2008)
19. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag, Germany (1999)
20. Gély, A.: Links between modular decomposition of concept lattice and bimodular decomposition of a context. In: Proceedings of international Conference on Concept Lattices and their Applications - CLA. pp. 393–403. Nancy, France (Oct 2011)
21. Grünwald, P.D.: The minimum description length principle. MIT press (2007)
22. Jota Resende, G., De Moraes, N., Dias, S., Marques Neto, H., Zárate, L.: Canonical Computational Models Based on Formal Concept Analysis for Social Network Analysis and Representation. In: International Conference on Web Services (ICWS) - IEEE. pp. 717–720 (June 2015)
23. Kardoš, F., Pócs, J., Pócsová, J.: On concept reduction based on some graph properties. Knowledge-Based Systems 93, 67–74 (2016)
24. Kauer, M., Krupka, M.: Subset-generated complete sublattices as concept lattices. Proceeding of the International Conference on Concept Lattices and their Applications - CLA pp. 11–22 (2015)
25. Klimushkin, M., Obiedkov, S., Roth, C.: Approaches to the Selection of Relevant Concepts in the Case of Noisy Data. In: Proceedings of international conference on Formal concept analysis - ICFCA. pp. 255–266. Springer Berlin, Heidelberg (2010)
26. Kuznetsov, S.: On stability of a formal concept. Annals of Mathematics and Artificial Intelligence 49, 101–115 (2007)
27. Kuznetsov, S.O., Makhalova, T.: On interestingness measures of formal concepts. arXiv preprint arXiv:1611.02646 (2016)
28. Kuznetsov, S.O., Makhalova, T.P.: Concept interestingness measures: a comparative study. Proceeding of the International Conference on Concept Lattices and their Applications - CLA pp. 59–71 (2015)
29. Kuznetsov, S.O., Poelmans, J.: Knowledge representation and processing with formal concept analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3(3), 200–215 (2013)
30. Li, J., Mei, C., Xu, W., Qian, Y.: Concept learning via granular computing: A cognitive viewpoint. Information Sciences 298, 447 – 467 (2015)
31. Medina, J.: Relating attribute reduction in formal, object-oriented and property-oriented concept lattices. Computers & Mathematics with Applications 64(6), 1992 – 2002 (2012)

32. Mouakher, A., Yahia, S.B.: QualityCover: Efficient binary relation coverage guided by induced knowledge quality. Information Sciences 355–356, 58 – 73 (2016)
33. Neto, S.M., Zárate, L.E., , M.A.J.S., Dias, S.M.: Minimal Cover of Implication Rules to Represent Two Mode Networks. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 211–218 (2015)
34. Pei, D., Mi, J.S.: Attribute reduction in decision formal context based on homomorphism. International Journal of Machine Learning and Cybernetics 2, 289–293 (2011)
35. Poelmans, J., Kuznetsov, S.O., Ignatov, D.I., Dedene, G.: Formal Concept Analysis in Knowledge Processing: a Survey on Models and Techniques. Expert Systems with Applications 40(16), 6601 – 6623 (2013)
36. Rice, M.D., Siff, M.: Clusters, Concepts, and Pseudometrics. Electronic Notes in Theoretical Computer Science 40(0), 323 – 346 (2001)
37. Roth, C., Obiedkov, S., Kourie, D.G.: Towards concise representation for taxonomies of epistemic communities. In: Proceeding of the International Conference on Concept Lattices and their Applications - CLA. pp. 205–218 (2006)
38. Ryssel, U., Distel, F., Borchmann, D.: Fast algorithms for implication bases and attribute exploration using proper premises. Annals of Mathematics and Artificial Intelligence 70(1-2), 25–53 (2014)
39. Snásel, V., Dahwa Abdulla, H., Polovincak, M.: Behavior of the Concept Lattice Reduction to visualizing data after Using Matrix Decompositions. In: International Conference on Innovations in Information Technology. pp. 392 –396 (nov 2007)
40. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with Titanic. Data and Knowledge Eng. 42, 189–222(34) (2002)
41. Taouil, R., Bastide, Y.: Computing Proper Implications. In: Proceedings of the International Conference on Conceptual Structures - ICCS. pp. 46–61. Stanford, CA US (Jul 2001)
42. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. I. Rival (Ed.): Ordered Sets pp. 445–470 (1982)
43. Yao, W., Han, S.E., Wang, R.: Lattice-theoretic contexts and their concept lattices via Galois ideals. Information Sciences 339, 1–18 (2016)
44. Zárate, L.E., Dias, S.M., Song, M.A.J.: FCANN: A new approach for extraction and representation of knowledge from ANN trained via Formal Concept Analysis. Neurocomputing 71, 2670–2684 (2008)
45. Zhang, S., Guo, P., Zhang, J., Wang, X., Pedrycz, W.: A completeness analysis of frequent weighted concept lattices and their algebraic properties. Data & Knowledge Engineering 81-82(0), 104 – 117 (2012)
46. Zhang, W., Wei, L., Qi, J.: Attribute reduction theory and approach to concept lattice. Science in China Series F: Information Sciences 48, 713–726 (2005)

Words Grow on Lattices: Analysing Sequences through FCA

Chris Nowak, Daniel Salmond, and Raymee Chau

Defence Science & Technology Group, Department of Defence, Australia
{chris.nowak, daniel.salmond, raymee.chau}@dsto.defence.gov.au

Abstract. This paper explores a novel approach to automatically discovering concepts from patterns in abstract strings, without reliance on prior knowledge of syntax or semantics.
Analysing sequences allows us to detect and predict patterns. When sequences take the form of streaming text, patterns take the form of words. Given text streams, recurring sequences can be detected, formal contexts formed, and FCA concepts and concept lattices generated.
Processing a text stream results in generating dynamically changing concept lattices representing, conceptually, the stream's initial segments of increasing length. Detected words become FCA attributes and they also dynamically change—they grow on the lattices.

1 Introduction

The act of communication is a process of generating and consuming streamed information. Streamed information can be characterised in terms of a sequence of symbols. For example, at the lowest resolution of a digital communications system, information is encoded as a sequence of bits whether stored in memory or transmitted via a digital waveform.

Low level sequences can be organised into sequences of higher order objects, such as packets or frames. In a non-cooperative communications channel, i.e. one in which the actor has limited visibility of the communications protocols, an actor may need to form its own higher abstractions of the observed sequence. Higher order abstractions allow more powerful processing such as predicting the behaviour of a communications channel so that a cognitive radio can anticipate segments of clear channel in which to opportunistically transmit.

Natural language is a valuable surrogate for more esoteric forms of communication as it contains protocol structure and implicit redundancy in order to make the act of communication resilient to noise and other forms of interference. Moreover, natural language and digital communications waveforms both have the goal of conveying information. For this reason, this paper shall consider the challenge of extracting higher order abstractions in the form of recurring sequences from streaming text as an analogue for extracting protocol structure from communications waveforms. Furthermore, the paper describes a method inspired by Formal Concept Analysis of representing these higher order abstractions as an information system that captures the structural relationships between the abstractions.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

2 Related Work

Sequence prediction is a well-studied problem. For example, recent innovations in Deep Learning [24] has been motivated by the demand for natural language processing, handwriting recognition and language translation, among other things. Each of these tasks can be formulated as a sequence prediction activity in which a sequence of inputs is ingest, such as speech, handwriting and language, and the prediction engine must generate (predict) the corresponding output. In cognitive radio, the challenge may be described more simply as ingesting channel state information and predicting channel state occupancy. Recurrent Neural Networks (RNNs) and specifically those implementing Long Short-Term Memory [13] have been demonstrated as being especially successful in handwriting recognition [9], speech-to-text processing [10] and language translation [28, 1].

Deep neural networks are neural networks with many hidden layers of neurons such that successive layers are activated by their previous layers. These networks implicitly develop models of the data upon which they are trained. Upon inspection of the hidden layers, it has been found that successive layers exhibit increasingly higher levels of abstraction such that there exists a hierarchy of abstraction. The authors regard these abstractions as an organically learned representation of the latent 'concepts' within the environment to which the deep neural networks are exposed. For example, the application of deep neural networks to facial recognition has revealed that lower layers learn to detect edges whereas higher layers detect objects such as eyes and noses [27, 23]. Alternatively, the adoption of *embedding* techniques [18, 20] has shown that deep neural network can also learn semantic relationships between inputs such that they produce a semantic map which can be interrogated.

The disadvantage of deep neural networks is that they require significant quantities of training data and careful tuning of hyper-parameters to achieve state-of-the-art performance. By contrast this paper is concerned with processing an online sequence of symbols, i.e. in the absence of training data, and extracting concepts on-the-fly.

Clustering algorithms provide an alternative method of recognising latent 'concepts' within data. There are many clustering algorithms,[1] including Principal Component Analysis, Independent Component Analysis, k-means clustering, and non-parametric Bayesian methods to name a few. For an exhaustive review the reader is directed to [14, 2].

One clustering approach of note is the application of Hierarchical Dirichlet Process Hidden Semi-Markov Models (HDP-HSMM) techniques [15] to modelling communications channels [16]. The HDP-HSMM is a non-parametric Bayesian method in which the sequential data is clustered into classes and a Hidden Semi-Markov Model (HSMM) is found that fits the transitions between the classes. The algorithm iteratively refines the number of classes and HSMM that best fits the data according to a fitting criterion. Although this approach

[1] One of the reviewers pointed out that PCA and ICA are unsupervised dimensionality reduction techniques, rather than clustering techniques.

may be regarded as extracting both concepts (i.e. clusters) and relationships from the sequential data, it suffers from the criticism levelled at many machine learning techniques, including deep neural networks, that the rationale for selecting concepts over others may not be transparent to the human observer. By contrast the Formal Concepts Analysis approach applies a logical framework to building a concept lattice to recurring objects within sequential data.

Searching for substrings in large strings could be seen as *external* to the FCA concept lattice processing. Nevertheless, such references as [5] (unsupervised learning of behaviours), [6] (sequence segmenting), [12] (unsupervised iterative Bayesian word segmenting), [11] (unsupervised morphology learning), [30] (sequence segmenting using Burrows-Wheeeler Transform), [22] (FCA linguistic processing), [29] (text segmenting, topic detection), [4] (concept stability), [7] (FCA substring scales) and [25] (pattern mining; change detection in data streams) are worth exploring. The reviewers suggested some other related work, including [8] (pattern structures), [3] (sequential data mining), [26] (discourse structures), [21] (FCA based information retrieval) and [17] (incremental computation of concept lattices).

3 Processing a sentence

One could select a *string,* assume that the string is a *sentence* and that it consists of *words*. There are sentences that are not good choices for such an exercise. For instance, the string (written in upper case, with blank spaces separating words) `THE QUICK BROWN FOX JUMPS OVER A LAZY DOG` is a (short) *pangram* and therefore its words, unsurprisingly, do not reoccur. Hence, if all we were given was the string: `THEQUICKBROWNFOXJUMPSOVERALAZYDOG` then the words would be impossible to detect (in fact, although some single characters do reoccur, no doubleton string reoccurs). However, it is not hard to come up with a sentence with recurring[2] words. Consider the following string:

`WHATSHECANDOHECANNOTDOANDWHATHECANDOSHECANNOTDO`

and note that a human reader could easily recognise and detect some words and guess that the (corresponding) sentence is: `WHAT SHE CAN DO HE CANNOT DO AND WHAT HE CAN DO SHE CANNOT DO`. But what words could a computer program detect, if all it was given was the string?

We refer to `WHATSHECANDOHECANNOTDOANDWHATHECANDOSHECANNOTDO` using `WHAT`ω`SHECANNOTDO`, with ω standing for `SHECANDOHECANNOTDOANDWHATHECANDO`, and therefore the original string is `WHAT` concatenated with ω concatenated with `SHECANNOTDO`. We will process `WHAT`ω`SHECANNOTDO` to see how words can be detected. There are multiple ways one can detect words in a string, where a word is understood as a substring that reoccurs. However, we will attempt to use Formal Concept Analysis (FCA) and therefore try to associate the string with FCA *objects* and *attributes*. The following stages of processing are employed:

[2] We use the verb *recur* as meaning *reoccur* (multiple times), but without expecting regular intervals between the word occurrences.

1. finding *reoccurring strings*;
2. choosing *objects* (and *attributes*);
3. forming *contexts*;
4. calculating *concepts* and *concept lattices*.

The above stages of processing are described in Sections 3.1–3.4. We have written a python program that takes `WHATωSHECANNOTDO` as its input and generates outputs depicted in Tables 1–4 (of Sections 3.1–3.3); hence, the python program generates *FCA contexts*. Then, the generated FCA contexts are used as inputs to FCA concept lattice calculation software available from `latviz.loria.fr`; the outputs—calculated *FCA concepts* and *FCA concept lattices*—are depicted in Table 5 and Figures 1–2 (of Section 3.4).

3.1 Finding recurring strings

When processing the stream of symbols `WHATωSHECANNOTDO`, the string `W` is processed, then the string `WH`, and so on, until the whole string is processed. To demonstrate results of the processing, we will consider the string of length 47, i.e., the whole string `WHATSHECANDOHECANNOTDOANDWHATHECANDOSHECANNOTDO` (to which we refer using `WHATωSHECANNOTDO`); shorter strings, such as an initial fragment of length 29, i.e., the string `WHATSHECANDOHECANNOTDOANDWHAT`, could also be considered. When the whole string `WHATωSHECANNOTDO` is considered, the search for recurring substrings is depicted in Table 1.

Table 1 depicts processing the string `WHATωSHECANNOTDO` by indicating recurring substrings with $+$ and $*$ signs at cells of a 47×47 matrix with rows and columns numbered $0, \ldots, 46$; the diagonal of the matrix and the cells above the diagonal allow us to represent whether the corresponding substrings reoccur.[3] If a substring starting at position i and ending at position j (where $j \geq i$) reoccurs then the corresponding cell is marked with $+$, or with $*$ if this is the first occurrence of the substring. For instance, a $*$ at row 0 and column 3 indicates that the string starting at 0 and ending at 3 (i.e., the substring `WHAT`) reoccurs and $(0, 3)$ is the first occurrence of the substring; there is another occurrence of the substring at $(25, 28)$, marked with $+$ at the corresponding cell.

All recurring substrings are listed in Table 2 in a form of a list, where each element of the list is a pair: `[string, list of the string's occurrences]`. For instance, the first element of the list is `['WHAT',[(0,3),(25,28)]]`, indicating that the string `WHAT` occurs at `(0,3)` and at `(25,28)`.

Given the string `WHATSHECANDOHECANNOTDOANDWHATHECANDOSHECANNOTDO` as input, the python program `fromStringToWordsAndOccurrences.py` (listed below) produces the output presented in Table 2. Note that Table 2 contains exactly the same information as Table 1, with Table 1 providing a simple visualisation of processing the input stream, and Table 2 simply listing recurring *words*—and their *occurrences*.

[3] For related string processing, cf. Lempel-Ziv-Welch compression algorithm dictionaries [31] and maximal substrings in suffix trees [7].

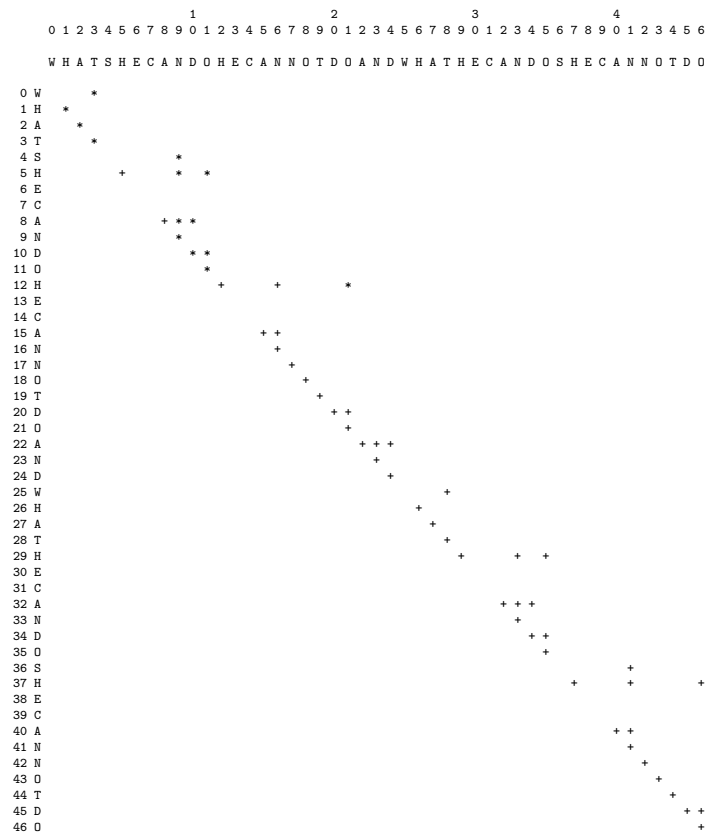


Table 1. WHATωSHECANNOTDO: recurring substrings.

Code for finding words and their occurrences

```
# fromStringToWordsAndOccurrences.py
def replace(prev,new,all):
  newAll = []
  for pointerToItemInAll in range (0,len(all)):
    current = all[pointerToItemInAll]
    if  not(current == prev):
      newAll.append(current)
    else:
      newAll.append(new)
  all = newAll
  return all
def coversOnItems(occurrence1,occurrence2):
  occurrence1beg = occurrence1[0]
  occurrence2beg = occurrence2[0]
  occurrence1end = occurrence1[1]
  occurrence2end = occurrence2[1]
  if (occurrence1beg <= occurrence2beg) and \
     (occurrence2end <= occurrence1end):
    return True
  else:
    return False
```

```
['WHAT',      [( 0, 3),(25,28)]]
['H',         [( 1, 1),( 5, 5),(12,12),(26,26),(29,29),(37,37)]]
['A',         [( 2, 2),( 8, 8),(15,15),(22,22),(27,27),(32,32),(40,40)]]
['T',         [( 3, 3),(19,19),(28,28),(44,44)]]
['SHECAN',    [( 4, 9),(36,41)]]
['HECAN',     [( 5, 9),(12,16),(29,33),(37,41)]]
['HECANDO',   [( 5,11),(29,35)]]
['AN',        [( 8, 9),(15,16),(22,23),(32,33),(40,41)]]
['AND',       [( 8,10),(22,24),(32,34)]]
['N',         [( 9, 9),(16,16),(17,17),(23,23),(33,33),(41,41),(42,42)]]
['D',         [(10,10),(20,20),(24,24),(34,34),(45,45)]]
['DO',        [(10,11),(20,21),(34,35),(45,46)]]
['O',         [(11,11),(18,18),(21,21),(35,35),(43,43),(46,46)]]
['HECANNOTDO',[(12,21),(37,46)]]
```

Table 2. WHATωSHECANNOTDO: *words*—and their *occurrences.*

```
['WHAT',      [( 0, 3),(25,28)]]
['H',         [( 0, 3),( 4, 9),( 5,11),(12,21),(25,28),(29,35),(36,41),(37,46)]]
['A',         [( 0, 3),( 4, 9),( 5,11),(12,21),(22,24),(25,28),(29,35),(36,41),(37,46)]]
['T',         [( 0, 3),(12,21),(25,28),(37,46)]]
['SHECAN',    [( 4, 9),(36,41)]]
['HECAN',     [( 4, 9),( 5,11),(12,21),(29,35),(36,41),(37,46)]]
['HECANDO',   [( 5,11),(29,35)]]
['AN',        [( 4, 9),( 5,11),(12,21),(22,24),(29,35),(36,41),(37,46)]]
['AND',       [( 5,11),(22,24),(29,35)]]
['N',         [( 4, 9),( 5,11),(12,21),(22,24),(29,35),(36,41),(37,46)]]
['D',         [( 5,11),(12,21),(22,24),(29,35),(37,46)]]
['DO',        [( 5,11),(12,21),(29,35),(37,46)]]
['O',         [( 5,11),(12,21),(29,35),(37,46)]]
['HECANNOTDO',[(12,21),(37,46)]]
```

Table 3. WHATωSHECANNOTDO: *attributes*—and their *objects.*

```
def coversOnLists(occurrences1,occurrences2):
  allOccurrences2areCovered = True
  for ptrToOccurr2 in range(0,len(occurrences2)):
    occurrToBeCovered = occurrences2[ptrToOccurr2]
    occurrIsCovered = False
    for ptrToOccurr1 in range(0,len(occurrences1)):
      occurrToBeUsedAsCover = occurrences1[ptrToOccurr1]
      isCovered = coversOnItems(occurrToBeUsedAsCover,occurrToBeCovered)
      if isCovered:
        occurrIsCovered = True
        break
    if not(occurrIsCovered):
      allOccurrences2areCovered = False
      break
  return allOccurrences2areCovered
def absorbs(D1,D2):
  string1 = D1[0];  occurrences1 = D1[1]
  string2 = D2[0];  occurrences2 = D2[1]
  if string2 in string1 and coversOnLists(occurrences1,occurrences2):
    return True
  else:
    return False
def occurrences(substring,string):
    occurrences = []
    for pointer in range(0,len(string)-len(substring)+1):
        pointedSubstring = string[pointer:pointer+len(substring)]
        if pointedSubstring == substring:
            occurrences.append((pointer,pointer+len(substring)-1))
    return occurrences
```

```
def main():
  listOfReoccuringSubstrings = []
  string = "WHATSHECANDOHECANNOTDOANDWHATHECANDOSHECANNOTDO"
  for row in range(0,len(string),1):
    for col in range(row,len(string),1):
      substring = string[row:col+1]
      occurrencesOfSubstring = occurrences(substring,string)
      numberOfOccurrences = len(occurrencesOfSubstring)
      if numberOfOccurrences == 1:
        break
      else:
        substringAndItsOccurrences = []
        substringAndItsOccurrences.append(substring)
        substringAndItsOccurrences.append(occurrencesOfSubstring)
        if listOfReoccuringSubstrings == []:
          listOfReoccuringSubstrings.append(substringAndItsOccurrences)
        else:
          sizeOfListOfReoccuringSubstrings = len(listOfReoccuringSubstrings)
          for pointerToSubstringAndOccurrences in range(0,sizeOfListOfReoccuringSubstrings):
            newSubstringAndOccurrences = substringAndItsOccurrences
            prevSubstringAndOccurrences = \
              listOfReoccuringSubstrings[pointerToSubstringAndOccurrences]
            allSubstringsAndOccurrences = listOfReoccuringSubstrings
            newDropped = False
            newReplacedPrev = False
            newToBeAppended = False
            if absorbs(prevSubstringAndOccurrences,newSubstringAndOccurrences):
              newDropped = True
            if absorbs(newSubstringAndOccurrences,prevSubstringAndOccurrences):
              allSubstringsAndOccurrences = replace(prevSubstringAndOccurrences,\
                newSubstringAndOccurrences,allSubstringsAndOccurrences)
              newReplacedPrev = True
            if newDropped:
              break
            if newReplacedPrev:
              break
            if pointerToSubstringAndOccurrences == sizeOfListOfReoccuringSubstrings - 1:
              newToBeAppended = True
          if newToBeAppended:
            allSubstringsAndOccurrences.append(newSubstringAndOccurrences)
          listOfReoccuringSubstrings = allSubstringsAndOccurrences
  print "string = ", string
  print "listOfReoccuringSubstrings = ", listOfReoccuringSubstrings
main()
```

3.2 Choosing objects

Table 2 connects *(abstract) strings* (such as `WHAT`) with their specific occurrences (such as `(0,3)` and `(25,28)`), at specific locations in the whole string. One could think that those *abstract strings* are *words* and could be employed as *FCA attributes*; then one could also think that the specific occurrences (of the *words*, or *attributes*) could be employed as *FCA objects*. But then one could immediately see that the occurrences listed in Table 2 are all *different*: they are *not shared!*

The approach just described would yield trivial, uninteresting FCA contexts and concept lattices. Fortunatelly, there is a simple solution: to use only *maximal occurrences* (maximal w.r.t. the substring relation). Table 2 is therefore modified by replacing the occurrences with maximal occurrences containing them; the result of the modification is shown in Table 3. The *maximal occurrences* listed in Table 3 are used as *FCA objects*. For instance, the occurrence `(5,9)` (of `HECAN`) will not be used as an object associated with (word, or attribute) `HECAN`;

	HECANNOTDO	HECANDO	SHECAN	HECAN	WHAT	AND	AN	DO	A	N	H	O	D	T
(12,21) = 12HECANNOTDO	×			×			×	×	×	×	×	×	×	×
(37,46) = 37HECANNOTDO	×			×			×	×	×	×	×	×	×	×
(5,11) = 05HECANDO		×		×		×	×	×	×	×	×	×	×	
(29,35) = 29HECANDO		×		×		×	×	×	×	×	×	×	×	
(4, 9) = 04SHECAN			×	×			×		×	×	×			
(36,41) = 36SHECAN			×	×			×		×	×	×			
(0, 3) = 00WHAT					×				×		×			×
(25,28) = 25WHAT					×				×		×			×
(22,24) = 22AND						×	×		×	×			×	

Table 4. WHATωSHECANNOTDO: formal context.

instead, the maximal occurrences, namely the occurrence (4,9) (of SHECAN) and the occurrence (5,11) (of HECANDO) will be used as objects. The *abstract strings* (such as WHAT, without a reference to its *location*) are *words* and are employed as *FCA attributes*; the maximal occurrences are employed as *FCA objects*. Hence, Table 3 lists *attributes*—and their *objects*. Note that Table 3 contains exactly the same information as Table 4.

3.3 Forming contexts

At the end of Section 3.2, we said that Table 3 lists *attributes* and their *objects*. Therefore, Table 3 specifies a *formal context*. This formal context is presented, in a form of a *cross-table*, in Table 4. In the context, the objects have been renamed (to have easier to "understand" object names): we refer to the objects that WHAT has as 00WHAT (occurrence of WHAT starting, within the whole string, at symbol number 0) and 25WHAT (occurrence of WHAT starting at symbol number 25)—rather than (0,3) and (25,28).

Finding reoccurring strings (Section 3.1), choosing objects (Section 3.2), and forming contexts (described in this section) have all been implemented in a python program. The obtained formal contexts have been used to calculate *formal concepts* and *concept lattices*; this is described in Section 3.4.

```
["05HECANDO","29HECANDO","22AND","12HECANNOTDO","37HECANNOTDO","04SHECAN","36SHECAN"]
["AN","N","A"]

["12HECANNOTDO","37HECANNOTDO"]
["H","AN","N","HECANNOTDO","T","A","D","DO","O","HECAN"]

["12HECANNOTDO","37HECANNOTDO","00W","25W"]
["H","T","A"]

["12HECANNOTDO","37HECANNOTDO","00W","25W","05HECANDO","29HECANDO","22AND","04SHECAN","36SHECAN"]
["A"]

["05HECANDO","29HECANDO","22AND","12HECANNOTDO","37HECANNOTDO"]
["AN","N","A","D"]

["00W","25W"]
["H","T","A","WHAT"]

["05HECANDO","29HECANDO","12HECANNOTDO","37HECANNOTDO"]
["H","AN","N","A","D","DO","O","HECAN"]

["05HECANDO","29HECANDO","12HECANNOTDO","37HECANNOTDO","04SHECAN","36SHECAN"]
["H","AN","N","A","HECAN"]

["05HECANDO","29HECANDO","12HECANNOTDO","37HECANNOTDO","04SHECAN","36SHECAN","00W","25W"]
["H","A"]

["05HECANDO","29HECANDO"]
["H","HECANDO","AN","N","AND","A","D","DO","O","HECAN"]

["05HECANDO","29HECANDO","22AND"]
["AN","N","AND","A","D"]

[]
["AN","N","HECANNOTDO","T","A","D","WHAT","DO","O","HECAN","H","HECANDO","AND","SHECAN"]

["04SHECAN","36SHECAN"]
["H","AN","N","SHECAN","A","HECAN"]
```

Table 5. WHATωSHECANNOTDO: concepts.

3.4 Calculating concepts and drawing concept lattices

The *formal contexts* have been described in Section 3.3; they can be used to calculate *formal concepts* and *concept lattices*. The formal contexts have been used as input to FCA lattice calculating software available from `latviz.loria.fr`.

When the context presented in Table 4 is used as input, the concepts presented in Table 5 are obtained. The *concept lattice* (built from the concepts of Table 5) is shown as the last lattice of Figure 2, with *reduced labeling*, and showing only the attributes (words).

Section 4 provides a brief description of how the concept lattice *grows* when initial segments of the whole string are processed, starting with the first symbol in the stream of symbols, then the first two symbols, and so on. A *stream (of symbols, or characters)* of length n would form a string $s_{n-1} = c_0 c_1 \dots c_{n-1}$. Then STEP 0 of processing the stream will correspond to processing the string $s_0 = c_0$ and STEP i to processing the string $s_i = c_0 \dots c_i$; we will consider some of the STEPS 0–46 of processing the string WHATωSHECANNOTDO of length 47. Figures 1–2 depict the concept lattices for selected steps of the processing.

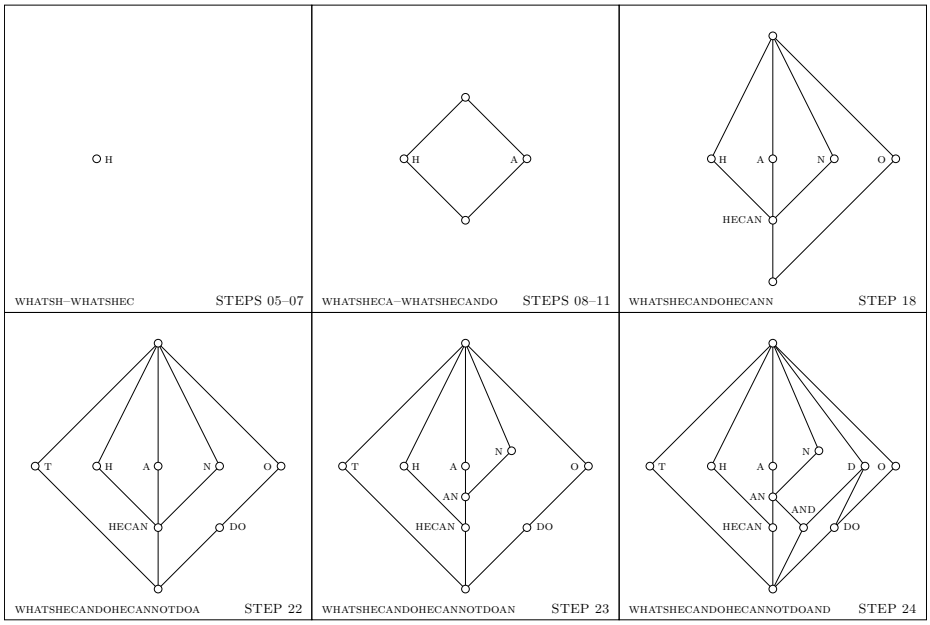


Fig. 1. WHATωSHECANNOTDO: concept lattices for STEPS 5–7, 8–11, 18, 22, 23, 24.

4 Growing words, growing lattices

Processing a *stream* (of text) is different than processing a *string.* A stream is a sequence of symbols, or *characters* arriving at times forming a temporal sequence. Let $t_0, t_1, \ldots, t_{i-1}$ be *timepoints*, such that $t_0 < t_1 < \ldots < t_{i-1}$. Suppose that a character c_j "arrives" at time t_j (for j in $(0, \ldots, i-1)$); at time t_{i-1} the *stream* of processed characters allows us to build a string $s_{i-1} = c_0 \ldots c_{i-1}$. At time t_i a new character c_i arrives and the previous string s_{i-1} is extended to $s_i = s_{i-1}.c_i$. An important aspect of stream processing is its iterative character: after processing the string s_{i-1} we wait for a character c_i and when it arrives we update the processing of the string s_{i-1} to have the string $s_i = s_{i-1}.c_i$ processed. This includes updating the following: *position* i (of the "reading head" that has just read the character that arrived), *character* c_i, *string* s_i, *alphabet* Σ_i (of unique characters), *set of attributes* or *words* M_i, *set of objects* G_i, *context* (G_i, M_i, I_i) and *ordered set of concepts* $\mathfrak{B}_i = \mathfrak{B}(G_i, M_i, I_i)$. The relevant questions include: how do objects and attributes change, and how do the contexts, concepts and concept lattices change?

In Figures 1 and 2 it is shown how the lattices *grow*, and how the attributes (words) change. Notice that a *reduced labelling* is used, showing only attributes, or words (to simplify the discussion, we refer to nodes, or concepts, using the attribute labels). A word can *grow* and during the *growing process* it *absorbs* its initial segments. For instance, we obtain the following sequence of words: W, WH, WHA, WHAT—and WHAT *absorbs* all its inital segments. This is properly imple-

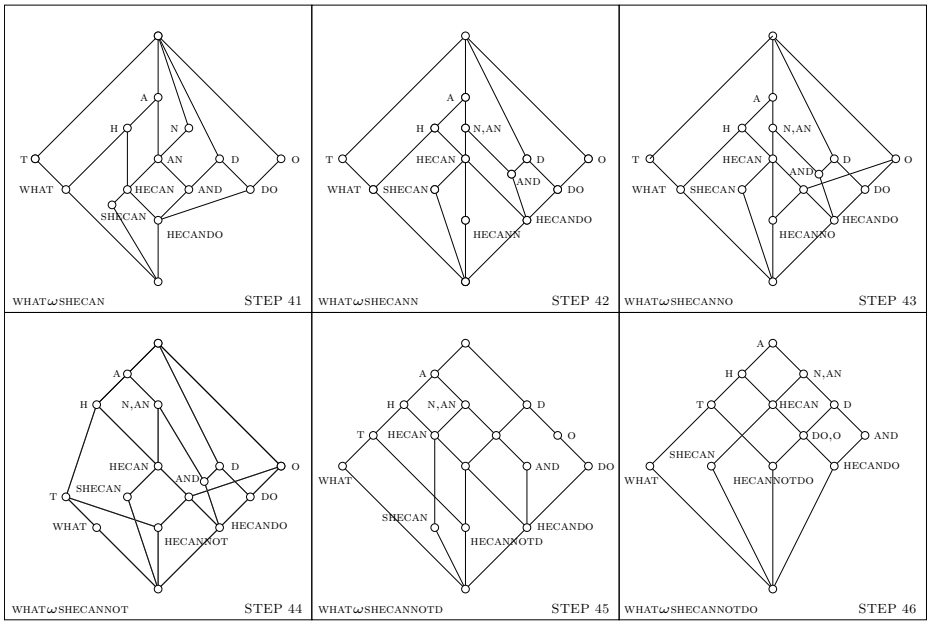


Fig. 2. `WHATωSHECANNOTDO`: concept lattices for STEPS 41–46.

mented in the python code: for a word w_1 to absorb a word w_2, the occurrences of w_1 must also "absorb" (or *cover*) the occurrences of w_2; for example, if there was a sequence `WHERE` in the processed string then `WHAT` would not absorb `WH` because its occurrences would not cover the occurrences of `WH` in `WHERE`.

STEPS 05–07, 08-11, 18, 22, 23, 24 are depicted in Figure 1;
STEPS 00–04: no string reoccurs: there are no attributes, no objects, no words; $G_i = M_i = \varnothing$ (for $i = 0, \ldots, 4$); $\Sigma_0 = \{\texttt{W}\}, \ldots, \Sigma_4 = \{\texttt{W}, \texttt{H}, \texttt{A}, \texttt{T}, \texttt{S}\}$;
STEPS 05–07: a string `H` reoccurs—the first detected *word*; $\Sigma_5 = \Sigma_4$;
STEPS 08–11: `A` appears; hence, two (singleton) words have been detected;
. . .
STEP 22: list of attributes: `HECAN`, `DO`, `T`, `H`, `A`, `N` and `O`;
STEP 23: `AN` appears as a meet of `A` and `N`, above `HECAN`;
STEP 24: `D` reappears and `AND` appears; $\texttt{D} \wedge \texttt{O} = \texttt{DO}$, $\texttt{D} \wedge \texttt{AN} = \texttt{AND}$;
. . .
STEPS 41–46 are depicted in Figure 2.
STEP 41: `SHECA` grows to `SHECAN`, `HECA` is absorbed by `HECAN`;
STEP 42: `HECANN` appears, `N` and `AN` merge (below `A`);
STEP 43: `HECANN` grows to `HECANNO`;
STEP 44: `HECANNO` grows to `HECANNOT`, `T` becomes a subconcept of `A`;
STEP 45: `HECANNOT` grows to `HECANNOTD`;
STEP 46: `HECANNOTD` grows to `HECANNOTDO`; `A` becomes the top concept.

Some comments follow. A word grows and (usually, but not always) absorbs its initial segments. A word never *disappears*: the only way for a word to disappear is to be absorbed by another word (in this case the absorbed word is a segment of the absorbing word). It is also possible for a word that disappeared to reappear. As the set of detected words grows (with the words themselves growing), the lattice necessarily grows (to accomodate the enlarged set of words). But the lattice does not grow monotonically: it can contract, when some words disappear by being absorbed by other words, or some nodes are dropped while other nodes merge or shift It seems desirable to investigate how words and word lattices change and grow, as a complete catalogue of possible changes—or *transitions*—would facilitate the understanding of the dynamic behaviour of word lattices.

5 Conclusion

FCA processing for word detection is being explored, with most processing already implemented. It remains to be seen whether it provides an alternative to HMM and NN processing, or whether some *hybrid* systems including FCA processing as components could be built.[4] For example, passing the word concept extracted by FCA through a neural network for sequence prediction could be used to generate predictions of future sequence behaviour.

Two main questions are the following.

1. Would an FCA based string processing aimed at word detection *scale up* for large alphabets and long streams of symbols?
2. Could an FCA based system be an alternative to NN and HMM based systems, or could it form a useful component of a *hybrid system?*

Regarding the first question, it is expected that *performance* might be an issue because the intention is to work with large alphabets[5] and long strings.

Regarding the second question, even if an FCA based system performs worse than the alternatives, it might provide a valuable *conceptual analysis* tool.

The performance issues are related to the cost of searching for substrings in large strings (to obtain *words* and their *occurrences*, cf. Table 2) and the cost of generating concept lattices. Searching for substrings could be seen as *external* to the FCA concept lattice processing. Nevertheless, the references listed in Section 2 are worth exploring.

Issues of interest, but not considered here, include detecting *higher level behaviours* (requiring detecting not only *words* but also *sentences*, or *topics*) and handling *noise* (possibly using *rough sets* [19]). Regarding the latter issue—*noise*—one should not expect all detectable words to be exact matches, some might differ from the *pattern words* only slightly and we don't want to miss them. Regarding the former issue—*higher level patters*—detecting them would

[4] Our work on HMM and NN based processing is progressing, but there are no results available yet.

[5] The expected size of the alphabet is in order of (potentially) millions of symbols.

provide a method for fast pattern detection, as it would allow *context switching*. It seems that the following could be attempted. The input stream contains *characters* $(c_i^{(1)})_i$ and we are detecting *strings*—*attributes* which are *words* representing *patters*—$(s_j^{(1)})_j$. The characters $(c_i^{(1)})_i$ and the strings $(s_j^{(1)})_j$ could be referred to as *first level* characters and strings. As we progress with detecting $(s_j^{(1)})_j$, we can form *second level* characters by putting $c_j^{(2)} = s_j^{(1)}$, i.e., a second level character is a first level pattern. Some sequences of second level characters would reoccur, forming *second level* strings, or *second level* patterns $(s_k^{(2)})_k$, where $s_k^{(2)}$—a second level pattern—is a recurring string over the alphabet $(c_j^{(2)})_j$ (the alphabet $(c_j^{(2)})_j$ is a set of detected first level patterns). Conceptually, detecting higher level patterns would be simple, and it should drastically improve performance and allow to handle large alphabets and string; However, no reportable experimentation has been performed.

For FCA *word detection*, or *behaviour prediction*, to be useful—whether as a stand-alone system or as an FCA-based component of a hybrid system—fast calculation and drawing of concept lattices is required. We are searching for good ways of achieving this.

Detecting and predicting patterns in ubiquitous data streams is an important and challenging task for knowledge processing in general, and FCA based processing (including *visualising* patterns) in particular.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR abs/1409.0473 (2014)
2. Berkhin, P.: Survey of Clustering Data Mining Techniques (2002), Accrue Software
3. Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S.O., Napoli, A., Raïssi, C.: On mining complex sequential data by means of FCA and pattern structures. Int. J. Gen. Syst. 45(2), 135–159 (2016)
4. Buzmakov, A., Kuznetsov, S.O., Napoli, A.: Scalable Estimates of Concept Stability. In: Int'l Conf. on Formal Concept Analysis (ICFCA). Springer (2014)
5. Chua, S., Marsland, S., Guesgen, H.W.: Unsupervised Learning of Human Behaviours. In: Proceedings Twenty-Fifth AAAI Conference on AI. AAAI (2011)
6. Cohen, P., Adams, N., Heeringa, B.: Voting experts: An unsupervised algorithm for segmenting sequences. Intelligent Data Analysis (2007)
7. Ferré, S.: The efficient computation of complete and concise substring scales with suffix trees. In: Int'l Conf. on Formal Concept Analysis (ICFCA). Springer (2007)
8. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Delugach, H.S., Stumme, G. (eds.) Conceptual Structures: Broadening the Base. 9th International Conference on Conceptual Structures, ICCS 2001. Springer (2014)
9. Graves, A.: Generating sequences with recurrent neural networks. CoRR abs/1308.0850 (2013), http://arxiv.org/abs/1308.0850
10. Graves, A., Mohamed, A., Hinton, G.E.: Speech recognition with deep recurrent neural networks. CoRR abs/1303.5778 (2013), http://arxiv.org/abs/1303.5778
11. Hammarström, H., Borin, L.: Unsupervised learning of morphology. Computational Linguistics 37(2), 309–350 (June 2011)

12. Heymann, J., Walter, O., Haeb-Umbach, R., Raj, B.: Iterative Bayesian word segmentation for unsupervised vocabulary discovery from phoneme lattices. In: Int'l Conf. on Acoustic, Speech and Signal Processing. IEEE (2014)
13. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (November 1997)
14. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. 31(3), 264–323 (September 1999)
15. Johnson, M.J., Willsky, A.S.: Bayesian Nonparametric Hidden Semi-Markov Models. Journal of Machine Learning Research (JMLR) 14, 673–701 (2013)
16. Kokalj-Filipovic, S., Goodman, J., Acosta, C.B., Stantchev, G.: I/O HSMM: Learning behavioral dynamics of a cognitive wireless network node from spectrum sensing. In: 2016 Annual Conference on Information Science and Systems. IEEE (2016)
17. Kriegel, F.: Incremental computation of concept diagrams (2014)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
19. Nowak, C.: Rough Decision Systems (2017), DSTG Tech. Report (under review)
20. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.K.: Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. CoRR abs/1502.06922 (2015)
21. Poelmans, J., Ignatov, D.I., Viaene, S., Dedene, G., Kuznetsov, S.O.: Text mining scientific papers: A survey on FCA-based information retrieval research. In: ICDM 2012, LNAI 7377. pp. 273–287. Springer (2012)
22. Priss, U.: Linguistic Applications of Formal Concept Analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. Foundations and Applications. Springer (2005)
23. Ranjan, R., Patel, V.M., Chellappa, R.: HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. CoRR abs/1603.01249 (2016), http://arxiv.org/abs/1603.01249
24. Schmidhuber, J.: Deep learning in neural networks. Neural Networks 61 (2015)
25. Siebes, A.: MDL in Pattern Mining. A Brief Introduction to Krimp. In: Int'l Conf. on Formal Concept Analysis (ICFCA). Springer (2014)
26. Strok, F., Galitsky, B., Ilvovsky, D., Kuznetsov, S.O.: Pattern structure projections for learning discourse structures. In: AIMSA 2014. Springer (2014)
27. Sun, Y., Liang, D., Wang, X., Tang, X.: DeepID3: Face Recognition with Very Deep Neural Networks. CoRR abs/1502.00873 (2015), http://arxiv.org/abs/1502.00873
28. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR abs/1409.3215 (2014), http://arxiv.org/abs/1409.3215
29. Tatar, D., Kapetanios, E., Sacarea, C., Tanase, D.: Text segments as constrained formal concepts. In: 12th Int'l Symp. on Symbolic and Numeric Algorithms for Scientific Computing (2010)
30. Wang, J., Yang, X., Wang, B., Zhu, H.: Memory-Aware BWT by Segmenting Sequences to Support Subsequence Search. In: Web Technologies and Applications: 14th Asia-Pacific Web Conference (2012)
31. Welch, T.: A Technique for High-Performance Data Compression. Computer 17(6), 8–19 (1984)

Mining Triclusters of Similar Values in Triadic Real-Valued Contexts

Dmitrii Egurnov[1] ✉
(0000-0002-8195-1670) egurnovd@yandex.ru,
Dmitry Ignatov[1]
(0000-0002-6584-8534) dignatov@hse.ru, and
Engelbert Mephu Nguifo[2]
(0000-0001-6502-0019) mephu@isima.fr

[1] National Research University Higher School of Economics, Moscow, Russia
[2] Université Clermont Auvergne, CNRS, LIMOS, Clermont-Ferrand, France

Abstract. Analysis of polyadic data (for example, multi-way tensors and n-ary relations) becomes more and more popular task nowadays. While several datamining techniques exist for (numeric) dyadic contexts, their extensions to the triadic case are not obvious, if possible at all. In this work, we study development of the ideas of Formal Concept Analysis for processing three-dimensional data, namely the so called OAC-triclustering (from Object, Attribute, Condition). Among several known methods, we have reasonably selected the most effective one and used it to propose an algorithm NOAC-triclustering for mining triclusters of similar values in real-valued triadic contexts. We have also proposed a second simple algorithm, Tri-K-Means, based on clustering algorithm K-Means, for the purpose of comparison. The experimental part demonstrates application of the algorithms to both computer-generated and real-world data.

Keywords: Three-way data mining, Triadic Formal Concept Analysis, Triclustering, OAC-triclustering, Real-valued context

1 Introduction

Analysis of polyadic data has a rather long history. Thus, in 1966, L. Tucker proposed a multidimensional extension of factor analysis for three-way tensors [1]. Now, tensor and matrix decomposition approaches play a crucial role in machine learning where so called latent factors, compact data representation and dimensionality reduction are needed [2].

There is a complementary approach to such decompositions that aims at finding homogeneous patterns (e.g., dense submatrices or subtensors) in object-attribute matrices and tensors of higher order. Namely, biclustering. The term was coined in [3] but dates back to the seminal work of Hartigan [4] on direct clustering.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

It seems that Formal Concept Analysis, which is aimed at analysis of object-attribute relational data, is able to solve both of the aforementioned tasks for specific cases of input data. Thus, it is possible to find optimal decompositions of three-dimensional Boolean tensors by means of certain subsets of formal concepts [5]; formal concepts can be viewed as biclusters and their n-ary extensions such as triconcepts can be treated as triclusters [6, 7]. In data mining and FCA community there were several attempts to extend FCA-like pattern mining techniques to searching for patterns in numeric data. Thus, in [8], the authors proposed mining of biclusters (bi-sets) under user-defined constraints, while in [9] the authors searched for numeric biclusters by means of Triadic Formal Concept Analysis (3-FCA). Several authors proposed n-ary extensions of FCA for pattern mining with closed n-sets both exact [10–12] and approximate [13, 14].

It is interesting, that historically triadic formal concepts came to the stage even before formal treatment of Triadic Formal Concept Analysis by Lehman and Wille [15]; in fact, one of the first application of 3-FCA deals with recall data in a study of social perception [16].

The goal of this work is two-fold: to prove important properties of the previously introduced OAC-triclustering and to propose a new subsequent approach, NOAC-triclustering, for mining triclusters of similar values in real-valued triadic contexts. We have also proposed a second simple algorithm, Tri-K-Means, based on clustering algorithm K-Means, for the purpose of comparison. The experimental part demonstrates application of the algorithms to both computer-generated and real-world data.

The paper is organised as follows. In Section 2, we recall several basic notions of Triadic FCA. Section 3 discusses multimodal (or multi-way) extensions of clustering approach with three dimensions simultaneously from the most general perspective. In Section 4, we recall the definition of OAC-triclusters as a relaxation of triconcepts from Section 2. In Section 5, we prove several important properties of possible variations of OAC-tricluster forming operators that justify our choice of the concrete OAC-tricluster definition. Section 6 proposes a version of OAC-triclustering for numeric (and real) data based on similar values as well as modification of triadic extension of conventional K-Means clustering. In Section 7, we describe real and synthetic datasets selected for experimentation. Section 8 reports main experimental results of comparison of both algrorithms. Section 9 concludes the paper and outlines possible directions of future work.

2 Triadic Formal Concept Analysis

In Triadic Formal Concept Analysis we deal with triadic formal contexts. They are very similar to traditional Formal Contexts, but include three dimensions, or modalities: objects, attributes and conditions [15].

Definition 1. *Let G, M and B be arbitrary sets. The subset of their Cartesian product defines a triadic relation $I \subseteq G \times M \times B$. The triple $\mathbb{K} = (G, M, B, I)$ is called a triadic formal context, or tricontext. The sets G, M and B are called set of objects, set of attributes, and set of conditions, respectively.*

When there is a triple $(g, m, b) \subseteq I$, where $g \in G$, $m \in M$, and $b \in B$, it is said that "the object g has the attribute m under the condition b".

Like in Formal Concept Analysis we use concept-forming (prime) operators, but because modality of our case is higher there exists six variations of those operators:

$$\theta_G : 2^G \to 2^M \times 2^B, \qquad \theta_{G,M} : 2^G \times 2^M \to 2^B,$$
$$\theta_M : 2^M \to 2^G \times 2^B, \qquad \theta_{G,B} : 2^G \times 2^B \to 2^M,$$
$$\theta_B : 2^B \to 2^G \times 2^M, \qquad \theta_{M,B} : 2^M \times 2^B \to 2^G.$$

These variations are called triadic concept-forming (or prime) operators. They are assorted by the number of sets in their input into two groups: 1-set (triadic) prime operators and 2-set (triadic) prime operators.

For example, if $X \subseteq G$, $Y \subseteq M$ for a given tricontext $\mathbb{K} = (G, M, B, I)$, then $\theta_{G,M}(X, Y) = (X, Y)' = \{b \in B \mid \forall (g, m) \in X \times Y : (g, m, b) \in I\}$. The remaining operators for 2-sets, $\theta_{G,B}(\cdot, \cdot)$, $\theta_{M,B}(\cdot, \cdot)$, are defined similarly. Further, we use the same prime-based notation for all the three operators: $(\cdot, \cdot)'$.

Therefore, triadic formal concept is defined in the following way:

Definition 2. *A triple of sets (X, Y, Z), where $X \subseteq G$, $Y \subseteq M$, $Z \subseteq B$, is called triadic formal concept (or triconcept) iff three conditions hold: $(X, Y)' = Z$, $(X, Z)' = Y$ and $(Y, Z)' = X$.*

The first and the second elements of the triple inherit their names from the dyadic case, which are respectively extent and intent, while the third component is called modus (modi in plural). As well as in the dyadic case, triadic formal concept can be interpreted as a maximal cuboid of positive values (or crosses) in the Boolean matrix representation of the formal context, possibly under suitable permutations of elements of the dimensions. In set notation, the statement is equivalent to maximality $X \times Y \times Z \subseteq I$ w.r.t. $\subseteq$ order over G, M, B.

The set of all triadic formal concepts of the triadic formal context $\mathbb{K}$ can be organized into a structure called concept trilattice $\mathfrak{T}(\mathbb{K})$, however unlike in the dyadic case the extents (respectively intents and modi) does not form a closure system since two different triconcepts may have the same extent, but their intent and modus parts may be incomparable in terms of set inclusion.

For polyadic extension of FCA see [17].

3 Multi-way Clustering

The methods of FCA and 3-FCA, in particular, have found numerous successful applications in different fields, but sometimes they may be too strict to operate on real-world data. Big datasets are prone to contain missing values, errors and noise, which may lead to losing some part of relevant information in the output. The clustering approach studied in this chapter is supposed to be more flexible and applicable to corrupted data.

3.1 Triclustering

A formal definition of tricluster is usually adjusted for each specific problem as it depends on the problem statement, data types, method of triclustering, and desired output. In this part we will give the most general definition.

Definition 3. *Let $A_{n\times k\times l}$ be a three-dimensional binary matrix (or tensor). Let sets $G = \{g_1, g_2, \ldots, g_n\}$, $M = \{m_1, m_2, \ldots, m_k\}$ and $B = \{b_1, b_2, \ldots, b_l\}$ be the index sets of A. Then for some arbitrary sets $X \subseteq G$, $Y \subseteq M$ and $Z \subseteq B$ the submatrix $A_{XYZ} = \{a_{xyz} \mid x \in X, y \in Y, z \in Z\}$ is called a tricluster. The sets X, Y and Z are called respectively extent, intent and modus of the tricluster.*

Triadic formal concepts may be considered as a special case of triclusters, because they form a three-dimensional structure without zeros ("holes") in the triadic formal context. Therefore, methods of 3-FCA are part of the triclustering methods. However, most datasets have little distortions, that may affect the output of the algorithms, so it is considered useful to relax the absolute density condition, allowing some number of empty cells in the clusters.

Various constraints may be applied to triclusters. Usually they are structure requirements, cardinality restrictions on extent, intent and modus, and limitations of other parameters. For example, the most common conditions, which eliminate small and meaningless structures from the output, are the minimal support condition ($|X| \geq s_X, |Y| \geq s_Y, |Z| \geq s_Z$) and minimal density threshold:

$$\rho(A_{XYZ}) = \frac{\sum_{i=1}^{|X|}\sum_{j=1}^{|Y|}\sum_{k=1}^{|Z|} a_{x_i y_j z_k}}{|X||Y||Z|} \geq \rho_{min}$$

The main application of triclustering is in the field of gene expression analysis and biological data mining [18, 19]. There are also works dedicated to the use of triclustering in recommender systems [20] [21], folksonomies analysis [6], and social network analysis [22].

3.2 Triadic Multivalued Clustering

Another step forward from traditional formal contexts, where table representations contain only values from the set $\{0, 1\}$, will be triadic multivalued contexts. In such contexts the data is not binary, but may take any value from some possibly infinite set W.

Definition 4. *Triadic multivalued context is a tuple $\mathbb{K} = (G, M, B, W, I, V)$, where G, M and B are respectively the sets of objects, attributes and conditions. The set W is called the set of values. Ternary relation $I \subseteq G \times M \times B$ defines which triples of the context have values. The value function $V : I \to W$ associates a triple $(g, m, b) \in I$ with its corresponding value $w \in W$.*

When $V(g, m, b) = w$ it is said that "the object g has the value w on the attribute m under the condition b".

In its general form, the definition of a tricluster does not differ from the binary case.

Definition 5. *Triple (X, Y, Z), where $X \subseteq G$, $Y \subseteq M$ and $Z \subseteq B$, is called a triadic multivalued cluster (multivalued tricluster).*

Multivalued Context Analysis is used to process numerical data. This type of data is common in biological problems, specifically microarray gene expression analysis [18], [23]. In this work we only consider cases when $W \subseteq \mathbb{R}$. Thence the studied methods are called Numerical.

4 OAC-triclustering

The main triclustering method studied within FCA framework previously is OAC-triclustering [13]. It is the result of the extension of the OA-biclustering algorithm [24] on the triadic case. The two existing variations of this method are very similar, but one relies on box operators, while another uses prime operators [13]. Let us describe them in detail.

4.1 Box Operator Based OAC-triclustering

The box operator based variant of OAC-triclustering was chronologically the first. It was introduced in [7]. The method utilizes the following idea:

Let $\mathbb{K} = (G, M, B, I)$ be a triadic context. For triplets $g \in G$, $m \in M$, $b \in B$ of the context, where $(g, m, b) \in I$, we generate triclusters by applying so-called box operators. The triple to which the operators are applied is called the generating triple, or generator. Further, we simplify the used set notation for both prime and box operators and write $a^{\square}$ and a' instead of $\{a\}^{\square}$ and $\{a\}'$, but readers should keep in mind, that the operators can also take sets as input.

First, we should give definitions for prime operators, that take 1-set input, as box operators rely on them (consider $X \subseteq G$,$Y \subseteq M$,$Z \subseteq B$):

$$X' = \{(m, b) \in M \times B \mid \forall \tilde{g} \in X : (\tilde{g}, m, b) \in I\}$$

$$Y' = \{(g, b) \in G \times B \mid \forall \tilde{m} \in Y : (g, \tilde{m}, b) \in I\}$$

$$Z' = \left\{(g, m) \in G \times M \mid \forall \tilde{b} \in Z : (g, m, \tilde{b}) \in I\right\}$$

For a fixed triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ the box operators are defined in the following way:

$$\tilde{g}^{\square} := \left\{g \mid \exists m : (g, m) \in \tilde{b}' \vee \exists b : (g, b) \in \tilde{m}'\right\}$$

$$\tilde{m}^{\square} := \left\{m \mid \exists g : (g, m) \in \tilde{b}' \vee \exists b : (m, b) \in \tilde{g}'\right\}$$

$$\tilde{b}^{\square} := \left\{b \mid \exists g : (g, b) \in \tilde{m}' \vee \exists m : (m, b) \in \tilde{b}'\right\}$$

The outputs of the operators are called the box sets.

Definition 6. *For a triple* $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ *a triple of sets* $T = (\tilde{g}^{\square}, \tilde{m}^{\square}, \tilde{b}^{\square})$ *is called a box operator based OAC-tricluster. The components of the triple are called respectively extent, intent and modus. The triple* $(\tilde{g}, \tilde{m}, \tilde{b})$ *is called a generating triple of the tricluster* T*.*

Figure 1 illustrates the addition condition for an element $g \in G$ to be included into $\tilde{g}^{\square}$. The element will be added to the box set, if the gray zone contains at least one cross.

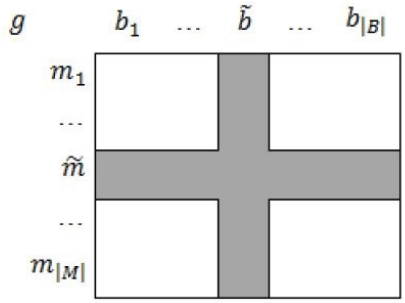


Fig. 1: Addition of g to $\tilde{g}^{\square}$

The method consists of iteration over all of the triples in the relation I of the triadic context $\mathbb{K}$, generation of box operator based triclusters for each of them and storing the triclusters in a set $\mathcal{T}$. The tricluster T should be added to the set $\mathcal{T}$ only if it has not been discovered earlier, as some triplets may result in the same triclusters. Minimal support condition and density threshold may improve performance by elimination of less important results from the output.

4.2 Alternative Box Operator Definition

The alternative definition for box operators utilizes possible changes of quantifiers and logical operators for obtaining different sets and, therefore, different triclusters. The general definition of box operators has the following form:

$$\tilde{g}^{\square} := \left\{ g \;\middle|\; \exists\, m : (g,m) \in \tilde{b}' \;\Diamond\; \exists\, b : (g,b) \in \tilde{m}' \right\}$$

$$\tilde{m}^{\square} := \left\{ m \;\middle|\; \exists\, g : (g,m) \in \tilde{b}' \;\Diamond\; \exists\, b : (m,b) \in \tilde{g}' \right\}$$

$$\tilde{b}^{\square} := \{ b \;\mid\; \exists\, g : (g,b) \in \tilde{m}' \;\Diamond\; \exists\, m : (m,b) \in \tilde{g}' \}$$

Here the symbol $\Diamond$ signifies variability between logical operators $\wedge$ and $\vee$. The 1-set prime operators remain the same, as they were given in the previous section.

This yields two variations of box operators. They are distinguished by characters, which substitute optionality symbol, and are called respectively $\vee$-box and $\wedge$-box operators. Let us investigate their structure.

1. The $\vee$-box variation is the simplest one, as it is identical to the classical box operators. In order for an element to be added to the box set, the slice corresponding to the element should contain at least one cross in the gray area, as shown in Figure 2a.
   The aforementioned general definition of the box operators contains 1-set prime operators, which makes it complicated and difficult to use in theoretical work. For this reason, we introduce a detailed definition for $\vee$-box operators:

$$\tilde{g}^{\square}_{\vee} := \left\{ g \mid \exists\, m : (g, m, \tilde{b}) \in I \ \vee \ \exists\, b : (g, \tilde{m}, b) \in I \right\}$$

$$\tilde{m}^{\square}_{\vee} := \left\{ m \mid \exists\, g : (g, m, \tilde{b}) \in I \ \vee \ \exists\, b : (\tilde{g}, m, b) \in I \right\}$$

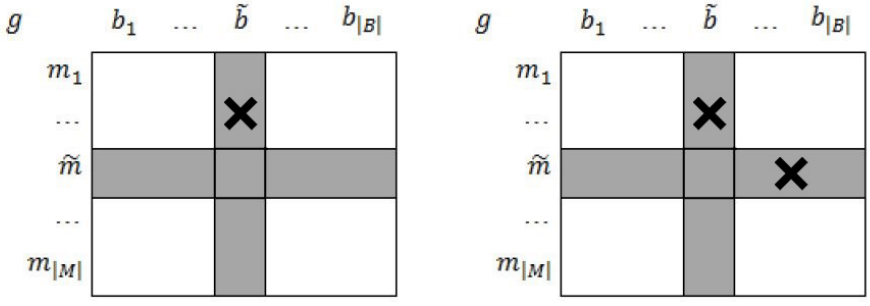
$$\tilde{b}^{\square}_{\vee} := \{ b \mid \exists\, g : (g, \tilde{m}, b) \in I \ \vee \ \exists\, m : (\tilde{g}, m, b) \in I \}$$

2. The $\wedge$-box operators impose stricter conditions for an element to be included in the box set. The corresponding slice needs to have at least one cross in both lines of the gray area (see Figure 2b). This way of forming triclusters is supposed to give higher density. The detailed definition is the following:

$$\tilde{g}^{\square}_{\wedge} := \left\{ g \mid \exists\, m : (g, m, \tilde{b}) \in I \ \wedge \ \exists\, b : (g, \tilde{m}, b) \in I \right\}$$

$$\tilde{m}^{\square}_{\wedge} := \left\{ m \mid \exists\, g : (g, m, \tilde{b}) \in I \ \wedge \ \exists\, b : (\tilde{g}, m, b) \in I \right\}$$

$$\tilde{b}^{\square}_{\wedge} := \{ b \mid \exists\, g : (g, \tilde{m}, b) \in I \ \wedge \ \exists\, m : (\tilde{g}, m, b) \in I \}$$



(a) $\vee$-box case (b) $\wedge$-box case

Fig. 2: $\Diamond$-box addition conditions

4.3 Prime Operator Based OAC-triclustering

The prime algorithm [25] uses a simpler way of generating triclusters. It corresponds to the dyadic method of mining OA-biclusters described in [24]. The OAC-triclusters found by this approach have similar to the OA-biclusters cross-like structure.

Like in the previous method, the triclusters are generated by applying 2-set prime operators to the pairwise combinations of the components of each triple in the context. The motivation for development of this method was to improve quality of the resulting triclusters in terms of density and simplicity of the structure. Now we will draw the definition for prime operator based OAC-triclusters:

Let $\mathbb{K} = (G, M, B, I)$ be a triadic context. 2-set prime operators have the following form (consider $X \subseteq G$,$Y \subseteq M$,$Z \subseteq B$):

$$(X, Y)' = \{b \in B \mid \forall \tilde{g} \in X, \tilde{m} \in M : (\tilde{g}, \tilde{m}, b) \in I\}$$

$$(X, Z)' = \left\{m \in M \mid \forall \tilde{g} \in X, \tilde{b} \in B : (\tilde{g}, m, \tilde{b}) \in I\right\}$$

$$(Y, Z)' = \left\{g \in G \mid \forall \tilde{g} \in X, \tilde{m} \in M : (g, \tilde{m}, \tilde{b}) \in I\right\}$$

The outputs of the operators are called the prime sets.

Definition 7. *For a triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ a triple of sets $T = ((\tilde{m}, \tilde{b})', (\tilde{g}, \tilde{b})', (\tilde{g}, \tilde{m})')$ is called a prime operator based OAC-tricluster. The components of the triple are called respectively extent, intent and modus. The triple $(\tilde{g}, \tilde{m}, \tilde{b})$ is called a generating triple of the tricluster T.*

As mentioned before these triclusters have star-like structure in the 3-dimensional table representation of the formal context (under appropriate permutation of rows, columns and layers).

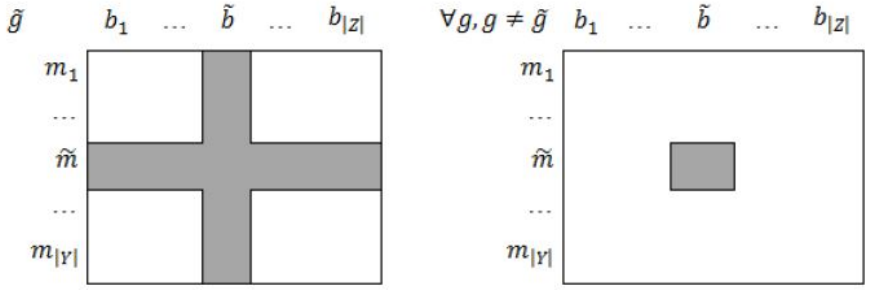


Fig. 3: Prime operator based tricluster structure

Figure 3 shows the structure of a prime operator based tricluster $T = (X, Y, Z)$, generated from a triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$. The gray zone contains crosses.

Contents of the white zone are unknown and define the density (quality) of the tricluster.

The algorithm is identical to the box operator based one: it iterates over all the triplets of the context $\mathbb{K}$, generates triclusters T-s and adds them to the resulting set $\mathcal{T}$, if they were not found on the previous steps.

5 Relations of the OAC-triclustering Operators

In this section we discuss the relations between the variations of box operators and the prime operator, as well as the triclusters that they produce. We propose several ordering lemmas that describe the relations, and later are used to organize the operators in a simple system. We consider the lemmas only for one dimension, for simplicity. The formulations and proofs for other dimensions can be developed in a similar way, due to intrinsic symmetry between the dimensions. The proofs for the following statements are rather simple, so we omit them in this paper.

Lemma 1. *The box set generated by the $\vee$-box operator contains as subset the box set generated by the corresponding $\wedge$-box operator from the same generating triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$, but the contrary is not always true.*

Lemma 2. *The box set generated by the $\wedge$-box operator contains as subset the prime set generated by the prime operator from the same generating triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$, but the contrary is not always true.*

Theorem 1 (Nesting Order of tricluster components generated from the same generating triple). *The box and prime sets generated by the prime operators and the variations of box operators from the same generating triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ are ordered in the following way with respect to set inclusion:*

$$(\tilde{m}, \tilde{b})' \subseteq \tilde{g}_{\wedge}^{\square} \subseteq \tilde{g}_{\vee}^{\square}$$
$$(\tilde{g}, \tilde{b})' \subseteq \tilde{m}_{\wedge}^{\square} \subseteq \tilde{m}_{\vee}^{\square}$$
$$(\tilde{g}, \tilde{m})' \subseteq \tilde{b}_{\wedge}^{\square} \subseteq \tilde{b}_{\vee}^{\square}$$

Corollary 1. *The triclusters built with the box and prime sets generated by the operators from the same generating triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ inherit the same nesting order with respect to component-wise inclusion $\sqsubseteq$:*

$$T' \sqsubseteq T_{\wedge}^{\square} \sqsubseteq T_{\vee}^{\square}.$$

These findings may help to analyze and explain the changes in number and quality of the triclusters found by different methods in the same triadic context.

6 Methods of Numerical Triclustering

In this section, two methods of numerical triclustering for finding triclusters of similar values are proposed. The first one is an extension of prime operator based OAC-triclustering for real-valued contexts. The second one is a classical clustering algorithm K-Means, for which we have developed a special metric in order to operate in multivalued triadic contexts.

6.1 Numerical OAC-triclustering

The NOAC (Numerical OAC) method was developed from the prime operator based OAC-triclustering by adjusting it for mining numerical triclusters of similar values in numerical triadic contexts. It has a parameter δ, which defines possible deviations of values inside the extracted triclusters.

Let $\mathbb{K} = (G, M, B, R, I, V)$ be real-valued triadic context. We iterate over the generating triples $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$, build numerical triclusters and try adding them to the resulting set of triclusters $\mathcal{T}$. The modified prime operators utilize the parameter δ, therefore we have decided to call them δ-operators.

$$(\tilde{g}, \tilde{m})^{\delta} = \left\{ b \;\mid\; (\tilde{g}, \tilde{m}, b) \in I \wedge |V(\tilde{g}, \tilde{m}, b) - V(\tilde{g}, \tilde{m}, \tilde{b})| < \delta \right\}$$

$$(\tilde{g}, \tilde{b})^{\delta} = \left\{ m \;\mid\; (\tilde{g}, m, \tilde{b}) \in I \wedge |V(\tilde{g}, m, \tilde{b}) - V(\tilde{g}, \tilde{m}, \tilde{b})| < \delta \right\}$$

$$(\tilde{m}, \tilde{b})^{\delta} = \left\{ g \;\mid\; (g, \tilde{m}, \tilde{b}) \in I \wedge |V(g, \tilde{m}, \tilde{b}) - V(\tilde{g}, \tilde{m}, \tilde{b})| < \delta \right\}$$

Definition 8. *δ-operator based OAC-tricluster for a generating triple $(\tilde{g}, \tilde{m}, \tilde{b}) \in I$ is a triple of sets $T = \left((\tilde{m}, \tilde{b})^{\delta}, (\tilde{g}, \tilde{b})^{\delta}, (\tilde{g}, \tilde{m})^{\delta}\right)$. The elements of the triple T are called the intent, extent and modus of the tricluster.*

As in binary case, the tricluster contains three-dimensional cross of elements with similar values, which maximal difference with the central element does not exceed δ.

6.2 Tri-K-Means

To compare the method from the previous section with some alternative, we use the classical clustering algorithm K-Means with special metric, which allows it to operate in multivalued triadic contexts and mine triclusters of similar values.

Let $\mathbb{K} = (G, M, B, W, I, V)$ be multivalued tricontext. We use the following formula to compute distance between triples $t_1 = (g_1, m_1, b_1)$ and $t_2 = (g_2, m_2, b_2)$:

$$D(t_1, t_2) = |V(t_1) - V(t_2)| + \gamma * ([g_1 \neq g_2] + [m_1 \neq m_2] + [b_1 \neq b_2])$$

Here γ is a parameter that defines the priority of closeness of the values inside tricluster over inclusion of new elements. The expression $[a_1 \neq a_2]$ defines a predicate, which is equal to 0, when the coordinates of the triples are equivalent in the corresponding dimension, and 1 otherwise.

As a result we have k usual clusters of triples. Let $H \subseteq I$ be a set of triples representing one cluster. We transform it into a tricluster $T = (\{g \mid \exists(g, m, b) \in H\}, \{m \mid \exists(g, m, b) \in H\}, \{b \mid \exists(g, m, b) \in H\})$. This way the extent, intent and modus of the tricluster contain the element of the corresponding dimension, if at least one triple in H has the element. We call the algorithm Tri-K-Means.

A well-known weak point of K-Means algorithm is strong dependency of quality of clustering on right number of clusters k and initial distribution of centriods. Moreover, our approach may lose accuracy during the transfer of clusters into triclusters. However lower estimated computing time may give it some advantage over the NOAC method.

7 Datasets

We assessed performance of the algorithms by applying them to two different datasets: a pseudo-random generated and a real world dataset available from GroupLens project.

7.1 Pseudo-random Dataset

Three sets of input data were generated.

- Initial context containing two full cuboids 10x10x10 and 5x6x4 with values respectively 3 and 7. The total count of triples is 1120.
- Contexts with missing values, derived from the initial context. The percentage of missing values increases from 10% to 90% with 10% step.
- Contexts with noise, derived from the initial context. The deviation upper bound takes values 0.1, 0.5, 0.9, 1, 1.5, 2.

These dataset are used to assess the ability of studied algorithms to process simple data with known perturbations. The findings of these experiments help to understand how algorithms cope with such rather simple difficulties. Further, it could advice which method to use in each specific case.

7.2 Real World Dataset

The GroupLens project collects data from the movie recommendation site MovieLens, where thousands of people rate and tag movies[26]. We used the 100k dataset[3], which contains information about 100 000 5-star scale ratings of 1700 movie titles in 19 genres from 1000 different users. The total count of triples is 212595 and density is 0,00658. The data also provides which movie belong to which genre. Each movie title may have several genres associated with it. We treat users as a set of objects, movies as a set attributes and genres as a set of conditions. The triples are constructed from a user id, movie id of a movie rated by the user and one of the genres of the movie. The ratings are used as values of the context. However, the rating of the movie does not depend on its genre, so all the triples corresponding to a user-movie pair have the same value. This feature of the context is its main drawback or at least peculiarity.

[3] http://grouplens.org/datasets/movielens/

8 Experimental Verification

During the testing period we have discovered, that most appropriate comparison can be made when the parameter γ of Tri-K-Means method is equal to the parameter δ of NOAC in pairs of related experiments.

8.1 Quality Criteria

The triclusters in the outputs of the algorithms were evaluated by density and variance of values. For the resulting tricluster sets the measures are the number of triclusters, average density and average variance of values in the triclusters.

Let $T = (X, Y, Z)$ be some tricluster in a multivalued tricontext $\mathbb{K} = (G, M, B, W, I, V)$. We compute density of the tricluster as a ratio of number of triples $(g, m, b) \in I$ to its geometrical volume:

$$\rho(T) = \frac{|I \cap X \times Y \times Z|}{|X||Y||Z|}$$

If we consider each tricluster as an independent sample $S(T) = \{V(g, m, b) \mid (g, m, b) \in I \cap X \times Y \times Z\}$, then unbiased estimate of the variance is:

$$\sigma^2(S) = \frac{\sum_{i=1}^{|S|} \left(s_i^2 - \bar{S}^2\right)}{|S| - 1}$$

The tricluster satisfy the condition of similar values with the parameter δ, if the standard deviation of the tricluster does not exceed this parameter:

$$\sigma(S(T)) \leq \delta.$$

8.2 Experiments with Missing Values

Table 1 contains the results of experiments with contexts with missing values. Tri-K-Means shows perfect resistance to noise in incomplete data. NOAC demonstrates opposite results, failing to discover initial cuboids even with 20% loss rate. However, the number of triclusters is almost equal to the number of left triples in each case, which means most of them generated a unique tricluster, possibly very similar to one of the initial cuboids. This problem may be solved by merging similar triclusters and eliminating smaller ones.

In these experiments Tri-K-Means was set with parameters $k = 2$, $\gamma = 0$ and NOAC with $\delta = 0$.

8.3 Noise Tolerance Experiments

For noise resistance experiments, the initial context has been distorted with different deviations. In these experiments we have used Tri-K-Means with parameters $k = 2$, $\gamma = 1$ and NOAC with parameter $\delta = 1$.

Tri-K-means	# triclusters	0 2	10 2	20 2	30 2	40 2	50 2	60 2	70 2	80 2	90 2
	Cuboids found	+	+	+	+	+	+	+	+	+	+
NOAC	# triclusters	2	603	862	771	664	554	469	329	239	105
	Cuboids found	+	+	-	-	-	-	-	-	-	-

Table 1: Results of experiments with missing values

Table 2 contains the results of noise tolerance experiments. It shows that both methods deal with small deviations well. When the deviation grows Tri-K-Means starts to lose its accuracy due to variance. On 1.5 mark its output does not satisfy the condition of similar values any more. NOAC shows better results, but generates numerous triclusters on high deviation values. It loses the condition of similar values, when deviation reaches two. At this point the ranges of values in the initial cuboids start to intersect ([1;5] and [5;9]).

	Tri-K-Means		NOAC	
Deviation	# triclusters	Avg. Variance	# triclusters	Avg. Variance
0	2	02		0
0.1	2	0.0031	2	0.0031
0.5	2	0.0775	2	0.0775
0.9	2	0.2767	562	0.2622
1	2	0.3424	967	0.3221
1.5	2	1.5556	1115	0.7573
2	2	2.9508	77	1.2769

Table 2: Results of experiments with noise

8.4 Real World Dataset Experiments

Due to high execution time we limited number of triples in the context to 100 000.

Table 3 contain the results of experiments with GroupLens dataset. In the related experiments, parameter k of Tri-K-Means is set to the number of triclusters found by NOAC. Parameters γ and δ are equal to 1. NOAC algorithm has minimal density threshold set to 0.5 and minimal support threshold in extent and intent set to 4, in order to reduce computational time and improve quality of extracted triclusters. In our experiments Tri-K-means algorithm produces the same set of triclusters and stops after not more than 3 steps, which proves our assumption about the convergence of Tri-K-Means algorithm. Figure 4 shows

that Tri-K-Means time consumption grows slower with increase of the number of the input triples and after certain point is almost linear.

The output of numerical OAC-triclustering algorithms can be interpreted as interest clubs of users, who give similar rating to the same set of movies. In this case, the missing values in the triclusters may be used by some recommender system, making an assumption, that new ratings fill "holes" in triclusters, satisfying the condition of similar values.

		NOAC		Tri-K-Means	
# triples	# triclusters	Avg. Variance	Avg. Density	Avg. Variance	Avg. Density
10 000	13	0.4831	0.5269	1.0447	0.0126
20 000	47	0.6840	0.5208	0.8264	0.0717
30 000	101	0.7947	0.5399	0.8152	0.1057
40 000	153	0.7667	0.5436	0.7006	0.1253
50 000	259	0.8186	0.5509	0.6286	0.2041
60 000	372	0.8240	0.5471	0.5657	0.2300
70 000	618	0.8200	0.5434	0.4854	0.3190
80 000	864	0.8265	0.5471	0.4339	0.3777
90 000	1135	0.8545	0.5508	0.4303	0.3929
100 000	1421	0.8672	0.5527	0.4087	0.4422

Table 3: NOAC and Tri-K-Means results on the GroupLens dataset

9 Conclusion

This work consists of two main parts. The first one is dedicated to the methods of OAC-triclustering. We described two kinds of tricluster generating operators. One of them has an alternative definition that yields four different variations of the operator. We investigated the relations between the operators and proposed a set of ordering lemmas that establish a nesting order of triclusters generated by the operators. These findings theoretically prove advantages of certain operators over the others.

In the second part of this work, numerical extension of OAC-triclustering was studied. We proposed two algorithms for finding triclusters of similar values in real-valued triadic contexts. One is based on prime operators from the previous part. The other is a well-known clustering algorithm K-Means equipped with a specific metric in order to operate in numerical triadic contexts. The experiments have revealed weak and strong sides of the proposed algorithms.

In our further studies, we would like not only to compare the proposed approach with a modified version of TriMax for triadic numeric data from [9] and Fenster from [14], but also to address such application as collaborative filtering

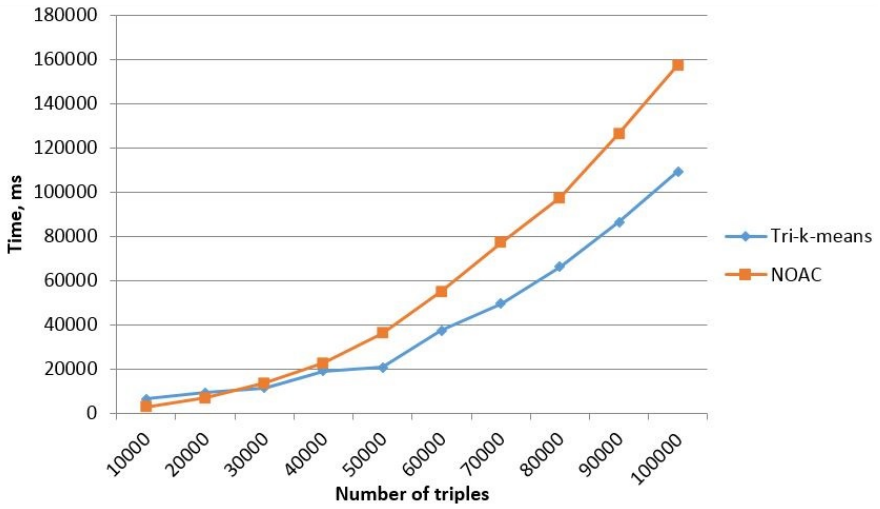


Fig. 4: Execution time against number of triples

in recommender systems [20, 12] and gene expression analysis [18] where numeric triclusters are used.

Acknowledgments We would like to thank our colleagues, Bernhard Ganter, Sergei Kuznetsov, Boris Mirkin, Rokia Missaoui, Loïc Cerf, Jean-Francois Boulicaut, Amedeo Napoli, Lhouari Nourine, Mehdi Kaytoue and Sadok Ben Yahia for their piece of advice and useful prior communication.

The article was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'. The study was implemented in the Laboratory of Intelligent Systems and Structural Analysis.

This work was partially supported by the French LabEx project IMobS3, Innovative Mobility: Smart and Sustainable Solutions. The second author was also partially supported by Russian Foundation for Basic Research.

References

1. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3) (1966) 279–311
2. Cichocki, A., Lee, N., Oseledets, I.V., Phan, A.H., Zhao, Q., Mandic, D.P.: Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. Foundations and Trends in Machine Learning **9**(4-5) (2016) 249–429
3. Mirkin, B.: Mathematical Classification and Clustering. Kluwer, Dordrecht (1996)

4. Hartigan, J.A.: Direct Clustering of a Data Matrix. Journal of the American Statistical Association **67**(337) (1972) 123–129
5. Belohlávek, R., Glodeanu, C.V., Vychodil, V.: Optimal factorization of three-way binary data using triadic concepts. Order **30**(2) (2013) 437–454
6. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: TRIAS - an algorithm for mining iceberg tri-lattices. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China. (2006) 907–911
7. Ignatov, D.I., Kuznetsov, S.O., Magizov, R.A., Zhukov, L.E.: From triconcepts to triclusters. In: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing - 13th International Conference, RSFDGrC 2011, Moscow, Russia, June 25-27, 2011. Proceedings. (2011) 257–264
8. Besson, J., Robardet, C., Raedt, L.D., Boulicaut, J.: Mining bi-sets in numerical data. In: Knowledge Discovery in Inductive Databases, 5th International Workshop, KDID 2006, Berlin, Germany, September 18, 2006, Revised Selected and Invited Papers. (2006) 11–23
9. Kaytoue, M., Kuznetsov, S.O., Macko, J., Napoli, A.: Biclustering meets triadic concept analysis. Ann. Math. Artif. Intell. **70**(1-2) (2014) 55–79
10. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.: Closed patterns meet n-ary relations. TKDD **3**(1) (2009) 3:1–3:36
11. Nataraj, R.V., Selvan, S.: Article:closed pattern mining from n-ary relations. International Journal of Computer Applications **1**(9) (February 2010) 9–13 Published By Foundation of Computer Science.
12. Trabelsi, C., Jelassi, N., Yahia, S.B.: Scalable mining of frequent tri-concepts from folksonomies. In: Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Proceedings, Part II. (2012) 231–242
13. Ignatov, D.I., Gnatyshak, D.V., Kuznetsov, S.O., Mirkin, B.G.: Triadic formal concept analysis and triclustering: searching for optimal patterns. Machine Learning **101**(1-3) (2015) 271–302
14. Cerf, L., Besson, J., Nguyen, K., Boulicaut, J.: Closed and noise-tolerant patterns in n-ary relations. Data Min. Knowl. Discov. **26**(3) (2013) 574–619
15. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings. (1995) 32–43
16. Krolak-Schwerdt, S., Orlik, P., Ganter, B. In: TRIPAT: a Model for Analyzing Three-Mode Binary Data. Springer Berlin Heidelberg, Berlin, Heidelberg (1994) 298–307
17. Voutsadakis, G.: Polyadic concept analysis. Order **19**(3) (2002) 295–304
18. Zhao, L., Zaki, M.J.: Tricluster: An effective algorithm for mining coherent clusters in 3d microarray data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005. (2005) 694–705
19. Li, A., Tuck, D.: An effective tri-clustering algorithm combining expression data with gene regulation information. Gene regulation and systems biology **3** (2009) 49–64
20. Nanopoulos, A., Rafailidis, D., Symeonidis, P., Manolopoulos, Y.: Musicbox: Personalized music recommendation based on cubic analysis of social tags. IEEE Trans. Audio, Speech & Language Processing **18**(2) (2010) 407–412

21. Jelassi, M.N., Yahia, S.B., Nguifo, E.M.: A personalized recommender system based on users' information in folksonomies. In: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume. (2013) 1215–1224
22. Gnatyshak, D., Ignatov, D.I., Semenov, A., Poelmans, J.: Gaining insight in social networks with biclustering and triclustering. In: Perspectives in Business Informatics Research - 11th International Conference, BIR 2012, Nizhny Novgorod, Russia, September 24-26, 2012. Proceedings. (2012) 162–171
23. Fazendeiro, P., de Oliveira, J.V.: Observer-biased analysis of gene expression profiles. In: Big Data Analytics in Bioinformatics and Healthcare. IGI Global (2015) 117–137
24. Ignatov, D.I., Kuznetsov, S.O., Poelmans, J.: Concept-based biclustering for internet advertisement. In: 12th IEEE International Conference on Data Mining Workshops, ICDM Workshops, Brussels, Belgium, December 10, 2012. (2012) 123–130
25. Gnatyshak, D., Ignatov, D.I., Kuznetsov, S.O.: From triadic FCA to triclustering: Experimental comparison of some triclustering algorithms. In: Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013. (2013) 249–260
26. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. TiiS **5**(4) (2016) 19:1–19:19

Classification of Demographic Sequences Based on Pattern Structures and Emerging Patterns

Danil Gizdatullin[1]✉(0000 − 0001 − 8291 − 3724), Dmitry I. Ignatov[1]✉(0000 − 0002 − 6584 − 8534), Ekaterina Mitrofanova[1] (0000 − 0002 − 3322 − 5922), and Anna Muratova[1]

National Research University Higher School of Economics, Moscow, Russia
{dgizdatullin,dignatov,emitrofanova,amuratova}@hse.ru

Abstract. We present recent results of studies in application of sequence-based pattern structures and emerging patterns to analysis of demographic sequences in Russia. This study is performed on data of 11 generations from 1930 till 1984 for the panel of three waves of the Russian part of Generation and Gender Survey, which took place in 2004, 2007, and 2011. The main goal is to develop methods of extracting emerging patterns (EP) with the following restrictions: the obtained patterns need to be (closed) frequent contiguous prefixes of the input sequences. These constraints were required by demographers for proper interpretation and understanding of early life course events that lead to adulthood. To fulfil this requirement we used modified FP-trees [1][1] based on pattern structures of contiguous prefixes. After extraction of EP we use CAEP[2] classifier to predict gender of respondents using their demographic sequences of the first life course events. The best results in terms of TPR-FPR have been obtained for large values of minimum growth-rate parameter (with some objects left without classification).

Keywords: demographic sequences, pattern structures, sequence mining, emerging patterns, emerging sequences, machine learning

1 Introduction and related work

The analysis of demographic sequences is a popular and promising direction of study in demography [2,3]. The life courses of people consist of the chains of events in different spheres of life. Scientists are interested in the transition from the analysis of separate events and their interrelation to the analysis of the whole sequences of the events. However, this transition is slowing by the technical peculiarity of working with sequences. As of today, demographers and sociologists do not have an available and simple instrument of such analysis.

[1] to specify our version we refer to it as frequent prefix trees rather than frequent pattern ones

[2] Classification by Aggregating Emerging Patterns

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

Some demographers possessing programming skills are successfully making sequence analysis [4,5,6] and developing statistical methods [7,8,9,10,11], but the majority of the social scientists have the only option to cooperate with other scientists to extract knowledge from demographic data. Commonly, demographers rely on simple statistics, but sophisticated sequence analysis techniques only start to emerge in this field [12]. Since traditional statistical methods used in the domain cannot face the emerging needs of demography, demographers start showing a great interest in techniques from machine learning and pattern mining [13,14].

Human demographic behaviour can be very different varying over generations, gender, education level, religious views etc., however, hidden similarities can be found and generalised by specially designed techniques.

In our previous paper [14], we used SPMF[3] [15] for mining frequent sequences and our program to find emerging patterns in them. However, the implemented sequence mining techniques rely on the definition of subsequence in general: it brings results which are hard to interpret. Our colleagues, demographers, asked us to find contiguous prefixes patterns because they are interested in full parts of people's life trajectories without gaps.

So, one of the goals of this study is to develop methods of extracting emerging patterns with the following restrictions: the obtained patterns need to be (closed) frequent contiguous prefixes of the input sequences.

The main goal of our paper was to find rules (patterns) that discern demographic behaviour of different groups of people. The classification itself is rather a means but not the goal. Thus good classification results only conviced us that the prefix-based classifier is applicable to the problem. From this point of view, black-box approaches like SVM and artificial neural networks do not match the task; they can be better in prediction but do not produce interpretable patterns.

The paper is organized as follows. In Section 2, we describe our demographic data. Section 3 introduces basic definitions and prefix-based contiguous subsequences in terms of pattern structures combined with emerging patterns. Experimental results are reported in two subsections of Section 4. Section 5 concludes the paper. Some implementation details can be found in Appendix.

2 Problem Statement and Demographic Data

The dataset for the study is obtained from the Research and educational group for Fertility, Family formation and dissolution of HSE [4]. We use the panel of three waves of the Russian part of Generation and Gender Survey (GGS), which took place in 2004, 2007 and 2011 [5]. The dataset contains records of 4857 respondents (1545 men and 3312 women). The gender imbalance of the dataset is caused by

[3] Sequential Pattern Mining Framework: `http://www.philippe-fournier-viger.com/spmf/`

[4] `http://www.hse.ru/en/demo/family/`

[5] This part of GGS "Parents and Children, Men and Women in Family and in Society" is an all Russia representative panel sample survey: `http://www.ggp-i.org/`

the panel nature of the data: the leaving of the survey by the respondents is an uncontrollable process. That is why the representative waves combined in a panel with the structure less close to the general sample.

In the database, for each person the following information is indicated: date of birth, gender (male, female), generation, type of education (general, higher, professional), locality (city, town, village), religion (yes, no), frequency of religious events' attendance (once a week, several times in a week, minimum once a month, several times in a year or never) and the date of significant events in their lives such as: first job experience, completion of education of the highest level, leaving the parental house, first partnership, first marriage, birth of the first child. There are eleven generations: from first (those who was born in 1930–34) through eleventh (1980–84) generations.

There is a variety of questions that demographers would like to answer, for example:
– What is the difference between men and women in terms of demographic behaviour?
– What is the difference between generations in terms of demographic behaviour?

There are many different variations of similar questions and all they in fact need a proper means of pattern mining to answer.

3 Sequence mining and emerging patterns

3.1 Basic definitions

A **prefix-based contiguous subsequence** (or prefix) of a sequence $s = \langle s_1, \ldots, s_k \rangle$ is the sequence $s_1 = \langle s'_1, \ldots, s'_{k'} \rangle$ where $s_i = s'_i$ for $i \leq k' \leq k$.
The **relative support**, $rsup(s, T)$, of a prefix subsequence s in the set of sequences T is the number of sequences in T that start with s divided by $|T|$.

Formally, given a set of sequences, where each sequence is a list of itemsets ordered by time, the problem amounts to find all frequent prefix-based contiguous subsequences that appear a sufficient number of times, i.e. greater than a user-specified minimum support threshold ($minsup$) [16].
A **frequent prefix** is a prefix-based contiguous subsequence which satisfies minimum support threshold (minsup).
A **frequent closed prefix** is a frequent prefix-based contiguous subsequence such that it is not a prefix of another prefix with the same support.

Let us introduce several definitions related to Pattern Structures.

Pattern structures were introduced in paper [17]. They expand capabilities of Formal Concept Analysis to more complex structures than a binary context.

A triple $(G, \underline{D}, \delta)$ is a **pattern structure** where G is a set of objects, $\underline{D} = (D, \sqcap)$ is a complete meet-semilattice of descriptions, and $\delta : G \to D$ maps an object to its description.

The usage of pattern structures for sequence mining has already been successfully demonstrated in [18].

The lattice operation in the semilattice $\sqcap$ corresponds to the similarity between two descriptions.

The Galois connection for a pattern structure $(G, (D, \sqcap), \delta)$ is defined as follows:

$$A^\diamond := \bigsqcap_{g \in A} \delta(g), \text{ for } A \subseteq G \text{ and } d^\diamond := \{g \in G | d \sqsubseteq \delta(g)\}, \text{ for } d \in D.$$

As in FCA, the Galois connection makes a correspondence between sets of objects and descriptions which are more complex than subsets of attributes. Given a subset of objects A, operation $A^\diamond$ returns the description that is common to all objects in A. Given a description d, $d^\diamond$ is the set of all objects whose description subsumes d. More precisely, the partial order on $D(\sqsubseteq)$ is defined w.r.t. the similarity operation $\sqcap$: $c \sqsubseteq d \Leftrightarrow c \sqcap d = c$.

A **pattern concept** of the pattern structure $(G, (D, \sqcap), \delta)$ is a pair (A, d) where $A \subseteq G$, $d \in D$, and $A^\diamond = d, d^\diamond = A$; A is called the (pattern concept) extent and d is called the pattern (concept) intent.

3.2 Pattern structures in the demographic context

Let us introduce $\sqcap$ operation which returns the maximal common prefix of two sequences. Let S be a set of sequences and D be the set of all their prefix-based descriptions.

Let $(S, (D, \sqcap), \delta)$ be a pattern structure related to our demographic problem.

For each $s \in S = \{1, \ldots, n\}$ operation $\delta(s)$ returns the sequence description of object s. For example, if we have two sequences s_1 and s_2:

$$s_1 = \langle a, b, c, d \rangle \text{ and } s_2 = \langle a, b, c, e \rangle.$$

If we apply operation $\diamond$ to subset of sequences from S, then we will obtain the maximal common prefix for these sequences. For example, for we have

$$\{s_1, s_2\}^\diamond = \langle a, b, c \rangle.$$

The operation $\diamond$ on description $d = \langle a, b, c \rangle$ in our case is given as

$$d^\diamond = \{s_1, s_2\} = \{s \in S \mid d \sqsubseteq s\},$$

$d \sqsubseteq s$ means that d is the prefix of subsequence s. In other words, that operation $\diamond$ applied to the description d returns subset of objects from S that have the prefix subsequence d.

Let (A, d) be a pattern concept of the pattern structure $(S, (D, \sqcap), \delta)$ with

$$A^\diamond = d, d^\diamond = A \text{ where } A \subseteq S \text{ and } d \in D.$$

Let us discuss the representation of pattern concepts in the related prefix-tree (see section 3.5) for the original pattern structure and how they can be found.

As an example consider the set of sequences, S:

$$s_1 : \langle a, b, c, d \rangle;$$

$$s_2 : \langle a, b, c \rangle;$$

$$s_3 : \langle a, b, d \rangle.$$

Let us draw the corresponding prefix-tree in Fig. 1.

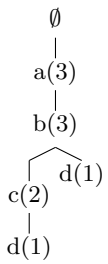


Fig. 1. The prefix-tree representation of S; the support of an event is shown in parenthesis in the related node

We can distinguish pattern concepts relevant to this example:

$$(\{s_1, s_2, s_3\}, \langle a, b \rangle);$$

$$(\{s_1, s_2\}, \langle a, b, c \rangle);$$

$$(\{s_1\}, \langle a, b, c, d \rangle);$$

$$(\{s_3\}, \langle a, b, d \rangle).$$

We can say that any path from the root to the top node, which support is more than the support of its descendants, corresponds to the concept of the original pattern structures.

3.3 Hypotheses as pattern intents

Let us recall the classification problem in terms of pattern structures [17]. For each object (individual) there is a target attribute (e.g., gender, for binary classification problem) on which we want to classify that individual. Then we split our pattern structures in two, positive and negative ones, respectively: $K_\oplus = (S_\oplus, \underline{D}, \delta_\oplus)$ and $K_\ominus = (S_\ominus, \underline{D}, \delta_\ominus)$. Also, we have examples S_τ with unknown value of the target attribute.

Now the Galois closure operators are denoted as $A^\oplus, A^{\oplus\oplus}$, respectively, for the positive pattern context; similarly for the negative one.

Let us define the positive and negative hypotheses. A pattern intent of the pattern structure $K_\oplus$ ($K_\ominus$) $H \sqsubseteq D$ is a positive (negative) hypothesis if H is

not a subset of the pattern intent of any negative (positive) examples $s \in S_\ominus$ ($s \in S_\oplus$):

$$\forall s \in S_\ominus (s \in S_\oplus) : H \not\sqsubseteq s^\ominus (H \not\sqsubseteq s^\oplus).$$

Eventually, the hypothesis is the pattern intent of a formal concept, which is found only in the objects of only one class.

Let $h_\oplus$ and $h_\ominus$ be hypotheses of positive and negative pattern structures, respectively. When we look at a new object, we have four scenarios:

- $\exists h_\oplus, h_\oplus \sqsubseteq \delta_\tau(g_\tau)$, and $\nexists h_\ominus, h_\ominus \sqsubseteq \delta_\tau(g_\tau)$ then a new object is classified as positive;
- $\nexists h_\oplus \sqsubseteq \delta_\tau(g_\tau)$, and $\exists h_\ominus, h_\ominus \sqsubseteq \delta_\tau(g_\tau)$ then a new object is classified as negative;
- $\exists h_\oplus \sqsubseteq \delta_\tau(g_\tau)$, and $\exists h_\ominus, h_\ominus \sqsubseteq \delta_\tau(g_\tau)$ then the object is classified contradictory, i.e. we can not say to which class it belongs to;
- $\nexists h_\oplus \sqsubseteq \delta_\tau(g_\tau)$, and $\nexists h_\ominus \sqsubseteq \delta_\tau(g_\tau)$ then the object remains undetermined, i.e. we can not say to which class it belongs to.

3.4 Emerging patterns in pattern structures

Also, we introduce the notion of emerging prefix subsequences in terms of pattern structures. Such emerging pattern is specific for one class, but not specific for its counterpart.

This feature is implemented via the ratio of the pattern supports for different classes. This ratio is called **growth rate**. Then the growth rate of a pattern p on the positive and negative pattern structures of $K_\oplus$ and $K_\ominus$ is defined as:

$$GR(p, K_\oplus, K_\ominus) = \frac{rsup_{K_\oplus}(g)}{rsup_{K_\ominus}(g)}.$$

Patterns are selected by specifying a minimum growth rate as in [19]. That is, we set the minimum ratio of growth, for which we want to select patterns:

$$GR(p, K_\oplus, K_\ominus) \geq \theta.$$

For each class we compute its score as follows. Let s be a new object which we want to classify, then score for positive class is equal to:

$$score_\oplus(s) = \frac{\sum_{p \in P, p \sqsubseteq \delta(s)} GR(p, K_\oplus, K_\ominus)}{median(GR(S_\oplus))}.$$

We have selected the set of patterns P based on the minimal growth rate. Then we sum GR-s over all prefix-based patterns $p \in P$ that are included in the description for the new object and get the score. Further we normalize score by dividing to the median of growth-rate for the current class. This scheme relates to the method in the article [20].

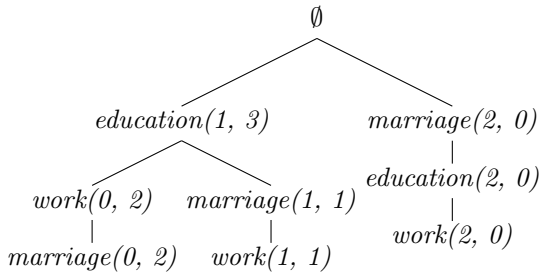
3.5 Prefix-tree structure

As we need to find prefix subsequences it is a good idea to use a prefix-tree [21]. Usually, a node in a prefix-tree is associated with a certain string but in our tree structure each node is associated with only one symbol (in case there are no simultaneous events).

Let us have a look at the example of sequences.

Men sequences:	Women sequences:
$\langle\{education\},\{work\},\{marriage\}\rangle$	$\langle\{education\},\{marriage\},\{work\}\rangle$
$\langle\{education\},\{work\},\{marriage\}\rangle$	$\langle\{marriage\},\{education\},\{work\}\rangle$
$\langle\{education\},\{marriage\},\{work\}\rangle$	$\langle\{marriage\},\{education\},\{work\}\rangle$

Now we build a prefix-tree for these sequences. In each node we store the event and the number of sequences associated with this prefix for Class 0 (woman) and Class 1 (man).



For example, we can see that according to the prefix-tree one woman and three men share prefix $\langle\{education\}\rangle$. All three men's sequences start with *education* and one women's sequence starts with *education*. As for the prefix

$\langle\{education\},\{marriage\},\{work\}\rangle$

according to the prefix-tree, one man and one woman have this prefix. Based on this tree-like structure we can efficiently compute support and growth-rate of each sequence.

4 Experiments and results

To perform experiments with pattern-based classification, we use Python and Contiguous Sequences Analysis library implemented by the first author[6].

4.1 Classification by gender

After discussion with demographers, we have set minimal relative support to 0.09. We have received the following prefix-based contiguous patterns that meet a minimum of 9% of all respondents:

[6] `https://github.com/DanilGizdatullin/ContiguousSequencesAnalysis`

Table 1. Women's patterns

Pattern	$rsupp$
$\langle\{work\}\rangle$	0.287
$\langle\{work\},\{education\}\rangle$	0.120
$\langle\{separation\}\rangle$	0.283
$\langle\{education\}\rangle$	0.239
$\langle\{education\},\{work\}\rangle$	0.168
$\langle\{separation\},\{education\}\rangle$	0.110
$\langle\{sep...\},\{edu...\},\{work\}\rangle$	0.097

Table 2. Men's patterns

Pattern	$rsupp$
$\langle\{work\}\rangle$	0.329
$\langle\{work\},\{education\}\rangle$	0.155
$\langle\{separation\}\rangle$	0.266
$\langle\{education\}\rangle$	0.276
$\langle\{education\},\{work\}\rangle$	0.103
$\langle\{separation\},\{education\}\rangle$	0.199
$\langle\{sep...\},\{edu...\},\{work\}\rangle$	0.099

By thoughtful inspection, we can conclude that the beginning of human life trajectories do not depend strongly on the gender and the beginning of the most popular paths are the same for both sexes.

We have split all our data into two groups: a training set and a test set as a percentage of 66.5%–33.5%.

We have selected the same minimum support threshold for both classes, 0.004; it means that the pattern must be found in trajectories at least five men and nine women. Then we made classification with different minimum threshold values for growth rates $\{1.5, 2, 5, 7\}$ for men and $\{1.5, 2, 5, 7, \infty\}$ for women.

The graphs below show the results and skyline in TPR-FPR (true positive rate, false positive rate), TPR-NCPR (true positive rate, non-classified positive rate), NCPR-FPR (non-classified positive rate, false positive rate) on axes (Fig. 2,3).

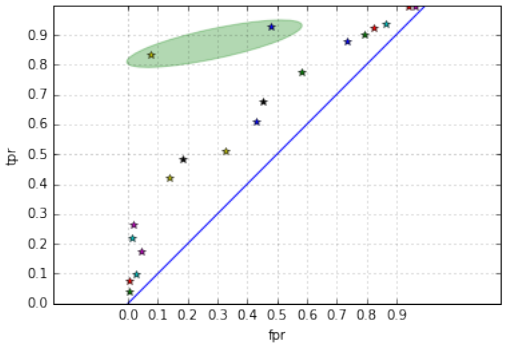


Fig. 2. TPR-FPR plot with two of Pareto-optimal results from the skyline in the oval.

We have chosen the same value for both the minimum supports of two classes, 0.004. It means that the pattern must share at least five men and nine women. We have performed several classifications with different minimum values of the growth rate from $\{1.2, 1.5, 2, 3, 5, 7, \infty\}$.

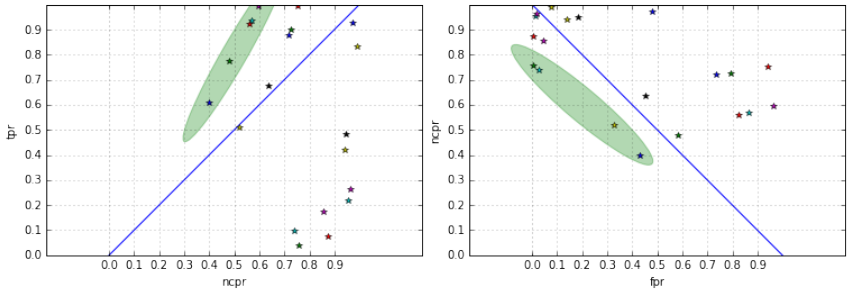


Fig. 3. TPR-NCPR and NCPR-FPR plots along with their skyline points in the ovals.

Since in demographic setting, it is important to identify interesting discriminative patterns, we do not try to solve the problem of classification per se. Thus, several objects from the test set have not been assigned to any class. For example, in the experiment with the best TPR-FPR metric, we cover over 1% of people in the test sample. It can be concluded from the results obtained that the interesting discriminative patterns for some class relative to the other have a small cover. Moreover, we can conclude the average behavior of men and women has not that strong differences in general, but there are local groups of both classes that behave sufficiently different.

The best quality of classification have been reached with the minimum value of the growth-rate (7, ∞). It corresponds to the following emerging patterns:

Table 3. Women's patterns in the test set

Pattern	Growth rate	$rsupp$
$\langle\{work, separation\}, \{marriage\}, \{children\}, \{education\}\rangle$	∞	0.006
$\langle\{separation, partner\}, \{marriage\}\rangle$	∞	0.005
$\langle\{separation, partner\}, \{marriage\}\rangle$	∞	0.005
$\langle\{work, separation\}, \{marriage\}, \{children\}\rangle$	∞	0.008
$\langle\{work, separation\}, \{marriage\}\rangle$	∞	0.009

We have got seven sequences that separate men and women most differently: four of them identify women, the remaining three describe men. The growth rate ∞) shows that all the prefixes for women are typical only for them; the growth rate for mens sequences varies between 10.6 and 12.7, which means these sequences are 10-12 times more indicative for men than for women.

The professional interpretations given by the third co-author, a demographer, are as follows. Within the sequences identifying women, the first event is "separation", which occurs simultaneously with another event: "work" (three cases out of four) and "partner" (one case out of four). The second event for women is

Table 4. Men's patterns in the test set

Pattern	Growth rate	$rsupp$
$\langle\{education\},\{marriage\},\{work\},\{children\},\{separation\}\rangle$	10.6	0.006
$\langle\{education\},\{marriage\},\{work\},\{children\}\rangle$	12.7	0.007
$\langle\{edu...\},\{work\},\{partner\},\{marriage\},\{sep...\},\{children\}\rangle$	10.6	0.006

"marriage", the third (when it is presented) is "children", and the fourth (when it is presented) is "education".

The presented results for women show that they prefer to start their adult life with separation. Only in one case separation connected with having a partner. In other cases, we have an image of an independent woman, who got a job and left parents. The second step in all the cases is marriage. Here we see how an independent (financially and from parents) woman creates her own family and give a birth to a child. The longest sequence contains an event "finishing an education of the highest level". Only after four important socioeconomic and sociodemographic events our typical woman finishes her education.

The first event in the sequences indicative for men is an education. Unlike women, who obtaining their education only after fulfilling almost all events, men are getting education earlier. It can show not only the priority in education for men and women, but also the difference in the level of the highest step of the finished education: the lower the level, the less the age of obtaining an education.

The second event in the three sequences specifying men is marriage (two cases) or work (one case). As a woman, a man tends to create his family quite soon, but unlike woman, who is already very independent at this step, a man has only an education. Men, whose second event is "marriage", are obtaining their first jobs next and then becoming fathers. In the longest sequence, they are leaving parental houses as the final step in the transition to adulthood.

Men who have "work" as the second event, demonstrate different events in their sequences: they have the first partner, then they get married, leave parental house, and only after all other events become parents.

4.2 Classification by generation

In this experiment, we search for emerging patterns for different generations of the same gender. The first class 0 features people who were born between 1924 and 1959. The second class 1 contains people who were born between 1960 and 1984.

First, let us find emerging patterns patterns for women from different generations. We have 940 women from class 0 and 1364 women from class 1. We need to tune two parameters: the first is minimal support and the second is minimal growth rate.

Let us tune the minimal support parameter (Table 5).

Table 5. Tuning of minimal support for women

minsup	accuracy	TPR	FPR	NCR non-classification rate
0.001	0.682	0.707	0.331	0.255
0.004	0.683	0.703	0.316	0.333
0.01	0.668	0.710	0.332	0.399
0.025	0.660	0.648	0.298	0.540
0.04	0.660	0.616	0.278	0.606
0.05	0.652	0.646	0.312	0.641
0.1	0.651	1.0	1.0	0.884

As we can see, minimal support can change sufficiently only non-classification rate, and slightly affect accuracy, TPR and FPR.

We have chosen 0.004 as a minimal support and tuned minimal growth rate.

Table 6. Tuning of minimal growth rate for women

minGrowthRate	accuracy	TPR	FPR	NCR
1.5	0.683	0.655	0.297	0.102
2	0.692	0.703	0.316	0.333
3	0.766	0.747	0.217	0.684
5	0.751	0.821	0.347	0.848
7	0.777	0.848	0.333	0.891

We have decided to choose minGrowthRate as 2 since it covers 66 percent of test data and also provides good results in terms of accuracy, TPR and FPR.

Since we have had quite many emerging patterns from this data, we consider only patterns with the biggest growth rate and support.

Table 7. Patterns for women of old generation

Pattern	$Growthrate$	$rsupp$
$\langle\{work\}, \{separation\}\rangle$	1.85	0.38
$\langle\{work\}, \{marriage, separation\}\rangle$	3.92	0.08

As we can see from Tables 7-8, the main differences in the behavior of women of different generations are the tendency to obtain education and a tendency to work and then separate from their parents in old generations.

Table 8. Patterns for women of younger generation

Pattern	$Growthrate$	$rsupp$
$\langle\{education\}\rangle$	1.84	0.26
$\langle\{education\},\{work\}\rangle$	3.92	0.08

Let us find emerging patterns for men from different generations. Again we report tuning of minimal support:

Table 9. Tuning of minimal support for men

minsup	accuracy	TPR	FPR	NCR
0.001	0.701	0.667	0.266	0.271
0.004	0.704	0.667	0.262	0.338
0.01	0.723	0.671	0.232	0.442
0.025	0.719	0.651	0.218	0.590
0.04	0.706	0.536	0.165	0.712
0.05	0.718	0.627	0.208	0.764
0.08	0.710	0.0	0.0	0.944

Again, minimal support can change sufficiently only non-classification rate.

We have chosen 0.01 as a minimal support and tuned minimal growth rate.

Table 10. Tuning of the minimal growth rate for men

minGrowthRate	accuracy	TPR	FPR	NCR
1.2	0.638	0.510	0.242	0.050
1.5	0.670	0.591	0.260	0.171
2	0.723	0.671	0.232	0.442
3	0.754	0.627	0.144	0.664
5	0.744	0.625	0.152	0.845
7	0.836	0.808	0.138	0.901

The patterns with the biggest growth rate and support are reported in Tables 11,12.

Table 11. Patterns for men of old generation

Pattern	Growth rate	$rsupp$
$\langle\{work\}, \{marriage, separation\}, \{education\}\rangle$	13.52	0.025
$\langle\{work\}, \{marriage\}, \{separation\}\rangle$	22.87	0.042
$\langle\{work\}, \{marriage\}, \{separation\}, \{education\}\rangle$	∞	0.0208

Table 12. Patterns for men of younger generation

Pattern	Growth rate	$rsupp$
$\langle\{education\}, \{work\}, \{separation\}, \{marriage\}, \{children\}\rangle$	10.58	0.020
$\langle\{education\}, \{work\}, \{separation, partner\}, \{marriage\}\rangle$	8.65	0.016
$\langle\{education\}, \{marriage, separation\}\rangle$	7.69	0.015

As in the previous experiment with the subsample of women, the main difference is a tendency to obtain education; thus, men of new generation demonstrates this tendency.

5 Conclusion

The main result of our work is the fitting and usage of different pattern mining approaches including pattern structures to the analysis of demographic sequences. The following conclusions can be drawn from the first results of this work:

1. In this paper, the usage of sequence analysis methods to problems of demographic trajectories of early life events has been studied.
2. A new method based on pattern structures for the analysis of special type of patterns required by demographers (prefix-based and contiguous) have been proposed and implemented.
3. Patterns of behavior for different classes of respondents were obtained and interpreted for the most recent and clean demographic material available for Russia.
4. Classifier based on pattern structures and emerging patterns have been designed and tested.

According to the demographers involved in the project, the work is very important for the further development of the application of pattern mining for demographic analysis of sequence data. Thus, the next planned steps includes following direction:

– using similarity [22] and kernel-based approaches [23] for demographic sequence mining;
– (sub)sequence clustering, in particular, based on pattern structures;
– pattern mining and rule-based approaches for next event prediction [14] competitive with black box approaches like recurrent neural networks;
– comprehensive trajectory visualisation within cohorts [24];
– analysing sequences of statuses like $\langle\{studying, single\}, \{working, single\}\rangle$;
– analysis of matrimonial and reproductive biographies, migration studies, etc.

Acknowledgments We would like to thank our colleagues, Sergei Kuznetsov, Alexey Buzmakov, Jaume Baixeries, and Mehdi Kaytoue for their piece of advice on pattern structures and sequence mining, Guozhu Dong for the interest to our work with emerging patterns, and to our colleagues from Institute of Demography at the National Research University Higher School of Economics.

The paper was prepared within the framework of the Academic Fund Program at HSE in 2016 (grant No 16-05-0011 "Development and testing of demographic sequence analysis and mining techniques") and supported within the framework of a subsidy granted to the HSE by the Government of the Russian Federation for the implementation of the Global Competitiveness Program. The second author was partially supported by Russian Foundation for Basic Research.

References

1. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery **8**(1) (2004) 53–87
2. Aisenbrey, S., Fasang, A.E.: New life for old ideas: The second wave of sequence analysis bringing the course back into the life course. Sociological Methods & Research **38**(3) (2010) 420–462
3. Billari, F.C.: Sequence analysis in demographic research. Canadian Studies in Population **28**(2) (2001) 439–458.
4. Aassve, A., Billari, F.C., Piccarreta, R.: Strings of adulthood: A sequence analysis of young british womens work-family trajectories. European Journal of Population **23**(3/4) (2007) 369–388
5. Braboy Jackson, P., Berkowitz, A.: The structure of the life course: Gender and racioethnic variation in the occurrence and sequencing of role transitions. Advances in Life Course Research (9) (2005) 55–90
6. Worts, D., Sacker, A., McMunn, A., McDonough, P.: Individualization, opportunity and jeopardy in american womens work and family lives: A multi-state sequence analysis. Advances in Life Course Research **18**(4) (2013) 296–318
7. Abbott, A., Tsay, A.: Sequence analysis and optimal matching methods in sociology: Review and prospect. Sociological Methods & Research (2000)
8. Billari, F., Piccarreta, R.: Analyzing demographic life courses through sequence analysis. Mathematical Population Studies **12**(2) (2005) 81–106
9. Billari, F.C., Fürnkranz, J., Prskawetz, A.: Timing, Sequencing, and Quantum of Life Course Events: A Machine Learning Approach. European Journal of Population **22**(1) (2006) 37–65

10. Gauthier, J.A., Widmer, E.D., Bucher, P., Notredame, C.: How Much Does It Cost? Optimization of Costs in Sequence Analysis of Social Science Data. Sociological Methods & Research **38**(1) (2009) 197–231
11. Ritschard, G., Oris, M.: Life course data in demography and social sciences: Statistical and data-mining approaches. Adv. in Life Course Research **10** (2005) 283–314
12. Gabadinho, A., Ritschard, G., Mller, N.S., Studer, M.: Analyzing and Visualizing State Sequences in R with TraMineR. J. of Stat. Software **40**(4) (4 2011) 1–37
13. Blockeel, H., Fürnkranz, J., Prskawetz, A., Billari, F.C.: Detecting temporal change in event sequences: An application to demographic data. In: Principles of Data Mining and Knowledge Discovery, 5th Eur. Conf., PKDD 2001. (2001) 29–41
14. Ignatov, D.I., Mitrofanova, E., Muratova, A., Gizdatullin, D.: Pattern mining and machine learning for demographic sequences. In: Knowledge Engineering and Semantic Web, KESW 2015, Proceedings. (2015) 225–239
15. Fournier-Viger, P., Lin, J.C., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The SPMF open-source data mining library version 2. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III. (2016) 36–40
16. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. (1995) 3–14
17. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: Harry S. Delugach and Gerd Stumme, editors, Conceptual Structures: Broadening the Base, volume 2120 of Lecture Notes in Computer Science. (2001)
18. Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S.O., Napoli, A., Raïssi, C.: On Projections of Sequential Pattern Structures (with an Application on Care Trajectories). In: 10th Int. Conf. on Concept Lattices and Their Applications. (2013) 199–208
19. Dong, G., Zhang, X., Wong, L., Li, J.: CAEP: classification by aggregating emerging patterns. In: Discovery Science, 2nd Int. Conf., DS '99. (1999) 30–42
20. Dong, G., Li, J.: Efficient mining of emerging patterns: Discovering trends and differences. In: Proc. of the Fifth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining. KDD '99, ACM (1999) 43–52
21. de la Briandais: File searching using variable length keys. In: File searching using variable length keys. Proc. Western J. Computer Conf. pp. 295-298. Cited by Brass. Rene. (1959)
22. Egho, E., Raïssi, C., Calders, T., Jay, N., Napoli, A.: On measuring similarity for sequences of itemsets. Data Min. Knowl. Discov. **29**(3) (2015) 732–764
23. Elzinga, C.H., Wang, H.: Versatile string kernels. Theoretical Computer Science **495** (2013) 50 – 65
24. Jensen, A.B., Moseley, P.L., Oprea, T.I., Ellesøe, S.G., Eriksson, R., Schmock, H., Jensen, P.B., Jensen, L.J., Brunak, S.: Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients. Nature Communications **5** (06 2014) 4022 EP –

Appendix

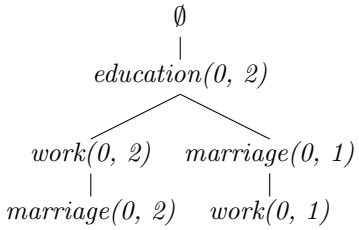
Pseudocode and Complexity

Prefix-tree building. Let us start with the prefix-tree building procedure. As an input we have the set of sequences and their labels. At first we create the

root node. It should be empty. Then we iterate through all sequences and try to match a full path from the root to the end of the sequence. If we meet a dead end having a non-matched suffix subsequence, we create a new node (and all the remaining events in the current sequence should be added as new subsequent nodes). Along the way we increment all counters associated with the traversed nodes, one for each class.

First, we start from sequence $\langle\{education\}, \{work\}, \{marriage\}\rangle$ (Step 1). Then we add the same sequence $\langle\{education\}, \{work\}, \{marriage\}\rangle$ (Step 2). The third sequence in men's data is different $\langle\{education\}, \{marriage\}, \{work\}\rangle$ (Step 3).

Step 3.



Algorithm 1 Sequence tree building

```
 1: procedure SequencesTree(S, L)
 2:     T ← {∅} // Initial prefix tree
 3:     cn ← ∅
 4:     for s ∈ S do
 5:         l = label(s)
 6:         for e ∈ s do
 7:             c ← Find(e, cn.children)
 8:             if c ≠ ∅ then
 9:                 c.counter[l] ← c.counter[l] + 1
10:                 cn ← c
11:             else
12:                 cn.children.append(newC)
13:                 newC.element ← e
14:                 newC.counter[l] ← 1
15:                 cn ← newC

 1: function Find(e, N)
 2:     for n ∈ N do
 3:         if n.element = e then
 4:             return n
 5:         return None
```

Description of the algorithm. Algorithm 1 has two procedures. The main procedure is SequencesTree and its auxiliary function is Find. Find procedure takes an element, e, and nodes, N as input. Each element is a part of an

input sequence and nodes are taken from the tree. Each node in the tree has an element field. It contains a part, single event or complex event (itemset) of a sequence. The goal of this procedure is to return the node with the wanted element. This procedure visits element fields in nodes and compares them to the element that we want to find. SequencesTree procedure is more complex. At first, it creates an empty tree, T (line 2). The current node, cn, is a node we are working at in a particular moment. The first cn is the root node (line 3). Then we iterate over the sequences, S and their associated labels via $label(s)$ from values in L. We start from the first element of a sequence and try to find it in the children of current node (lines 5–7). Now we have two possible options (line 8). The first option is enabled when we find a node with the wanted element. In this case, the counter of the found node for the label of the sequence is incremented (line 9). And the current node changes to its child node with the wanted element, c (line 9). The second option is enabled if we cannot find a node with the wanted element. In this case we create a new child, $newC$ for the current node (line 12), then its element field is set equal to e and the counter is initialised by 1 for the corresponding label (lines 13–14). Finally, the current node is changed to the new node (line 15).

Time complexity. Let n be the size of the training set and m be the number of different events in it.

In line 4, we go through all the data; it takes n times. Then in line 6 we go through all the elements in a sequence. The maximum length of sequence is m. Then in Find procedure we iterate through all children of a node. The maximum number of children nodes is $m-1$. Thus the total time complexity is $O(n \cdot m^2)$. In our case m is a small value (about 7-10 events) and can be considered as a constant. Then the time complexity is $O(n)$.

Space complexity. The space complexity is equal to the number of nodes in a tree. The worst case is when all n sequences are different; i.e. all n sequences do not have the same prefix. In this case space complexity will be $O(n \cdot m)$.

Classification by patterns. After performing of SequencesTree procedure on an input data we have the prefix-tree with absolute support value for each node and label. Then we can classify a new portion of sequences from the same domain. At first we make preprocessing by PrecomputeGrowthRate procedure. In this procedure we compute relative support and growth rate for each node. After that we use ClassifySequence function to predict the label of a new sequence. At first the function initialises $Score$ for $Classes$, sfc, and the current node, cn, variables. Then it iterates through all elements of the sequence (line 4). Next it traverses all children of the current node (line 5) and searches for continuation of the sequence in the prefix-tree (line 6). Then for each label, if this sub-sequence satisfies the minimum thresholds for support and growth rate, we increase the score (lines 7–10). Then the current node is changed to a child node (line 11). And so on until the end of the sequence. Finally, the function returns arg max of $Score$ over $Classes$.

Algorithm 2 Classify Sequence

```
1: function ClassifySequence(T, s, l, Classes, minSup, minGR)
2:     sfc ← [0, 0]
3:     cn ← T.root
4:     for e ∈ s do
5:         for c ∈ cn.children do
6:             if c.element = e then
7:                 for l ∈ Classes do
8:                     if (c.support[l] > minSup) and (c.GR[label] > minGR) then
9:                         sfc[l] = sfc[l] + n.GR[l]
10:                cn ← c
    return argMax(sfc)

1: procedure PrecomputeGrowthRate(T, Classes, soc)
2:     soc=size(Classes) // soc is the number of classes
3:     for n ∈ T do // iterate over the tree nodes
4:         for l ∈ Classes do // iterate over the labels of classes
5:             n.support[l] ← (n.counter[l] / soc[l])
6:     for n ∈ T do
7:         for l ∈ C do // GR is a growth-rate attribute
8:             n.GR[l] ← (n.support[l] / n.support[counterpartL])
```

Time complexity. Let k be the length of a sequence for classification. In PrecomputeGrowthRate we need to iterate through the tree's nodes two times for each class label. We consider the situation where only 2 different classes: $O(n \cdot m)$. In ClassifySequence we iterate through the elements of the sequence and children of nodes for each label: $O(k \cdot m)$.

Direct and Ordered Direct Bases Updates for One-set Extensions of a Closure System

Kira Adaricheva[1] and Taylor Ninesling[2]

[1] Department of Mathematics, Hofstra University, Hempstead, NY 11549, USA
kira.adaricheva@hofstra.edu
[2] Hofstra University, Hempstead, NY 11549, USA
tninesling1@pride.hofstra.edu

Keywords: Closure system, Horn-to-Horn belief revision, Singleton Horn Extension Problem, direct basis, D-basis, ordered direct basis

1 Singleton Horn Extension problem

The dynamic update of evolving knowledge bases and ontologies is a routine procedure in the realm of Artificial Intelligence. These applications require tractable representations, such as Horn logic or various versions of descriptive logic. The interest in Horn logic is easily explained by the fact that the reasoning in Horn logic is effective, while the reasoning in general propositional logic is intractable.

If some knowledge base is represented by a (definite) Horn formula Σ in variables $X = \{x_1, \ldots, x_n\}$, then the set of its models $\mathcal{F}_\Sigma$ forms a lower subsemilattice in 2^X, which is often referred to as a *closure system* on X, or a *Moore family* on X. Alternately, one can associate with Σ a *closure operator* φ on X, so that models from $\mathcal{F}_\Sigma$ are exactly the closed sets of φ. Also, Σ can be interpreted as a set of *implications* defining the closure operator φ. The general connections between Horn formulas (in propositional and first order logic), closure operators and their models were surveyed recently in [1].

The knowledge base requires an update if some of the models expire or the new models need to be incorporated into the existing base. In the current work we tackle the problem of re-writing the Horn formula Σ, when a new model A has to be added to the family $\mathcal{F}_\Sigma$. We will refer to it as the Singleton Horn Extension (SHE) Problem. To avoid misconception, we note that adding set A may result in adding more than just single set to $\mathcal{F}_\Sigma$. Some proper subsets of A may be added as well, which are intersections of A with members of $\mathcal{F}_\Sigma$. This is due to the requirement that the updated family of models must be described by a Horn formula as well, see the classical result in [9]. If the closure operator is encoded through the formal context, the SHE problem arises when adding new rows to the context. The problem was earlier addressed in the general framework of closure systems, including the FCA framework in [11] and the framework of Horn-to-Horn belief revision in [4].

In our work we considered two special cases of the SHE problem: when formula Σ is given by the *canonical direct basis* of implications defining closure

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

operator φ, and when it is given by its refined form of the *D-basis*. We will assume that one needs an algorithmic solution that provides at the output an updated formula $\Sigma^*(A)$ that is canonical direct, or, respectively, the D-basis of the extended closure system.

These two cases will be addressed in the next two sections. The last section is devoted to the results of algorithmic implementations and testing on various closure systems.

2 Update of the canonical direct basis of implications

In [4], the SHE problem was addressed in the case when the formula Σ describing the knowledge base is assumed to be a conjunction of *prime implicates* of the Horn belief set. Translating this into the language of closure systems, one would call Σ *the canonical direct basis*, a type of implicational basis surveyed in [6].

Recall that a formula/implicational set $\Sigma = \{C \to d : C \cup \{d\} \subseteq X\}$ is called *direct* for closure operator φ on X, if, for any $Y \subseteq X$,

$$\varphi(Y) = Y \cup \{d : (C \to d) \in \Sigma, C \subseteq Y\}.$$

In other words, the closure of any subset Y can be computed by checking the bodies (left sides) of implications of Σ with respect to set Y and expanding Y by their consequents when possible. Each implication of Σ is attended only once during the process. Recall that the computation of the closure of Y is generally performed through multiple iteration of Σ and expansion of Y, see the theoretical background in [13], or through the algorithm known as Forward Chaining [7] or LinClosure [8]. The canonical direct basis is the smallest implicational set contained in all direct bases defining the same closure operator φ on X, see [6].

The algorithmic solution for the SHE problem in [4] was given in the form of *body-building* formula $\Sigma(A)$, which was produced given a set of implications/formula Σ that forms the canonical direct basis of a closure system, and a new set A that needs to be added to the closure system $\mathcal{F}_\Sigma$. The formula came up as a consequence to earlier work [10], where the body-building formula was provided to a special extension of the closure system, namely, to the one corresponding to the *saturation operator* φ^* associated with given operator φ. The necessary background for the saturation operator can be found in [5].

In our current work we analyse further the solution for the one-set extension of a closure system. One core observation is the following

Lemma 1. *If Σ is any direct basis for the closure operator φ on X, and A a new set added to closure system $\mathcal{F}_\Sigma$, then $\Sigma(A)$ is also direct.*

It turns out that without the assumption about the directness of Σ, the body-building formula $\Sigma(A)$ may lack implications to define the updated closure system. Also, if Σ is the canonical direct basis, formula $\Sigma(A)$ as given in [4] may not be the canonical direct basis of the updated closure system.

We describe an effective algorithm to produce the modified version $\Sigma^*(A)$ of body-building formula, which is canonical direct , when Σ is canonical direct.

For this aim, we propose the new data structure for Σ that is designed for a quick update should a new set A be incorporated into the closure system. The algorithm builds $\Sigma^*(A)$ in the upgraded data-structure to keep it ready for the fast future updates.

3 Update of the *D*-basis of implications

The other part of the work is devoted to the algorithmic solution for the case when Σ is the D-basis for the closure operator φ and updated formula $\Sigma^*(A)$ is expected to be the D-basis of the expanded closure system.

The D-basis was introduced in [3] as a refined and shorter version of the canonical direct basis: the former is a subset of the latter, while the D-basis still possesses a form of the directness property known as *ordered direct* [3]. The closure of any subset Y can be computed attending the implications of the D-basis Σ only once, when it is done in the specific order.

The part of the basis containing implications $x \to y$, i.e. implications with only two variables from X, is called *binary*, and it plays a special role in the computation of the closures using the D-basis.

We will assume that binary part of the basis Σ^b is transitively closed, i.e. if $a \to b$ and $b \to c$ are in Σ^b, then $a \to c$ is also in Σ^b.

We will denote by $Y_\downarrow$ the set $\{c \in X : (y \to c) \in \Sigma^b \text{ for some } y \in Y\}$.

Then the closure of $Y \subseteq X$ with respect to closure operator φ associated with D-basis Σ is given by some modification of that procedure for the canonical direct basis:

$$\varphi(Y) = Y_\downarrow \cup \{d : (C \to d) \in \Sigma, C \subseteq Y_\downarrow\}.$$

We describe an effective algorithm for the solution of SHE problem, when the implicational basis Σ is the D-basis of the associated closure operator φ, and $\Sigma^*(A)$ is the D-basis of the extended closure system. The update of Σ will require additional procedure we call an *A-lift* before the standard body-building technique applies.

We will also discuss the open problems related to the D-basis and its characterization.

4 Algorithmic solutions and testing

We will present the results of code implementations of two algorithms discussed in section 2 and 3 and their testing on some bench-mark examples developed in earlier code implementations for the D-basis outputs. In [12], the algorithm produces the D-basis, when the input is some set of implications defining the closure operator. In [2], the closure operator is encoded in the context, and the extraction of the D-basis is done by reduction to known solutions of the hypergraph dualization problem.

We will present the bounds of algorithmic complexity and compare them with the actual time distributions based on parameters such as the sizes of the input and output.

References

1. K. Adaricheva, J.B.Nation, *Bases of closure systems*, in Lattice Theory: Special Topics and Applications, v.2, G. Grätzer and F. Wehrung, eds. Springer, Basel, 2016.
2. K. Adaricheva, J.B.Nation, *Discovery of the D-basis in binary tables based on hypergraph dualization*, Theor. Comp. Science **658** (2017), 307–315.
3. K. Adaricheva, J.B. Nation and R. Rand, *Ordered direct implicational basis of a finite closure system*, Disc. Appl. Math. **161** (2013), 707–723.
4. K. Adaricheva, R. Sloan, B. Szörenyi and G.Turan, *Horn belief contraction: remainders, envelopes and complexity*, in *Proc. KR'12*, 107–115.
5. N. Caspard and B. Monjardet, *The lattices of closure systems, closure operators, and implicational systems on a finite set: a survey*, Disc. Appl. Math. **127** (2003), 241–269.
6. K. Bertet and B. Monjardet, *The multiple facets of the canonical direct unit implicational basis*, Theoretical Computer Science **411** (2010), 2155–2166.
7. W. Dowling and J.H. Gallier, *Linear-time algorithms for testing the satisfiability of propositional Horn formulae*, J. Logic Programming **3** (1984), 267–284.
8. H. Mannila and K.J. Räihä, The design of relational databases, Addison-Wesley, 1992.
9. J. McKinsey, *The decision problem for some classes of sentences without quantifiers*, J. of Symbolic logic **8** (1943), 61–76.
10. M. Langlois,R. Sloan, B. Szörenyi and G.Turan, *Horn complements: towards Horn-to-Horn belief revision*, in *Proc. AAAI* 2008, 466–471.
11. S. Rudolph, *Succintness and tractability of closure operator representations*, Theor. Comp. Science **658** (2017), 327–345.
12. E. Rodríguez-Lorenzo, K. Adaricheva, P. Cordero, M. Enciso and A. Mora, *From an Implicational System to its Corresponding D-basis*, in *Proc.CLA* 2015, 217–228.
13. M. Wild, *A theory of finite closure spaces based on implications*, Adv. Math. **108** (1994), 118–139.

Local Generation of AOC-posets: Reducing the Complexity of Conceptual Navigation for SPLE Product Selection

Alexandre Bazin[1], Jessie Carbonnel[2], and Giacomo Kahn[1]

[1] LIMOS & Université Clermont Auvergne, France
giacomo.kahn@isima.fr, contact@alexandrebazin.com
[2] LIRMM, CNRS & Université de Montpellier, France
jessie.carbonnel@lirmm.fr

Abstract. Exploratory search allows to progressively discover a dataspace by browsing through a structured collection of documents. In this paper, we provide techniques to reduce the complexity of FCA-based exploratory search by using of AOC-posets to achieve conceptual navigation. Also, we outline algorithms to enable an on-demand generation of AOC-posets. This work is motivated by the necessity to devise more flexible methods to perform product selection in software product line engineering.

Keywords: Formal Concept Analysis, AOC-poset, Conceptual Navigation, Software Product Line Engineering, Product Selection.

1 Introduction

Exploratory search is an information retrieval strategy that aims at guiding a user into a space of existing documents he is unfamiliar with, to help him select the one that best suits his needs. Lattice structures were among the first structures used to support information retrieval processes [2], and their usage was later generalised to formal concept analysis theory (FCA)[1]. The concept lattice offers a convenient structure to do exploratory search, where navigating from concept to concept by selecting or deselecting attributes (*conceptual navigation*) emulates iterative modifications of the document descriptor selection (the user query), and thus of the current research state. Neighbour concepts (i.e., direct super-/sub-concepts) represent the minimal possible modifications of the current query and therefore permit to navigate through the dataspace by minimal steps [2]. Therefore, a concept lattice depicts all possible queries a user can formulate. However, FCA-based exploratory search raises some problems, mainly because of the size (in terms of number of concepts) of the generated lattices, which are well known to grow exponentially with the size of the input data.

In this document, we propose a new and more scalable approach to perform conceptual navigation, that relies on local generation of AOC-posets, a partial sub-order of concept lattices. This alternative conceptual structure represents and structures the minimal set of queries that are necessary to perform

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

conceptual navigation. We outline algorithms to identify neighbour concepts in AOC-posets, i.e., determining upper and lower covers of a given concept in an AOC-poset. Our work is motivated by an application in the field of software product line engineering, for an activity called product selection.

2 Motivation

Software product line engineering (SPLE) [3] is a development paradigm that aims to efficiently create and manage a collection of related software systems, based on factorisation and exploitation of a common set of artefacts. A central point of SPLE is the modelisation of the common parts and the variants contained in the related software systems, called the *variability*. This variability is represented by variability models, which are the traditional starting points to perform information retrieval operations in SPLE, including product selection, an important task that consists in guiding the user into selecting the functionalities he wants in the final derived software system. A combination of these functionalities respecting all the constraints expressed in the variability model is called a *valid configuration*, and corresponds to a derivable software system.

Current approaches for product selection rely on the variability model's structure to automatically deploy configurators; however, these methods are too stiff considering that they do not allow the user to change his final configuration without having to start again the product selection process, or to see which other configurations are similar to his. We propose to apply exploratory search in the context of product selection to complement these methods and offer a more flexible selection. Reducing the complexity of the underlying conceptual structure along with its generation time is crucial to be able to conceive applications using conceptual navigation in this context.

3 Local Generation of AOC-poset

FCA is a mathematical framework based on the notion of formal context as a condensed representation of a lattice. A *formal context* $K = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ is a triple, where $\mathcal{O}$ is the set of objects, $\mathcal{A}$ the set of attributes and $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ a binary relation. The application of FCA permits to extract from a context K a finite set of *formal concepts* through the use of two *derivation operators* $\cdot' : 2^{\mathcal{O}} \mapsto 2^{\mathcal{A}}, and\ \cdot' : 2^{\mathcal{A}} \mapsto 2^{\mathcal{O}}$. Thus, $O' = \{a \in \mathcal{A} \mid \forall o \in O, (o,a) \in \mathcal{R}\}$ and $A' = \{o \in \mathcal{O} \mid \forall a \in A, (o,a) \in \mathcal{R}\}$. A formal concept C is a pair (E, I) with $E \subseteq \mathcal{O}$ and $I \subseteq \mathcal{A}$, representing a maximal set of objects that share a maximal set of common attributes. $E = I'$ is the concept's *extent* (denoted $Ext(C)$), and $I = E'$ is the concept's *intent* (denoted $Int(C)$). The set of all concepts extracted from K together with the extent set-inclusion order forms a lattice structure called a *concept lattice*. We simplify the representation of intents and extents in the lattice by displaying each attribute (resp. object) only once in the structure, in the lowest (resp. the greatest) concept having this attribute (resp. object). We say that these concepts *introduce* an element. A

concept introducing at least one attribute is called an *attribute-concept* (AC), and a concept introducing at least one object is called an *object-concept* (OC). A concept can introduce both (*attribute-object-concept* (AOC)), or it can introduce neither of them (*plain-concept*). In some applications, one can benefit from only generating the sub-order restricted to the *introducer* concepts instead of the whole concept lattice. This smaller structure (in terms of number of concepts) is called an *Attribute-Object-Concept partially ordered set* (AOC-poset). While a concept lattice can have up to $2^{min(|\mathcal{A}|,|\mathcal{O}|)}$ concepts, the associated AOC-poset cannot exceed $|\mathcal{O}| + |\mathcal{A}|$ concepts.

AOC-posets restrict the set of possible queries a user can formulate to the minimal queries required to perform conceptual navigation. Neighbour concepts in a concept lattice represent minimal possible modifications a user can make to the current query, and therefore offer a dataspace in which one can navigate in minimal steps. Concept lattices allow to select and deselect non-cooccurrent attributes one by one. AOC-posets do not preserve the minimal step query refinement/enlargement property, but factorise the possible query modification steps to keep the most prevalent ones.

Local generation consists in generating only the part of the structure we are interested in. Here, we outline algorithms to retrieve the neighbourhood of a given concept of an AOC-poset. Exploration can start from the top concept (i.e., the most general query), but it is possible that the user already has partial knowledge of the configuration, and it is then necessary to start from a concept in the AOC-poset; the problem is thus to compute the upper and lower covers of a given concept C in the poset.

Let us start with computing the upper covers. We are looking for the smallest ACs or OCs greater than the input. We start out by computing the smallest ACs greater than C. They can be obtained by computing the concepts $(\{a\}', \{a\}'')$ for each attribute $a \in Int(C)$. We remark that $(\{a_1\}', \{a_1\}'') \geq (\{a_2\}', \{a_2\}'')$ if and only if $a_1 \in \{a_2\}''$. As such, the smallest ACs are the ones that are computed from attributes that do not appear in the closures of other attributes. Once we have the smallest ACs, we want to compute the smallest OCs that are between them and C. This means that we are looking for concepts $(\{o\}'', \{o\}')$ such that o is in the extent of one of the ACs we have and $\{o\}' \subset Int(C)$. We remark once again that $(\{o_1\}', \{o_1\}'') \geq (\{o_2\}', \{o_2\}'')$ implies $o_2 \in \{o_1\}''$ and that the closures of some objects give us information on OCs that can't be minimal.

Algorithm 1 computes the upper neighbours of the input concept in the AOC-poset. The first loop computes the closure of single attributes. Each closure allows us to remove attributes that correspond to non-minimal ACs. The resulting set R contains the intents of the ACs that are both super-concepts of C and minimal for this property. The second loop constructs the set O of objects that are in the extent of an element of R but not in the extent of C. The third loop removes the objects of O that cannot possibly be introduced by a superset of C and, finally, the fourth loop removes the objects of O that produce non-minimal OCs. The ACs that are no longer minimal are also removed. Computing the lower covers is done using the same algorithm, exchanging the roles of attributes and objects.

Algorithm 1: UPPER COVERS

Input: A concept C
Output: The upper covers of C in the AOC-poset

$A \leftarrow Int(C)$
foreach $a \in A$ **do**
  $Y \leftarrow \{a\}''$
  $A \leftarrow A \setminus \{Y \setminus \{a\}\}$
$R \leftarrow \{\{a\}''|a \in A\}$
$O \leftarrow \emptyset$
foreach $S \in R$ **do**
  $X \leftarrow S'$
  $O \leftarrow O \cup (X \setminus Ext(C))$
foreach $o \in O$ **do**
  if $o' \not\subset Int(C)$ **then**
    $O \leftarrow O \setminus \{o\}$
foreach $o \in O$ **do**
  $T = \{S|(S \in R) \wedge (o \in S')\}$
  $R \leftarrow R \setminus T$
  $Y \leftarrow \{o\}''$
  if $\exists p \in O$ *such that* $p \in Y$ **then**
    $O \leftarrow O \setminus \{o\}$
$R \leftarrow \{(\{o\}'', \{o\}') \mid o \in O\}$
return R

4 Conclusion

We adress the problem of providing a more scalable and praticable techniques to perform conceptual navigation with formal concept analysis. This is motivated by the problem of product selection, a software product line engineering task that can benefit from these techniques to complement current methods. We used AOC-posets, the concept lattice sub-hierarchy restricted to introducer concepts, as a smaller, condensed alternative to concept lattices that preserve objects and attributes taxonomy. To avoid generating the whole sub-hierarchy, we outline algorithms to enable local generation of AOC-posets by computing the neighbourhood of any concept in the AOC-poset.

References

1. Ganter, B., Wille, R.: Formal concept analysis - mathematical foundations. Springer (1999)
2. Godin, R., Saunders, E., Gecsei, J.: Lattice model of browsable data spaces. Information Sciences 40(2), 89–116 (1986)
3. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering - Foundations, Principles, and Techniques. Springer (2005)

Logical Information Systems for Linguistic Data: TermLis

Annie Foret

IRISA and Université de Rennes1
Campus de Beaulieu 35042 Rennes, France
Annie.Foret@irisa.fr

Abstract. Logical Concept Analysis [1] (LCA) extends the Formal Concept Analysis [4] (FCA) approach. Recently, this approach has been undertaken for terminology, a workflow has been developed to go from XML data to a logical information context. Through experiments on specific resources, facet designs have been tuned to facilitate the search and control on the data. We will consider several usages of such contexts and illustrate benefits of the approach.

1 Introduction

This proposal aims to show how linguistic data can become easier to exploit through the *logical information systems* approach: whereas such data are not always easy to use without assistance or expertise, logical information systems are especially designed to offer a flexible browsing of data when organized as a *logical context*.

TermLis is an application of Logical Information Systems to terminological resources (http://www-semlis.irisa.fr/software/). It has been introduced in [3,2]. It was illustrated initially using the FranceTerme resource[1], containing terms of different scientific and technical fields; we have carried on this development, considering the EuroVoc resource[2], a Multilingual Thesaurus of the European Union. In these data, each *concept* is related to a domain, with an attempt to avoid *ambiguities*.

We transduced parts of the EuroVoc XML data, producing *modular* contexts to be loaded in a *logical context management system*; we use `Camelis`[3] showing three actionable windows: a set of objects, a property index and a textual query (a logical formula). In this process, we consider terms (viewed as terminological concept labels) as objects, and for *the property index*, we favored *semantic facets* (logical properties attached to objects) identical or similar to those initially devised for FranceTerme.

We propose several scenarios, illustrating benefits of the approach, applied on both linguistic data, transduced as logical contexts, then loaded in `Camelis`.

[1] http://www.culture.fr/franceterme, see https://fr.wikipedia.org/wiki/FranceTerme
[2] http://eurovoc.europa.eu/
[3] to our knowledge, it is the only logical context management system, it is available at http://www.irisa.fr/LIS/ferre/camelis/

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

2 Some key aspects

A transducer methodology

In order to adapt the transducer of [2] to the new set of files in XML format, we followed its general workflow shown in Figure 1 (on the left) and the facet modelling in [3]. We also relied on the data schema (in DTD format) of the new resource. We could keep the overall facet structure, as illustrated in the next screenshots in Figure 2. The workfow outputs (modular) logical contexts to be loaded in a Logical Information System.

Logical Information System capabilities

LCA generalizes Database and Hierarchical systems, as illustrated on Figure 1 on the right; the figure also follows the `Camelis` interface that enables three interaction modes and shows synchronized windows: after an initial logical context is loaded, interactions lead to a state/subcontext displayed by: its formula/query on the top, links in the navigation index on the left, and its objects on the right.

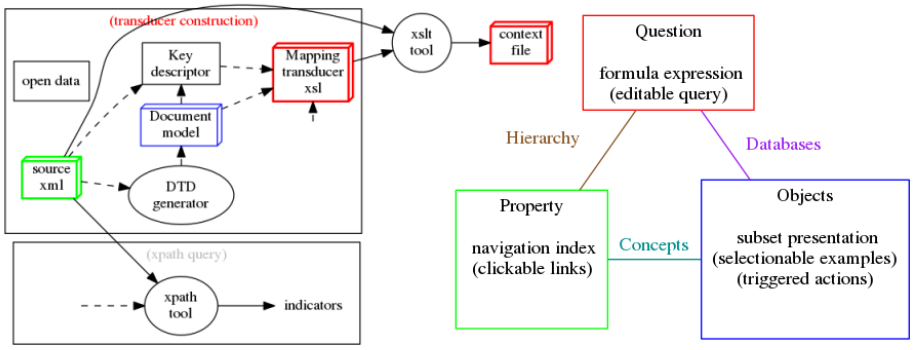


Fig. 1. Workflow architecture (on the left) and LIS-LCA overview (on the right)

3 Logical contexts, multiple facets and search examples

We consider and discuss several usages and interaction scenarios, on the contexts obtained as transductions from terminological resources.

Property index and sub-context cardinalities. In the property index-tree, we may choose an order for displaying a given facet. This is useful for example to read directly which `Equivalent _en` correspond to the greatest number of French terms (`package` on the left part of Figure 3). This highlights potential ambiguities, with a

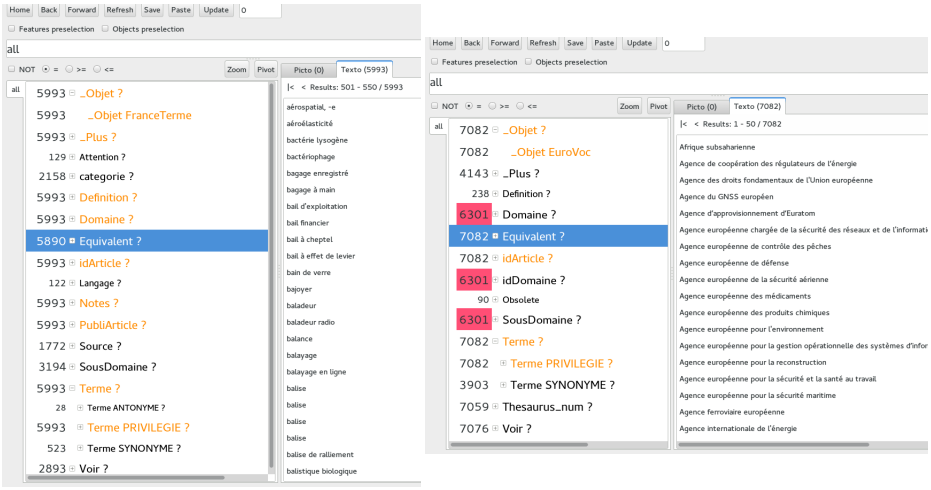


Fig. 2. FranceTerme (left) and EuroVoc (right) TermLis contexts in `Camelis`

degree indicator, using both languages [4]. The dynamic links (domains, etc.) then provide hints to disambiguate, as in Figure 3 on the right, showing a subcontext after a selection in the property index-tree (`Camelis` displays three windows about a same subcontext). Links of a same color characterize the same set of objects (concept) which provides further hints.

Other kinds of ambiguities can be highlighted, such as the `ABS` abbreviation in FranceTerme, with these steps: (1) open a facet and select, moving to a state showing the query: `_Plus EquivalentVariante _en is "ABS"` , and two french terms as objects: `antiblocage de sécurité` whose `Equivalent _en is "Anti-lock Braking System"` and `titre adossé à des créances titrisées` whose `Equivalent _en is "asset-backed security"` (2) open the `Domaine` facet, which displays two domains: `Automobile, Finances.`

We can explore variants and false-friends. Other kinds of search are possible: focused search on elements and exceptions, in some cases the query at the top may act as a useful summary.

Data types. Data types other than attributes and strings can be handled, such as dates, allowing for more or less fine-grained selections. This can be illustrated on FranceTerme publications dates with facet `PubliArticle date`.

4 Refinements and user preferences

We can adapt facets using rules. For example, Domains and SubDomains data have been translated for FranceTerme in a context of update rules of the form:
`rule_extr Domaine is "Acoustique" -> Domain is "Acoustic"`

[4] no ambiguities of that first type are found in EuroVoc (displaying card 1); other kinds on EuroVoc can be shown, for example with `_Plus EquivalentVariante _en is "ESA"`

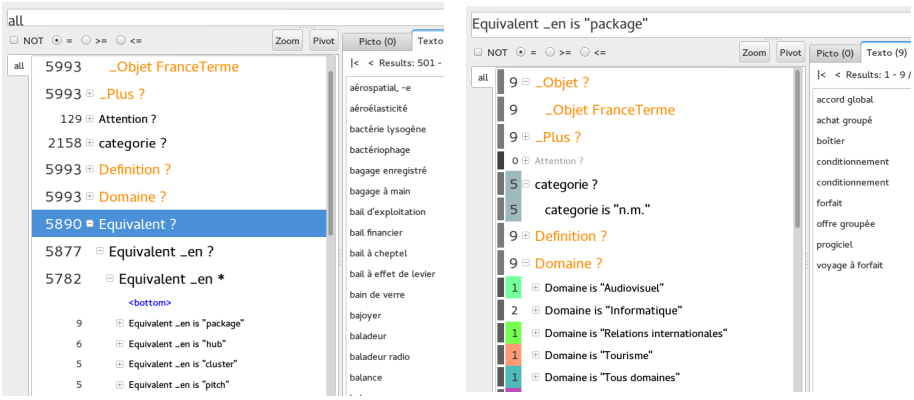


Fig. 3. Facet `Equivalent _en` opened, by card desc - then subcontext after selection (right)

when loaded in the context, properties on the right hand side are added to all objects verifying the left hand side. As another example, the linguistic features in FranceTerme displayed in the `categorie` facet, can be harmonized adding a set of rules and axioms (as a context file for `Camelis`) that render them in a hierarchy of regular categories. The context file modularity also enables to load parts or variants of the context.

Linking data and resources (using actions)

Relations between objects are rendered by properties `Voir` ... ("see") in the index. Another kind is linkage is made through actions associated to objects. We generated connections to: - a parser; a web link to another terminological resource for French (CNRTL, http://www.cnrtl.fr/); an XML link to a subpart of the source file. This action list is not exhaustive and can be adapted.

References

1. Sébastien Ferré and Olivier Ridoux. Introduction to logical information systems. *Inf. Process. Manage.*, 40(3):383–419, 2004.
2. Annie Foret. A logical information system proposal for browsing terminological resources. In Thierry Poibeau and Pamela Faber, editors, *Proceedings of the 11th International Conference on Terminology and Artificial Intelligence, Universidad de Granada, Granada, Spain, November 4-6, 2015.*, volume 1495 of *CEUR Workshop Proceedings*, pages 51–59. CEUR-WS.org, 2015.
3. Annie Foret. Termlis : un contexte d'information logique pour des ressources terminologiques. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, pages 642–643, Caen, France, June 2015. Association pour le Traitement Automatique des Langues.
4. Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.

Mining Relevant Interval Rules

Thomas Guyet[1], René Quiniou[2], and Véronique Masson[3]

[1] AGROCAMPUS-OUEST/IRISA-UMR 6074
[2] Inria, Centre de Rennes
[3] University Rennes-1/IRISA-UMR 6074

Abstract. This article extends the method of Garriga et *al.* for mining relevant rules to numerical attributes by extracting interval-based pattern rules. We propose an algorithm that extracts such rules from numerical datasets using the interval-pattern approach from Kaytoue et *al.* This algorithm has been implemented and evaluated on real datasets.

Keywords: rule learning, interval patterns, relevant rules, closed patterns

1 Introduction

Garriga et *al.* [2] proposed a method to extract relevant association rules from labeled itemsets. We extend the work of Garriga et *al.* to numerical attributes using the pattern mining approach of Kaytoue et *al.* [3] which is based on FCA (Formal Concept Analysis). Kaytoue et *al.* [3] proposed to extend the mining of frequent closed interval pattern to numerical data. Our work bridges the gap between these two approaches to extract relevant interval pattern rules.

2 Closed Interval Patterns

Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a fixed set of n features. We represent a training example as a tuple of real values $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, where $x_i \in Dom(f_i)$, with an associated class label. The tuple stores one value per feature of $\mathcal{F}$. We consider two-class learning problems where the set of examples E is divided in positives (P) and negatives (N) such that $E = P \cup N$ and $P \cap N = \emptyset$. Multi-class problems can be transformed in two-class learning problems.

An n-dimensional interval pattern is a tuple of intervals $\langle [l_i, u_i] \rangle_{i \in [1,...n]}$, where $l_i, u_i \in \mathcal{M}_i \subset \mathbb{R}$, $l_i \leq u_i$ and $\mathcal{M}_i$ is an ordered finite set of modalities (*i.e.* each $\mathcal{M}_i$ is a set of feature values, *i.e.* $Dom(f_i)$, or a subset of values $\mathcal{M}_i \subset Dom(f_i)$). An interval pattern $P = \langle [l_i, u_i] \rangle_{i \in [1,...n]}$ covers a tuple $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, denoted $\boldsymbol{x} \sqsubseteq P$, iff $\forall i \in [1, ...n],\ l_i < x_i \leq u_i$.

Let $X = \langle [l_i, u_i] \rangle_{i \in [1,...n]}$ and $Y = \langle [l'_i, u'_i] \rangle_{i \in [1,...n]}$ be two n-dimensional interval patterns. We define $X \sqcup Y = \langle [\min(l_i, l'_i), \max(u_i, u'_i)] \rangle_{i \in [1,...n]}$. Furthermore, $X \sqsubseteq Y$ iff $\forall i \in [1, ...n],\ [l_i, u_i] \subseteq [l'_i, u'_i]$. This definition extends the previous one for tuple covering considering a value v as a singleton interval $[v, v]$.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

Let $X \rightarrow +$ be a positive rule where X is an interval pattern. True positives are positive examples covered by the rule: $TP(X) = \{\boldsymbol{e} | \boldsymbol{e} \in P \wedge \boldsymbol{e} \sqsubseteq X\}$. False positives are negative examples covered by the rule: $FP(X) = \{\boldsymbol{e} | \boldsymbol{e} \in N \wedge \boldsymbol{e} \sqsubseteq X\}$. True negatives are negative examples not covered by the rule: $TN(X) = \{\boldsymbol{e} | \boldsymbol{e} \in N \wedge \boldsymbol{e} \not\sqsubseteq X\}$. $supp(X)$, the support of pattern X is defined as $supp(X) = |\{\boldsymbol{e} | \boldsymbol{e} \in E \wedge \boldsymbol{e} \sqsubseteq X\}|$. We also define $supp^+(X) = |TP(X)|$ and $supp^-(X) = |FP(X)|$. $supp^+$ is antimonotone w.r.t the $\sqsubseteq$ relation and $supp^-$ is monotone w.r.t $\sqsubseteq$. This means that $\forall X, Y,\ X \sqsubseteq Y,\ supp^+(X) \leq supp^+(Y)$ and $supp^-(Y) \leq supp^-(X)$.

The learning task consists in constructing all interval patterns X such that $supp^+(X) > minsup$ and $supp^-(X) < maxfp$ where $minsup$ and $maxfp$ are given parameters.

From the practical point of view of data mining algorithms, closed patterns are the largest patterns (w.r.t. a partial order $\sqsubseteq$ on the set of patterns, denoted $\mathcal{P}$) among patterns occurring in the exact same set of examples. Formally, a set $X \in \mathcal{P}$ is closed when there is no other set $Y \in \mathcal{P}$ such that $X \sqsubset Y$ (*i.e.* $Y \sqsubseteq X \wedge Y \neq X$) and $supp(X) = supp(Y)$. Closed patterns are interesting because they carry the same information as the total set of frequent patterns.

Kaytoue et *al.* [3] have investigated the problem of mining frequent closed interval patterns with Formal Concept Analysis (FCA). They proposed the MinIntChange algorithm which enumerates all frequent closed frequent patterns. It starts from the most generic interval pattern that covers all the examples: $IP = \langle [\min(\mathcal{M}_i), max(\mathcal{M}_i)] \rangle_{i \in 1 \ldots n}$. Then, each interval pattern is specialized applying minimal changes on the left or on the right of the interval.

3 Mining Relevant Interval-Rules

The theory of relevancy, described in [4], aims mainly at reducing the hypothesis space by eliminating irrelevant features. This theory has been used by Garriga et *al.* [2] to extract relevant features in example database where an example is a tuple of symbolic features. Here, we extend the definition of relevancy of Garriga et *al.* [2] to the relevancy of interval patterns. First, we define two closure operators, Γ^+ and Γ^-, that respectively stand for the closure of interval pattern on P (positive examples) and on N (negative examples).

Definition 1 (Relevancy of an interval pattern) *Let X and Y be two interval patterns. X is more relevant than Y iff $\Gamma^+(Y) = \Gamma^+(X \sqcup Y)$ and $\Gamma^-(X) = \Gamma^-(X \sqcup Y)$.*

Thus, similar results as those of Garriga et *al.* [2] can be deduced about the characterization of the space of relevant interval patterns.

Theorem 1. *Let X and Y be two interval patterns. If $\Gamma^+(Y) = X$ and $Y \neq X$ then Y is less relevant than X.*

Theorem 2. *Let X and Y be two different closed interval patterns such that $X \sqsubset Y$. Then, we have that Y is less relevant than X iff $\Gamma^-(X) = \Gamma^-(Y)$.*

Algorithm 1 Closed interval rule mining algorithm. P is the set of positive examples, N is a set of negative examples and $\mathcal{M}$ is the set of modalities.

1: $FCIP \leftarrow$MinIntChange(P, $\mathcal{M}$)
2: **for** $(X, Y) \in FCIP$ **do**
3: **if** $FP(X) = FP(Y)$ **and** $X \sqsubset Y$ **then**
4: $FCIP \leftarrow FCIP \setminus \{Y\}$
5: **end if**
6: **end for**

The first theorem shows that the relevant rules $X \rightarrow +$ are those for which the interval pattern X is closed over the positive examples. According to the second theorem, in case of similar negative supports, the interval pattern with largest intervals is preferred. Proofs for Theorems 1 and 2 may be deduced from proofs on features sets [2].

Algorithm 1 is based on these theorems to extract the relevant interval patterns. The first step of the algorithm is to extract $FCIP$, the set of frequent interval patterns closed over the positives. Then, line 3 prunes irrelevant patterns in accordance with Theorem 2. For any closed interval pattern $Y \in FCIP$, if there exists another closed interval pattern X such that both have the same support in the negatives (*i.e.* same number false-positives) and such that $X \sqsubset Y$ then Y is removed.

The size of the interval patterns search space is $O\left(m^{2\times n}\right)$ where n is the number of features and m is the number of modalities $\mathcal{M}_i$ of one attribute. Thus, we are facing a memory usage constraint. Keeping all the frequent concept in memory requires a large memory. This memory issue is classically encountered in formal concept analysis but it becomes harder when the number of modalities increases.

To tackle the issue of memory usage, we reduce the modalities to a subset $\mathcal{M}_i$ of a fixed maximal size, defined by parameter *eqmod*. The overall rule mining algorithm has not to be modified. There are several methods to reduce the number of modalities. We choose to extract the equi-probable intervals from the positives examples.

4 Implementation and Results

We evaluated our algorithm on three UCI datasets [1] (*Haberman*, *Iris* and *Vertebral column*). The algorithm is implemented in C++. Experiments are conducted on an Intel Core-I5 with 8GB of RAM with Linux system.

For all experiments in this section, $fpmax = 10\%$ and $eqmod = 10$. Figure 1 illustrates the number of closed interval patterns in positive examples, the number of frequent and accurate rules; and the number of relevant interval pattern rules. We can see that the computing times (see Figure 2) are strongly correlated to the number of patterns.

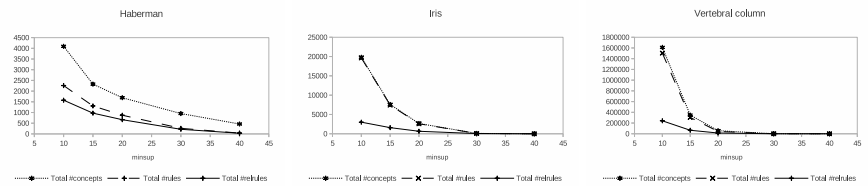


Fig. 1. Number of closed interval patterns in positives, number of rules satisfying $minsup$ and $maxfp$; and number of relevant interval-rules w.r.t. minimal support.

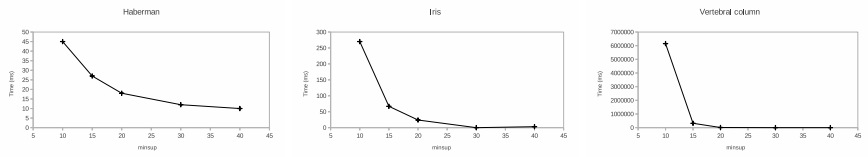


Fig. 2. Computing time (in millisecond) w.r.t. minimal support.

Even for small data such as the Iris dataset, the number of patterns is high for low thresholds (≈ 3000) but the number of relevant patterns is significantly lower than the total number of closed rules. Moreover, the number of patterns increases exponentially with the number of modalities.

5 Conclusions

We have presented a new algorithm for extracting relevant rules from a numerical dataset. It offers a wider choice of possibly interesting rules for experts. The number of extracted patterns is high but more representative of the input dataset whereas standard algorithms such as CN2 or Ripper select a priori a very limited set of rules simply based on covering and accuracy criteria. Future work will be devoted to proposing additional selection criteria which enable the expert to express his/her preferred set of relevant rules.

References

1. K. Bache and M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.
2. Gemma C. Garriga, Petra Kralj, and Nada Lavrač. Closed sets for labeled data. *Journal of Machine Learning Research*, 9:559–580, 2008.
3. Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Revisiting numerical pattern mining with formal concept analysis. In *Proceedings of International Join Conference on Artificial Intelligence (IJCAI)*, pages 1342–1347, 2011.
4. Nada Lavrač and Dragan Gamberger. Relevancy in constraint-based subgroup discovery. In Jean-François Boulicaut, Luc Raedt, and Heikki Mannila, editors, *Constraint-Based Mining and Inductive Databases*, volume 3848 of *Lecture Notes in Computer Science*, pages 243–266. Springer Berlin Heidelberg, 2006.

Navigation and Exploration Tool for Polyadic FCA

Levente Lorand Kis, Christian Săcărea, and Diana Troancă

Babeş-Bolyai University Cluj-Napoca
kis_lori@yahoo.com, csacarea@cs.ubbcluj.ro, dianat@cs.ubbcluj.ro

Abstract. Formal concept analysis (FCA) is a powerful mathematical tool that allows deriving concept hierarchies from large sets of data in order to analyze data and derive meaningful information from it [3]. FCA finds practical application in fields including data mining, text mining, machine learning, knowledge management, semantic web, software development, chemistry, biology and many more. FCA works with contexts and concept lattices derived from them. Since navigation in three-dimensional spaces is rather difficult, one method of navigation in tricontexts uses local projections and reduces the triadic contexts to multiple dyadic contexts. The purpose of this paper is to present `FCA Tools Bundle`, which is a collection of tools for dyadic and triadic formal concept analysis. Furthermore, in this paper, we describe the architecture of the tool and the technologies used in its implementation.

Keywords: FCA, Triadic Context, Navigation Tool, Concept Lattice

1 Introduction

Formal concept analysis is a field of applied mathematics having order theory as a foundation and its main focus towards data analysis. The basic data structure is a formal context described a set of objects, a set of attribute and a relation between them. In order to obtain relevant information about clusters of objects and attributes, the context needs to be parsed and a concept lattice generated. The concept lattice displays the formal concepts of a context in an order diagram which in turn allows analysis and navigation of the data in order to discover more knowledge about it. For triadic contexts this approach becomes more difficult, therefore another method was defined that enables local navigation in such contexts. This navigation method in tricontexts based on local projections is implemented and incorporated in the `FCA Tools Bundle`.

The tools implemented in the `FCA Tools Bundle` are meant to help users analyze large datasets and derive useful information from them.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

2 FCA Tools Bundle

`FCA Tools Bundle`[1] is a web based application which currently implements features for dyadic and triadic formal concept analysis. The main purpose of the tool is to enable the user to visualize formal contexts of different dimensions in several formats and to interact with them. The tool offers some public contexts for users who want to test the functionalities and, in addition, users can import their own dyadic or triadic contexts. The allowed formats for importing a formal context are `cxt` and `csv`. The `csv` file must contain only the tuples of the relation from which the context can be reconstructed. The key features offered by the `FCA Tools Bundle` are: storage and visualization of n-dimensional contexts, generation of the n-adic concepts, dyadic context exploration based on a concept lattice, triadic context exploration using dyadic projections on different perspectives, and exploration of concepts in an n-adic context by user defined constraints.

3 Concept Generation

One of the base functionalities of the website is the storage the contexts. In order to ensure privacy, the uploaded contexts are private by default and bound to the user's account. As mentioned previously, a context can be imported in several formats but it can also be manually created if it is a relatively small context. For larger contexts the import method is preferred for the simple reason that it is faster. Once a context has been saved, it can be accessed at any time and exported in different formats. If the context is made public it can also be accessed and explored by other users. The basic features offered by `FCA Tools Bundle` for a context are visualizing the context details, i.e. object set, attribute set, incidences and the subsequent concept set, and to compute and visualize the concept lattice of the context.

We have implemented two different algorithms for concept generation. For computing formal concepts of dyadic contexts, we use `In-Close2` [1]. However, if one wants to explore multi-dimensional datasets, it becomes more complex to generate fast formal concepts for higher dimensions. Therefore, `FCA Tools Bundle` uses `Data-Peeler` [2] for computing formal concepts of n-adic contexts ($n \geq 3$).

4 Dyadic Context Exploration

The usual and most efficient way of exploring a dyadic context is by computing and drawing its concept lattice. This functionality is implemented in the website using JavaScript in order to present and draw the concept lattice in a simple and efficient way. There are multiple approaches for this task but drawing a "good" concept lattice is difficult and it can be subjective. Some of the aesthetics

[1] fca-tools-bundle.com

criteria by which two concept lattices can be compared are: the number of line crossings, the number of line breaks, angle between incident lines, symmetries, compactness.

There are multiple different methods of drawing a concept lattice, some of which are better for certain contexts while being less efficient on others. Among the methods of representing a concept lattice we recall: *Drawing by hand*, which yields the best results but it is slow and increasingly difficult with the size of a concept lattice. *The layered approach*, which assigns each node a level based on the distance from the top node. From top to bottom each level is reserved a layer on which nodes will be drawn. After that, each node will be drawn on the layer of its respective level. The nodes on a layer are sorted in a manner such that the line crossings between two layers will be minimized. *Force directed approach*, which assigns each node a level much like the layered approach but the position of the nodes and the distance between them is computed based on multiple forces of repulsion and attraction between the nodes. Basically each node has a repulsion force that pushes other nodes away from it while each line between the nodes acts like a spring and draws them close to each other.

`FCA Tools Bundle` uses the force-directed approach because it seems to yield the best results, but it also allows the user to move the nodes freely on their level and pin them into fixed positions in order to increase the visibility and the readability of the concept lattice.

5 Triadic Context Exploration

For a triadic datasets, the trilattice representation of triconcepts is no longer satisfactory. Besides displaying a triconcept list, `FCA Tools Bundle` implements a navigation paradigm based on local projections, which was introduced in a previous paper and has not been implemented by any previous tool [5]. In this navigation paradigm, one can choose a set of elements (from the same set of objects/attributes/conditions) to project on and visualize the concept lattice of the dyadic projection. This functionality can be used to navigate among different projections of a threedimensional dataset.

The first step of the navigation is to choose a triconcept. Next, in order to obtain a local projection, one must choose one of the dimensions to project on, that is: the extent, the intent or the modus. The component along which the projection is done is called the perspective. After choosing a perspective, the projected dyadic context is built and the dyadic concept lattice is computed. It can be proven that every dyadic concept of the projected context corresponds to exactly one triconcept [5].

The navigation is done by selecting one of the dyadic concepts of the dyadic context and computing the associated triadic concept. By choosing different perspectives of that triadic concept one can switch the projection and navigate in a different dyadic concept lattice. Therefore, the whole tricontext can be explored.

6 Concept Search Tool

Besides navigation on dyadic and triadic contexts one other main functionality of `FCA Tools Bundle` is represented by the concept search tool. This tool implements user defined constraints by including and excluding objects/attributes. A key feature is that constraints definition is generalized in a way that works on any n-adic context, even on large datasets. This allows for exploration of very big contexts where one cannot generate the list of all its concepts. The algorithm for the Concept Search Tool was analyzed and described in more details in two previous papers [4, 6].

While it is slow for large contexts it slowly gets faster due to an efficient caching system. The experience is also enhanced by multiple small functionalities like bookmarks which allow the user to bookmark not only a found concept but also the search path used to reach that concept.

7 Conclusions

`FCA Tools Bundle` was built with the idea of providing a user with a wide range of tools for exploring and analyzing contexts via an FCA approach. The tools range from common functionalities, like concept lattice display, to new and innovative ones like the triadic context exploration via dyadic projections or concept search in n-dimensional contexts. The website UI was also built with usability and practicality in mind.

References

1. Andrews, S.: In-Close2, a High Performance Formal Concept Miner. In: Andrews, S., Polovina, S., Hill, R., Akhgar, B. (eds.) Proc. of ICCS 2011. LNCS, vol. 6828, pp. 50–62. Springer (2011)
2. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Closed Patterns Meet n-ary Relations. ACM Transactions on Knowledge Discovery from Data, TKDD 3(1), 3:1–3:36 (2009)
3. Ganter, B., Wille, R.: Formal Concept Analysis - Mathematical Foundations. Springer (1999)
4. Rudolph, S., Săcărea, C., Troancă, D.: Membership constraints in formal concept analysis. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina. pp. 3186–3192. AAAI Press (2015)
5. Rudolph, S., Săcărea, C., Troancă, D.: Towards a Navigation Paradigm for Triadic Concepts. In: Baixeries, J., Săcărea, C., Ojeda-Aciego, M. (eds.) Proc. of ICFCA 2015. LNCS, vol. 9113, pp. 232–248. Springer (2015)
6. Rudolph, S., Săcărea, C., Troancă, D.: Conceptual navigation for polyadic formal concept analysis. In: Mercier-Laurent, E., Kayakutlu, G., Owoc, M.L. (eds.) Proceedings of the 4th Workshop on Artificial Intelligence for Knowledge Management, AI4KM 2016, co-located with the International Joint Conference on Artificial Intelligence, IJCAI 2016. pp. 35–41 (2016)

Program for Conceptual Navigation in Relational Databases

Jens Kötters

IT University of Copenhagen

1 Outline

A program for navigation in relational databases is presented. To enable navigation, the relational database is first transformed into a relational context family [3] by means of conceptual scaling; unlike in [3], we shall also use formal contexts for the representation of relations, thereby treating a relational context family as a many-sorted version of a power context family [6]. Figure 1 shows a relational database, and Fig. 2 shows a relational context family obtained by scaling. The context family has two *object contexts* of sorts `author` and `book` and a *relation context* of the sort `book`×`author`.

Author

name	nationality	date of birth
Lewis Carroll	British	1832-01-27
Virginia Woolf	British	1882-01-25
Douglas Adams	British	1952-03-11
J. K. Rowling	British	1965-07-31
Stephen King	American	1947-09-21
Dan Brown	American	1964-06-22

Book

title	author	publication date
Alice in Wonderland	Lewis Carroll	1865-11-26
To the Lighthouse	Virginia Woolf	1927-05-05
The Hitchhiker's Guide to the Galaxy	Douglas Adams	1979-10-12
Harry Potter and the Deathly Hallows	J. K. Rowling	2007-07-21
The Casual Vacancy	J. K. Rowling	2012-09-27
The Shining	Stephen King	1977-01-28
Doctor Sleep	Stephen King	2013-09-24
The Da Vinci Code	Dan Brown	2003-03-18
Inferno	Dan Brown	2013-03-14

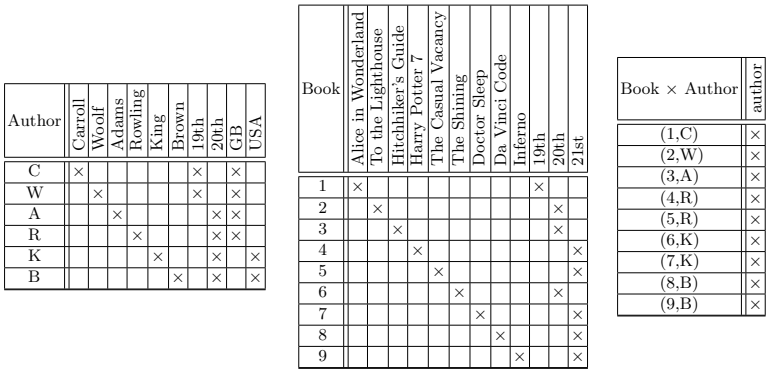
Fig. 1. Database

2 Specifications

2.1 User Interface

The user interface consists mainly of three parts:

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.



Author	Carroll	Woolf	Adams	Rowling	King	Brown	19th	20th	GB	USA
C	×						×		×	
W		×					×		×	
A			×					×	×	
R				×				×	×	
K					×			×		×
B						×		×		×

Book	Alice in Wonderland	To the Lighthouse	Hitchhiker's Guide	Harry Potter 7	The Casual Vacancy	The Shining	Doctor Sleep	Da Vinci Code	Inferno	19th	20th	21st
1	×									×		
2		×									×	
3			×								×	
4				×								×
5					×							×
6						×					×	
7							×					×
8								×				×
9									×			×

Book × Author	author
(1,C)	×
(2,W)	×
(3,A)	×
(4,R)	×
(5,R)	×
(6,K)	×
(7,K)	×
(8,B)	×
(9,B)	×

Fig. 2. Relational Context Family

Intension Graph An attribute-labeled graph formalizing a conjunctive query.
Result Table The result table for the database query expressed by the graph.
Navigation Options Incremental changes to the graph.

Figure 3 (left) shows an intension graph which asks for all 20th-century-born authors who have published in the 21st century (as known to the database). The rectangles are sorted object variables (with the sort stated inside the node) and circles represent relations (the sort is implied by the connections). Rectangles *can* and circles *must* be labeled by one or more attributes of the respective sort context. Each node contains a unique node ID of the form `#n` which serves as a variable name. A valid assignment for the graph would e.g. be `#1=Dan Brown` and `#3=Inferno` (assignments for relation nodes are implicit, here `#2=(Inferno,Dan Brown)`). The *marked* nodes (colored gray) designate the output columns. So the result table for the example graph has only one column (Fig. 3, right). Navigation options are described in Sect. 2.3.

2.2 Program State

If we consider the relational context family fixed, program state is defined by the intension graph exclusively. State transitions are given by the navigation options. Every time the state changes, the result table and navigation options are updated automatically.

2.3 Navigation Options

2.3.1 Refinement Options

R1 *Create Node* Creates an isolated object node of a given sort. This would usually only be used as the first action when building a query. The generated node is marked by default.

R2 *Add attribute* Adds an attribute of the proper sort to an object or relation node.
R3 *Adjoin edge* Creates a new relation node and connects the i-th edge with an existing object node of the proper sort. Each of the remaining edges is connected to a new, unmarked object node.
R4 *Mark node* Includes a column for a given object node in the result table.
R5 *Merge nodes* Identifies two object nodes in the graph.

The attribute labels for each graph node x are collected in the set $\mathrm{def}(x)$. In addition, let $\mathrm{ext}(x)$ denote the set of valid assignments for the node x. The *node context* of a node x of sort s is the subcontext of the sort context (G_s, M_s, I_s) with object set $\mathrm{ext}(x)$ and attribute set $\{m \in M_s \mid \mathrm{ext}(x) \cap m^{I_s} \neq \emptyset \wedge m \notin \mathrm{def}(x)\}$. The attributes of the node context are the attribute choices for action **R2** (note that the result table changes only when the chosen attribute is not common to all objects in the node context). The *extended node context* of a node x of sort s contains in addition a column r_i for each relational attribute r which expects objects of type s in the i-th position. An object g has the attribute r_i iff $(g_1, \ldots, g_n)$ has attribute r for some objects $g_1, \ldots, g_n$ with $g = g_i$. We may call these attributes *roles*. The roles define the choices for action **R3**. The extended node contexts for nodes `#1` and `#3` in the example above are shown in Fig. 4. The possible choices for action **R5** are obtained from the result table.

2.3.2 Generalization Options

G1 *Remove attribute* Removes an attribute from an object or relation node. If this is applied to a relation node with a single attribute, the relation node is deleted. As a result, the number of connected components may increase. If a connected component contains only unmarked nodes, it is irrelevant to the output and the whole component is deleted.
G2 *Unmark node* Removes the column associated with a given object node from the result table.

2.4 Architecture

The program is implemented as a web application. The front end is written in Javascript (using jQuery for better DOM accessibility and browser independence). Intension graphs are represented by SVG graphs, which can be directly integrated into the webpage with the new HTML5 `<svg>` tag. The Python backend answers API calls `get_sorts` (which returns all objects sorts in the relational context family, needed for action **R1**) and `solve`, which takes an intension graph and returns the result table and refinement options. Program support for translating relational databases into relational context families would be an important feature but is not supported yet.

3 Related Work

Similar applications are described in [2], [1] and [4]. Relevant theoretical background is given in [5], but is not required to understand the navigation concept.

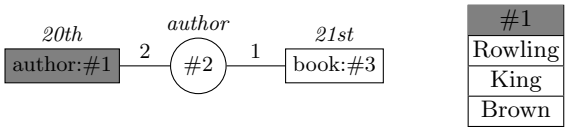


Fig. 3. Intension graph (left) and its result table (right)

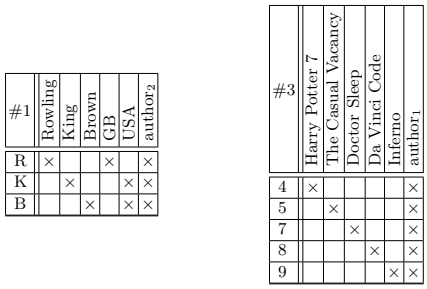


Fig. 4. Extended node contexts (used for query refinement)

References

1. Azmeh, Z., Huchard, M., Napoli, A., Hacene, M.R., Valtchev, P.: Querying relational concept lattices. In: Napoli, A., Vychodil, V. (eds.) Proc. of the 8th Intl. Conf. on Concept Lattices and their Applications (CLA'11). CEUR Workshop Proceedings, vol. 959, pp. 377–392. CEUR-WS.org (2011)
2. Ferré, S.: Conceptual navigation in RDF graphs with SPARQL-like queries. In: Kwuida, L., Sertkaya, B. (eds.) Proceedings of ICFCA 2010. LNCS, vol. 5986, pp. 193–208. Springer, Heidelberg (2010)
3. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Annals of Mathematics and Artificial Intelligence 49(1-4), 39–76 (2007)
4. Kötters, J.: Object configuration browsing in relational databases. In: Valtchev, P., Jäschke, R. (eds.) Proceedings of ICFCA 2011. LNCS, vol. 6628, pp. 151–166. Springer (2011)
5. Kötters, J.: Intension graphs as patterns over power context families. In: Huchard, M., Kuznetsov, S. (eds.) Proceedings of CLA 2016. CEUR Workshop Proceedings, vol. 1624. CEUR-WS.org (2016)
6. Wille, R.: Conceptual graphs and formal concept analysis. In: Lukose, D., Delugach, H.S., Keeler, M., Searle, L., Sowa, J.F. (eds.) Proceedings of ICCS 1997, 5th International Conference on Conceptual Structures. LNCS, vol. 1257, pp. 290–303. Springer, Heidelberg (1997)

Lattice Miner - A Formal Concept Analysis Tool

Rokia Missaoui and Kevin Emamirad

Université du Québec en Outaouais, Gatineau QC J8X 3X7, Canada
{rokia.missaoui, emak01}@uqo.ca

Abstract. Lattice Miner 2.0 is the latest release of a Formal Concept Analysis (FCA) software tool for the construction, visualization and manipulation of concept lattices. The newly added procedures include the computation of implications with negation as well as the production of triadic association rules, including implications.

1 Introduction

Lattice Miner is an FCA-based prototype developed in the LARIM research laboratory at *Université du Québec en Outaouais* under the supervision of the first author. It allows the construction of concept lattices and association rules (including implications) from a given formal context $\mathbb{K} := (G, M, I)$ that links the set G of objects to a set M of attributes through a binary relation I. The focus of Lattice Miner is on pattern (knowledge) visualization, exploration, querying and approximation through a lattice representation of either a flat or a nested structure. Lattice Miner is a public-domain Java tool whose main functions include all low-level operations and structures for the representation and manipulation of input data, lattices and association rules. The interface offers a context editor and concept lattice manipulator to assist the user in a set of interactive operations. The architecture of the tool is open and modular enough to allow the integration of new features and facilities. Lattice Miner 2.0 [4] adds new facilities to the previous release by integrating the computation of implications with negation as well as triadic association rules.

The rest of the paper is organized as follows. In Section 2 we recall triadic concept analysis and triadic association rules and provide screen shots of these rules. Section 3 briefly recalls the theorem behind computing implications with negation and offers an illustrative example together with a screen capture. Finally, Section 4 presents our future work.

2 Triadic Association Rules

Triadic concept analysis was originally introduced by Lehmann and Wille [3] as an extension to FCA, to analyze data described by three sets K_1 (objects), K_2 (attributes) and K_3 (conditions) together with a 3-ary relation $Y \subseteq K_1 \times K_2 \times K_3$. $\mathbb{K} := (K_1, K_2, K_3, Y)$ is called a *triadic context* (see Table 1). A triple (a_1, a_2, a_3) in Y means that object a_1 possesses attribute a_2 under condition a_3.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

$\mathbb{K}$	P	N	R	K	S
1	abd	abd	ac	ab	a
2	ad	bcd	abd	ad	d
3	abd	d	ab	ab	a
4	abd	bd	ab	ab	d
5	ad	ad	abd	abc	a

Table 1. A triadic context $\mathbb{K} := (K_1, K_2, K_3, Y)$

According to Biedermann [1], a *triadic implication* has the form $(A \to D)_C$ and holds if "*whenever A occurs under all conditions in C, then D also occurs under the same conditions*". Later on, Ganter and Obiedkov [2] extended Biedermann's definition by proposing three types of implications: *attribute*×*condition* implications (AxCIs), *conditional attribute* implications (CAIs), and *attributional condition* implications (ACIs).

An *attribute*×*condition implication* has the form $A \to D$, with A, D subsets of $K_2 \times K_3$. Such implications (rather dyadic) are extracted from the binary context $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$ where $(a_i, (a_j, a_k)) \in Y^{(1)} :\Leftrightarrow (a_i, a_j, a_k) \in Y$.

A *conditional attribute* implication takes the form: $A \xrightarrow{C} D$, where A and D are subsets of K_2, and C is a subset of K_3. It means that *A implies D under all conditions in C*. In particular, the implication holds for any subset in C. Such implication is then linked to Biedermann's definition of triadic implication as follows [2]: $A \xrightarrow{C} D \iff (A \to D)_{C_1}$ for all $C_1 \subseteq C$.

In a dual way, an *attributional condition* implication is an exact association rule of the form $A \xrightarrow{C} D$, where A and D are subsets of K_3, and $C \subseteq K_2$.

By extending the definition of *conditional attribute* implications to association rules, a *Biedermann conditional attribute* association rule BCAAR $(A \to D)_C$ $[s, c]$ holds, where A and D are subsets of K_2, and C is a subset of K_3, if D is true for C whenever A is true for C with a support s and a confidence c.

All these variants of triadic implications and association rules are produced by our tool using procedures in [5,7] as illustrated by Figure 1.

3 Implications with negation

To compute implications with negation from a given context $K = (G, M, I)$, the following theorem in [6] exploits the set $\Sigma_{\mathcal{K}}$ of key-based implications of the context $K|\tilde{K}$[1] to infer implications with a non-null support.

Theorem 1. *Let* $Ax \subseteq M\tilde{M}$. *Then,* $Ax \to M\tilde{M} \backslash Ax$ [0] $\Leftrightarrow A \to \tilde{x}$ [*sup*] *where* $sup = |A'|/|G|$ *and G is a set of objects.*

[1] The context $K|\tilde{K}$ is the apposition of K and its complementary $\tilde{K}$ whose attributes are in $\tilde{M}$ (negative attributes).

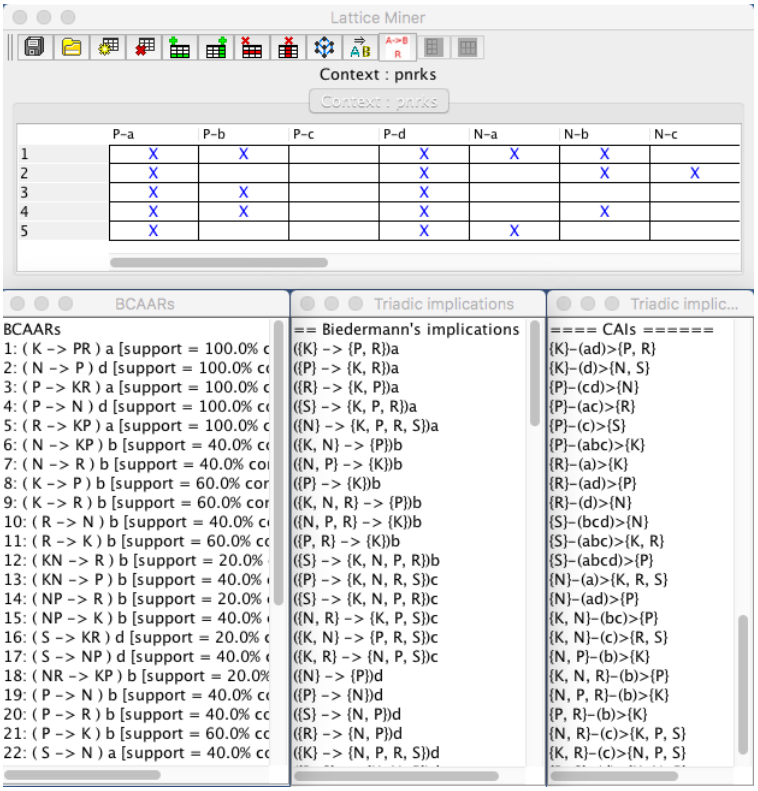


Fig. 1. Top: The dyadic context $\mathbb{K}^{(1)}$ extracted from the triadic context shown in Table 1. **Left**: Biedermann triadic association rules. **Middle**: Biedermann implications. **Right**: Conditional attribute implications.

This is done in Lattice Miner by (i) first computing the context $K|\tilde{K}$, (ii) identifying the left-hand side of key-based implications, which are the minimal generators of the infimum, and (iii) finally generating the whole set of implications (with or without negation) by creating for each key Ax, $p = |Ax|$ implications where each element in Ax is negated and shifted from the left to the right of the generated implication.

For example, $\tilde{d}ef \rightarrow M\tilde{M}$ [0] leads to three implications, namely $ef \rightarrow d$ [0.16], $\tilde{d}e \rightarrow \tilde{f}$ [0.16] and $\tilde{d}f \rightarrow \tilde{e}$ [0.16]. Such implications appear in Figure 2.

4 Conclusion and Further Development

We have presented two main additions to our prototype, namely the computation of implications with negation and the identification of triadic association rules. We plan to have a Web-based release of Lattice Miner 2.0. This will bring the following advantages: (i) the application can be used from a laptop or a tablet

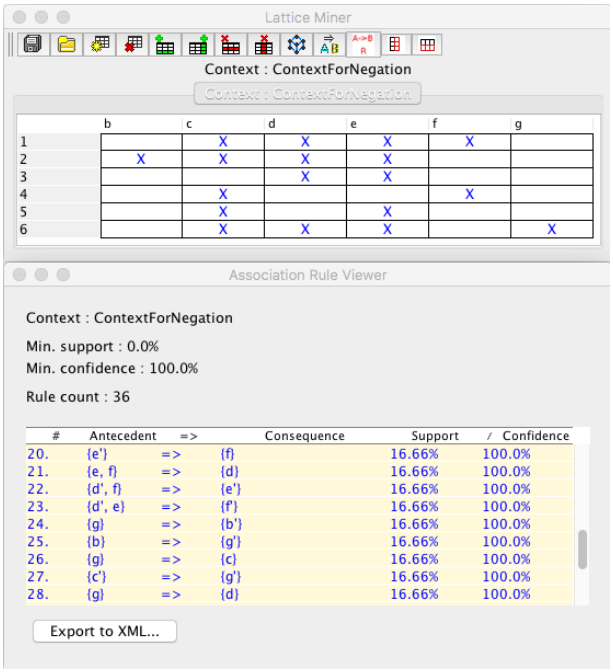


Fig. 2. Implications with negation

without installing any further software, (ii) it can be deployed to a private or public cloud to benefit from powerful software and hardware resources, and (iii) the opportunity to speed up the development of this project by using modern tools and technologies.

References

1. Biedermann, K.: How triadic diagrams represent conceptual structures. In: ICCS. pp. 304–317 (1997)
2. Ganter, B., Obiedkov, S.A.: Implications in triadic formal contexts. In: ICCS. pp. 186–195 (2004)
3. Lehmann, F., Wille, R.: A triadic approach to formal concept analysis. In: ICCS. pp. 32–43 (1995)
4. Missaoui, R., Emamirad, K.: Lattice Miner 2.0. `https://github.com/LarimUQO/lattice-miner/` (2017), [Online; accessed April 12, 2017]
5. Missaoui, R., Kwuida, L.: Mining triadic association rules from ternary relations. In: Formal Concept Analysis - 9th International Conference, ICFCA 2011, Nicosia, Cyprus, May 2-6, 2011. Proceedings. pp. 204–218 (2011)
6. Missaoui, R., Nourine, L., Renaud, Y.: Computing implications with negation from a formal context. Fundam. Inform. 115(4), 357–375 (2012)
7. Rodriguez-Lorenzo, E., Cordero, P., Enciso, M., Missaoui, R., Mora, A.: An axiomatic system for conditional attribute implications in triadic concept analysis. International Journal of Intelligent Systems DOI:10.1002/int.21888 (February 2017)

FACT - A Tool for Temporal Formal Concept Analysis

Beniamin Movileanu, Christian Săcărea, Diana-Florina Şotropa*

Babeş-Bolyai University
movileanubeniamin@gmail.com,{csacarea@math.,diana.halita@}ubbcluj.ro

Abstract. Formal Concept Analysis offers an elegant, intuitive and powerful graphical representation of landscapes of knowledge as concept lattices. In this paper, we report about the current state of `FACT`, a tool for temporal data analysis and knowledge discovery. `FACT` is a web-based application which allows an online interaction with larger data sets in order to explore and analyze data conceptually. It uses concept lattices in order to extract knowledge from the data set. After presenting the tool itself we shortly describe an example and present the planning for further development.

Keywords: Life tracks, Temporal Concept Analysis, Web logs analysis, Conceptual structures, User behavior, Attractors

1 Introduction

During the last decades, many theories have been developed, aiming at the investigation of temporal phenomena. One of them is the particular approach to Temporal Concept Analysis (TCA) introduced by Karl Erich Wolff in [3]. He uses FCA as a frame for describing concept analysis of data with a temporal layer.

The FCA researchers community agrees upon the necessity of having powerful software tools for formal context handling and lattice representations. There is a long list of FCA related software tools[1] but a freely available tool for TCA, which is handy and easy to use is missing. In this paper we aim to fill this gap and present the current state of `FACT` which is a tool for Temporal Concept Analysis. We motivate the implementation of the tool suite and we describe the implemented features in Section 2. Here, we also describe the architecture of the tool and the technologies as well as the external tools used in implementation. Finally, Section 3 describe some of the future directions for the development of the tool.

Conceptual time systems have been introduced by K. E. Wolff in [3] in order to investigate conceptual structures of data enhanced with a time layer. Basically, conceptual time systems are many-valued contexts, comprising a time part and

* Diana Şotropa was supported by Tora Trading Services private scholarship.
[1] http://www.upriss.org.uk/fca/fcasoftware.html

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

an event part, which are subject to conceptual scaling, unveiling the temporal development of the analyzed data, object trajectories and life tracks. K. E. Wolff defined states as objects concepts of time points and transitions in order to introduce the definition of the life track of a conceptual time system. K. E. Wolff states in [3] that programs which represent complex temporal systems have to be developed in order to easily understand and visualize data and connections between data.

2 `FACT` - Motivation, Description and Features

The main motivation of developing this tool comes from practical applications of FCA. Some data have a natural temporal structure and can usually be interpreted as many-valued contexts which are scaled using the ToscanaJ Suite [1].

In some previous research, we have investigated different types of user behavior in educational platforms. Paper [2] emphasize how TCA methods might help in discovering users behavior by considering different points in time. Besides data from educational platforms, many other data sets have an inherently temporal layer and the lack of easy to use software solutions for analyzing temporal data contributes to a relatively small number of reported applications of TCA.

The main goal of conceptual knowledge discovery is to support a human-centered process of knowledge discovery by providing a visualization of data based on the visualization of underlying conceptual structures. Therefore, `FACT` was designed in order to support visualization of temporal behavior of data. Hence, it currently implements current features for Temporal Concept Analysis. The main purpose of the tool is to enable the user to visualize the temporal perspective of his data, projected on a conceptual landscape (such as concept lattice, or user defined browsing scenarios). Every user identifies himself by a login system and has access to his own data, as well as to a small number of public contexts which can be used as toy examples.

There are multiple technologies used for the implementation. For the persistence is used `postgresql` - to store the private conceptual data for further analysis, as well as the authentication data. The `Cytoscape` library is used to display the concept lattice on the web page and to display the temporal development of data on it.

The tool offers the following features:

- load CSX and SQL files;
- visualize and navigate through the contexts;
- store contexts;
- user can dynamically filter data after some SQL fields;
- indicate the nodes and edges based on the temporal data.
- diagrams can be exported in the Portable Networks Graphic (png);
- after computing the concept lattice, the y coordinate of the nodes is locked; the nodes are only allowed to move freely in their corresponding layer.

We have chosen Java as a platform for the new development of this software tool. Java offers virtual machines for most common operating systems and since it is very common as programming language in universities, it makes the project more accessible for many students. Java is used to process the data and send the result via JSON to Cytoscape. The `FACT` tool requires the previous use of the ToscanaJ suite to build a conceptual schema file. Other than Siena, which requires preprocessing by a quite rare tool, Cernato, `FACT` is self-contained. Once a conceptual schema has been imported and some scales have been chosen for visualization, `FACT` draws the corresponding order diagrams indicating all objects stored in the database in their relationships to the attributes of the scale. The novelty comes also from the front-end for temporal conceptual systems. It allows the user to navigate through the data and to analyze specific sets of objects by activating scales that interpret relevant aspects of the given data from a temporal perspective using a simple graphical interface. The screenshot of `FACT` in Figure 1 exemplifies this interface.

Usability and accessibility is another goal of this tool development. Hence it has been developed as an open source project. Furthermore, it is made public as a web site which requires registration, but no other installation process which makes it easy to use.

Last but not least, revival of software development for FCA is in our opinion, a major objective for our community, to which we strongly adhere.

In order to exemplify the use of FACT tool we will consider our previous research on the use of an e-learning platform. Different users patterns were identified and formalized in [2]. They are called attractors and are defined as sets of scales in conceptual time systems. An attractor is a conceptual structure having a clear meaning and which is considerably influencing the users behavior in the educational platform. Some attractors are intended, i.e., encoded in the conceptual structure by the educator itself, while some others are not intended and are occurring at some particular points in time. Students, while browsing the e-learning platform, adhere to some attractors or not, showing thus particular browsing habits.

A behavioral attractor is a conceptual scale which reflects the habits of a user while visiting an educational platform at a specific point in time. The behavioral pattern represents the event part of the conceptual time system at the specific time granule. In order to build users' life tracks we set the time granularity to week level and mark the temporal trajectory of the student through different behaviors (eg. `Ls-LP-LT-LA`). Then, we superimpose them on the concept lattice of the corresponding behavioral attractor. Behavioral attractors are unintended attractors, which are crystallizing behavioral patterns showing how users are using the resources, independently to the intention of the educator. Figure 1 presents how user F is using the educational content considered for this particular behavior (`Ls-LP-LT-LA`) on the entire semester. This behavior is a navigational behavior and emphasize how students are using the educational content presented on the platform. In this specific context we may observe how and when user F is visiting lectures page, different information about lecture

papers given during lectures or laboratory pages which presents the laboratory theory or assignments.

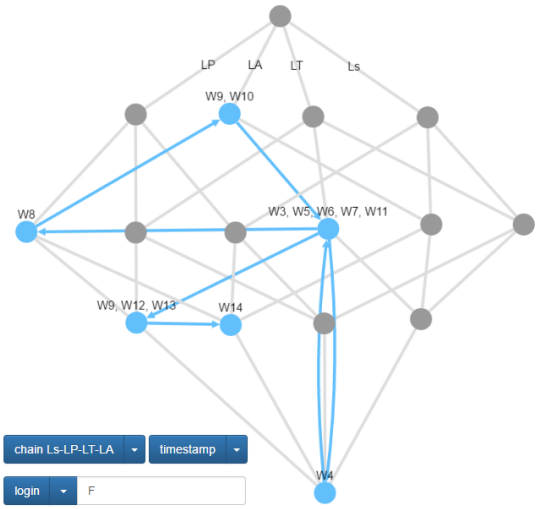


Fig. 1. Life track of user F while navigating the platform wrt. `Ls-LP-LT-LA` behavior

3 Conclusions and Further Research

Temporal Concept Analysis is the theory of temporal phenomena described with tools of Formal Concept Analysis. The development of the tool is ongoing and there are multiple functionalities that we plan to implement in the future. The tool will be developed according to users needs and feedback. In conclusion, we believe that the presented version of `FACT` brings an important contribution to the collection of FCA tools, by implementing functionalities of visualization and navigation in temporal conceptual systems.

References

1. Peter Becker and Joachim Hereth Correia. The toscanaj suite for implementing conceptual information systems. In *Formal Concept Analysis, Foundations and Applications*, pages 324–348, 2005.
2. Diana-Florina Şotropa Beniamin Movileanu, Christian Săcărea. An investigation of user behavior in educational platforms using temporal concept analysis. In *Proceedings of the 14th International Conference on Formal Concept Analysis*, 2017.
3. Karl Erich Wolff. Temporal concept analysis. In *ICCS-2001 International Workshop on Concept Lattices-Based Theory, Methods and Tools for Knowledge Discovery in Databases, Stanford University, Palo Alto (CA)*, pages 91–107, 2001.

Exploration of Textual Sequential Patterns

Hedi-Théo Sahraoui[1], Pierre Holat[2],
Peggy Cellier[3], Thierry Charnois[2], and Sébastien Ferré[1]

[1] Université de Rennes 1, IRISA, firstname.lastname@univ-rennes1.fr
[2] Université de Paris Nord, LIPN, firstname.lastname@lipn.univ-paris13.fr
[3] INSA Rennes, IRISA, Peggy.Cellier@irisa.fr
http://tal.lipn.univ-paris13.fr/sdmc/

1 Introduction

The extraction of regularities in texts is important for several natural language processing tasks. For instance, in information extraction, the regularities can allow to discover linguistic patterns [4] or to study the stylistics of authors [9]. When looking for those regularities, some specificities of textual data have to be taken into account: the sequentiality of the data (i.e., the order between words), the different levels of abstractions (i.e., words, lemma, Part-Of-Speech (POS) tags) and specific constraints (e.g., "the regularities have to contain a verb"). SDMC (Sequential Data Mining under Constraints) [3, 2][4] is a sequential pattern mining tool that deals with all those requirements. From a text, the tool extracts regularities called *sequential patterns*, i.e sequences of words, lemmas, and POS tags that frequently appear together in the text. In order to extract such patterns mixing different levels of abstraction, each word in the text is represented by itself but also by its lemma and its POS tags. In addition, SDMC allows to apply constraints to filter the extracted patterns: widespread constraints in data mining like minimum frequency (support) but also text-specific constraints like *"contains a verb"*.

A well-known drawback of pattern mining is the huge number of patterns that can be extracted. Even if SDMC manages the computation issue through constraints, the set of extracted patterns can be very large and hard to assess for users. In a previous work [5], the authors have used the Logical Information Systems (LIS) [6] paradigm to explore a set of patterns. The main advantage of this approach is that users can benefit from their background knowledge to navigate through the patterns.

In this paper we show how we have instantiated the LIS paradigm into SDMC to help users deal with their patterns and their texts. Indeed, we propose to explore the sequential patterns that appear in a text with a visualization of the sentences where those patterns occur. That exploration functionality is available online in the "Concordancier" menu as "Navigation dans les motifs". To illustrate the exploration, we use the French book "Le Petit Prince"[5].

[4] http://tal.lipn.univ-paris13.fr/sdmc/
[5] "Le Petit Prince", Antoine de Saint-Exupéry, 1943.

In: K. Bertet, D. Borchmann, P. Cellier, S. Ferré (Eds.): Supplementary Proceedings of ICFCA, Rennes, France, 2017.

Fig. 1. The textual interface of SDMC

2 Sequential Pattern Mining and SDMC

Sequential pattern mining [1] is a data mining technique that aims at discovering correlations between events through their order of appearance. It is an important field of data mining with broad applications (e.g. biology, marketing), and many algorithms to extract frequent sequential patterns [10, 8, 11]. In the context of the extraction of sequential patterns from texts, a *sequence* is an ordered list of distinct words also called *items*. Note that when considering different levels of abstraction for a word, a sequence is an ordered list of itemsets where each item represents some information about the word (e.g., the word itself, its lemma or a POS tag). The *support* of a sequence S in a text is the number of sentences in the text containing S. Given a mininimum support threshold *minsup*, the problem of frequent sequential pattern mining is to find the complete set of sequences whose support is greater or equal to *minsup*.

SDMC (Sequential Data Mining under Constraints) [3, 2] is an online sequential pattern mining tool with two user interfaces: one for mining textual data, and another for mining any kind of dataset. Figure 1 shows the first interface. SDMC handles various types of constraints which are not only numerical (e.g., the support constraint) but also symbolic and syntactic (e.g., "the pattern has to contain a verb"). These multiple constraints enable the user to express a large scope of knowledge to focus on interesting textual sequential patterns.

3 Logical Information Systems (LIS)

Logical Information Systems (LIS) [6] are a paradigm of information retrieval that combines querying and navigation. LIS are formally based on Logical Con-

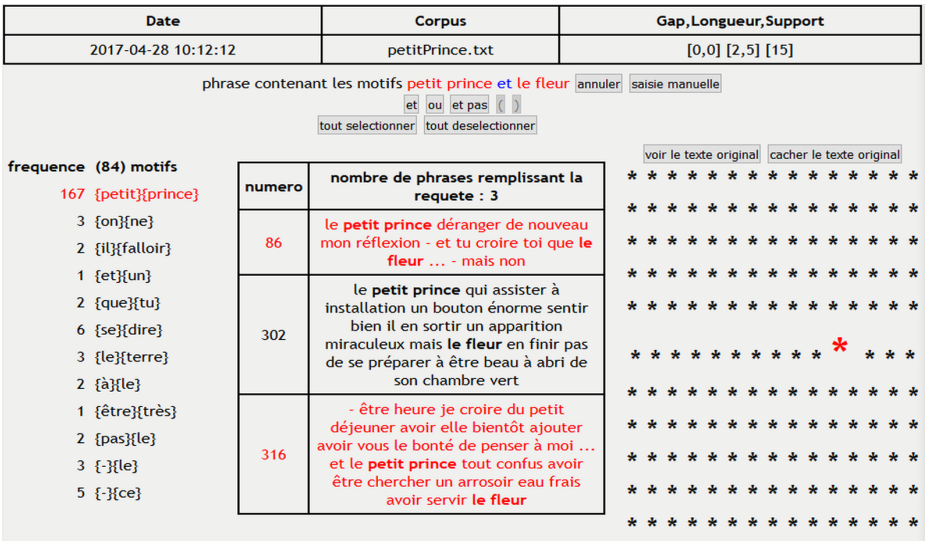


Fig. 2. The LIS interface of SDMC to explore a text and its patterns

cept Analysis (LCA), a logical generalization of Formal Concept Analysis [7]. In LCA, objects are described by logical formulas rather than sets of attributes. The concept lattice serves as a navigation structure where each concept is a navigation place. Because of its huge size, LIS only show a local view of the concept lattice, centered on each navigation place. A local view has three components: (a) the query that is a Boolean combination of descriptors, (b) the extent that is the set of objects matching the query, and (c) the index that is the set of descriptors that occur in the extent, along with their relative frequency. The index descriptors can be used as navigation links to modify the query, and hence reach related concepts: e.g., adding a descriptor to reach a more specific concept.

4 LIS Exploration Interface in SDMC

We have instantiated the LIS paradigm to textual sequential patterns by considering sentences as objects, and sequential patterns as descriptors. This has been implemented and integrated into SDMC as a new user interface. Figure 2 shows that LIS interface in SDMC.

On the top of the screen, information about the extraction are given: the date of the extraction, the corpus (text), and the values of numerical constraints used for pattern extraction: [min,max] gap size between words in patterns, [min,max] length of the extracted patterns, and minimum support. The query appears just below those information. It is a Boolean combination of patterns. Here, it is a conjunction of two patterns: "phrases contenant les motifs **petit prince** et **le fleur**" (in English "sentences that contain patterns **little prince** and **the**

flower") which means that the user has selected pattern "**petit** followed by **prince**" and pattern "**le** followed by **fleur**".

On the main part of the screen, there are 3 parts. On the left part, there is the index, i.e. textual patterns that can be selected or added to the query. On the middle part, there is the extent, i.e. the sentences that match the query. In the example, three sentences (82, 302, and 316) contain both patterns **petit prince** and **le fleur**. On the right part, a text view is displayed where the selected sentences appear in bold and red. The tool proposes three kinds of text views: the text itself, a compact version where the sentences are replaced by stars (as shown on the figure), and a void view (useful for very long texts).

5 Conclusion

In this paper we have presented an interface to explore regularities extracted from text, called *textual sequential patterns*. The exploration is based on a conceptual navigation over the set of all patterns in the logical information systems framework, a logical version of formal concept analysis. The exploration interface is available through the online tool SDMC.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE. pp. 3–14. IEEE Computer Society (1995)
2. Béchet, N., Cellier, P., Charnois, T., Crémilleux, B.: SDMC : un outil en ligne d'extraction de motifs séquentiels pour la fouille de textes (2013)
3. Béchet, N., Cellier, P., Charnois, T., Crémilleux, B.: Sequence mining under multiple constraints. In: Wainwright, R.L., Corchado, J.M., Bechini, A., Hong, J. (eds.) ACM Symposium on Applied Computing. pp. 908–914. ACM (2015)
4. Cellier, P., Charnois, T., Plantevit, M., Rigotti, C., Crémilleux, B., Gandrillon, O., Kléma, J., Manguin, J.: Sequential pattern mining for discovering gene interactions and their contextual information from biomedical texts. J. Biomedical Semantics 6, 27 (2015)
5. Cellier, P., Ferré, S., Ducassé, M., Charnois, T.: Partial Orders and Logical Concept Analysis to Explore Patterns Extracted by Data Mining. In: Conceptual Structures for Discovering Knowledge. pp. 77–90. Springer, Berlin, Heidelberg (Jul 2011)
6. Ferré, S., Ridoux, O.: Introduction to logical information systems. Inf. Process. Manage. 40(3), 383–419 (Jan 2004)
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer-Verlag New York, Inc. (1997)
8. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: ICDE (2001)
9. Quiniou, S., Cellier, P., Charnois, T., Legallois, D.: What about Sequential Data Mining Techniques to Identify Linguistic Patterns for Stylistics? In: Int. Conf. on Computational Linguistics and Intelligent Text Processing. LNCS, Springer (2012)
10. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: EDBT (1996)
11. Zaki, M.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)

Author Index

Adaricheva, Kira 67

Bazin, Alexandre 71

Carbonnel, Jessie 71
Cellier, Peggy 99
Charnois, Thierry 99
Chau, Raymee 17

Dias, Sérgio 1

Egurnov, Dmitry 31
Emanirad, Kevin 91

Ferré, Sébastien 99
Foret, Annie 75

Gizdatullin, Danil 49

Guyet, Thomas 79

Holat, Pierre 99

Ignatov, Dmitry 31, 49

Kahn, Giacomo 71
Kis, Levente Lorand 83
Kötters, Jens 87

Masson, Véronique 79
Mephu Nguifo, Engelbert 31
Missaoui, Rokia 91
Mitrofanova, Ekaterina 49
Movileanu, Beniamin 95
Muratova, Anna 49

Ninesling, Taylor 67
Novak, Krzysztof 17

Quiniou, René 79

Săcărea, Christian 83, 95
Sahraoui, Hedi-Théo 99
Salmond, Daniel 17
Song, Mark 1
Şotropa, Diana-Florina 95

Troancă, Diana 83

Vieira, Newton 1

Zárate, Luis 1