

Conception et implémentation de nouvelles fonctionnalités dans un prototype de fouille de données

Kévin Emamirad

Université du Québec en Outaouais
Département d'informatique et d'ingénierie

emak01@uqo.ca

Sous la supervision de Prof. Rokia Missaoui

mercredi 31 mai 2017

Plan de la présentation

- 1 Introduction
- 2 Rappels
- 3 Les outils de l'analyse formelle de concepts
- 4 Production d'implications avec négation
- 5 Enrichissement du module de génération des règles triadiques
- 6 Conclusion

Introduction

- L'analyse formelle de concepts (AFC) (Ganter et Wille 1999) est un formalisme de représentation des connaissances et de fouille de données qui est
 - ▶ utilisé dans divers domaines (informatique, linguistique, sociologie, biologie, etc.), et
 - ▶ produit des visualisations graphiques des structures inhérentes aux données sous forme de treillis de concepts (Galois).
- Au cours des deux dernières décennies, on a vu apparaître plusieurs d'outils d'analyse formelle de concepts tels *ToscanaJ*, *ConExp*, *Coron*, *Java Lattices*, et *Lattice Miner*.

Introduction

Objectifs

- But : Enrichir *Lattice Miner* (Roberge 2007), un prototype de fouille de données qui exploite l'AFC avec les objectifs suivants :
 - ① production d'implications avec négation (Missaoui, Nourine et Renaud 2012),
 - ② enrichissement du module de génération des règles triadiques pour obtenir exhaustivement et précisément les trois formes d'implications triadiques définies par (Ganter et Obiedkov 2004),
 - ③ validation intensive de la procédure de production de la base d'implications de (Guigues et Duquenne 1986) et de la procédure de calcul des relations de flèches. Cette dernière est utile dans le processus de décomposition de contextes formels (Viaud et al. 2015), et
 - ④ amélioration de la convivialité de l'interface usager.

Rappels

Rappels I

Analyse formelle de concepts

Définition 1 : Contexte formel

Soit $\mathbb{K} = (G, M, I)$ un contexte formel où G , M et I sont respectivement un ensemble d'objets, une collection d'attributs et une relation binaire entre G et M . L'expression $(g, m) \in I$ ou encore glm signifie que l'objet g possède l'attribut m .

	a	b	c	d	e	f
1	×		×		×	
2		×		×		×
3			×	×		
4		×	×			×
5	×	×				×
6	×	×		×	×	

Tableau 1: Exemple de contexte \mathbb{K}

Rappels II

Analyse formelle de concepts

Définition 2 : Concept formel

Un *concept formel* c est une paire d'ensembles $c := (A, B)$ avec $A \subseteq G$, $B \subseteq M$, $A = B'$ et $B = A'$, où A' est l'ensemble des attributs partagés par les objets dans A et B' est l'ensemble des objets ayant tous leurs attributs dans B . Les sous-ensembles A et B sont appelés respectivement l'extension et l'intention du concept c . Les valeurs de A' et B' sont obtenues comme suit : $A' := \{m \in M \mid g \text{ l m } \forall g \in A\}$ et $B' := \{g \in G \mid g \text{ l m } \forall m \in B\}$.

Rappels

Analyse formelle de concepts

Définition 3 : Sous-ensemble fermé

Un sous-ensemble X est fermé si $X'' = X$. Un concept objet pour l'objet g est une paire de la forme $\gamma(g) = (g'', g')$ alors que le concept attribut pour l'attribut m est $\mu(m) = (m', m'')$.

- La paire $(\{3\}, \{c, d\})$ forme un concept objet pour 3 car $\gamma(3) = (3'', 3') = (\{3\}, \{c, d\})$.
- La paire $\mu(e) = (e', e'') = (\{1, 6\}, \{a, e\})$ forme un concept attribut pour e .

Rappels III

Analyse formelle de concepts

Définition 4 : Treillis de concepts

Un *treillis de concepts* $\underline{\mathfrak{B}}(G, M, I)$ est un treillis résultant de l'ordre partiel existant entre les concepts du contexte $\mathbb{K} = (G, M, I)$.

$$(A, B) \leq (C, D) \Leftrightarrow A \subseteq C \text{ et } D \subseteq B$$

(A, B) est alors un sous-concept ou prédécesseur immédiat de (C, D) alors que ce dernier est un successeur de (A, B) .

Rappels IV

Analyse formelle de concepts

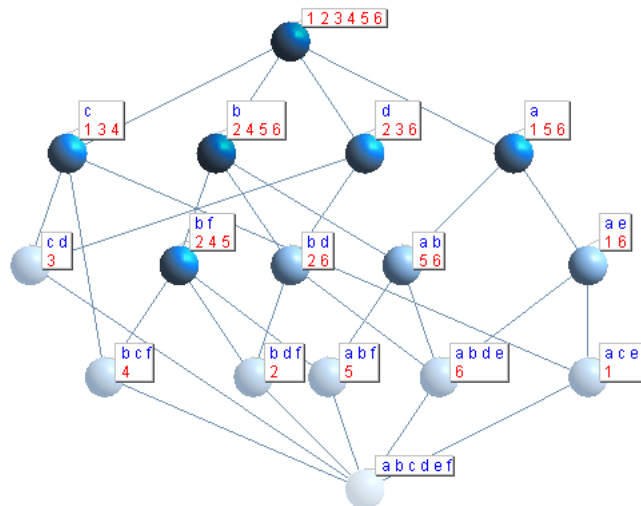


Figure 1: Treillis avec étiquetage complet pour le contexte du tableau 1 10 / 60

Rappels V

Analyse formelle de concepts

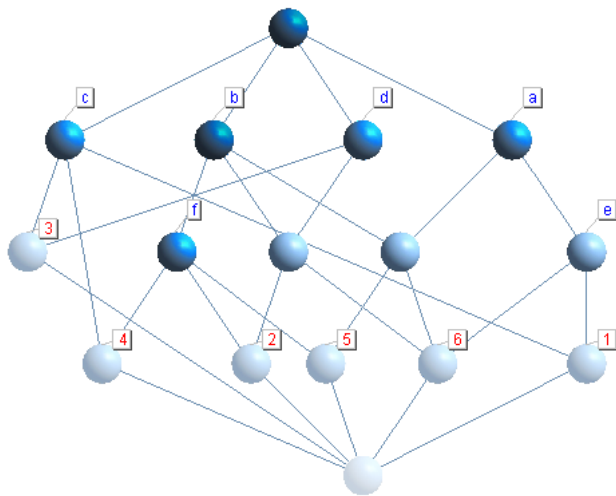


Figure 2: Treillis avec étiquetage réduit pour le contexte du tableau 1

Rappels VI

Analyse formelle de concepts

Définition 5 : sup-irréductible et inf-irréductible

On rappelle qu'un concept est dit *sup-irréductible* (ou *join-irréductible*) si et seulement si il possède un unique prédécesseur et il est dit *inf-irréductible* (ou *meet-irréductible*) s'il admet un unique successeur.

- Par exemple, le concept $(245, bf)$ est *inf-irréductible* c'est un concept attribut.

Rappels VII

Règles d'association et implications

Définition 6 : règle d'association

$$r : Y \rightarrow Z \text{ [sup, conf]}$$

r est une règle d'association avec

- Y et Z sont des sous-ensembles d'attributs appelés *itemsets*,
 - $Y \cap Z = \emptyset$,
 - *sup*, le support est $Prob(Y \cup Z)$ (proportion des objets ayant simultanément les attributs Y et Z), et
 - *conf*, la confiance est $Prob(Z/Y)$ (probabilité d'avoir Z lorsque Y est présent dans le contexte \mathbb{K}).
-
- Par exemple, $\{e\} \rightarrow \{a\}$ [33%, 100%]

Rappels VIII

Règles d'association et implications

Définitions 7 : itemsets et générateurs

- Un *itemset* est dit *fréquent* si la proportion d'objets le possédant est au moins égale au support minimum défini par l'utilisateur.
- Un *itemset* Y est dit *fermé* si $Y = Y''$. Cela signifie que Y est une intention d'un concept formel.
- Un *générateur* Y (Pfaltz et Taylor 2002 ; Pasquier et al. 1999) d'un itemset fermé (intention) Z est un ensemble minimal de Z tel que $Y'' = Z$.
- Par exemple, $\{e\}$ est un générateur de $\{a, e\}$, on a donc $\{e\} \rightarrow \{a\}$ [33%, 100%].

Rappels XII

Règles d'association et implications

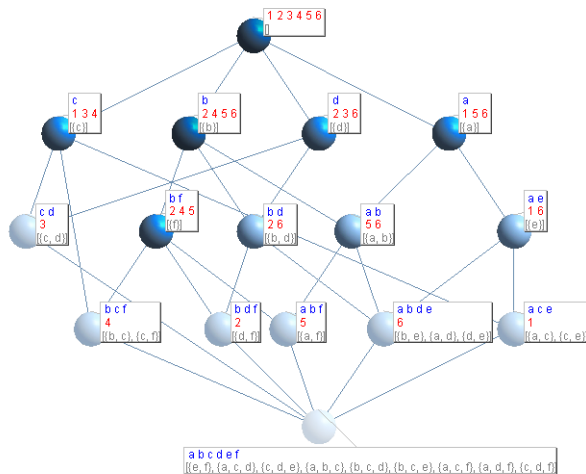


Figure 3: Treillis de concepts avec indication des générateurs

Rappels IX

Règles d'association et implications

Définition 8 : implication et base générique

$$r : g_i \rightarrow \text{Int}(c) \setminus g_i$$

r est une implication ou règle exacte avec

- c un concept formel,
- $G = \{g_1, \dots, g_i, \dots, g_n\}$ l'ensemble de ses générateurs,
- $\text{Int}(c)$ représente l'intention du concept c ,
- $\text{support}(c)$ est la proportion d'objets contenus dans l'extension de c ,
- $\text{support}(r) = \text{support}(c)$ et
- $\text{confiance}(r) = 100 \%$

Une base générique (Pasquier et al. 1999) d'implications est une représentation relativement concise d'implications de la forme précédente.

Rappels X

Règles d'association et implications

Définition 9 : règle approximative

$$r : g_i \rightarrow \text{Int}(c_j) \setminus \text{Int}(c)$$

r est une règle approximative avec

- c un concept formel,
 - $P = \{c_1, \dots, c_j, \dots\}$ l'ensemble des prédécesseurs de c ,
 - $\text{Int}(c)$ représente l'intention du concept c ,
 - g_i un générateur de $\text{Int}(c)$,
 - $\text{support}(r) = \text{support}(c_j)$, et
 - $\text{confiance}(r) = \text{support}(c_j) / \text{support}(c)$
-
- Par exemple, $\{b\} \rightarrow \{f\}$ [50%, 75%]

Rappels XI

Règles d'association et implications

Définitions 10 : pseudo-intent et base de Guigues-Duquenne

Un ensemble $P \subseteq M$ est un *pseudo-intent* de (G, M, I) si et seulement si $P \neq P''$ et si $Q'' \subseteq P$ est vraie pour tout pseudo-intent $Q \subseteq P$, $Q \neq P$.
L'ensemble des implications de la forme :

$$\mathcal{L} := \{ \mathcal{P} \rightarrow \mathcal{P}'' \setminus \mathcal{P} \mid \mathcal{P} \text{ pseudo-intent} \}$$

est appelé base de Guigues-Duquenne (*stem base*) et reconnue comme étant minimale.

À titre d'exemple, $\{a, c\}$ est un pseudo-intent dont le fermé est $\{a, c, e\}$, d'où l'implication $\{a, c\} \rightarrow \{e\}$

Rappels XIV

Règles d'association et implications

	a	b	c	d	e	f	\tilde{a}	\tilde{b}	\tilde{c}	\tilde{d}	\tilde{e}	\tilde{f}
1	×		×		×			×		×		×
2		×		×		×	×		×		×	
3			×	×			×	×			×	×
4		×	×			×	×			×	×	
5	×	×				×			×	×	×	
6	×	×		×	×				×			×

Figure 4: $\mathbb{K}|\tilde{\mathbb{K}}$: apposition du contexte \mathbb{K} avec son complémentaire $\tilde{\mathbb{K}}$

- Le contexte complémentaire de \mathbb{K} est $\tilde{\mathbb{K}} = (G, \tilde{M}, G \times M \setminus I)$ avec \tilde{M} l'ensemble des attributs négatifs.

Rappels XV

Relations flèches

Définition 11 : Relations flèches

La relation entre l'objet g et l'attribut m dans un contexte formel se présente sous l'une des quatre formes suivantes :

- $g \updownarrow m$ si $\gamma(g) \not\leq \mu(m)$, $\gamma(g) \leq m^+$, et $g^- \leq \mu(m)$
- $g \uparrow m$ si $\gamma(g) \not\leq \mu(m)$, $\gamma(g) \leq m^+$, et $g^- \not\leq \mu(m)$
- $g \downarrow m$ si $\gamma(g) \not\leq \mu(m)$, $\gamma(g) \not\leq m^+$, et $g^- \leq \mu(m)$
- $g \circ m$ si $\gamma(g) \not\leq \mu(m)$, $\gamma(g) \not\leq m^+$, et $g^- \not\leq \mu(m)$

où g^- représente le prédécesseur immédiat du concept objet (sup-irréductible) $\gamma(g)$ et m^+ représente le successeur immédiat du concept attribut (inf-irréductible) $\mu(m)$.

Rappels XVI

Relations flèches

	a	b	c	d	e	f
1	×	↖↗	×	↖↗	×	↙↘
2	↖↗	×	↖↗	×	↙↘	×
3	↖↗	↖↗	×	×	↙↘	↙↘
4	↖↗	×	×	↖↗	↙↘	×
5	×	×	↖↗	↖↗	↖↗	×
6	×	×	↖↗	×	×	↖↗

Figure 5: Relation flèches pour le contexte \mathbb{K}

Par exemple, $3 \downarrow e$ car

- $\gamma(3) \not\leq \mu(e)$ avec $\gamma(3) = (3, cd)$ et $\mu(e) = (16, ae)$
- $\gamma(3) \not\leq e^+$ avec $e^+ = (156, a)$
- $3^- \leq \mu(e)$ avec $3^- = (\emptyset, abcdef)$

Les outils de l'analyse formelle de concepts

Les outils de l'analyse formelle de concepts

Galicia

- Galicia¹ a été développé en Java par Pekto Valchev et ses collaborateurs (Université de Montréal).

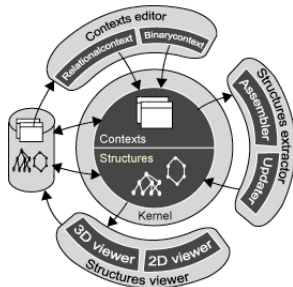


Figure 6: Cycle de vie du treillis Galicia

Quelques fonctionnalités de Galicia :

- Affichage du treillis en trois dimensions
- Manipulation du contexte et du treillis
- Plusieurs procédures de construction du treillis : Bordat, Nourine, Godin et *Next-Closure*
- Connexion à des bases de données SQL pour sauvegarder les contextes

1. <http://www.iro.umontreal.ca/~galicia/>

Les outils de l'analyse formelle de concepts

Le système Coron

Le système Coron² (Szathmary 2006) est une boîte à outils qui a été développée au laboratoire LORIA³

- Développé avec Java et Perl
- En ligne de commandes
- Version actuelle de ce système est 0.8 et date de janvier 2010
- Compatible avec les systèmes d'exploitation Unix, macOS et Windows
- Implémente plusieurs algorithmes de fouille de données tels qu'Apriori, Close, Aclose, Carpathia, Charm, Next-Closure ou NFI-Simple.
- Simple d'utilisation, modulaire et documentation riche
- Code source non disponible

2. <http://coron.loria.fr/site/index.php>.

3. Laboratoire lorrain de recherche en informatique et ses applications, Nancy, France

Les outils de l'analyse formelle de concepts

Librairie Java Lattices

La librairie *Java Lattices*⁴ (Bertet et al. 2014) a été développée par le laboratoire L3i⁵.

- Permet de générer des treillis de Galois et des règles d'association
- Code source libre et disponible sur la plate-forme GitHub
- Entièrement écrit avec Java
- La librairie est compatible avec Maven (outil Java pour gérer les dépendances)
- Client en ligne de commandes disponible
- Algorithme *Limited Object Access (LOA)* (Demko et Bertet 2011)

4. <https://thegalactic.github.io/java-lattices>

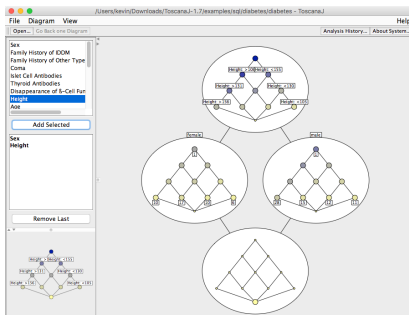
5. Laboratoire Informatique, Image et Interaction, université de La Rochelle, France

Les outils de l'analyse formelle de concepts

ToscanaJ

ToscanaJ (Becker, Hereth et Stumme 2002) est une réimplémentation avec le langage de programmation Java de l'outil d'AFC connu sous le nom de « *Toscana* ». Cet outil a la particularité d'être le premier outil à offrir l'affichage des diagrammes imbriqués (*nested line diagrams*) comme on peut le voir avec la figure 7.

Figure 7: ToscanaJ : Diagrammes imbriqués



Les outils de l'analyse formelle de concepts

Concept Explorer

*Concept Explorer*⁶ (*ConExp*) est un outil souvent utilisé par les chercheurs en AFC.

- Interface graphique complète en Java Swing
- Édition et transformation de contextes
- Construction de treillis de concepts
- Code distribué librement
- Base Guigues-Duquenne
- Génération des règles d'association et l'exploration des attributs
- Génération des relations flèches
- Réduction du contexte

6. <http://conexp.sourceforge.net/>

Les outils de l'analyse formelle de concepts

Concept Explorer

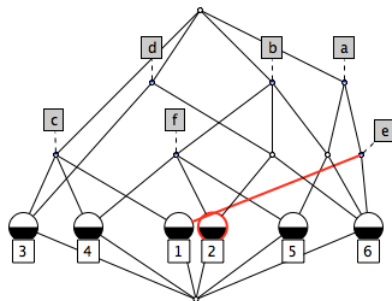


Figure 8: *ConExp* : Affichage du treillis

- L'affichage du treillis se fait selon un étiquetage réduit seulement.
- *ConExp* utilise un algorithme de construction du treillis avec le minimum d'intersections et les collisions sont affichées en rouge.

Les outils de l'analyse formelle de concepts

Lattice Miner

- Logiciel en Java pour la visualisation et la manipulation de treillis
- Initialement conçu par Geneviève Roberge dans le cadre de son mémoire de maîtrise.
- Continuellement enrichi par les membres de l'équipe et plusieurs stagiaires ingénieurs français.
- Aussi bien dans sa version initiale que dans sa version révisée *Lattice Miner* est un logiciel libre disponible sur *SourceForge*⁷ et GitHub⁸.

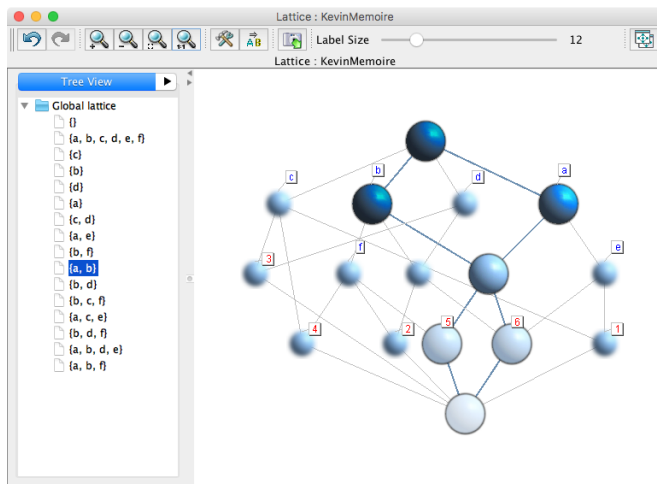
7. <https://sourceforge.net/projects/lattice-miner/>

8. <https://github.com/LarimUQO/lattice-miner>

Les outils de l'analyse formelle de concepts

Lattice Miner

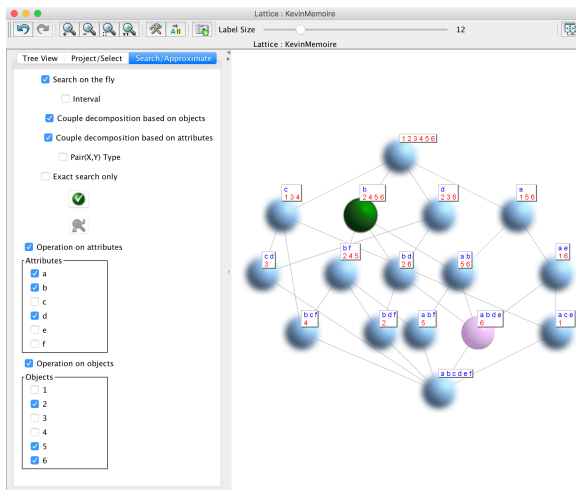
Figure 9: Génération du treillis de concepts dans *Lattice Miner*



Les outils de l'analyse formelle de concepts

Lattice Miner

Figure 10: Approximation dans *Lattice Miner*



Les outils de l'analyse formelle de concepts

Lattice Miner

Figure 11: Affichage de l'apposition d'un contexte avec son complémentaire

Lattice Miner

Context : KevinMemoire

Context : KevinMemoire

	a	b	c	d	e	f	a'	b'	c'	d'	e'	f'
1	X		X		X			X		X		X
2		X		X		X	X		X		X	
3			X	X			X	X			X	X
4		X	X			X	X			X	X	
5	X	X				X			X	X	X	
6	X	X		X	X				X			X

Les outils de l'analyse formelle de concepts

Lattice Miner

Figure 12: Base générique d'implications avec possibilité de redondance

Association Rule Viewer

Context : KevinMemoire

Min. support : 10.0%

Min. confidence : 100.0%

Rule count : 11

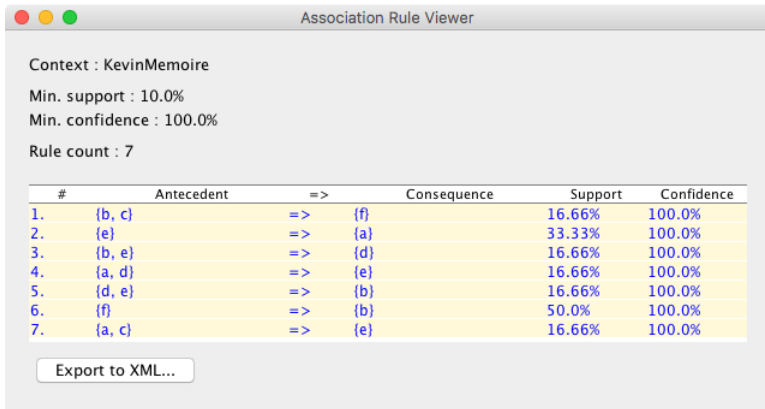
#	Antecedent	=>	Consequence	Support	Confidence
1.	{a, c}	=>	{e}	16.66%	100.0%
2.	{c, e}	=>	{a}	16.66%	100.0%
3.	{b, c}	=>	{f}	16.66%	100.0%
4.	{c, f}	=>	{b}	16.66%	100.0%
5.	{e}	=>	{a}	33.33%	100.0%
6.	{f}	=>	{b}	50.0%	100.0%
7.	{b, e}	=>	{a, d}	16.66%	100.0%
8.	{a, d}	=>	{b, e}	16.66%	100.0%
9.	{d, e}	=>	{a, b}	16.66%	100.0%
10.	{a, f}	=>	{b}	16.66%	100.0%
11.	{d, f}	=>	{b}	16.66%	100.0%

Export to XML...

Les outils de l'analyse formelle de concepts

Lattice Miner

Figure 13: Base générique d'implications sans redondance



Context : KevinMemoire
 Min. support : 10.0%
 Min. confidence : 100.0%
 Rule count : 7

#	Antecedent	=>	Consequence	Support	Confidence
1.	{b, c}	=>	{f}	16.66%	100.0%
2.	{e}	=>	{a}	33.33%	100.0%
3.	{b, e}	=>	{d}	16.66%	100.0%
4.	{a, d}	=>	{e}	16.66%	100.0%
5.	{d, e}	=>	{b}	16.66%	100.0%
6.	{f}	=>	{b}	50.0%	100.0%
7.	{a, c}	=>	{e}	16.66%	100.0%

Export to XML...

Les outils de l'analyse formelle de concepts

Lattice Miner

Figure 14: Relations flèches dans *Lattice Miner*

Lattice Miner

Context : KevinMemoire

Context : KevinMemoire

	a	b	c	d	e	f
1	X	↓	X	↓	X	✓
2	↓	X	↓	X	✓	X
3	↓	↓	X	X	✓	✓
4	↓	X	X	↓	✓	X
5	X	X	↓	↓	↓	X
6	X	X	↓	X	X	↓

Production d'implications avec négation

Production d'implications avec négation

Définition 15 : Catégories d'implications

$$r : Y \rightarrow Z \text{ [sup]}$$

r est une implication du contexte $\mathbb{K}|\tilde{\mathbb{K}}$ tel que :

- Y et Z sont des sous-ensembles de $M \cup \tilde{M}$
- $Y \cap Z = \emptyset$

Les implications appartiennent à une des catégories suivantes :

- implications *purement négatives* avec $M \cap (Y \cup Z) = \emptyset$,
- implications *purement positives* avec $\tilde{M} \cap (Y \cup Z) = \emptyset$, et
- implications *mixtes* tel que $M \cap (Y \cup Z) \neq \emptyset$ et $\tilde{M} \cap (Y \cup Z) \neq \emptyset$.

- $\{b'\} \rightarrow \{d'\}$ est *purement négative*

- $\{b, c, e'\} \rightarrow \{a'\}$ est *mixte*

- $\{e\} \rightarrow \{c\}$ est *purement positive*

- La confiance est de 100%.

Production d'implications avec négation

Définition 16 : Implications basées sur les clés (*key-based implication*)

$$r : Y \rightarrow M \cup \tilde{M} \setminus Y [0]$$

- Y une clé dans le contexte $\mathbb{K}|\tilde{\mathbb{K}}$, ce qui signifie que Y implique tous les attributs dans $M \cup \tilde{M}$
- Le support d'une telle implication est évidemment nul car il n'existe aucun itemset Y impliquant tous les attributs dans $M \cup \tilde{M}$.
- $\{c', d\}$ est une clé, on n'a jamais c' et d en même temps.

Production d'implications avec négation

Solution

Théorème

*Soit Ax avec x un attribut de $M\tilde{M}$ et $A \subseteq M\tilde{M}$. Alors, $Ax \rightarrow M\tilde{M} \setminus Ax$ [0]
 $\Leftrightarrow A \rightarrow \tilde{x}$ [sup] avec $\text{sup} = |A'|/|G|$ où A' est l'ensemble d'objets possédant les attributs contenus dans A et G est l'ensemble des objets du contexte \mathbb{K} .*

- Ces implications avec clé de la forme $Ax \rightarrow M\tilde{M} \setminus Ax$ ont un support nécessairement nul puisque même si un objet possède tous les attributs dans M , il n'en possède aucun dans \tilde{M} .
- On déplace un attribut à la fois.
- Par exemple, $\{c', e\}$ nous donne les implications $\{e\} \rightarrow \{c\}$ et $\{c'\} \rightarrow \{e'\}$

Production d'implications avec négation

Algorithme 1 : Génération des clés de l'infimum du treillis $\mathbb{K}|\tilde{\mathbb{K}}$

```

1: Procédure
2: Entrée : Contexte initial  $\mathbb{K} := (G, M, I)$ 
3: Sortie :  $\mathcal{K}$  : Les générateurs de l'infimum du treillis du contexte  $\mathbb{K}|\tilde{\mathbb{K}}$ 
4:  $INT \leftarrow \emptyset$  {Ensemble des intentions des co-atomes}
5:  $O \leftarrow \emptyset$  {Objets traités}
6: while  $G \neq O$  do
7:   for all  $o \in G$  mais  $o \notin O$  do
8:      $X \leftarrow o''$  {Calcul du fermé de l'objet  $o$ }
9:      $Y \leftarrow Intent(o) \cup (M \setminus Intent(o))^\sim$ 
10:     $O := O \cup X$ 
11:     $INT := INT \cup \{Y\}$ 
12:   end for
13: end while
14:  $\mathcal{K} \leftarrow JEN(M \cup \tilde{M}, INT)$ 
15: return  $\mathcal{K}$ 

```

Production d'implications avec négation

	a	b	c	d	e
1	×				
2	×		×		
3	×	×			
4		×			
5	×		×		×
6	×	×	×		×
7		×	×	×	
8		×	×		×

Figure 15: Exemple de contexte \mathbb{K}

- L'exécution de l'algorithme 1 nous donne les clés suivantes :
 $\{a, a'\}$, $\{a', b'\}$, $\{b, b'\}$, $\{a, d\}$,
 $\{b', d\}$, $\{d, d'\}$, $\{c, c'\}$, $\{c', d\}$,
 $\{c', e\}$, $\{d, e\}$, $\{e, e'\}$,
 $\{a, b, c, e'\}$, $\{b, c, d', e'\}$,
 $\{a', c, d', e'\}$.
- Après avoir écarté les clés triviales représentant un attribut et sa négation (ex. $\{a, a'\}$), on retient les éléments suivants :
 $\{a', b'\}$, $\{a, d\}$, $\{b', d\}$, $\{c', d\}$,
 $\{c', e\}$, $\{d, e\}$, $\{a, b, c, e'\}$,
 $\{b, c, d', e'\}$, $\{a', c, d', e'\}$.

Production d'implications avec négation

Création des implications avec négation à partir des clés candidates

Candidats	$\{a', b'\}$	$\{a, d\}$	$\{b', d\}$
Règles	$\{b'\} \rightarrow \{a\}$ $\{a'\} \rightarrow \{b\}$	$\{d\} \rightarrow \{a'\}$ $\{a\} \rightarrow \{d'\}$	$\{d\} \rightarrow \{b\}$ $\{b'\} \rightarrow \{d'\}$
Candidats	$\{c', e\}$	$\{d, e\}$	$\{a, b, c, e'\}$
Règles	$\{e\} \rightarrow \{c\}$ $\{c'\} \rightarrow \{e'\}$	$\{e\} \rightarrow \{d'\}$ $\{d\} \rightarrow \{e'\}$	$\{b, c, e'\} \rightarrow \{a'\}$ $\{a, c, e'\} \rightarrow \{b'\}$ $\{a, b, e'\} \rightarrow \{c'\}$ $\{a, b, c\} \rightarrow \{e\}$
Candidats	$\{b, c, d', e'\}$	$\{a', c, d', e'\}$	$\{c', d\}$
Règles	$\{c, d', e'\} \rightarrow \{b'\}$ $\{b, d', e'\} \rightarrow \{c'\}$ $\{b, c, e'\} \rightarrow \{d\}$ $\{b, c, d'\} \rightarrow \{e\}$	$\{c, d', e'\} \rightarrow \{a\}$ $\{a', d', e'\} \rightarrow \{c'\}$ $\{a', c, e'\} \rightarrow \{d\}$ $\{a', c, d'\} \rightarrow \{e\}$	$\{d\} \rightarrow \{c\}$ $\{c'\} \rightarrow \{d'\}$

Production d'implications avec négation

Intégration de la solution dans *Lattice Miner*

Figure 16: Affichage des implications avec négation

Association Rule Viewer

Context : ExempleNeg

Min. support : 0.0%

Min. confidence : 100.0%

Rule count : 24

#	Antecedent	=>	Consequence	Support	Confidence
1.	{b'}	=>	{a}	37.5%	100.0%
2.	{a'}	=>	{b}	37.5%	100.0%
3.	{d}	=>	{a'}	12.5%	100.0%
4.	{a}	=>	{d'}	62.5%	100.0%
5.	{d}	=>	{b}	12.5%	100.0%
6.	{b'}	=>	{d'}	37.5%	100.0%
7.	{d}	=>	{c}	12.5%	100.0%
8.	{c'}	=>	{d'}	37.5%	100.0%
9.	{e}	=>	{c}	37.5%	100.0%
10.	{c'}	=>	{e'}	37.5%	100.0%
11.	{e}	=>	{d'}	37.5%	100.0%
12.	{d}	=>	{e'}	12.5%	100.0%
13.	{b, c, e'}	=>	{a'}	12.5%	100.0%
14.	{a, c, e'}	=>	{b'}	12.5%	100.0%
15.	{a, b, e'}	=>	{c'}	12.5%	100.0%
16.	{a, b, c}	=>	{e}	12.5%	100.0%
17.	{c, d', e'}	=>	{b'}	12.5%	100.0%
18.	{b, d', e'}	=>	{c'}	25.0%	100.0%
19.	{b, c, e'}	=>	{d}	12.5%	100.0%
20.	{b, c, d'}	=>	{e}	25.0%	100.0%
21.	{c, d', e'}	=>	{a}	12.5%	100.0%
22.	{a', d', e'}	=>	{c'}	12.5%	100.0%
23.	{a', d', e'}	=>	{c}	12.5%	100.0%

Enrichissement du module de génération des règles triadiques

Enrichissement du module de génération des règles triadiques

Analyse triadique de concepts

L'analyse triadique de concepts a été introduite par (Lehmann et Wille 1995) comme une extension à l'analyse formelle de concepts.

Définition 17 : Contexte triadique

Soit un contexte triadique $\mathbb{K} := (K_1, K_2, K_3, Y)$

- avec K_1 , K_2 et K_3 représentant respectivement un ensemble d'objets, un ensemble d'attributs et un ensemble de conditions, et
- $Y \subseteq K_1 \times K_2 \times K_3$ une relation ternaire entre les trois ensembles.

Le triplet (a_1, a_2, a_3) dans Y signifie que l'objet a_1 possède l'attribut a_2 sous la condition a_3 (voir le tableau 2).

Le but d'une telle analyse est d'identifier des concepts et des implications triadiques.

Enrichissement du module de génération des règles triadiques

Analyse triadique de concepts

Définition 18 : Concept triadique

Un concept triadique d'un contexte $\mathbb{K} := (K_1, K_2, K_3, Y)$ est un triplet (A_1, A_2, A_3) avec $A_1 \subseteq K_1$, $A_2 \subseteq K_2$, $A_3 \subseteq K_3$ et $A_1 \times A_2 \times A_3 \subseteq Y$. Il représente un cuboïde plein de 1. Les sous-ensembles A_1 , A_2 et A_3 sont appelés respectivement l'extension, l'intention et le mode (*modus*) du concept.

- Par exemple, $(12345, PRK, a)$ et $(14, PN, bd)$ sont des concepts triadiques.

Enrichissement du module de génération des règles triadiques

Analyse triadique de concepts

\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

Tableau 2: Un contexte triadique $\mathbb{K} := (K_1, K_2, K_3, Y)$

- $(12345, PRK, a)$ et $(14, PN, bd)$ sont des concepts
- $(135, PN, d)$ n'est pas un concept car il est non maximal puisque son extension peut être augmentée pour aboutir au concept $(12345, PN, d)$ tout en respectant la relation ternaire.

Enrichissement du module de génération des règles triadiques

Implications triadiques

Définition 19 : Implications triadiques (Biedermann 1997)

Une implication triadique de la forme $(A \rightarrow D)_C$ est vraie si chaque fois que A se produit pour l'ensemble des conditions dans C , alors D se produit également dans les mêmes conditions.

- Exemple, $(\{K\} \rightarrow \{P, R\})_a$

Enrichissement du module de génération des règles triadiques

Implications triadiques (Ganter et Obiedkov 2004)

Définition 19 : Implication *attribut*×*condition*

Une implication *attribut*×*condition* ($A \times C \rightarrow I$) a la forme $A \rightarrow D$, avec A, D des sous-ensembles de $K_2 \times K_3$. De telles implications sont extraites du contexte binaire $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$ où $(a_i, (a_j, a_k)) \in Y^{(1)} :\Leftrightarrow (a_i, a_j, a_k) \in Y$.

$\mathbb{K}^{(1)}$	P				N				R				K				S			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
1	1	1		1	1	1		1	1		1		1	1			1			
2	1			1		1	1	1	1	1		1				1				1
3	1	1		1				1	1	1			1	1			1			
4	1	1		1		1		1	1	1			1	1						1
5	1			1	1			1	1	1		1	1	1			1			

Tableau 3: Contexte dyadique $\mathbb{K}^{(1)} := (K_1, K_2 \times K_3, Y^{(1)})$ extrait de \mathbb{K}

Enrichissement du module de génération des règles triadiques

Implications triadiques (Ganter et Obiedkov 2004)

Définition 20 : Implication *attributs conditionnels* (CAI)

Une implication d'*attributs conditionnels* (CAI) est de la forme $A \xrightarrow{C} D$

- avec A et D des sous-ensembles de K_2 ,
- et C un sous-ensemble de K_3 .

L'implication signifie que A implique D pour la totalité ou tout sous-ensemble des conditions de C . Une telle implication est alors liée à la définition de l'implication triadique de Biedermann comme suit :

$A \xrightarrow{C} D \iff (A \rightarrow D)_{C_1}$ pour tout $C_1 \subseteq C$, y compris pour tout élément atomique de C .

- Par exemple, $\{K\} \xrightarrow{ad} \{P, R\}$

Enrichissement du module de génération des règles triadiques

Implications triadiques (Ganter et Obiedkov 2004)

Définition 21 : Implication *conditions attributionnels* (ACI)

Une implication de type conditions attributionnels (ACI) est de la forme $A \xrightarrow{C} D$, où A et D sont des sous-ensembles de K_3 , et C est un sous-ensemble de K_2 .

Enrichissement du module de génération des règles triadiques

Implications triadiques (Ganter et Obiedkov 2004)

Définition 22 : Règle d'association d'attributs conditionnels à la Biedermann (BCAAR)

Une règle BCAAR $(A \rightarrow D)_C [s, c]$ est vraie si chaque fois que $A \subseteq K_2$ est vraie sous l'ensemble de conditions (et non nécessairement un sous-ensemble de) $C \subseteq K_3$, alors $D \subseteq K_2$ est vraie pour C avec un support s et une confiance c .

- Par exemple, $(K \rightarrow P)_b [60\%, 75\%]$

Enrichissement du module de génération des règles triadiques

Algorithme de calcul des implications triadiques CAI et ACI

En partant de la définition même d'une implication CAI qui indique que $A \xrightarrow{C} D$ est vraie ssi $(A \rightarrow D)_k$ pour tout $k \in C$, nous allons procéder comme suit :

- 1 créer autant de contextes dyadiques qu'il y a de conditions dans K_3
- 2 calculer les implications à la Biedermann au niveau des chacune des conditions
- 3 placer les implications avec un même antécédent dans un même baquet (*bucket*) puis décomposer chaque implication avec p attributs dans la conclusion en p implications avec un seul attribut dans la conséquence
- 4 cumuler les conditions pour toutes les implications ayant le même antécédent et la même conséquence
- 5 regrouper les conclusions des implications ayant le même antécédent
- 6 calculer le support des implications obtenues.

Enrichissement du module de génération des règles triadiques

Algorithme de calcul des implications triadiques CAI et ACI

En partant de la définition même d'une implication CAI qui indique que $A \xrightarrow{C} D$ est vraie ssi $(A \rightarrow D)_k$ pour tout $k \in C$, nous allons procéder comme suit :

- 1 créer autant de contextes dyadiques qu'il y a de conditions dans K_3
- 2 calculer les implications à la Biedermann au niveau des chacune des conditions
- 3 placer les implications avec un même antécédent dans un même baquet (*bucket*) puis décomposer chaque implication avec p attributs dans la conclusion en p implications avec un seul attribut dans la conséquence
- 4 cumuler les conditions pour toutes les implications ayant le même antécédent et la même conséquence
- 5 regrouper les conclusions des implications ayant le même antécédent
- 6 calculer le support des implications obtenues.

Conclusion

Contributions :

- Vérification des procédures de la base d'implications de Guigues-Duquenne ;
- Vérification des procédures de calcul des relations de flèches ;
- Production d'implications avec négation ;
- Enrichissement du module de génération des règles triadiques ;
- Mise à jour de Lattice Miner avec la dernière version de Java 8, support de Maven, et documentation technique ;
- Article accepté dans les actes supplémentaires pour la conférence ICFCA 2017 (Missaoui et Emamirad 2017) ;

Travaux futurs :

- Procédure qui permet de calculer le fermé d'un groupe d'attributs Y avec ou sans un ensemble C de conditions ;
- Exploiter le parallélisme dans un environnement MapReduce ;
- Version Web *Lattice Miner* ;

References I



Jean-Louis Guigues et Vincent Duquenne. « Familles minimales d'implications informatives résultant d'un tableau de données binaires ». In : *Mathématiques et Sciences Humaines* 95 (1986), p. 5–18.



Fritz Lehmann et Rudolf Wille. « A Triadic Approach to Formal Concept Analysis ». In : *Proceedings of the Third International Conference on Conceptual Structures: Applications, Implementation and Theory*. 1995, p. 32–43.



Klaus Biedermann. « How Triadic Diagrams Represent Conceptual Structures ». In : *ICCS*. 1997, p. 304–317.



Bernhard Ganter et Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., 1999.

References II



Nicolas Pasquier et al. « Efficient Mining of Association Rules Using Closed Itemset Lattices ». In : *Information Systems* 24.1 (1999), p. 25–46.



Peter Becker, Joachim Hereth et Gerd Stumme. « ToscanaJ: An Open Source Tool for Qualitative Data Analysis ». In : *Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD'2002)*. Juil. 2002, p. 1–2.



J. Pfaltz et C. Taylor. « Scientific Discovery through Iterative Transformations of Concept Lattices ». In : *Proceedings of the 1st International Workshop on Discrete Mathematics and Data Mining*. Avr. 2002, p. 65–74.



Bernhard Ganter et Sergei A. Obiedkov. « Implications in Triadic Formal Contexts ». In : *ICCS*. 2004, p. 186–195.

References III



Laszlo Szathmary. « Symbolic Data Mining Methods with the Coron Platform ». *Thèse de doctorat. Univ. Henri Poincaré - Nancy 1, France, nov. 2006.*



Geneviève Roberge. « Visualisation des résultats d'une fouille de données dans les treillis de concepts ». *Mém.de mast. Université du Québec en Outaouais, 2007.*



Christophe Demko et Karell Bertet. « Generation Algorithm of a Concept Lattice with Limited Object Access ». *In : Proceedings of The Eighth International Conference on Concept Lattices and Their Applications. 2011, p. 239–250.*



Rokia Missaoui, Lhouari Nourine et Yoan Renaud. « Computing Implications with Negation from a Formal Context ». *In : Fundam. Inf. 115.4 (déc. 2012), p. 357–375.*

References IV



Karell Bertet et al. *java-lattices: a Java library for lattices computation*. <http://thegalactic.org>. 2014.



Jean-François Viaud et al. « Décomposition sous-directe d'un treillis en facteurs irréductibles ». In : *Journées francophones d'Ingénierie des Connaissances IC 2015*. collection AFIA. Rennes, France, juin 2015.



Rokia Missaoui et Kévin Emamirad. « Lattice Miner 2.0: A Formal Concept Analysis Tool ». In : *Supplementary Proc. of ICFCA, Rennes, France, 2017*. 2017, p. 91–94.

Questions ?