UNIVERSITE GASTON BERGER

Master2 Informatique

Memoire de fin de cycle

Présenté par:

PEKPASSI Digonaou

29 Juillet 2015

Table des matières

1 Etat de l'art

1.1 Notions de base

1.1.1 Pré requis

Definition 1. ***Relation binaire***
Une relation binaire $\mathcal{R}$ sur un ensemble E est un sous-ensemble du produit cartésien $E \times E$ ie un ensemble de couples (x, y) d'éléments de E. Nous noterons $x\mathcal{R}y$ pour indiquer que le couple (x,y) appartient à la relation $\mathcal{R}$. Une relation binaire peut être :

- *réflexive si* $\forall x \in E,\ x\mathcal{R}x$
- *irréflexive si* $\forall x \in E,\ \neg(x\mathcal{R}x)$
- *symétrique si* $\forall x, y \in E,\ x\mathcal{R}y \Rightarrow y\mathcal{R}x$
- *antisymétrique si* $\forall x, y \in E,\ x\mathcal{R}y \wedge y\mathcal{R}x \Rightarrow x = y$
- *asymétrique si* $\forall x, y \in E,\ x\mathcal{R}y \Rightarrow \neg(y\mathcal{R}x)$
- *complète si* $\forall x, y \in E, x\mathcal{R}y \vee y\mathcal{R}x$
- *transitive si* $\forall x, y \in E, x\mathcal{R}y \wedge y\mathcal{R}z \Rightarrow x\mathcal{R}z$
- *négativement transitive si* $\forall x, y \in E, \neg(x\mathcal{R}y) \wedge \neg(y\mathcal{R}z) \Rightarrow \neg(x\mathcal{R}z)$

Definition 2. ***Relation d'indifférence*** *Etant donnée une relation $\mathcal{R}$ sur un ensemble E, la relation d'indifférence noté $\tilde{\mathcal{R}}$ est définie pout out $x, y \in E$ par $\tilde{\mathcal{R}}six\mathcal{R}y$ et $y\mathcal{R}x$.*

Definition 3. ***Relation d'incompatibilité*** *Etant donnée une relation $\mathcal{R}$ sur un ensemble E, la relation d'incompatibilité notée $||_{\mathcal{R}}$ est définie pour tout $x, y \in E$ par : $x||_{\mathcal{R}}y$ si $\neg(x\mathcal{R}y)$ et $\neg(y\mathcal{R}x)$*

Definition 4. ***Relation de préférence*** *Une relation de préférence sur un ensemble d'éléments E est une relation binaire.*
Dans ce cas, étant donné une relation de préférences notée $\succ_{\mathcal{R}}$:

1. *$x \succ_{mathcalR} y$ signifie que x est strictement préféré à y,*
2. *$x\tilde{\mathcal{R}}y$ signifie que x et y sont également préférés, et enfin,*
3. *$x||_{\mathcal{R}}y$ signifie qu'il n'y a ni indifférence, ni préférence entre x et y et on dit que x et y ne sont pas comparables.*

Definition 5. ***Relation d'ordre*** *On appelle relation d'ordre sur un ensemble E, toute relation binaire sur E qui est réflexive, antisymétrique et transitive.*

Si la relation d'ordre est complète, elle est dite ordre total sinon l'ordre est partiel.

Definition 6. ***Préordre***
Un préordre sur un ensemble E est une relation binaire réflexive et transitive.

Á une relation d'ordre on peut associer une relation obtenue en ôtant de celle-ci les couples d'éléments identiques.

Definition 7. *(Relation d'ordre strict) Une relation d'ordre strict sur un ensemble E est une relation binaire irréflexive et transitive. L'ordre strict est dit faible, s'il est partiel et négativement transitif.*

1.1.2 Les différents types de représentation de préférences qualitatives

Dans ce qui suit, nous considérons l'ensemble E comme étant l'ensemble des tuples $t = (u_1, u_2, ..., u_d)$ avec $u_i \in dom(A_i)$ de $R(A_1, A_2, ..., A_d)$. Nous définissons suivant cela une relation de préférence ($\succ_p$) sur un schéma R par un ensemble de tuples (t_i, t_j) de r. Par conséquent $t_i \succ_p t_j$ signifiera que t_i est préféré à t_j sous $\succ_p$.

Formulation de Jan Chomicki

CHOMICKI a dans l'article [**?**] établit la notion de **relation de préférence** $\succ$ comme un sous ensemble de $Dom(A) \times Dom(A)$. Ceci veut intuitivement dire que $\succ$ est une relation binaire entre des transactions d'une même instance r du schémas relationnel $\mathcal{R}$. Ainsi on a t_1 est préféré à t_2 suivant $\succ$ si $t_1 \succ t_2$. De même il a définit la notion de formule de préférence $C(t_1, t_1)$ comme étant une une formule de premier ordre définissant une relation de préférence $\succ_C$ de telle sorte que $t_1 \succ_C t_2 \equiv C(t_1, t_2)$. Cette formule est définie ci dessous.

Definition 8. *Une formule de préférence suivant le formalisme de CHOMICKY est une formule représentée sous la forme normale disjonctive DNF sans les quantificateurs. Ainsi on a :*

$$C(t_1, t_2) = \bigvee_{i=1..k} (\bigwedge_{j=1..l} f_{ij})$$

où f_{ij} sont des formules atomiques sous la forme $x\delta y$ ou $x\delta c$ avec x et y des variables d'un attribut de R correspondants respectivement à t_1 et t_2. c quand à lui est une constante. δ prend les valeurs $=, \neq$ dans le cas où les éléments x, y, c sont de type quelconque (Exemple : $x = y, x = a, y \neq a$), et $\leqslant, \geqslant, <, >$ dans le cas où on a à faire à des valeurs numériques

(Exemple : $x < y$,$x < a$,$y \geqslant a$).

Example 1

— Soit deux transactions $t_1 = \{x_1, x_2, x_3\}$ et $t_2 = \{y_1, y_2, y_3\}$ d'une instance de la relation R(Plat,TypePlat, TypeVin). On a qu'ils vérifient la formule de préférence C si et seulement si

$$\begin{aligned} C(t_1, t_2) \equiv\ & (x_1 = y_1 \wedge x_2 = "fish" \wedge y_2 = "fish" \wedge x_3 = "white" \wedge y_3 = "red") \\ & \vee (x_1 = y_1 \wedge x_2 = "meat" \wedge y_2 = "meat" \wedge x_3 = "red" \wedge y_3 = "white") \end{aligned}$$

est vérifié. Cette condition veut dire que dans le cas où c'est un plat à base de poisson, le client préfère le vin blanc au vin rouge et dans le cas où c'est un plat à base de viande, le client préfère le vin rouge au vin blanc.

— En prenant en compte les valeurs numériques, soit deux transactions $t_1 = \{x_1, x_2, x_3\}$ et $t_2 = \{y_1, y_2, y_3\}$ d'une instance de la relation R(TypeFilm,Annee, Acteur). On a $t_1 \succ_C t_2$ ssi :

$$\begin{aligned} C(t_1, t_2) \equiv\ & (x_1 = y_1 \wedge x_2 > 2010 \wedge y_2 < 2000) \\ & \vee (x_1 = y_1 \wedge x_2 = y_2 \wedge x_3 = \text{"Sylvester Stalone"} \wedge y_3 = \text{"Pierce Brosnan"}) \end{aligned}$$

Cette règle veux tout simplement dire que le client préfère pour des films de même type, des films au delà de 2010 aux films produits avant 2000, et si les films sont de même type et de meême anné, il préfère ceux avec l'acteur Sylvester Stalone à ceux avec l'acteur Pierce Brosnan.

Formulation de Nic Wilson

Le formalisme de Wilson se fonde sur la logique des préférences conditionnelles. Une règle de préférence conditionnelle a sa forme générale comme suit : "si [conjonction de conditions élémentaires] alors [décision]"

La partie condition indique dans quelle condition la règle est appliquée et la partie décision montre le choix à opérer sur les valeurs si la partie condition est satisfaite. Plus précisément, le formalisme proposé par Wilson est défini de la façon suivante :

Definition 9. *Soit $\mathcal{A}$ un ensemble d'attributs, $U \subseteq \mathcal{A}$, a_{i1} et $a_{i2} \in dom(A_i)$, $A_i \notin U$. Une règle de préférence contextuelle est sous la forme $u : a_{i1} \succ a_{i2}[W]$ tel que $u \in dom(U)$ et*

W un sous-ensemble de $S = \mathcal{A} - (U \cup A_i)$.

Cette formulation peut être réécrite suivant le formalisme de CHOMICKY (Définition 8) de la façon suivante (dans cette situation on a $p = u : a_{i1} \succ a_{i2}[W]$) :
$t_i \succ_p t_j$ ssi $t_i[S - W] = t_j[S - W] \wedge t_i[U] = t_j[U] = u \wedge t_i[A_i] = a_i 1 \wedge t_j[A_i] = ai$

De façon intuitive cette règle de préférence contextuelle indique que tout tuple t_i contenant u et a_{i1} est préféré à tout tuple t_j contenant u et a_{i2} indépendamment des valeurs des attributs W dans $dom(W)$, sachant que les valeurs dans $dom(S - W)$ sont les mêmes pour les deux tuples.

Par cette formulation, l'exemple 1 peut se réécrire de la façon suivante :

Example 2

Dans le cas d'un schéma relationnel $\mathcal{R} = \{Langage, Title, Director, Actor\}$ on a l'exemple suivant
$Langage = English \succ Langage = French[Title, Director, Actor]$

Cette règle indique la préférence des films Anglais aux films Français sans tenir compte du titre du film, du directeur du film ou bien de l'acteur, pour tout couple de film ayant la même année de production.

Formulation de Werner Kießling

Dans les travaux de [Kie02], l'auteur propose un langage formel pour modéliser les préférences qui forment une relation d'ordre partiel strict. Pour cela, l'auteur propose un certain nombre de constructeurs de base qui sont des patrons de préférence, qui dès qu'ils sont instanciés donnent des préférences de base. Ces constructeurs sont caractérisés par plusieurs arguments dont le premier indique les noms des attributs concernés par le constructeur et les autres arguments indiquant les caractéristiques de l'ordre partiel stricte $< P$ qui sera appliqué sur les attributs énoncés dans le premier argument. Dans ce travail, nous présenterons deux types de constructeurs de base à savoir les **constructeurs de base non numériques** et les **constructeurs de base numériques**.

Les constructeurs de base non numériques

Leurs définition ne prend pas en compte d'opérations numériques.
Exemple :

Préférence POS : POS(A, POS-set)
P est une préférence POS si :

$$x < Pyssix \notin POS - set \wedge y \in POS - set$$

Ceci veut dire que parmis deux valeurs de A, la valeur préférée est celle qui appartient à $POS-set \subset dom(A)$. Autrement si aucune des valeurs n'est dans $POS-set$, alors une des deux est accepable. (Ex :scénario d'un choix de voiture : POS(transmission, automatique))

Préférence POS/NEG : POS/NEG(A,POS-set ;NEG-set)
P est appelé préférence POS/NEG si :

$$x < Pyiff(x \in NEG-set \wedge y \notin NEG-set) \vee (x \notin NEG-set \wedge x \notin POS-set \wedge y \in POS-set)$$

Ceci veut dire que parmi deux valeurs, la préférée est celle qui appartient à $POS-set \subset A$, autrement si aucune des deux valeurs n'est dans $POS-set$, et qu'une d'elle n'appartient pas à $NEG-set \subset A$, elle est la préférée, autrement si les deux appartiennent à $NEG-set$ alors une d'elles est choisie arbitrairement. (Ex : scénario de choix de voiture : POS/NEG(couleur,jaune ;gris)

Les constructeurs de base non numériques
Leur définition prend en compte des opérations numériques.
Exemple :

Préférence AROUND : AROUND(A, z)
Etant donné $z \in dom(A), pourtout v \in dom(A)$, on défini $distance(v, z) := abs(v - z)$. P est appelé préférence AROUND, si :

$$x < Pyssidistance(x, z) > distance(y, z)$$

Ceci veut dire que parmi deux valeurs de A, si l'une d'elle est z, elle est la valeur préférée. Autrement, si aucune des valeurs n'est z, la plus préférée est celle qui iest la plus proche de Z e terme de distance. (Ex : scénario de choix de voiture : AROUND(prix, 40000))

Préférences LOWEST, HIGHEST : LOWEST(A),HIGHEST(A)

P est appelé préférence LOWEST, si :

$$x < Pyssix > y$$

P est appelé préférence HIGHEST, si :

$$x < Pyssix < y$$

Ceci veut dire qu'une valeur désirée doit être la plus élevé (basse) possible. (Ex : scénario choix de voiture : HIGHEST(puissance).

1.2 Les différents modes de combinaison de préférences

Après avoir présenté précédemment les différentes formulations des préférences dans la littérature, nous allons à présent étudier les différentes méthodes de combinaisons de ces préférences afin de former des modèles de préférence.

1.2.1 Composition de préférences prioritaires

Elle est définie comme suit :

Definition 10. *Soient P_x et P_y deux relations de préférence définies sur le même schéma relationnel R. La relation de préférence prioritaire $\succ_{P_x}$ & $\succ_{P_y}$ est définie sur R par :*

$$\forall t_i, t_j \text{ d'une instance de } R, t_i \succ_{P_x} \& \succ_{P_y} t_j \text{ ssi } (t_i \succ_{P_x} t_j) \vee (t_i \sim_{P_x} t_j \wedge t_i \succ_{P_y} t_j).$$

Ceci veut dire qu'on utilise la préférence P_x si elle est applicable et autrement on utilise P_y.

Example 3

Soient deux relations de préférence $\succ_{P_x}$ et $\succ_{P_y}$ tel que :

— $t_i \succ_{P_x} t_j$, ssi $t_i[genre] =' Drama' \wedge t_j[genre] =' Comedy'$.

— $t_i \succ_{P_y} t_j$, ssi $t_i[Language] =' French' \wedge t_j[Language =' English']$.

La relation de préférence prioritaire $\succ_{P_x}$ & $\succ_{P_y}$ est alors définie comme suit :

$t_i \succ_{P_x} \& \succ_{P_y} t_j$ *ssi* $(t_i[genre] =' Drama' \wedge t_j[genre] =' Comedy') \vee (t_i[genre] \neq' Drama' \wedge t_i[Language] =$

1.2.2 Réseaux de préférences conditionnelles (CP-nets)

Un CP-nets est une approche de représentation des préférences fondée sur un représentation graphique. Ceci se fait à l'aide de deux notions à savoir :

— la notion de préférence conditionnelle ;

— la notion de ceteris paribus.

Soit $V = \{X_1, X_2, ..., X_i\}$ La notion de préférence conditionnelle est d'indiquer pour chaque attribut X_i ses attributs parents $Pa(X_i)$, c'est à dire les autres attributs dont il dépend par rapport aux préférences de l'utilisateur. Ainsi lorsque l'on a une instantiation u de $Pa(X_i)$, on sait quel valeur de X_i est préférée à l'autre suivant u. La notion de ceteris paribus dans le cas d'une préférence conditionnelle est que pour un attribut X_i, le choix au niveau de cet attribut dépend de ses parents $Pa(X_i)$ indépendamment des valeurs fixées au niveau des autres attributs $Y = V - Pa(X_i) \cup X_i$.

En se basant sur ces notions, nous formons au niveau de chaque attribut X_i, ce qu'on appelle **Table de Préférence Conditionnelle CPT** (sur la condition du ceteris paribus). Celle ci énumère les différentes préférences au niveau des valeurs de X_i suivant les valeurs des attributs de $Pa(X_i)$.

Example 4

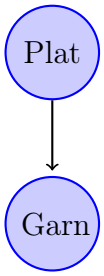
Cas des préférences au niveau des plats de riz. L'utilisateur préfère accompagné le riz au poisson plutôt qu'à la viande, et dans le cas du spagetti, il préfère la viande au poisson.

Riz	Poisson$\succ$ Viande
Spaghetti	Viande$\succ$ Poisson

L'ensemble de ces tables de préférences conditionnelles vont former les CP-nets comme on peut le voir au niveau de l'exemple ci dessous.

Example 5

Meme genre que l'exemple précédent.



Poisson$\succ$ Viande	
Riz	Poisson$\succ$ Viande
Spaghetti	Viande$\succ$ Poisson

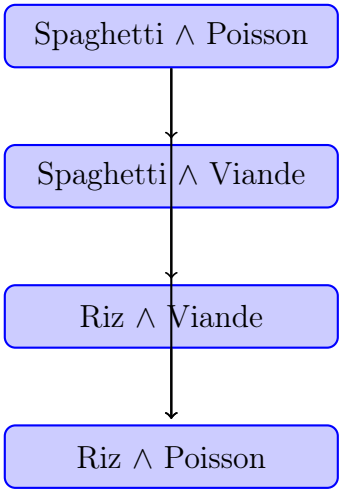
Ainsi nous définissons un CP-nets de la manière suivante :

Definition 11. *Un CP-net sur les attributs* $V = \{X_1, X_1, ..., X_n\}$ *est un graphe orienté sur les attributs* $X_1, X_1, ..., X_n$ *où chaque nœud* X_i *est annoté avec une table de préférence conditionnelle,* $CPT(X_i)$*. Chaque table de préférence conditionnelle* $CPT(A_i)$ *associe un ordre total* $\succ_u^i$ *pour chaque instanciation* u *de* $Pa(X_i)$*.*

A partir d'un CP-net on peut alors schématiser le graphe de préférence induit.

Example 6

Le graphe de préférence induit par rapport à l'exemple précédent nous donne :



Sur cette figure, un arc partant d'un élément a vers un élément b indique que la préférence de b sur a put être déterminée directement par les CPTs du Cp-net. On voit que l'élément le plus en haut (Spaghetti ∧ Poisson) est l'élément le moins préféré tandis que l'élément le plus en bas (Riz ∧ Poisson) est l'élément le plus préféré.

Comparaison de deux transaction suivant un CP-net

Definition 12. *Prenons un CP-net* N *sur* V*,* X_i *un attribut de* V *et* $Pa(X_i)$ *les attributs parents de* X_i *dans* V*. Soit* $Y = V_(U \cup \{X_i\})$*. Soit* $\succ_u^i$ *l'ordre induit par* $CPT(X_i)$ *sur*

$Dom(X_i)$ pour toute initialisation $u \in Ass(U)$ de $Pa(X_i)$. Soit $\succ$ une relation de préférences sur $Asst(V)$.
On peut dire qu'une relation de préférences $\succ$ satisfait $\succ_u^i$ ssi on a, pour tout $y \in Asst(Y)$ et tout $x, x' \in Dom(X_i)$, $yxu \succ yx'u$ chaque fois que $x \succ_u^i x'$.
Une relation de préférence $\succ$ satisfait $CPT(X_i)$ ssi elle satisfait $\succ_u^i$ pour tout $u \in Asst(U)$.
Donc un CP-net N est satisfait par $\succ$ ssi celui ci satisfait chacun des préférences conditionnelles exprimées dans les CPT de N suivent la notion de ceteris paribus.

Definition 13. *Soit N un CP-net sur l'ensemble des attributs V. Soit $a, b \in Asst(V)$. N entraine $a \succ b$ et noté $N \models a \succ b$ ssi $a \succ b$ pour toute relation de préférence qui satisfait N.*

---

2 Résumé article Preference Learning : An Introduction de Johannes Furnkranz 1 and Eyke Hullermeier

Introduction

Le raisonnement à base de préférence est un domaine de recherche en Intelligence Artificielle. Les préférences sont des contraintes faibles qui permetttent une meilleure flexibilité en terme d'étude et deprédiction des choix utilisateurs. Ainsi on peut dans le cas où les conditions de recherche fixé par un utilisateur ne donnent pas des résultats consistants, se baser sur ces préférences afin de proposer d'autres résultats suivant ses préférences qui porteront aussi autant d'intérêt.

Il est à noter que l'quisition de préférences se base sur plusieurs principes à savoir :

— Les languages de modélisation et de formalisme

— Les méthodes d'apprentissage automatique, de découverte

Ainsi l'étude des préférences est un domaine qui fait l'objet de recherche dans les disciplines tel que l'apprentissage artificiel, la fouille de données, les systèmes de recommandation.

L'apprentissage de préférence, grosso modo parlant consiste à extraires des modèles de préférences de données empiriques.

Deux approches sont à distinguer en terme de modélisation de préférences à savoir

— Les fonctions utilités

— Les relations de préférence

Comparativement aux méthodes basées sur les fonctions utilité, l'apprentissage des relations de préférence se distingue fortement des méthodes classiques comme la classification ou la régression du fait qu'elle permet d'obtenir des structures complexes comme les classements et les relations d'ordre partielles plutôt que des valeurs.

Ce document parcours de façon générale les recherches en cours dans le domaine de l'apprentissage de préférences et alors permettra d'avoir une meilleure vue de ce domaine.

2.1 Tâches d'apprentiissage de prférences

La tâche d'apprentissage de classement est celle qui a le plus d'attention dans l'apprentissage de préférence.Dans cette section nous proposons une terminologie unifiée et claire pour les problèmes de classement les plus importants.

Dans le cadre des notations, nous allons utiliser une terminologie qui est souvent utilisée en apprentissage supervisée. Ainsi un objet caractérisant une donnée est appellé instance dénoté x et la classe à laquelle il est associé sera appelé label de classe. L'espace caractéristique des instances sera noté $\mathcal{X}$ et l'espace de sortie sera dénoté $\mathcal{Y}$. Les instances sont souvent représentées sous forme de vecteur caractéristique :
$x = (x_1, x_2, ..., x_m) \in \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times ... \times \mathcal{X}_m$

Nous distinguons trois types de classement à savoir :

— Le classement de labels

— Le classement d'instances

— Le classement d'objets

2.1.1 Le classement de labels

Le classement de labels est un type de classement prenant en compte un espace d'instances $\mathcal{X}$ et un ensemble fini de labels $\mathcal{Y} == \{y_1, y_2, ..., y_k\}$. Il consiste à apprendre un "classeur de labels" qui est une fonction définie sur $\mathcal{X} \Rightarrow S_y$ qui prend en entrée une instance et fournie en sortie une permutation de l'ensemble des labels suivant un ordre total. Ainsi le classement de label peut être vu comme une généralisation de la classification conventionnelle où un classement total

$$y_{\pi_x^{-1}(1)} \succ_x y_{\pi_x^{-1}(2)} \succ_x ... \succ_x y_{\pi_x^{-1}(k)}$$

est associé à une instance x au lieu d'un seul label de classe das le cas de la classification. π_x est une permutation de $\{1, 2, ..., k\}$ telle que $\pi_x(i)$ est la position du label $y_(i)$ dans le classement associé à x.

L'ensemble d'apprentissage pour le classement de labels typiquement consiste en un ensemble de préférences par paires de la forme $y_i \succ y_j$ indiquant que y_i est préféré à y_j pour l'instance x.

Comme exemple de situations où on a affaire à un classement de label, on peut parler du classement de la pertinence des types de rubriques par journal comme le sport, les technologies, la santé etc..

Soit :

- Un ensemble d'instances d'apprentissage $\{x_l | l = 1, 2, ..., n\} \subseteq \mathcal{X}$ (chaque instance peut ou pas être représentée par un vecteur).
- Un ensemble de labels $\mathcal{Y} = \{y_i | i = 1, 2, ..., k\}$
- Pour toute instance d'entraînement x_l, un enesmble de préférences de la forme $y_i \succ_{x_l} y_j$

Trouvons :

Une fonction de classement qui lie toute instance x à un classement $\succ_x$ de $\mathcal{Y}$ (i.e une permutation $\pi_x \in \mathcal{S}_k$)

Mesures de performance

- Erreur de classement (ex : basée sur les mesures de corrélation entre les rangs) comparant les classements prédits aux classements cibles.
- Erreur de position comparant le rang prédit au label cible.

2.1.2 Classement d'instance

Ce classement se fonde sur le principe d'une 'classification ordinale où une instance x appartient à une des classes de l'ensemble de classes ordonnées $\mathcal{Y} = \{y_1, y_2, ..., y_k\}$ tel que $y_1 < h_2 < ... < y_k$.

Le jeux de données d'apprentissage consistent en un ensemble $\mathcal{T}$ d'instances labellisées. Comme exemple on peut considérer la répartition d'articles suivant les catégories "rejeté", "faible rejet", "faible acceptation", "acceptation". Suivant les données d'apprentissage, le but est alors d'apprendre des fonctions de classement assignant un score à chaque

instance fournie en entrée. Ce score permet de classer les différentes instances. Ainsi des problèmes concrêts étudiés prennent en compte des classements basés sur des ensembles de deux classes (k=2) appelés **problèmes de classement bipartites** et des classements basés sur des ensembles de k classes appelés de **problèmes de classement k-partite ou multipartite**. Dans le cas des classements bipartites, en prennant deux instances x_1 et x_2 , avec le score donné par la fonction, on a que :

$$f(x_1) > f(x_2) \Rightarrow x_1 \text{ a le label supérieur et } x_2 \text{ le label inférieur}$$

Du coté du classement multipartite de même on a en prennant deux instances x_1 et x_2 , avec le score donné par la fonction, on a que :

$$f(x_1) > f(x_2) \Rightarrow x_1 \text{ a un label supérieur à celui de } x_2$$

Mais dans ce cas ci il est à prendre en compte l'écart entre les niveau des labels lors de l'évaluation de la précision de la méthode d'apprentissage.

Soit :

- Un ensemble d'instances d'apprentissage $\{x_l | l = 1, 2, ..., n\} \subseteq \mathcal{X}$ (chaque instance peut ou pas être représentée sous forme d'un vecteur).
- Un ensemble de labels $\mathcal{Y} = \{y_i | i = 1, 2, ..., k\}$ munis d'un ordre $y_1 < y_2 < ... < y_k$
- Pour toute instance d'entraînement x_l, un label associé y_l

Trouvons :

Une fonction de classement qui permet de classé un nouvel ensemble d'instances $\{x_j\}_{j=1}^{t}$ suivant leur degré de préférence

---

2.1.3 Classement d'objets

L'approche de Cohenref : article de Cohen :Learning to Order Things

L'approche de Cohen est caracérisée par la définition des **fonctions de préférence** PREF qui sont des fonctions définies sur $X \times X \rightarrow [0, 1]$ indiquant pour tout couple $(u, v) \in X \times X$, avec quel taux de recommandation l'un doit être préféré à l'autre. Ainsi si $PREF(u, v)$ tend vers 1, alors il y'a une forte recommandation que u soit préféré à v et si $PREF(u, v)$ tend vers 0, alors il y'a une forte recommandation que v soit préféré à u. Une valeur proche

de 1/2 est interprétée comme une abstenion de faire une recommandation.

Cette fonction de préférence est obtenue par la combinaison pondérée de N **fonctions de préférence primitive** $R_1, ..., R_N$. Elle est écrite alors de la manière suivante :

$$PREF(u, v) = \sum_{i=1}^{N} w_i R_i(u, v)$$

Chaque fonction de préférence primitive $R_i(u, v)$ est déjà disponible et prend les valeurs 1 si u est préféré à v, 0 si v est préféré à u et 1/2 autrement. Afin de déterminer les poids w_i alloué à chaque fonction de préférence primitive R_i, plusieurs tours sont effectués en fournissant à chaque tour t un ensemble d'instance X^t à classer. Après le classement de ces instances par la fonction de préférence de base, un feedback F^t lui est fourni en contenant des paires (u, v) indiquant quel élément u devrait être préféré à quel élément v dans le classement. Avecce feedback on peut définir la **perte** de la fonction de préférence R_i définie par :

$$Loss(R, F) = \frac{\sum_{(u,v)\in F}(1 - R(u, v))}{|F|} = 1 - \frac{1}{|F|}\sum_{(u,v)\in F} R(u, v)$$

Initialement $w_i^1 = 1/N$ mais par la suite, les valeurs de w_i^t par les itérations suivantes sont obtenus par la formule suivante :

$$w_i^(t + 1) = \frac{w_i^t . \beta^{Loss(R_i^t, F^t)}}{Z^t}$$

où $\beta \in [0, 1]$ est un paramètre et Z_t est une **constante de normalisation**.

Suite à l'obtention de la fonction de préférence finale fournissant ainsi un score à chaque couple d'un l'ensemble $X \times X$ fourni en entrée, le but est de fournir un ordre total de ces instances. Le processus de classement effectué à partir de l'évaluation des scores des couples (u, v) fourni par $PREF$ est NP-complet,. D'où la necessité d'utiliser un algorithme optimisant ce classement. Le but est d'avoir l'ordre O maximisant

$$\sum_{(u,v)\in O} PREF(u, v)$$

Ainsi il a été proposé un algorithme glouton décrit ci dessous :

Algorithme 1 : Algorithme glouton de classement (Cohen)

$O \leftarrow \emptyset$;
forall $x \in X_u$ **do**
  $score(x) \leftarrow \sum_{x' \in X_u} PREF(x, x')$;
while $X_u \neq \emptyset$ **do**
  $x_{top} \leftarrow argmax_x score(x)$;
  $O_u \leftarrow O_u \succ x_{top}$;
  $X_u \leftarrow X_u - x_{top}$;
  forall $x \in X_u$ **do**
    $score(x) \leftarrow score(x) - PREF(x, x_{top})$;
return O_u;

Algorithme RankBoost

C'est un algorithme proposé par Freund et al ref : livre preference learning qui est un dérivé de Adaboost, Algorithme de boosting connu dans le domaine de l'apprentissage. Sa spécificité est qu'il se base sur des ordre au lieu de se baser sur des scores dans le cas de Adaboost. Les données d'entrée de Rankboost sont des **fonctions de retour** $\phi(x_a, x_b)$ qui impliquent que $x_b \succ x_a$ si $\phi(x_a, x_b) > 0$. Ces fonctions de retour sont issu du processus d'appréciation fournie en retour de l'information de classement fournie par l'algorithme. De même en entré, il y'a un ensemble de fonctions appellées **ranking features** $f(x_i)$ qui fournie une information partielle sur l'ordre dans l'ensemble X des instances. Avec ces entrées, Rankboost retourne le classement final $H(x_i)$ qui fonctionne comme une fonction de score. Initialement, une distributino est calculée par $D_1(x_a, x_b) = max(\phi(x_a, x_b), 0)/Z_1$ pour tout $(x_a, x_b) \in X \times X$ où Z_1 est un coefficient de normalisation qui assure que $\sum_{x_a, x_b} D_1(x_a, x_b) = 1$ Ainsi pour chaque tour $t = 1, ..., T$, l'algorithme répète un processus de sélection du poids α_t et de la fonction d'apprentissage de base $h_t(x)$ de manière à minimiser la valeur de Z_t. Il existe plusieurs méthodes pour cette minimisation que nous n'allons pas expliciter (conf ...). A l'itération suivante on met à jour la distribution :

$$D_{t+1}(x_a, x_b) = \frac{1}{Z_t} \exp^{(} \alpha_t(h_t(X_a) - h_t(x_b))$$

Les fonctions d'apprentissage de base h_t sont caractérisées par le fait que $h_t(x_b) > h_t(x_a) \Rightarrow$

$x_b \succ x_a$. Après les T itérations, nous obtenos une fonction de score :

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Ainsi cette foncion permet de classer toutes les instances de X tel que :

$$\forall x_a, x_b \in X \times X, H(x_a) > H(x_b) \Rightarrow x_a \succ x_b$$

Classement à base de SVM

---

3 Définition et extraction des règles de préférence contextuelle

Soit $\mathcal{R}(R_1, R_2, ..., R_n)$ un schémas relationnel tel que pour chaque attribut R_i on note son domaine de valeurs par $Dom(R_i)$. Ainsi notons $Dom(\mathcal{R}) = Dom(R_1) \times Dom(R_2) \times ... \times Dom(R_n)$. On appelle une **transaction**, tout élément de $Dom(\mathcal{R})$. Soit $\mathbf{I} = \bigcup_1^n Dom(R_i)$, tout élément i tel que $i \in \mathbf{I}$, est appelé **item**. En prenant une transaction $T = i_1, i_2, ..., i_n$, tout éléménent I tel que $I \in T$ est appelé **itemset**. Une **base de transactions** est un ensemble de transactions, chacun associé à un identifiant unique. Une **paire de transactions** $\langle T_1, T_2 \rangle$ est un vecteur de transactions tel que $\langle T_1, T_2 \rangle \neq \langle T_2, T_1 \rangle$

Example 7

Soit un schémas relationnel $\mathcal{R}(Genre, Acteur, Annee)$. Leur domaines de valeurs peuvent être les suivants : $Dom(Genre) = \{Action, Aventure, Guerre, Comedie\}$, $Dom(Acteur) = \{PierceBrosman, SylvesterStalone, HarrisonFord\}$ et $Dom(Annee) = [1900; 2020]$. $Dom(A) = Dom(Genre) \times Dom(Acteur) \times Dom(Annee)$ Ainsi $T = \{Action, PierceBrosman, 2010\}$ est une transaction de $Dom(A)$. $T' = \{Action, 2010\} \in T$ est alors un itemset.

3.1 Préférences utilisateur

Definition 14. *(Préférence utilisateur)*
*Une **préférence utilisateur** est une paire de transactions $\langle T_1, T_2 \rangle$ qui spécifie que l'utili-*

sateur préfère T_1 à T_2. Elle est aussi écrite sous la forme $T_1 \succ T_2$.

Definition 15. *(Base de préférence)*
Soit $\mathcal{D}$ un ensemble de transactions. Une ***base de préférence*** *$\mathcal{P} = \mathcal{D} \times \mathcal{D}$ d'un utilisateur est un ensemble de préférences utilisateur.*

Definition 16. *(Règle de préférence contextuelle)*
Soit des itemsets C, X, Y. Une ***règle de préférence contextuelle*** *est une représentation de la forme $C \rightarrow X \succ Y$ signifiant que lorsque l'itemset C est observé, l'utilisateur préfère l'itemset X à l'itemset Y.*
C est appellé itemset contextuel, X est appellé itemset préféré et Y itemset non préféré

Example 8

La règle de préférence contextuelle $\{Action\} \rightarrow \{1990\} \succ \{2000\}$ signifie que dans le cas de films d'action, l'utilisateur préfère les films de 1990 aux films de 2000.
La règle de préférence contextuelle $\{MemeGenre\} \rightarrow \{Avant\ 2000\} \succ \{Apres\ 2000\}$ signifie que dans le cas où deux films sont de même genre, l'utilisateur préfère les films produits au delà de l'année 2000.

Definition 17. *Deux transactions sont* ***comparables*** *suivant la règle $C \rightarrow X \succ Y$ si les deux contiennent C et si une des deux contient X et l'autre Y.*

Definition 18. *On dit qu'une préférence $\langle T_1, T_2 \rangle$* ***supporte*** *une règle de préférence contextuelle $\mathcal{R} = C \rightarrow X \succ Y$ (noté $R \vdash^+ \langle T_1, T_2 \rangle$) si $C \in T_1 \cap T_2$, $X \in T_1 \backslash T_1 \cap T_2$ et $Y \in T_2 \backslash T_1 \cap T_2$.*

Definition 19. *On dit qu'une préférence $\langle T_1, T_2 \rangle$* ***contredit*** *une règle de préférence contextuelle $\mathcal{R} = C \rightarrow X \succ Y$ (noté $R \vdash^- \langle T_1, T_2 \rangle$) si $C \in T_1 \cap T_2$, $Y \in T_1 \backslash T_1 \cap T_2$ et $X \in T_2 \backslash T_1 \cap T_2$.*

Example 9

Avec $T_1 = \{Action, Pierce\ Brosman, 1990\}$ et $T_2 = \{Action, Sylvester\ Stalone, 2000\}$, on constate que $< T_1, T_2 >$ supporte la règle de préférence $\{Action\} \rightarrow \{1990\} \succ \{2000\}$.

De même on dit qu'une transaction T_1 est **préférée** à une transaction T_2 suivant une règle contextuelle $\mathcal{R} = C \rightarrow X \succ Y$ et noté $T_1 \succ_{\mathcal{R}} T_2$ si si $C \in T_1 \cap T_2$ (resp T_1 et T_2 vérifient C) et si $Y \in T_1 \backslash T_1 \cap T_2$ (resp seul T_1 vérifie Y) et $X \in T_2 \backslash T_1 \cap T_2$ (resp seul T_2 vérifie X).

Soit une règle de préférence r et une base de données de préférences $\mathcal{P}$.
Nous définissons par :

— $agree(r, \mathcal{P})$ l'ensemble des préférences qui supportent r ;

$$agree(r, \mathcal{P}) = \{\langle T_i, T_j \rangle \in \mathcal{P} / R \vdash^{+} \langle T_i, T_j \rangle\}$$

— $contradict(r, \mathcal{P})$ l'ensemble des préférences qui contredisent r ;

$$contradict(r, \mathcal{P}) = \{\langle T_i, T_j \rangle \in \mathcal{P} / R \vdash^{-} \langle T_i, T_j \rangle\}$$

— $cover(i, \mathcal{P})$ l'ensemble des préférences qui supportent et contredisent r

$$cover(r, \mathcal{P}) = agree(r, \mathcal{P}) \cup contradict(r, \mathcal{P})$$

Definition 20. *Le **support** d'une règle de préférence dans une base de préférence $\mathcal{P}$ est le rapport entre le nombre de préférences qui supportent la règle de préférence et le nombre total des préférences de la base de préférence :*

$$supp(r) = \frac{agree(r, \mathcal{P})}{|\mathcal{P}|}$$

Definition 21. *La **confiance** d'une règle de préférence quand à elle est le taux de transactions vérifiant la règle par rapport à la somme du nombre de transactions vérifiant et du nombre de transactions contredisant la règle.*

$$conf(r) = \frac{|agree(r, \mathcal{P})|}{|cover(r, \mathcal{P})|}$$

Definition 22. *(Règle de préférence fréquente) Une règle de préférence r est dite fréquente par rapport à un seuil σ si $supp(r) \geq \sigma$*

Definition 23. *(Règle de préférence interressante) Une règle de préférence est dite interressante pour un seuil de support σ et un seuil de confiance δ si elle est fréquente par rapport au seuil σ et si elle a une confiance $conf(r) \geq \delta$.*

Definition 24. *(Règle de préférence minimale) Une règle de préférence contextuelle $r = C \rightarrow X \succ Y$ est **minimale** par rapport à une base de préférences utilisateurs $\mathcal{P}$ si et seulement s'il n'existe aucune règle $r' = C' \rightarrow X' \succ Y'$ avec $r \neq r'$, tel que $C' \subseteq C$, $X' \subseteq X$ et $Y' \subseteq Y$ avec $supp_{\mathcal{P}}(r) = supp_{\mathcal{P}}(r')$ $conf_{\mathcal{P}}(r) = conf_{\mathcal{P}}(r')$.*

Definition 25. *(Modèle de préférences) Un **modèle de préférence** $\mathcal{M}_{\mathcal{P}}$ sur une base de préférence P est un ensemble trié et ordonné de règles de préférences contextuelles interressantes minimales.*

Dans le cadre d'un utilisateur, le modèle de préférence de la base de préférence de cet utilisateur est appelé **profile de préférences** de cet utilisateur. Ce profil de préférence est construit de façon à ce qu'il soit **précis** et **concis** par rapport à la base de préférences utilisateur. Le critère de concision est évalué par la cardinalité du profil de préférence (recherche de profils à faible cardinalité). Le critère de précision quand à lui est évalué par une **fonction de coût** que nous allons définir dans la suite de ce document.

On peut étendre les notations agree,contradict et cover à un ensemble de règles de préférences de la manière suivante :

- $agree(\Pi, \mathcal{P}) = \cup_{\pi \in \Pi} agree(\pi, \mathcal{P})$;
- $contradict(\Pi, \mathcal{P}) = \cup_{\pi \in \Pi} contradict(\Pi, \mathcal{P})$;
- $cover(\Pi, \mathcal{P}) = \cup_{\pi \in \Pi} cover(\Pi, \mathcal{P})$

Definition 26. *(fonction coût) Etant donné une base de préférences $\mathcal{P}$ et un ensemble de règles de préférences contextuelles Π, le coût de l'ensemble Π par rapport à $\mathcal{P}$ noté $Cost(\Pi, \mathcal{P})$ est défini par :*

$$cost(\Pi, \mathcal{P}) = \frac{|\mathcal{P} \setminus cover(\Pi, \mathcal{P})| + |contradict(\Pi, \mathcal{P})|}{|\mathcal{P}|}$$

La précision d'un profil de préférence est caractérisée par la minimisation de la fonction coût définie ci dessus.

4 Utilisation des motifs séquentiels dans le cadre des préférences contextuelles (approche Sprex)

Les travaux antérieurs se sont penchés sur des méthodes d'extraction de connaissance afin de les exploiter pour extraire les profils de préférence. C'est dans cette logique que le travail effectué au niveau de (Giacometti et al.) a permis de proposer un mode d'extraction de profil utilisateur inspiré de l'extraction des motifs séquentiels.Ils ont nommé cette approche **Sprex**(Sequence-pattern based preference rule extraction) Il a été ainsi adopté de nouvelles représentation des préférences utilisateurs inspirées de la erprésentation des motifs séquentiels. Nous allons dans les sous sections suivantes décrire cette nouvelle représentation et les différentes étapes de Sprex qui permettent de construire les profils de préférence (à l'aide de son module nommé **Sprex-Build**) et prédire les préférences (à l'aide de son module nommé **Sprex-Predict**).

4.1 Représentation séquentielle des préférences

Soit I l'ensemble de tous les items. Un item de préférence est une paire $L : i$ où $L \in \{C, P, N\}$ est un label (avec $C \equiv$ Contexte, $P \equiv$ Préféré, $N \equiv$ Non-préféré) et i un item de $\mathcal{I}$. Soient deux transactions T_1 et T_2 telles que $T_1 \succ T_2$. Dans leur cas, on défini par :

- **Itemset contextuel** l'ensemmble d'items de préférence $C = \{C : c_1, C : c_2, ..., C : c_k\}$ tel que $\{c_1, c_2, ..., c_k\} = T_1 \cap T_2$
- **Itemset préféré** l'ensemmble d'items de préférence $P = \{P : x_1, P : x_2, ..., P : x_k\}$ tel que $\{x_1, x_2, ..., x_k\} = T_1 \setminus T_2$
- **Itemset non préféré** l'ensemmble d'items de préférence $N = \{N : y_1, N : y_2, ..., N : y_k\}$ tel que $\{y_1, y_2, ..., y_k\} = T_2 \setminus T_1$

Ces types d'itemsets sont appelés **itemsets de préférence**. La **base de séquences de préférence** $\mathcal{P}$ correspondant à une **base de préférence** est l'ensemble des séquences de préférence qui correspondent à une préférence unique de la base de préférence. Dans ce cas on calcule le support *supp* d'une séquence de préférence s en comptant le nombre de

séquences de préférence s' de la base de séquence de préférence qui contiennent s.

$$supp(s) = \{s' \in \mathcal{P} / s \subset s'\}$$

On appelle **séquence de préférence** la séquence $\langle CPN \rangle$ qui est une liste ordonnée des itemsets C,P,N représentant la transaction de préférence $\langle T_1, T_2 \rangle$ avec $T_1 = C \cup P$ et $T_2 = C \cup N$.
Une règle de préférence contextuelle $C \rightarrow X \succ Y$ où $C = \{c_1, c_2, ..., c_k\}$, $X = \{x_1, x_2, ..., x_3\}$, $Y = \{y_1, y_2, ..., y_3\}$, peut être écrite sous forme de séquence de préférence $\langle C_C X_P Y_N \rangle$.

Soit la paire de transactions $\langle T_1, T_2 \rangle$ dont la séquence de préférence correspondante est $\langle I_C I_P I_N \rangle$. Ainsi :

$$C \rightarrow X \succ Y \; supporte \, \langle T_1, T_2 \rangle \Leftrightarrow \langle C_C X_P Y_N \rangle \subseteq \langle I_C I_P I_N \rangle$$

$$C \rightarrow X \succ Y \; contredit \, \langle T_1, T_2 \rangle \Leftrightarrow \langle C_C X_P Y_N \rangle \subseteq \langle I_C I_N I_P \rangle$$

Ces correspondances nous permettent de déduire que le support d'une règle de préférence $C \rightarrow X \succ Y$ est égal au support de la séquence de préférence $\langle C_C X_P Y_N \rangle$ qui lui correspond. De même le nombre de paires de préférence qui contredisent cette règle correspond au nombre de séquences de préférences qui supportent $\langle C_C Y_P X_N \rangle$.

4.2 Description de la construction du profil de préférence par Sprex-Build

Sprex-Build est la composante de Sprex dont le rôle est de permettre la construction d'un profil utilisateur (modèle de préférence) $\mathcal{M}_\mathcal{P}$ à partir d'une base de préférences utilisateur $\mathcal{P}$ en prennant en commpte le seuil minimal de support σ, le seuil minimal de confiance δ et en utilisant une fonction de modélisation π (choisie par l'utilisateur). Sprex build génère d'abord une base de séquence de préférences $\mathcal{P}_S$ à partir de la base des préférences de l'utilsateur $\mathcal{P}$ suivants les correspondances entre séquences de préférence et préférence qu'on a vu précédamment. Suite à cela, l'ensemble des séquences fréquentes de préférences $\mathcal{F}$ est extrait de $\mathcal{P}_S$ en sélectionnnant toutes les séquences fréquentes dont le support *supp* est au dessus du seuil σ en utilisant n'imorte quel algorithme d'extraction de motifs séquentiels.(dans le cas de Sprex c'est lapproche **Patterngrowth** qui est utilisée). Sprex-build calcule ensuite pour chaque séquence fréquente de préférence sa confiance $conf_{\mathcal{P}_S}$ et l'ajoute au modèle $\mathcal{M}$ à condition que $conf_{\mathcal{P}_S} \geq \delta$. Ensuite une fonction de

modélisation est utilisée pour construire le modèle de préférence final.
Voici c dessous ces étapes résumées sous forme d'algorithme.

Algorithme 2 : Sprex-Build

Entrées : l'ensemble des préférences de l'utilisateur $\mathcal{P}$, un seuil minimal de support δ, un seuil minimal de confiance σ et une fonction de modélisation π

Sorties : Modèle de règles de prférences $\mathcal{M}_{\mathcal{P}}$

début

$\mathcal{P}_S = \emptyset$;
pour chaque $\langle T, U \rangle \in \mathcal{P}$ **faire**
$C_c = \lambda_C(T \cap U), P_p = \lambda_P(T \setminus T \cap U), N_N = \lambda_N(U \setminus T \cap U)$;
$\mathcal{P}_S = \mathcal{P}_S \cup \langle C_C P_P N_N \rangle$;
$\mathcal{F} = FrequentSequenceMining(\mathcal{P}_S, \sigma)$;
$\mathcal{M} = \emptyset$;
pour chaque $s \in \mathcal{F}$ **faire**
si $conf_{\mathcal{P}_S} \geq \delta$ **alors**
$\mathcal{M} = \mathcal{M} \cup s$;
$\mathcal{M}_{\mathcal{P}} = \pi(sort(\mathcal{M}))$;
Retourner $\mathcal{M}_{\mathcal{P}}$;

Nous avions décrit antérieurement ce que c'est qu'un profil utilisateur et qu'il se doit de vérifier les critères de précision et de concision. Pour vérifier ces critères, Sprex le construit de la manière suivante :

— **A compléter**

4.3 Description de l'approche utilisée pour la prédiction des prafarences utiliateurs (Sprex-Predict)

Sprex-Predict est le module de Sprex permettant d'effectuer des prédictions entre deux préférences. Nous le décrivons dans la suite du document.

Etant donné un modèle de préférences $\mathcal{M}$, une **fonction de préférences** ρ retourne un score c entre 0 et 1 qui prédit entre les transactions T et U, celle que l'utilisateur préfère en s'inspirant du modèle $\mathcal{M}$.

— Si $c > 0.5$, il est prédit que l'utilisateur va préférer T à U ;

— Si $c < 0.5$, il est prédit que l'utilisateur va préférer U à T ;
— Autrement Sprex-Predict retourne qu'il y'a une indécision au niveau du choix du plus préféré.

La fonction de préférence utilise des règles du modèle afin de déterminer quelle est la transaaction la plus préférée. Une approche simple est d'utiliser la **meilleure règle** r_{best} du modèle qui permet de comparer T à U. Si T est préféré à U alors r_{best} retourne une valeur $conf_{\mathcal{P}}(R_{best})$ (>0.5), si U est préféré à T, r_{best} retournera $1 - conf_{\mathcal{P}}(r_b)$ (<0.5), autrement dans le cas de l'indécision ($T \sim U$), r_{best} va retourner la valeur 0.5.
Malheureusement cette fonction mêne souvent à l'indécision suivant une étude de (de Amo et al. 2012) d'où Sprex-Predict utilise une fonction de préférence basée sur le **vote par valeur**. Il se décrit de la sorte : c'est sous forme d'une élection où les transactions d'un ensemble $\mathcal{E}$ votent pour les candidats U ou T suivant r_{best} de telle façon que chacun accorde à chaque candidat la valeur retournée par r_{best}. Toutes ces valeurs sont sommées pour chaque candidat et celui ayant le plus haut score est le gagnant.

$$\rho_{vote} = \begin{cases} 1, si \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle T, V \rangle) > \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle U, V \rangle) \\ 0, si \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle T, V \rangle) < \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle U, V \rangle) \\ 0.5, si \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle T, V \rangle) = \sum_{V \in \mathcal{E}} \rho_{best}(\mathcal{M}, \langle U, V \rangle) \end{cases}$$

5 Extraction des préférences contextuelles dans notre approche

Le rôle de Sprex a consisté à exploiter l'efficacité des méthodes d'extraction des motifs séquentiels afin de l'appliquer à l'extraction des profils de préférence utilisateur. De même contrairement aux précédents travaux qui permettaient de comparer des items en fonction d'un contexte (repésentation des règles limitées à la forme $C \rightarrow i_1 \succ i_2$ avec i_1, i_2 des items et C un itemset), Sprex a permit d'améliorer l'approche en permettant d'étendre la comparaison à des itemsets(ensemble d'items) en fonction d'un contexte (représentation des règles sous la forme $C \rightarrow X \succ Y$ avec X,Y et C tous des itemsets). Cette extension des possibilités de comparaison, a permit l'enrichissement de l'expressivité des règles de préférences extraites et ainsi du profil utilisateur.

Suite à cela, différentes approches ont étés étudiées afin d'enrichir encore plus cette expressivité des règles. C'est dans cette logique que s'est déroulé notre travail. Il a consisté

à rehercher une formulation améliorée des règles de préférences afin de contribuer à l'amélioration de cette expressivité. Au cours des travaux, l'approche que nous avons adopté s'est inspiré d'un travail de l'article (Chomicky et al..... à compléter) qui exploitait l'utilisation de prédicats logiques afin de représenter les préférences.

5.1 Formules de préférence suivant les travaux de ([?])

Une partie du travail de [**?**] a porté sur l'étude de la représentation des préférences utilisateur en s'inspirant des formules de prédicat du premier ordre. Dans leur travail le terme **relations de préférences** est utilisé similairement au terme **règle de préférence** de l'approche Sprex. Nous allons nous contenter de garder le terme règle de préférence. Nous allons décrire par la suite la formulation qu'ils ont proposé.

Definition 27. *(formule de préférence) Une* ***formule de préférence*** $C(t_1, t_1)$ *est une* ***formule de premier ordre*** *(formule de prédicats) qui est utilisée pour définir une règle de préférence (relation de préférence)* $\succ_C$ *de telle sorte que* $t_1 \succ_C t_2 \equiv C(t_1, t_2)$*, c'est à dire la transaction* t_1 *est préférée à la transaction* t_2 *suivant la préférence* $\succ_C$ *si et seullement si elles vérifient la formule de prédicat* $C(t_1, t_2)$*.*

L'utilisation des formules de préférence nécessite de préciser le type d'attribut auquel les items (des transactions) appartiennent.
Ces formules de préférence sont écrites sous une **forme normale disjonctive**(DNF) excepté le fait que les quantificateurs ne sont pas utilisés contrairement aux formules des prédicats (dans la logique des prédicats). Ainsi elles se présentent comme suit :

$$C(t_1, t_2) = \bigvee_{i=1..k} \left(\bigwedge_{j=1..l} f_{ij} \right)$$

où f_{ij} $\forall i = 1..k,\ j = 1..l$ sont des **formules atomiques** (caractérisées par des contraintes de comparaison$=, \neq, <, >, \leq, \geq$) sous la forme $x\delta y$ ou $x\delta c$ avec x et y des variables (d'un type d'attribut donné) qui correspondent respectivement à la première transaction t_1 et à la deuxième transaction t_2. c quand à lui est une constante.
δ prend les valeurs

- $=$,$\neq$ dans le cas où les éléments x, y, c sont de type d'attribut quelconque (Exemple : $x = y$,$x = a$,$y \neq a$),

— $\leqslant, \geqslant, <, >$ dans le cas où on a à faire à des types d'attributs numériques (Exemple : $x < y$,$x < a$,$y \geqslant a$).

Example 10

— Soit deux transactions $t_1 = \{x_1, x_2, x_3\}$ et $t_2 = \{y_1, y_2, y_3\}$ d'une instance de la relation R(Plat,TypePlat, TypeVin). On a qu'ils vérifient la formule de préférence C si et seulement si

$$\begin{aligned} C(t_1, t_2) \equiv\ & (x_1 = y_1 \wedge x_2 = "fish" \wedge y_2 = "fish" \wedge x_3 = "white" \wedge y_3 = "red") \\ & \vee (x_1 = y_1 \wedge x_2 = "meat" \wedge y_2 = "meat" \wedge x_3 = "red" \wedge y_3 = "white") \end{aligned}$$

est vérifié. Cette condition veut dire que dans le cas où c'est un plat à base de poisson, le client préfère le vin blanc au vin rouge et dans le cas où c'est un plat à base de viande, le client préfère le vin rouge au vin blanc.

— En prenant en compte les valeurs numériques, soit deux transactions $t_1 = \{x_1, x_2, x_3\}$ et $t_2 = \{y_1, y_2, y_3\}$ d'une instance de la relation R(TypeFilm,Annee, Acteur). On a $t_1 \succ_C t_2$ ssi :

$$\begin{aligned} C(t_1, t_2) \equiv\ & (x_1 = y_1 \wedge x_2 > 2010 \wedge y_2 < 2000) \\ & \vee (x_1 = y_1 \wedge x_2 = y_2 \wedge x_3 = \text{"Sylvester Stalone"} \wedge y_3 = \text{"Pierce Brosnan"}) \end{aligned}$$

Cette règle veux tout simplement dire que le client préfère pour des films de même type, des films au delà de 2010 aux films produits avant 2000, et si les films sont de même type et de meême anné, il préfère ceux avec l'acteur Sylvester Stalone à ceux avec l'acteur Pierce Brosnan.

5.2 Formules de préférence suivant notre approche

Dans notre travail, nous allons nous inspirer de la représentation en formules de préférences décrite précédamment pour l'appliquer au cas des règles de préférence contextuelles. En effet nous voulons définir les formules de préférence de telle sorte qu'elles puissent représenter nos règles de préférence contextuelles. Dans la section précédante les formules de préférences ont été définies comme suit :

$$C(t_1, t_2) = \bigvee_{i=1..k} \left(\bigwedge_{j=1..l} f_{ij} \right)$$

Dans notre contexte, les formules de préférence vont seulement utiliser les connecteurs logiques $\wedge$. C'est à dire elles seront sous la forme :

$$P(t_1, t_2) = \bigwedge_{k \in E} P_{A_k}(t_1, t_2) \tag{1}$$

avec $E \subset \{1, 2, .., n\}$ où n est le nombre des différents types d'attributs A_k qui définissent les items des transactions t_1, t_2 et $P_{A_k}(t_1, t_2), k \in E$ est une conjonction de formules atomiques définissant les items pour chaque attribut $A_k, k \in E$ comme

Example 11

Pour deux transactions $t_1 = \{x_1, x_2\}$ et $t_2 = \{y_1, y_2\}$, on a que $t_1 \succ_P t_2$ ssi $P(t_1, t_2) = \{x_1 = y_1\} \wedge \{x_2 > 4\} \wedge \{y_2 < 3\}$ est vrai.

Après avoir définit les formules de préférence élémentaires, nous allons définir les formules de préférence.

Definition 28. *Une **formule de préférence élémentaire** est une conjonction de formules atomiques (définies dans [?]) construite à partir de la comparaison de deux items x,y de même type d'attribut.*

Example 12

Si on a deux items x et y de type année de valeurs 2000 et 2002, alors par comparaison, on obtient les formules de préférence élémentaires $x < y$, $x \neq y$, $x < 2003 \wedge y < 2003$, $x > 1999 \wedge y > 1999$ etc..

Nous fournissont ci dessous les types de formules de préférences élémentaires que nous allons prendre en compte dans notre travail. Soit $\mathcal{F}$ l'ensemble des formules de préférence élémentaires issues de la comparaisons entre deux items de t_1 et de t_2i ayant un même type d'attribut A_k donné.

Les formules de préférences élémentaires qui peuvent être déduites sont :

— Cas où les items de type A_k ont des valeurs numériques.Soit un item x de t_1 de type A_k et y un item de t_2 de type A_k.

  — Si x et y sont tous deux égaux à une valeur a alors les formules de préférences élémentaires extraites sont $x = y$ et $x = y \wedge x = a \wedge y = a$.

- Si x est différent de y alors la formule de préférence élémentaire $x \neq y$ est extraite.
  - Si $x < y$ alors la formule de préférence élémentaire $x < y$ est extraite.
  - Si $x > y$ alors la formule de préférence élémentaire $x > y$ est extraite.
- Soit a appartenant à l'ensemble des valeurs possibles des items de type A_k,
  - $x < a$ et $y < a$ alors la formule de préférence élémentaire $x < a \wedge y < a$ est extraite.
  - $x > a$ et $y > a$ alors la formule de préférence élémentaire $x > a \wedge y > a$ est extraite.
  - $x < a$ et $y \geq a$ alors la formule de préférence élémentaire $x < a \wedge y \geq a$ est extraite
  - $x > a$ et $y \leq a$ alors la formule de préférence élémentaire $x > a \wedge y \leq a$ est extraite.
  - $y > a$ et $x \leq a$ alors la formule de préférence élémentaire $y > a \wedge y \leq a$ est extraite.
  - $y < a$ et $x \geq a$ alors la formule de préférence élémentaire $y < a \wedge y \geq a$ est extraite.
  - $x = a$ et $y \neq a$ alors la formule de préférence élémentaire $x = a \wedge y \neq a$ est extraite.
  - $y = a$ et $y \neq a$ alors la formule de préférence élémentaire $y = a \wedge x \neq a$ est extraite.
- Cas où pour un type d'attribut donné, les items prennent des valeurs symboliques.

  - Si x et y sont tous deux égaux à une valeur a alors les formules de préférence élémentaires $x = y$ et $x = y \wedge x = a \wedge y = a$ sont extraites ;
  - Si x est différent de y alors la formule de préférence élémentaire $x \neq y$ est extraite.

5.2.1 Séquences de formules de préférence élémentaires

Avant d'aller plus en détail, nous faisons un rappel sur ce que sont les relations binaires symétriques et les relations binaires asymétriques.

Definition 29. *(relation binaire symétrique) Une relation binaire $\mathcal{R}$ est symétrique si :*

$$\forall(x,y) \in E^2, x\mathcal{R}y \Rightarrow y\mathcal{R}x$$

Definition 30. *(relation binaire asymétrique) Une relation binaire $\mathcal{R}$ est dite asymétrique si :*

$$\forall(x,y) \in E^2, x\mathcal{R}y \Rightarrow \neg y\mathcal{R}x$$

Definition 31. *(Formules de préférence élémentaires symétriques) Les formules de préférence élémentaires symétriques $C(t_1,t_2)$ sont caractérisées par la propriété de symétrie c'est à dire $C(t_1,t_2) \Rightarrow C(t_2,t_1)$.*

Example 13

la formule $F(x,y) \equiv (x=y)$ est symétrique car $F(x,y) \equiv (x=y) \Rightarrow (y=x) \equiv F(y,x)$, d'où $F(x,y) \Rightarrow F(y,x)$.

Definition 32. *(Formules de préférence élémentaires asymétriques) Les formules de préférence élémentaires asymétriques $P(t_1,t_2)$ (ou resp $N(t_1,t_2)$) sont caractérisées par la propriété d'asymétrie c'est à dire $P(t_1,t_2) \Rightarrow \neg P(t_2,t_1)$ (ou resp $N(t_1,t_2) \Rightarrow \neg N(t_2,t_1)$).*

Example 14

la formule $F(x,y) \equiv (x>y)$ est asymétrique car $F(x,y) \equiv (x>y) \Rightarrow \neg(y>x \vee y=x) \Rightarrow \neg(y>x) \wedge \neg(x=y) \Rightarrow \neg(y>x) \equiv \neg F(y,x)$, d'où $F(x,y) \Rightarrow \neg F(y,x)$.

Nous nous rappellons que l'approche Sprex a décrit les séquences de préférences comme étant un itemset composé de trois parties à savoir une partie *contextuelle* qui rassemble les items communs à t_1 et t_2 une la partie *préférée* distinguant les items qui sont dans t_1 et pas dans t_2 et une partie *non préférée* distinguant les items qui sont dans t_2 et pas dans t_1.
Dans notre cas où i s'agit de séquences de formules de préférences élémentaires, nous ne parlons pas d'items mais de formules de préférences élémentaires au niveau de la composition des séquences de préférences élémentaires.
Ainsi nous définissons différamment la composition de ces séquences de formules de préférence élémentaires. Elles sont décomposées en deux parties : une partie rassemblant les formules de préférences élémentaires symétriques, appellée contexte, et une partie rassemblant les formules de préférence élémentaires asymétriques, appellées préférence.

Cette approche s'explique par le fait que la partie contexte regroupe les formules vérifiées autant par la paire de transaction $\langle t_1, t_2\rangle$ que $\langle t_2, t_1\rangle$ (ce sont les formules de préférence élémentaires symétriques). La partie préférence permet de spécifier les formules de préférence élémentaires qui permettent de différencier la paire $\langle t_1, t_2\rangle$ de la paire $\langle t_2, t_1\rangle$ (ce sont les formules de préférence élémentaires asymétriques).

Ainsi nous définisons une séquence de formules de préférences élémentaires comme suit :

Definition 33. *(Séquence de formules de préférence élémentaire) Pour une paire de transaction $\langle t_1, t_2\rangle$, la séquence de formules de préférence élémentaires est la suite de préférences élémentaires $\langle C, PN\rangle$ tel que*

- $C \equiv \{F(t_1, t_2)/F(t_1, t_2) \in \mathcal{F}(t_1, t_2)$ *et symétrique*$\}$
- $PN \equiv \{F(t_1, t_2)/F(t_1, t_2) \in \mathcal{F}(t_1, t_2)$ *et asymétrique*$\}$

avec $\mathcal{F}(t_1, t_2)$ qui est l'ensemble des formules de préférence élémentaires extraites de la comparaison entre les items de t_1 et de t_2.

Voici un diagramme répertoriant les formules de préférence élémentaires réparties suivant qu'elles soient symétriques ou asymétriques :

Formules de préférence élémentaires symétriques (Contexte)

$$\begin{cases} C:\{x=y\} & \equiv & x=y \\ C:\{x\neq y\} & \equiv & x\neq y \\ C:\{x,y=a\} & \equiv & x=y\wedge x=a\wedge y=a \\ C:\{x,y<a\} & \equiv & x<a\wedge y<a \\ C:\{x,y>a\} & \equiv & x>a\wedge y>a \end{cases}$$

Formules de préférence élémentaires asymétriques (Préférences)

$$\begin{cases} P:\{x<y\} & \equiv & x<y \\ P:\{x>y\} & \equiv & x>y \\ P:\{x=a\} & \equiv & x=a\wedge x\neq y \\ P:\{x<a\} & \equiv & x<a\wedge y\geqslant a \\ P:\{x>a\} & \equiv & x>a\wedge y\leqslant a \\ N:\{y>a\} & \equiv & y>a\wedge x\leqslant a \\ N:\{y<a\} & \equiv & y<a\wedge x\geqslant a \\ N:\{y=a\} & \equiv & y=a\wedge x\neq y \end{cases}$$

Dans ce diagramme, nous avons à droite des équivalences ($\equiv$), les formules de préférences élémentaires et à gauche nous avons leur notation ($C:\{...\}$, $P:\{...\}$,$N:\{...\}$) qui sera adoptée lors de leur représentation au niveau des séquences de formules de préférence.

Nous pouvons à présent définir les formules de préférence qui vont caractériser les règles de préférence contextuelles qu'on aura à extraire.

Definition 34. *(Formules de préférence)*
Une formule de préférence définissant une règle de préférence suivant est une formule qui s'écrit sous la forme :

$$\mathcal{P}(t_1,t_2)=C(t_1,t_2)\wedge\ PN(t_1,t_2)$$

et qui équivaut à une règle de préférence contextuelle $C\rightarrow PN$ avec

$$C(t_1,t_2) = \bigwedge_{k\in E} C_{A_k}(t_1,t_2)\ \textit{conjonction de formules de préférence élémentaires symétriques}$$
$$PN(t_1,t_2) = \bigwedge_{k\in E} PN_{A_k}(t_1,t_2)\ \textit{conjonction de formules de préférence élémentaires asymétriques}$$

Ici $PN_{A_k}(t_1,t_2)$ est soit une formule de type $P_{A_k}(t_1,t_2)$ ou $N_{A_k}(t_1,t_2)$.

$C(t_1,t_2)$ sera définie comme la **partie contexte** de la formule de préférence contex-

tuelle et $PN(t_1, t_2)$ sera définie comme la **partie préférence** de la formule de préférence contextuelle.

Une formule de préférence $\mathcal{P}(t_1, t_2) = C(t_1, t_2) \wedge \ PN(t_1, t_2)$ peut être écrite sous forme d'une séquence de formule de préférences $\langle \lambda(C), \lambda(PN) \rangle$ avec $\lambda(C)$ est l'ensemble des formules de préférence élémentaires constituant $C(t_1, t_2)$ et $\lambda(PN)$ est l'ensemble des formules de préférence élémentaires constituant $PN(t_1, t_2)$.

Condition de non redondance :
Il est à noter que les P_{A_k} ne doivent pas contenir de redondance i.e si on considère F_k l'ensemble des formules élémentaires constituant P_{A_k} ($P_{A_k} = \bigwedge_{f \in F_k} f$), on a

$$F' \subsetneq F \Rightarrow \neg(\bigwedge_{f \in F'} f \Rightarrow P_{A_k}).$$

Le tableau ci dessous résume les différents cas évoqués en ce qui concerne les formules de préférence élémentaires :

	Valeurs Attr	Type Attr	Items de transaction de préférence
Attr monovalué	$x = a,\ y = a$	num/symb	$C : \{x = y\}$ $C : \{x, y = a\}$
		num	$C : \{x, y < c\},\ \forall c \vert c < a$ $C : \{x, y > c\},\ \forall c \vert c > a$
	$x = a,\ y = b$ avec $a \neq b$	num/symb	$P : \{x = a\}$ $N : \{y = b\}$ $C : \{x \neq y\}$
		num	$P : \{x < y\}$, si $a < b$ $P : \{x > y\}$, si $a > b$ $P : \{x < c\}$, si $a < b$ et $\forall c \vert a < c \leqslant b$ $P : \{x > c\}$, si $a > b$ et $\forall c \vert a > c \geqslant b$ $N : \{y < c\}$, si $a > b$ et $\forall c \vert b < c \leqslant a$ $N : \{y > c\}$, si $a < b$ et $\forall c \vert b > c \geqslant a$ $C : \{x, y < c\},\ \forall c \vert c < a \wedge c < b$ $C : \{x, y > c\},\ \forall c \vert c > a \wedge c > b$
Attr multivalué	$x \in X, y \in Y$	Set(symb)	$C : \{x, y = c\},\ \forall c \in X \cap Y$ $P : \{x = a\},\ \forall a \in X \backslash Y$ $N : \{y = b\},\ \forall b \in Y \backslash X$

5.3 Construction du profil utilisateur

Après avoir défini ce qu'est une séquence de formules de préférences, nous pouvons à présent utiliser cette formulation pour construire le profil utilisateur. En effet nous nous inspirons de Sprex afin de construire ce pofil. Nous utilisons l'algorithme de Sprex-Build tout en remplaçant l'étape qui permet la conversion des paires de transactions en séquences de préférence, par l'étape qui permettra la conversion des paires de transactions en séquence de formules de préférence.

Voici l'algorithme Sprex-Build-Formula de construction du profil obtenu dans ce cas (qui est une modification de Sprex-Build) :

Algorithme 3 : Sprex-Build-Formula

Entrées : l'ensemble des préférences de l'utilisateur $\mathcal{P}$, un seuil minimal de support δ, un seuil minimal de confiance σ et une fonction de modélisation π

Sorties : Modèle de règles de préférences $\mathcal{M}_{\mathcal{P}}$

début

$\mathcal{P}_S = \emptyset$;

pour chaque $\langle T, U \rangle \in \mathcal{P}$ **faire**

$\mathcal{P}_S = \mathcal{P}_S \cup SequenceFormula(\langle T, U \rangle)$

$\mathcal{F} = FrequentSequenceMining(\mathcal{P}_S, \sigma)$;

$\mathcal{M} = \emptyset$;

pour chaque $s \in \mathcal{F}$ **faire**

si $conf_{\mathcal{P}_S} \geq \delta$ **alors**

$\mathcal{M} = \mathcal{M} \cup s$;

$\mathcal{M}_{\mathcal{P}} = \pi(sort(\mathcal{M}))$;

Retourner $\mathcal{M}_{\mathcal{P}}$;

L'algorithme Sprex-Builds-Formula utilise la fonction SequenceFormula() qui permet la transformation d'une paire de transaction $\langle T, U \rangle$ en séquences de formules de préférences. Cette fonction est définie comme suit :

Algorithme 4 : SequenceFormula

Entrées : Paire de transactions $\langle T, U \rangle$

Sorties : Séquence de formules de préférence S

début

$S = \emptyset$;

pour chaque *Type d'attribut* A_k **faire**

pour chaque *paire d'items* $(x, y) \in T \times U$ *de type* A_k **faire**

$S = S \cup CompareFormula(x, y, A_k)$

CompareFormula quand à elle est la fonction qui effectue la comparaison entre deux items respectivement de T et U afin de générer les formules de préférences élémentaires

correspondantes. Elle est définie comme suis :

Algorithme 5 : CompareFormula

Entrées : x , y et l'attribut A_i auquel ils sont liés

Sorties : items issus de la comparaison entre x et y

début

- $\mathcal{E} = \emptyset$; `// Ensemble des items générés`
- **si** $x = y$ **alors**
  - Ajouter $P : \{x_{A_i} = y_{A_i}\}$ à $\mathcal{E}$;
  - Ajouter $P : \{x_{A_i} = y_{A_i} = x\}$ à $\mathcal{E}$;
  - **pour chaque** $c \in DomActif(A_i)$ *et* A_i *numérique* **faire**
    - **si** $x < c$ **alors**
      - Ajouter $C : \{x_{A_i}, y_{A_i} < c\}$ à $\mathcal{E}$;
    - **sinon si** $x > c$ **alors**
      - Ajouter $C : \{x_{A_i}, y_{A_i} > c\}$ à $\mathcal{E}$;
- **sinon si** $x \neq y$ **alors**
  - Ajouter $C : \{x_{A_i} \neq y_{A_i}\}$ à $\mathcal{E}$;
  - Ajouter $P : \{x_{A_i} = x\}$ à $\mathcal{E}$;
  - Ajouter $N : \{y_{A_i} = y\}$ à $\mathcal{E}$;
  - **si** A_i *numérique* **alors**
    - **si** $x > y$ **alors**
      - Ajouter $P : \{x_{A_i} > y_{A_i}\}$ à $\mathcal{E}$;
    - **si** $x < y$ **alors**
      - Ajouter $P : \{x_{A_i} < y_{A_i}\}$ à $\mathcal{E}$;
    - **pour chaque** $c \in DomActif(A_i)$ **faire**
      - **si** $x < c \wedge y \geqslant c$ **alors**
        - Ajouter $P : \{x_{A_i} < c\}$ à $\mathcal{E}$;
      - **sinon si** $x \leqslant c \wedge y > c$ **alors**
        - Ajouter $N : \{y_{A_i} > c\}$ à $\mathcal{E}$;
      - **sinon si** $x > c \wedge y \leqslant c$ **alors**
        - Ajouter $P : \{x_{A_i} > c\}$ à $\mathcal{E}$;
      - **sinon si** $x \geqslant c \wedge y < c$ **alors**
        - Ajouter $N : \{x_{A_i} < c\}$ à $\mathcal{E}$;
      - **sinon si** $x < c \wedge y < c$ **alors**
        - Ajouter $C : \{x_{A_i}, y_{A_i} < c\}$ à $\mathcal{E}$;
      - **sinon si** $x > c \wedge y > c$ **alors**
        - Ajouter $C : \{x_{A_i}, y_{A_i} > c\}$ à $\mathcal{E}$;
- retourner $\mathcal{E}$;

Cette fonction est équivalente à la description qu'on avait faite antérieurement sur les formules de préférence élémentaires qui peuvent re générées par la comparaison de deux items.

Les différentes étapes de la construction de profils de préférence iétant décrites, on peut décrire la phase de prédiction des préférences utilisateur à partir de ce profil obtenu.

5.4 Prédiction des préférences utilisateur

La prédiction des préférences de l'utilisateurs peut être faite à l'aide du profil construit précédamment. Nous voyons que le profil qui est un ensemble de règle de préférences représentées sous forme de séquences de formules de préférence, necessite que le couple de transaction à comparer soit d'abord convertit en séquences de formules de préférences. Ensuite on peut utiliser l'approche Sprex-Predict pour déterminer la préférence entre ce couple de transactions.

Notre travail décrit, nous pouvons passer à la phase d'implémentation afin de permette une évaluation de performance de l'approche adoptée.

5.5 Implémentation du travail

L'implémentation du travail a consisté à mettre en place un programme capable de :

— Construire les profils utilisateurs basés sur les séquences de formules de préférence ;
— Evaluer la qualité de prédiction des profils utilisateurs.

Afin de le réaliser, nous avons exploité deux programmes qui avaient été développé dans le cadre des recherches antérieurs.

— Le premier nommé Sprex a été développé afin d'implémenter l'algorithme de Sprex (Sprex-Build et Sprex-Predict) ;
— Le second est un programme surnommé Colico qui s'est basé sur un algorithme d'extraction de motifs séquentiels minimaux.

5.5.1 Le programme Sprex

Sprex est un programme qui a été développé pour le travail de l'aticle [?]. Les fonctionnalités principales sont :

- L'extraction de toutes les règles de préférence contextuelles ;
- L'extraction des règles de préférence contextuelles interressantes minimales. Ceci est fait en précisant les valeurs seuils du support et de la confiance.
- La construction d'un model de préférences dépendant d'une fonction de modélisation. En effet c'est la fonction de modélisation qui permet de filtrer et trier les règles de préférence contextuelles minimales afin de construire le modèle de préférences ;
- La prédiction les préférences utilisateurs dépendant d'une fonction de préférence. En effet la fonction de préférence permet d'exploiter le modèle de préférence pour comparer un couple de transactions de préférences.
- Le test de la qualité de ces prédictions des préférences utilisateurs qui sont réalisées. Plusieurs indicateurs sont évalués pour déterminer la qualité de prédiction. Exemple : la précision, le Rappel etc..

Initialement dans notre travail, il a été prévu d'utiliser le programme Sprex afin d'extraire les règles de préférence interressantes minimales, de construire ensuite le modèle de préférence à partr de ces règles et d'évaluer la qualité de la prédiction du modèle de préférence obtenu.
Mais nous avons été confrontés à un problème de saturation mémoire lors de la phase d'extraction des règles interressantes minimales à partir des séquences de formules de préférence. Ceci est dû au fait que Sprex a été utilisé pour le traitement des séquences de préférences qui ont une taille plus réduite que les séquences de formules de préférence formalisées dans notre travail, alors que plus la taille des séquence est longue, plus l'extraction de motifs minimaux prend de l'espace lorsqu'on est confronté à un algorithme d'extraction de motifs minimaux qui effectue l'extraction par niveau.
Ainsi nous nous avons choisit autre programme nommé Colico pour réaliser la phase d'extraction de règles de préférences minimales interressantes.

5.5.2 Le programme Colico

Colico est un programme qui a été développé par Arnault Soulet (en c++) afin de répondre au problème de saturation dû à l'extraction de motifs minimaux. Il se base sur les travaux de l'article [?] qui a proposé une méthode d'extraction de motifs minimaux avec l'avantage d'être polynomial-delay et polynomial-space.

Principe de l'approche de Colico Nous allons tout d'abord énoncer quelques fondements sur lequel se base l'algorithme de Colico.

Definition 35. *(Opérateur de couverture) Soit un ensemble d'objets $\mathcal{O}$, un opérateur de couverture $cov : 2^E \to 2^{\mathcal{O}}$ satisfait $cov(X \cup Y) = cov(X) \cap cov(Y)$ pour tout $X, Y \in 2^E$*

Definition 36. *(Opérateur canonique) Etant donné deux systèmes d'ensembles $(\mathcal{F}, E)$ et $(\mathcal{G}, E)$, un opérateur canonique $\phi : \mathcal{F} \cup \mathcal{G} \to \mathcal{F}$ est une fonction satisfaisant : (i) $X \in Y \Rightarrow \phi(X) \in \phi(Y)$ et (ii) $X \in \mathcal{F} \Rightarrow \phi(X) = X$ pour tout les ensembles $X, Y \in \mathcal{G}$.*

Definition 37. *(Système minimisable d'ensembles)*

Un système minimisable d'ensembles est un tuple $\langle(\mathcal{F}, \mathcal{E}, \mathcal{G}, cov, \phi)\rangle$ où :

- *$(\mathcal{F}, E)$ est un système d'ensembles fini et fortement accessible. Un ensemble acceptable dans $\mathcal{F}$ est appelé motif.*
- *$(\mathcal{G}, E)$ est un système d'ensembles fini et fortement accessible satisfaisant pour chaque couple d'ensembles acceptables $X, Y \in \mathcal{F}$ tels que $X \subseteq Y$ et chaque élément $e \in E$, $X \setminus \{e\} \in \mathcal{G}$. Un ensemble acceptable de $\mathcal{G}$ est appellé généralisation.*
- *$cov : 2^E \to 2^{\mathcal{O}}$ est un opérateur de couverture.*
- *$\phi : \mathcal{F} \cup \mathcal{G} \to \mathcal{F}$ est un opérateur canonique tel que pour chaque ensemble accepatble $X \in \mathcal{G}$, on ait $cov(\phi(X)) = cov(X)$.*

Definition 38. *(Motif minimal) Un motif X est minimal pour $\langle(\mathcal{F}, E), \mathcal{G}, cov, \phi\rangle$ ssi $X \in \mathcal{F}$ et pour chaque généralisation $Y \in \mathcal{G}$ telle que $Y \subset X, cov(Y) \neq cov(X)$. $\mathcal{M}(\mathcal{S})$ dénote l'ensemble des motifs minimaux.*

Theorem 5.1. *(Représentation condensée) L'ensemble des motifs minimaux est une représentation condensée de $\mathcal{F}$ adéquate pour le calcul de cov : pour tout motif $W \in \mathcal{F}$, il existe $Y \in X$ tel que $\phi(Y) \in \mathcal{M}(\mathcal{S})$ et $cov(\phi) = cov(X)$.*

Theorem 5.2. *(Systeme indépendant) Si un motif X est minimal pour S, alors tout motif $Y \in \mathcal{F}$ satisfaisant $Y \subseteq X$ est aussi minimal pour S.*

Lemma 5.3. *Si X n'est pas minimal, $\exists e \in X$ tel que $X \setminus \{e\} \in \mathcal{G}$ et $cov(X) = cov(X \setminus \{e\})$.*

Definition 39. *(Objets critiques) Pour un motif X, les objets critiques d'un élément $e \in X$, notés $\widehat{cov}(X, e)$ sont l'ensemble d'objets qui appartiennent à la couverture de X privé de e mais pas à la couverture de e : $\widehat{cov}(X, e) = cov(X \setminus e) \setminus cov(e)$.*

Property 5.3.1. *(Minimalité) $X \in \mathcal{F}$ est minimal si $\forall e \in X$ tel que $X \setminus e \in \mathcal{G}, \widehat{X, e} \neq \emptyset$*

Property 5.3.2. *L'égalité suivante est vraie pour tout motif $X \in \mathcal{F}$ et deux éléments $e, e' \in E$:*

$$\widehat{cov}(Xe', e) = \widehat{cov}(X, e) \cap cov(e').$$

Algorithme DEFME

Algorithme 6 : DEFME

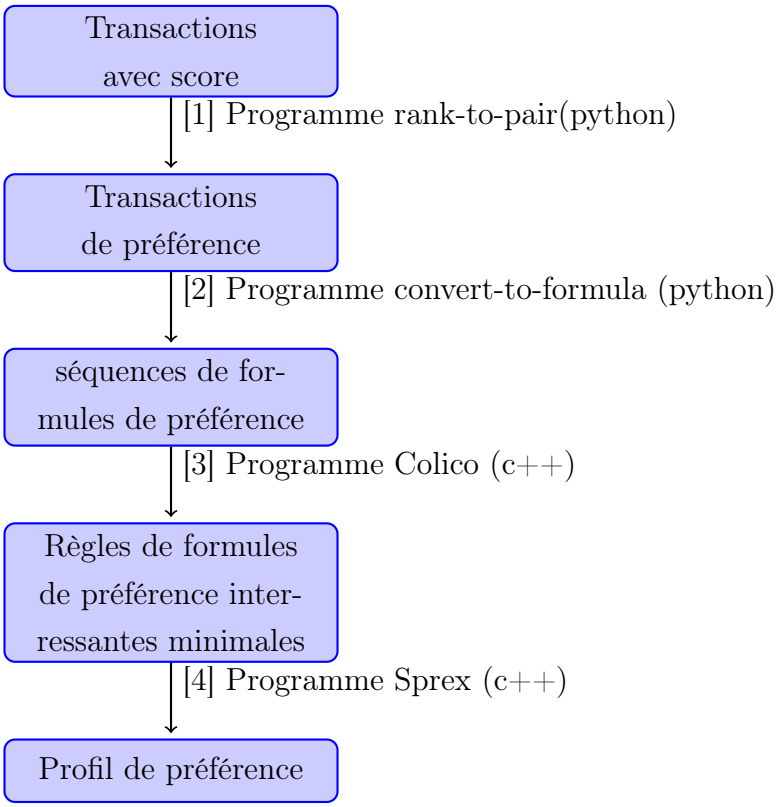
Entrées : X est un motif, tail est l'ensemble des items restant à utiliser pour générer les candidats. Valeurs initiales : $X = \emptyset$, $tail = E$.

Sorties : Calcul des motifs minimaux de manière incrmentale polynomiale

début

- **si** $\forall e \in X, \widehat{cov}(X, e) \neq \emptyset$ **alors**
  - print X;
  - **pour chaque** $e \in tail$ **faire**
    - **si** $Xe \in \mathcal{F}$ **alors**
      - $tail := tail \setminus \{e\}$;
      - $Y := Xe$;
      - $cov(Y) := cov(X) \cap cov(e)$;
      - $\widehat{cov}(Y, e) := cov(X) \setminus cov(e)$;
      - **pour chaque** $e' \in X$ **faire**
        - $\widehat{cov}(Y, e') := \widehat{cov}(X, e') \cap cov(e)$;
      - DEFME$(Y, tail)$;

L'extraction et la prédiction des préférences utilisateur a été réalisé avec des programmes écrits en c++ et python. Ainsi le côté implémentation des traitements s'est déroulé en plusieurs étapes comme le décrit le schémas ci dessous :



Nous disposions initialement d'une base de transactions utilisateur accompagné de leur scores représentant le degré de préférence de l'utilisateur. Nous devions alors convertire ces scores en préférences entre paires de transaction (U, T)i indiquant que la première transaction est préférée à la seconde. Après cette étape, nous avons converti les transactions de préférence en séquences de formules de préférence en conformité avec l'approche que nous avons adopté et suivant les méthodes de conversions que nous avions décrits avec les algorithmms précédents. Ensuite nous avons extrait les règles de formules de préférence interressantes minimales.

Nous avons bénéficié des ressources qui ont été utilisées dans les travaux antérieurs. Ainsi dans le cadre des travaux sur Sprex, un outil nommé Sprex avaait été développé afin de pouvoir Extraire les profils utilisateurs (par l'approche basée sur les séquences de préférence) et évaluer la qualité des préférences prédites à partir de ces profils. Sprex est un programme qui a été développé en C++ et utilise des librairies d'extraction de motifs séquentiel.

Le programme Sprex a été développé autant pour la construction de profils utilisateurs que pour à la prédiction des préférences.
Malheureusement nous avons été confrontés à un problème de saturation mémoire lors de

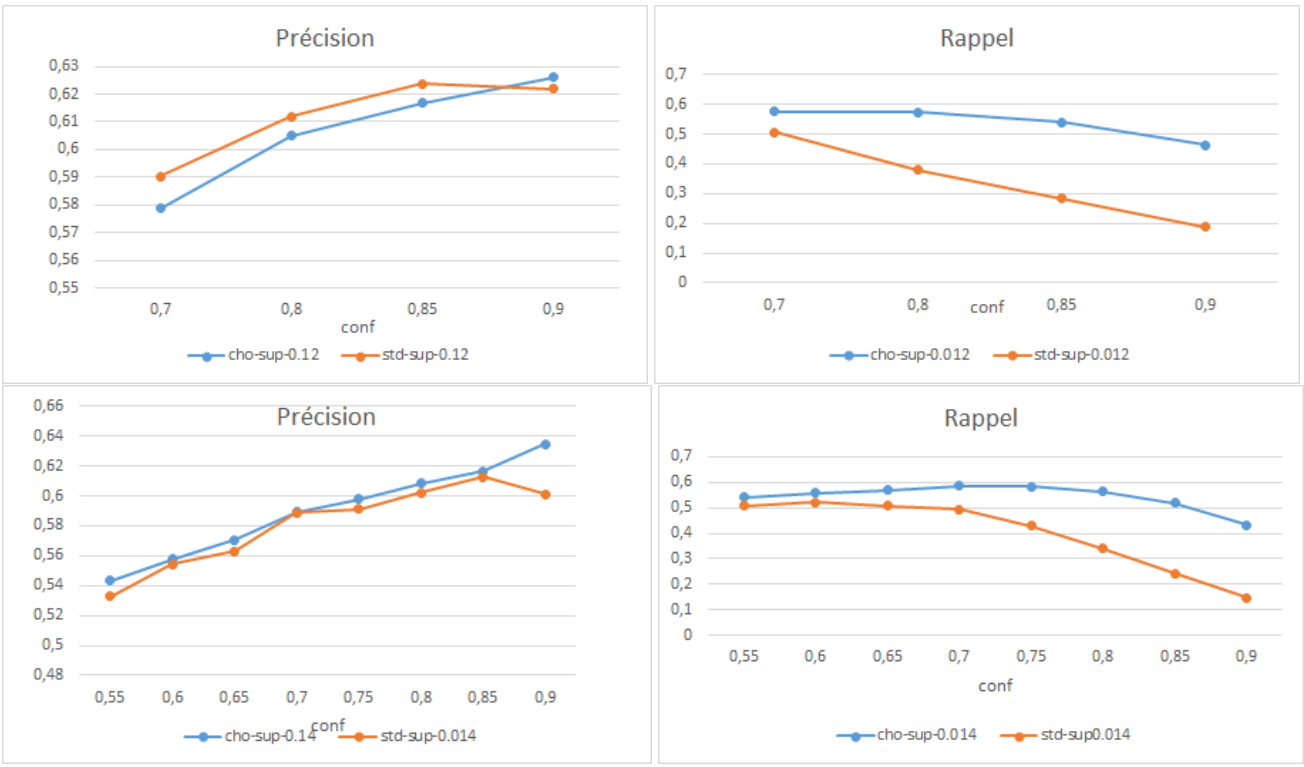
l'extraction des règles de préférence à partir des séquences de formules de préférence. En effet ce problème s'est avéré être dû au fait que les séquences de formules de préférence peuvent contenir un grand nombre d'éléments(formules de préférence élémentaires) alors que lors des travaux antérieurs ([]), les séquences de préférence quand à elles avaient un nombre d'éléments (items) beaucoup plus réduit et donc ils n'ont subit cette difficulté. Pour cela nous étions obligés d'utiliser un autre programme afin de gérer la partie consistant à extraire les règles de préférence interressantes minimales. Pour résumer :

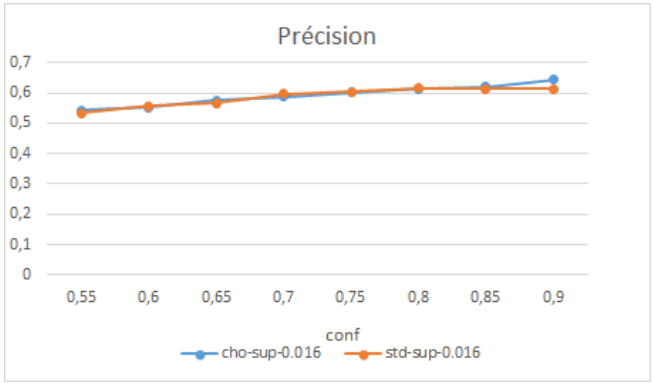
—

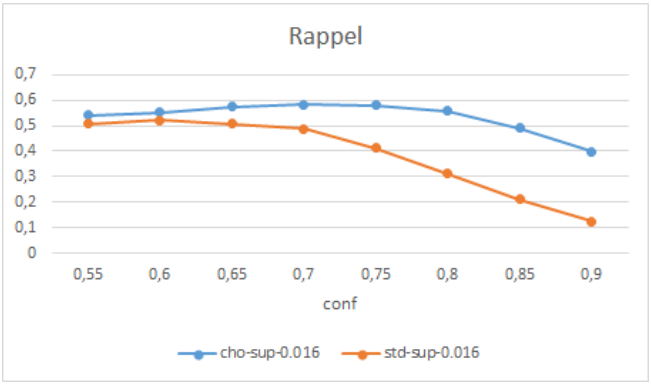
—

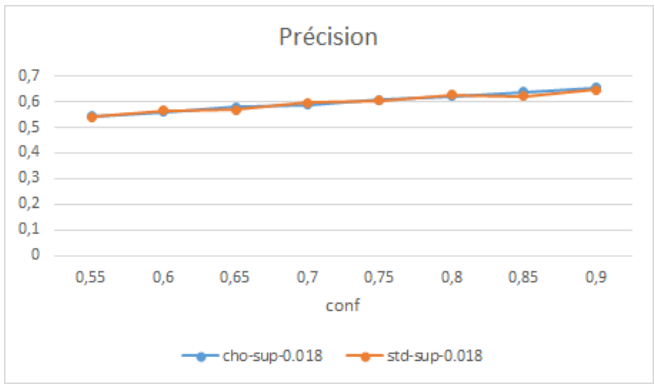
Initialement nous cherchions à utiliser Sprex autan pour l'extraction de profil que l'évaluation de la qualité des préférences. Nous nous somme confronté à un problème de saturation mémoire au niveau de Sprex dû au fait

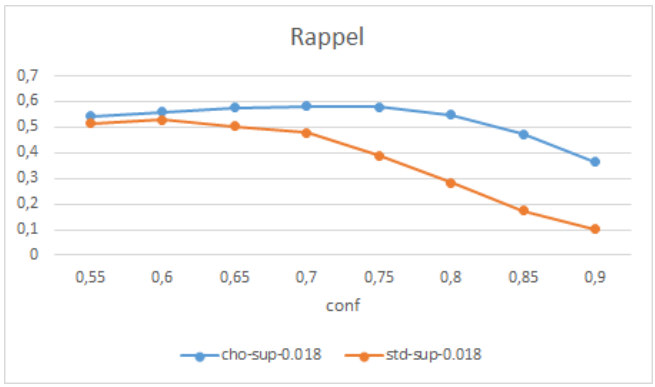
5.6 Evaluations expérimentales

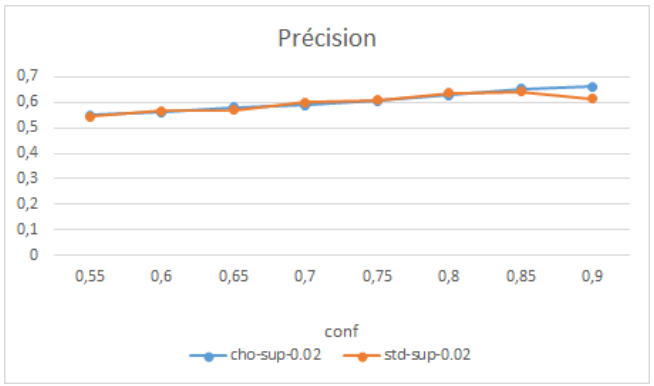


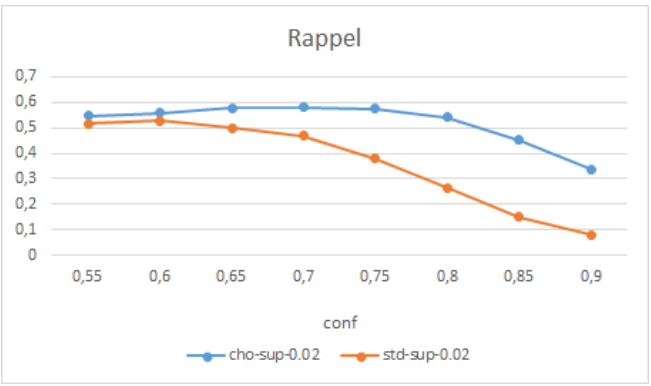






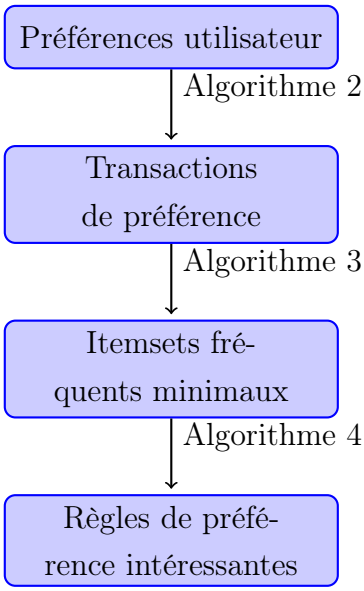






5.6.1 Interprétation

nous pouvons passer à la phase d'implémentation qui nous permettra alors d'éffectuer une comparaison de cette approche avec l'approche de Sprex.



Le diagramme ci dessus énonce les différentes étapes du processus permettant l'extraction des règles de préférence intéressantes. Ces étapes seront décrites par la suite. Nous allons tout d'abord définir le support d'un itemset et la confiance d'une règle.

5.6.2 Etapes de transformation des préférences utilisateurs en transaction de préférence

Cette étape consiste à transformer le couple $< t_1, t_2 >$ de préférences utilisateur en une transaction de formules que l'on appellera **transaction de préférence**.

Principe Il se fonde sur le formalisme de Chomicky appliqué aux règles de préférence contextuelles comme précédemment énoncé. Ainsi la prise en compte de la symétrie et de l'asymétrie dans cette situation, nous amène à définir ci-dessous les items élémentaires qui seront utilisés dans les transactions de préférence :

5.6.3 Etape de l'extraction des règles interressantes minimales des transactions de préférence

Cette étape peut utiliser toute méthode d'extraction d'itemsets intérresaants minimaux. Notre choix s'est porté sur une méthode d'extraction des itemsets fréquents minimaux décrite dans l'article [?], du faite du peu de mémoire dont elle nécessite pour l'extraction.

Principe

Elle consiste en un parcours en profondeur basé sur le principe du calcul des objets critiques. On a que

$$X \in \mathcal{F} \text{ est minimal ssi } \forall e \in X,\ \widehat{cov}(X, e) \neq \emptyset$$

avec $\widehat{cov}(X, e) = cov(X \setminus e) \setminus cov(e)$ et cov étant la couverture d'un itemset (nombre de transactions contenant l'itemset). Donc à chaque ajout d'un item à un itemset X lors d'un parcours en profondeur, l'évaluation de la minimalité du nouvel itemset Y obtenu se fait en utilisant l'égalité suivante :

$$\widehat{cov}(Xe', e) = \widehat{cov}(X, e) \cap cov(e')$$

Ainsi il suffit d'obtenir la couverture du nouvel item ajouté pour savoir si le nouvel itemset formé est minimal.

A partir de cette procédure, étant donné que la minimalité que nous utilisons est une minimalité basée sur le support et sur la confiance, nous allons procéder comme suit :

- Générer la base de transactions de préférences inverses et ajouter celle ci à la base de préférence initiale.
- Extraire à l'aide du procédé précédent, les itemsets minimaux suivant leur support
- Filtrer les itemsets obtenus suivant que leur support dans la base initiale soit supérieur au support seuil.

Cette procédure s'explique par le fait que si un itemset X est minimal dans la base globale, c'est que

$$\forall e \in X,\ supp(X \setminus e) \neq supp(X) \text{ ou } conf(X \setminus e) \neq conf(X)$$

5.7 Algorithme

Algorithme 7 : Algorithme d'extraction de règles interressantes

Entrées : Préférences utilisateur $\mathcal{P}$

Sorties : Règles de préférence interressantes $\mathcal{R}$

début

$\mathcal{P}_S = \emptyset$; `// Ensemble des transactions obtenues des tuples de préférence`

pour chaque $< T, U > \in \mathcal{P}$ **faire**

$\mathcal{P}_{TU} = \emptyset$; `// Ensemble des items`

pour chaque $A_i \in R$ **faire**

$x = T.A_i$; `// Val correspondante à l'attribut` A_i `de` T

$y = U.A_i$;

$\mathcal{P}_{TU} = \mathcal{P}_{TU} \bigcup$ Comparaison(x, y, A_i);

Ajouter P_{TU} à P_S;

$\mathcal{F}$ =ExtraireItemsetFréquents(P_S);

$\mathcal{R}$ =FiltrerRèglesInterressantes$(\mathcal{F})$;

Retourner $\mathcal{R}$;

5.8 Exemple d'extraction des règles de préférence

Nous allons étudier deux situations :

— Cas où le schémas relationel $\mathcal{R}(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n)$ n'a que des attributs symboliques ;

— Cas où le schémas relationnel $\mathcal{R}(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n)$ a des attributs symboliques et numériques.

5.8.1 Cas où le schémas relationnel n'a que des attributs symboliques

Prenons un schémas relationnel $\mathcal{R}(\mathcal{A}_1, \mathcal{A}_2)$ avec A_1 et A_2 de type symboliques. Soit la base de préférence ci dessous :

	t_1		t_2	
P_1	a	A	b	B
P_2	b	A	a	B
P_3	a	A	a	B
P_4	a	A	a	C
P_5	b	B	a	A

Passage des préférences vers les transactions de préférence

On a :

$P_1 \Rightarrow T_1 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = a\}, N : \{y_1 = b\}, C : \{x_2 \neq y_2\}, P : \{x_2 = A\}, N : \{y_2 = B\}, \}$

$P_2 \Rightarrow T_2 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = b\}, N : \{y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = A\}, N : \{y_2 = B\}, \}$

$P_3 \Rightarrow T_3 = \{C : \{x_1 = y_1\}, C : \{x_1, y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = A\}, N : \{y_2 = B\}, \}$

$P_4 \Rightarrow T_4 = \{C : \{x_1 = y_1\}, C : \{x_1, y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = A\}, N : \{y_2 = C\}, \}$

$P_5 \Rightarrow T_5 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = b\}, N : \{y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = B\}, N : \{y_2 = A\}, \}$

5.9 Determination des itemsets interressants minimaux avec supmin=2 et confmin=0.5

Itemsets interressants minimaux de taille 1 ++++++++++++++++++++++++

$I_4 = \{P : \{x_2 = A\}, \}$ sup=4.0 conf=0.8
$I_5 = \{N : \{y_2 = B\}, \}$ sup=3.0 conf=0.75
$I_6 = \{P : \{x_1 = b\}, \}$ sup=2.0 conf=0.666666666667
$I_7 = \{N : \{y_1 = a\}, \}$ sup=2.0 conf=0.666666666667

Itemsets interressants minimaux de taille 2 ++++++++++++++++++++++++

$I_{14} = \{C : \{x_1 \neq y_1\}, P : \{x_2 = A\}, \}$ sup=2.0 conf=0.666666666667
$I_{15} = \{C : \{x_1 \neq y_1\}, N : \{y_2 = B\}, \}$ sup=2.0 conf=0.666666666667
$I_{29} = \{P : \{x_2 = A\}, C : \{x_1 = y_1\}, \}$ sup=2.0 conf=1.0
$I_{30} = \{P : \{x_2 = A\}, C : \{x_1, y_1 = a\}, \}$ sup=2.0 conf=1.0

Pas d'itemsets interressant minimaux de taille 3

5.9.1 Cas où le schémas relationnel a des attributs numériques et symboliques

Prenons un schémas relationnel $\mathcal{R}(\mathcal{A}_1, \mathcal{A}_2)$ avec A_1 et A_2 de type symboliques. Soit la base de préférence ci dessous :

	t_1		t_2	
P_1	a	3	b	7
P_2	a	3	b	3
P_3	b	3	a	7
P_4	a	3	a	7
P_5	a	3	a	9
P_6	b	9	a	7
P_7	b	7	a	3

Transactions de preference obtenues

$P_1 \Rightarrow T_1 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = a\}, N : \{y_1 = b\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 3\}, N : \{y_2 = 7\}, P : \{x_2 < y_2\}, N : \{y_2 > 3\}, P : \{x_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$

$P_2 \Rightarrow T_2 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = a\}, N : \{y_1 = b\}, C : \{x_2 = y_2\}, C : \{x_2, y_2 = 3\}, C : \{x_2, y_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$

$P_3 \Rightarrow T_3 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = b\}, N : \{y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 3\}, N : \{y_2 = 7\}, P : \{x_2 < y_2\}, N : \{y_2 > 3\}, P : \{x_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$

$P_4 \Rightarrow T_4 = \{C : \{x_1 = y_1\}, C : \{x_1, y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 3\}, N : \{y_2 = 7\}, P : \{x_2 < y_2\}, N : \{y_2 > 3\}, P : \{x_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$

$P_5 \Rightarrow T_5 = \{C : \{x_1 = y_1\}, C : \{x_1, y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 3\}, N : \{y_2 = 9\}, P : \{x_2 < y_2\}, N : \{y_2 > 3\}, N : \{y_2 > 7\}, P : \{x_2 < 7\}, P : \{x_2 < 9\}, \}$

$P_6 \Rightarrow T_6 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = b\}, N : \{y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 9\}, N : \{y_2 = 7\}, P : \{x_2 > y_2\}, P : \{x_2 > 7\}, N : \{y_2 < 9\}, C : \{x_2, y_2 > 3\}, \}$

$P_7 \Rightarrow T_7 = \{C : \{x_1 \neq y_1\}, P : \{x_1 = b\}, N : \{y_1 = a\}, C : \{x_2 \neq y_2\}, P : \{x_2 = 7\}, N : \{y_2 = 3\}, P : \{x_2 > y_2\}, P : \{x_2 > 3\}, N : \{y_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$

5.10 Determination des itemsets interressants minimaux avec supmin=2 et confmin=0.5

Itemsets interressants minimaux de taille 1 ++++++++++++++++++++

$I_4 = \{P : \{x_2 = 3\}, \}$ sup=4.0 conf=0.8
$I_5 = \{N : \{y_2 = 7\}, \}$ sup=4.0 conf=0.8
$I_6 = \{P : \{x_2 < y_2\}, \}$ sup=4.0 conf=0.666666666667
$I_7 = \{N : \{y_2 > 3\}, \}$ sup=4.0 conf=0.8
$I_8 = \{P : \{x_2 < 7\}, \}$ sup=4.0 conf=0.8
$I_{13} = \{P : \{x_1 = b\}, \}$ sup=3.0 conf=0.6
$I_{14} = \{N : \{y_1 = a\}, \}$ sup=3.0 conf=0.6

Itemsets interressants minimaux de taille 2 ++++++++++++++++++++

$I_{32} = \{C : \{x_1 \neq y_1\}, P : \{x_2 = 3\}, \}$ sup=2.0 conf=0.666666666667
$I_{33} = \{C : \{x_1 \neq y_1\}, N : \{y_2 = 7\}, \}$ sup=3.0 conf=0.75
$I_{35} = \{C : \{x_1 \neq y_1\}, N : \{y_2 > 3\}, \}$ sup=2.0 conf=0.666666666667
$I_{36} = \{C : \{x_1 \neq y_1\}, P : \{x_2 < 7\}, \}$ sup=2.0 conf=0.666666666667
$I_{74} = \{C : \{x_2 \neq y_2\}, P : \{x_1 = b\}, \}$ sup=3.0 conf=0.75
$I_{75} = \{C : \{x_2 \neq y_2\}, N : \{y_1 = a\}, \}$ sup=3.0 conf=0.75
$I_{79} = \{P : \{x_2 = 3\}, N : \{y_2 = 7\}, \}$ sup=3.0 conf=0.75
$I_{83} = \{P : \{x_2 = 3\}, C : \{x_2, y_2 < 9\}, \}$ sup=3.0 conf=0.75
$I_{86} = \{P : \{x_2 = 3\}, C : \{x_1 = y_1\}, \}$ sup=2.0 conf=1.0
$I_{87} = \{P : \{x_2 = 3\}, C : \{x_1, y_1 = a\}, \}$ sup=2.0 conf=1.0
$I_{89} = \{N : \{y_2 = 7\}, P : \{x_2 < y_2\}, \}$ sup=3.0 conf=0.75
$I_{90} = \{N : \{y_2 = 7\}, N : \{y_2 > 3\}, \}$ sup=3.0 conf=0.75
$I_{91} = \{N : \{y_2 = 7\}, P : \{x_2 < 7\}, \}$ sup=3.0 conf=0.75
$I_{92} = \{N : \{y_2 = 7\}, C : \{x_2, y_2 < 9\}, \}$ sup=3.0 conf=0.75
$I_{93} = \{N : \{y_2 = 7\}, P : \{x_1 = b\}, \}$ sup=2.0 conf=1.0
$I_{94} = \{N : \{y_2 = 7\}, N : \{y_1 = a\}, \}$ sup=2.0 conf=1.0
$I_{100} = \{P : \{x_2 < y_2\}, C : \{x_2, y_2 < 9\}, \}$ sup=3.0 conf=0.75
$I_{103} = \{P : \{x_2 < y_2\}, C : \{x_1 = y_1\}, \}$ sup=2.0 conf=1.0
$I_{104} = \{P : \{x_2 < y_2\}, C : \{x_1, y_1 = a\}, \}$ sup=2.0 conf=1.0
$I_{107} = \{N : \{y_2 > 3\}, C : \{x_2, y_2 < 9\}, \}$ sup=3.0 conf=0.75

$I_{110} = \{N : \{y_2 > 3\}, C : \{x_1 = y_1\}, \}$ sup=2.0 conf=1.0
$I_{111} = \{N : \{y_2 > 3\}, C : \{x_1, y_1 = a\}, \}$ sup=2.0 conf=1.0
$I_{113} = \{P : \{x_2 < 7\}, C : \{x_2, y_2 < 9\}, \}$ sup=3.0 conf=0.75
$I_{116} = \{P : \{x_2 < 7\}, C : \{x_1 = y_1\}, \}$ sup=2.0 conf=1.0
$I_{117} = \{P : \{x_2 < 7\}, C : \{x_1, y_1 = a\}, \}$ sup=2.0 conf=1.0
$I_{127} = \{P : \{x_1 = b\}, P : \{x_2 > y_2\}, \}$ sup=2.0 conf=0.666666666667
$I_{130} = \{N : \{y_1 = a\}, P : \{x_2 > y_2\}, \}$ sup=2.0 conf=0.666666666667

Itemsets interressants minimaux de taille 3 ++++++++++++++++++++++

$I_{151} = \{C : \{x_1 \neq y_1\}, N : \{y_2 = 7\}, P : \{x_2 < y_2\}, \}$ sup=2.0 conf=0.666666666667
$I_{154} = \{C : \{x_2, y_2 < 9\}, C : \{x_1 \neq y_1\}, N : \{y_2 = 7\}, \}$ sup=2.0 conf=0.666666666667
$I_{160} = \{C : \{x_2, y_2 < 9\}, C : \{x_1 \neq y_1\}, P : \{x_2 < y_2\}, \}$ sup=2.0 conf=0.666666666667
$I_{197} = \{C : \{x_2, y_2 < 9\}, C : \{x_2 \neq y_2\}, P : \{x_1 = b\}, \}$ sup=2.0 conf=0.666666666667
$I_{198} = \{C : \{x_2, y_2 < 9\}, C : \{x_2 \neq y_2\}, N : \{y_1 = a\}, \}$ sup=2.0 conf=0.666666666667

Pas d'itemsets interressant minimaux de taille 4