REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE
N° : ……..

DOMAINE : Mathématiques et Informatique
FILIERE : INFORMATIQUE
OPTION : SIGL

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par : Roumili Amine

Intitulé

Étude sur l'algorithme ERMiner pour l'extraction des règles séquentielles

Soutenu devant le jury composé de :

Nom et prénom Enseignant	Université de M'sila	Président
Mr.Guesmia Salah	Université de M'sila	Rapporteur
……	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

Dédicace

A ma chère ***mère****, que nulle dédicace ne puisse exprimer Ce que je vous dois, pour votre bienveillances, votre affectation et votre soutien morale durant l'élaboration de ce travail, En témoignage de mon profond amour et mes sincères reconnaissances pour les efforts qu'ils ont consenti pour l'accomplissement de mes études,*

A toutes mes ***sœurs*** *et à tous les membres de ma famille .qui ont tous contribué par leurs encouragements ce travail.*

À mes très chers amis ***Khiero,Abd el hay et Mouhammed****.*

A tous mes collègues de la promo D013 surtout ***Islam, Lazhar, Nedjmo, Amir et Oussama*** *…*

Et finalement a tous Ceux qui m'ont soutenu tout au long de ce projet.

Je vous dédie ce modeste travail.

Roumili Amine

Remerciements

Avant tout on tient notre remerciement à notre dieu tout puissant de nous avoir donné la foi, la force et le courage.

Nous tenons à remercier d'abord notre encadreur, ***Mr.Guesmia Salah*** *pour avoir accepté de nous encadrer ainsi que pour son aide, ses conseils, et ses remarques objectives qui ont contribué à la réalisation de ce mémoire de fin d'étude.*

Nos remerciements les plus vifs s'adressent aussi aux Messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail.

Enfin nous tenons à remercier tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Table des matières

Chapitre 4 - Implémentation

Introduction générale

La découverte des modèles séquentiels intéressants dans les ordres est un problème fondamental dans l'exploitation de données. On a proposé beaucoup d'études pour les bases de données intéressantes d'extraction de modèles dans l'ordre. L'exploitation séquentielle de modèle est probablement le domaine de recherche de les plus populaires parmi elles. Elle se compose trouver des subséquences apparaître fréquemment dans un ensemble d'ordres. Cependant, savoir qu'un ordre apparaît n'est pas fréquemment suffisant pour faire des prévisions. Une alternative qui aborde le problème de la prévision est exploitation séquentielle de règle. Une règle séquentielle indique que si quelques articles se produisent dans un ordre, quelques autres articles sont susceptibles de se produire après avec une confiance ou une probabilité donnée. On a proposé deux types principaux de règles séquentielles. Le premier type est règles où l'antécédent et le conséquent sont les modèles séquentiels. Le deuxième type est des règles entre deux ensembles non commandés d'articles. En ce document nous considérons le deuxième type parce qu'il est plus général et on lui a montré pour fournir une exactitude considérablement plus élevée de prévision pour la prévision d'ordre dans quelques domaines.

D'ailleurs, une autre raison est que le deuxième type a été employé dans beaucoup de vraies applications telles que l'apprentissage en ligne, la simulation de fabrication, le contrôle de qualité, préextraction de page Web, la détection d'anti-modèle en service les systèmes basés, les systèmes inclus, l'analyse d'ordre d'alarme et la recommandation de restaurant. On a proposé plusieurs algorithmes pour le mien de ce type de règles séquentielles. CMDeo est un algorithme basé sur Apriori qui explore l'espace de recherche des règles utilisant une recherche en largeur. Un inconvénient important de CMDeo est qu'il peut produire d'une énorme quantité de candidats. Comme alternative, on a proposé l'algorithme de CMRules. Il se fonde sur la propriété que toutes les règles séquentielles doivent également être une règle d'association pour tailler l'espace de recherche des règles séquentielles. Il s'est avéré beaucoup plus rapide que CMDeo pour des ensembles de données clairsemés. Récemment, on a proposé l'algorithme de RuleGrowth. Il se fonde sur une approche de modèle-croissance pour éviter la génération de candidat. Il s'est avéré plus qu'un ordre de grandeur plus rapidement que CMDeo et CMRules. Cependant, pour des ensembles de données contenant des ordres denses ou longs, la représentation des deterioates de RuleGrowth rapidement parce qu'elle doit à plusieurs reprises effectuer des opérations coûteuses

de projection de base de données. Puisque l'extraction des règles séquentielles reste très informatique une tâche chère d'exploitation de données, une question importante de recherches est : « Pourrions-nous concevons des algorithmes plus rapides ? »

En ce document, nous abordons cette question en proposant l'ERMiner Il se fonde sur une représentation verticale de la base de données pour éviter d'exécuter la projection de base de données et l'idée nouvelle d'explorer l'espace de recherche des règles utilisant les classes d'équivalence de règles ayant le même antécédent ou conséquent. En outre, il inclut une structure de données SCM (Sparse Count Matrix) nommé (compte clairsemé Matrix) pour tailler l'espace de recherche. Le reste du papier est organisé comme suit. La section 2 définit le problème de l'exploitation séquentielle de règle et présente des définitions et des propriétés importantes. La section 3 décrit l'algorithme d'ERMiner. La section 4 présente l'expérimental étude. En conclusion, la section 5 présente la conclusion

Chapitre 1

Le data-mining et les règles d'association

Chapitre 1

Le data-mining et les règles d’association

1 Introduction

Dans le domaine du Data-Mining la recherche des règles d'association est une méthode populaire étudiée d'une manière approfondie dont le but est de découvrir des relations ayant un intérêt pour le statisticien entre deux ou plusieurs variables stockées dans de très importantes bases de données.

Un exemple de règle d'association extraite d'une base de données de ventes de supermarché est : « céréales, sucre → lait (support 7%, confiance50%) ». Cette règle indique que les clients qui achètent des céréales et du sucre ont également tendance à acheter du lait. La mesure de support définit la portée de la règle, c'est à dire la proportion de clients qui ont acheté les trois articles, et la mesure de confiance définit la précision de la règle, c'est à dire la proportion de clients qui ont acheté du lait parmi ceux qui ont acheté des céréales et du sucre. L'extraction de règles d'association consiste à extraire les règles dont le support et la confiance sont au moins égaux à des seuils minimaux de support et de confiance définis par l'utilisateur. Les règles d'association ont été utilisées avec succès dans de nombreux domaines, parmi lesquels l'aide à la planification commerciale, l'aide au diagnostic et en recherche médicale, l'amélioration des processus de télécommunications, l'organisation et l'accès aux sites Internet, et l'analyse d'images, de données spatiales, de données géographiques et de données statistiques**.**

2 Présentation du data mininig :

Le terme de Data Mining est souvent employé pour désigner l’ensemble des outils permettant à l’utilisateur d’accéder aux données volumineuses de l'entreprise et à en extraire de manière automatique des connaissances intéressantes et inconnues, imprévues, a priori. Nous restreindrons ici le terme de Data Mining aux outils ayant pour objet de générer des informations riches à partir des données de l’entreprise, notamment des données historiques, de découvrir des modèles implicites dans les données. Ils peuvent permettre par exemple à un magasin de dégager des profils de client et des achats types et de prévoir ainsi les ventes futures. Il permet d’augmenter la valeur des données contenues dans le DW (Datawarehouse).

Les outils d'aide à la décision, qu'ils soient relationnels ou **OLAP (on-line transaction processing)**, laissent l'initiative à l'utilisateur, qui choisit les éléments qu'il veut observer ou analyser. Au contraire, dans le cas du Data Mining, le système a l'initiative et découvre lui-même les associations entre données, sans que l'utilisateur ait à lui dire de rechercher plutôt dans telle ou telle direction ou à poser des hypothèses. Il est alors possible de prédire l'avenir, par exemple le comportement d'un client, et de détecter, dans le passé, les données inusuelles, exceptionnelles.

Ces outils ne sont plus destinés aux seuls experts statisticiens mais doivent pouvoir être employés par des utilisateurs connaissant leur métier et voulant l'analyser, l'explorer. Seul un utilisateur connaissant le métier peut déterminer si les modèles, les règles, les tendances trouvées par l'outil sont pertinentes, intéressantes et utiles à l'entreprise. Ces utilisateurs n'ont donc pas obligatoirement un bagage statistique important. L'outil doit donc soit être ergonomique, facile à utiliser et rendant transparentes toutes les formules mathématiques et termes techniques utilisés, soit permettre de construire une application « clé en main », rendant à l'utilisateur transparentes toutes les techniques utilisées. [1]

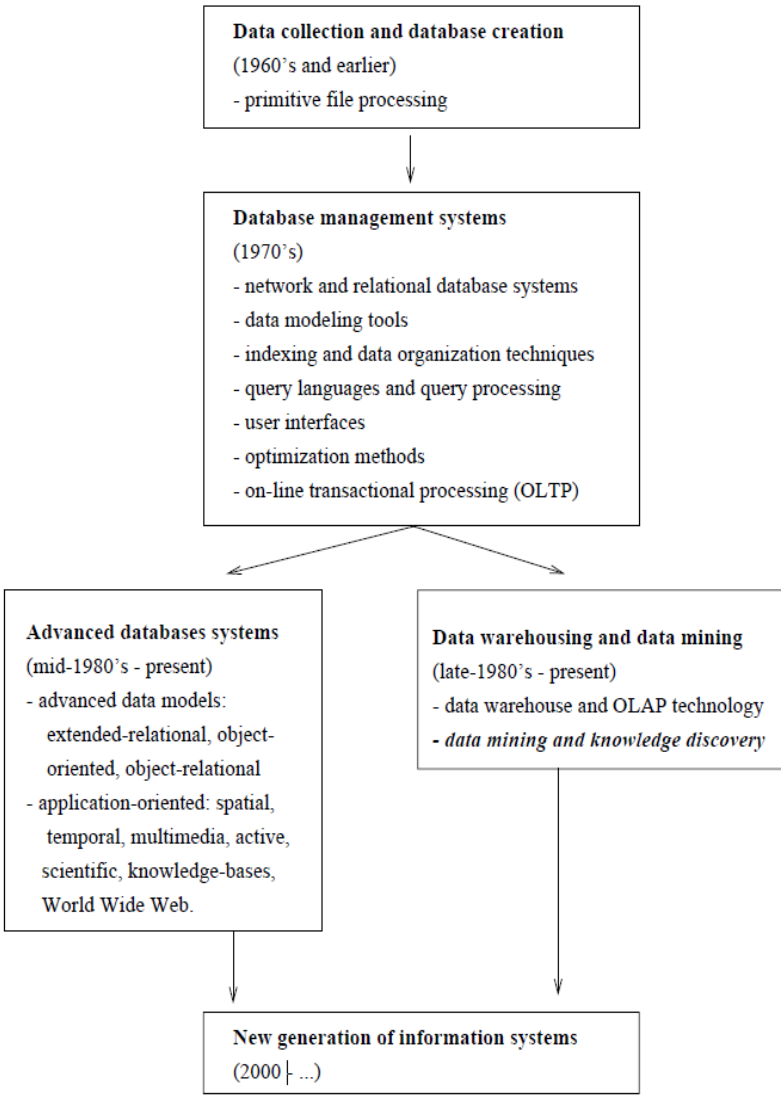

Figure 1.1 : L'évolution de la technologie de base de données.

On pourrait définir le data-mining comme une démarche ayant pour objet de découvrir des relations et des faits, à la fois nouveaux et significatifs, sur de grands ensembles de données. On devrait ajouter que la pertinence et l'intérêt du Data Mining sont conditionnés par les enjeux attachés à la démarche entreprise, qui doit être guidée par des objectifs directeurs clairement explicités ("améliorer la performance commerciale", "mieux cibler les prospects", "fidéliser la clientèle", "mieux comprendre les performances de production"...).

Nous appellerons Data Mining l'ensemble des techniques qui permettent de transformer les données en connaissances.

Il existe de nombreux autres termes qui ont un sens similaire ou légèrement différent de data mining, tels que **knowledge mining from databases (**l'exploration des connaissances à partir des bases), **knowledge extraction** (extraction de connaissances), **data/pattern analysis** (analyse de données / modèles), **data archaeology** (archéologie des données) et data **dredging** (dragage de données).

Beaucoup de gens traitent le data mining comme un synonyme d'un autre terme couramment utilisé, **"Knowledge Discovery in Databases"**, ou **KDD**. Alternativement, d'autres considèrent le data mining comme une étape essentielle dans le processus de la connaissance découverte dans les bases de données. La découverte de connaissances en tant que processus est illustrée à la figure 1.2 et consiste en une séquence itérative des étapes suivantes : [2]

- **Data cleaning (**Nettoyage des données**)** : pour supprimer le bruit ou les données non pertinentes).
- **Data integration** (intégration de données) : où plusieurs sources de données peuvent être combinées.
- **Data selection** (la sélection des données) : où les données pertinentes à la tâche d'analyse sont extraites de la base de données.
- **Data transformation** (transformation de données) : lorsque les données sont transformées ou consolidées dans des formes appropriées pour l'extraction par effectuer des opérations de résumé ou d'agrégation
- **Data mining** (l'exploration de données) : un processus essentiel où des méthodes intelligentes sont appliquées pour extraire des modèles de données.
- **Pattern evaluation** (évaluation de modèle) : pour identifier les modèles vraiment intéressants représentant la connaissance basée sur quelques mesures d'intérêt.
- **Knowledge presentation** (Présentation des connaissances) : lorsque des techniques de visualisation et de représentation des connaissances sont utilisées pour présenter les connaissances minées à l'utilisateur.

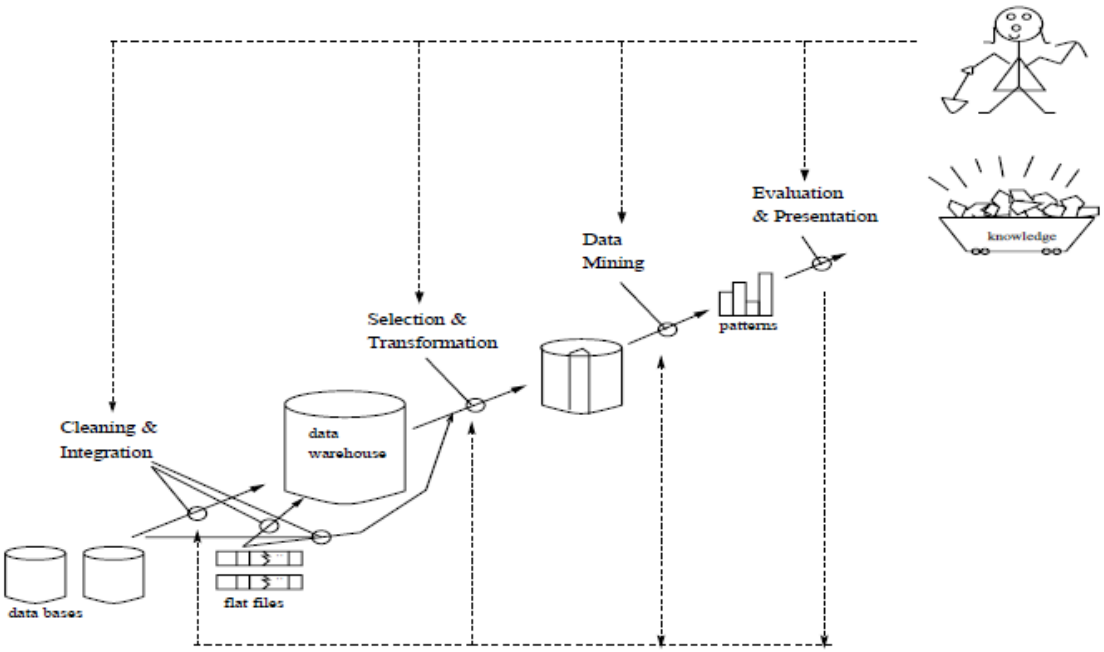

Figure 1.2 : Le data-mining en tant que processus de découverte des connaissances.

3 Les domaines de l'utilisation du data-mining :

Le développement récent de la fouille de données (depuis le début des années 1990) est lié à plusieurs facteurs : une puissance de calcul importante est disponible sur les ordinateurs de bureau ou même à domicile ; le volume des bases de données augmente énormément ; l'accès aux réseaux de taille mondiale, ces réseaux ayant un débit sans cesse croissant, qui rendent le calcul distribué et la distribution d'information sur un réseau d'échelle mondiale viable ; la prise de conscience de L'intérêt commercial pour l'optimisation des processus de fabrication, vente, gestion, logistique.

La fouille de données a aujourd'hui une grande importance économique du fait qu'elle permet d'optimiser la gestion des ressources (humaines et matérielles). Elle est utilisée par exemple :

- Organisme de crédit : pour décider d'accorder ou non un crédit en fonction du profil du demandeur de crédit, de sa demande, et des expériences passées de prêts.
- Optimisation du nombre de places dans les avions, hôtels, ... (sur- réservation).
- Organisation des rayonnages dans les supermarchés en regroupant les produits qui sont généralement achetés ensemble (pour que les clients n'oublient pas bêtement d'acheter

un produit parce qu'il est situé à l'autre bout du magasin). Par exemple, on extraira une règle du genre : ≪les clients qui achètent le produit X en fin de semaine, pendant l'été, achètent généralement également le produit Y≫.

- Organisation de campagne de publicité, promotions, ... (ciblage des offres) diagnostic médical : ≪les patients ayant tels et tels symptômes et demeurant dans des agglomérations de plus de 104 habitants développent couramment telle pathologie≫.
- Analyse du génome et bio-informatique plus généralement.
- Classification d'objets (astronomies, ...)
- Commerce électronique, recommandation de produits
- Analyser les pratiques et stratégies commerciales et leurs impacts sur les ventes.
- Moteur de recherche sur internet : fouille du web extraction d'information depuis des textes : fouille de textes
- Évolution dans le temps de données : fouille de séquences. [3]

4 Les tâches du Data-Mining :

Une multitude de problèmes d'ordre intellectuel, économique ou commercial peuvent être regroupés, dans leur formalisation, dans l'une des tâches suivantes :

- **La classification :** La classification consiste à examiner des caractéristiques d'un élément nouvellement présenté afin de l'affecter à une classe d'un ensemble prédéfini.
- **L'estimation :** Contrairement à la classification, le résultat d'une estimation permet d'obtenir une variable continue. Celle-ci est obtenue par une ou plusieurs fonctions combinant les données en entrée. Le résultat d'une estimation permet de procéder aux classifications grâce à un barème.
- **La prédiction :** La prédiction ressemble à la classification et à l'estimation mais dans une échelle temporelle différente. Tout comme les tâches précédentes, elle s'appuie sur le passé et le présent mais son résultat se situe dans un futur généralement précisé.
- **La Segmentation :** La segmentation consiste à segmenter une population hétérogène en sous populations homogènes. Contrairement à la classification, les sous-populations ne sont pas préétablies.

- **La description :** C'est souvent l'une des premières tâches demandées à un outil de Data-Mining. On lui demande de décrire les données d'une base complexe. Cela engendre souvent une exploitation supplémentaire en vue de fournir des explications.
- **L'optimisation :** Pour résoudre de nombreux problèmes, il est courant pour chaque solution potentielle d'y associer une fonction d'évaluation. Le but de l'optimisation est de maximiser ou minimiser cette fonction. Quelques spécialistes considèrent que ce type de problème ne relève pas du Data- Mining. [4]

5 Les techniques de Data-Mining :

Plusieurs techniques peuvent être inscrites dans le contexte du Data-Mining, on en cite :

- **Les arbres de décision :** ce sont des outils très puissants principalement utilisés pour la classification, la description ou l'estimation. Il s'agit d'une représentation graphique sous forme d'arbre de décision représentant un enchaînement hiérarchique de règles logiques qui permet de diviser la base d'exemples en sous-groupes.
- **Les réseaux de neurones :** inspirés de la biologie, les réseaux de neurones représentent une transposition simplifiée des neurones du cerveau humain. Ils sont utilisés dans la prédiction et la classification.
- **Les algorithmes génétiques :** ce sont des méthodes d'optimisation de fonctions basée sur les mécanismes génétiques pour élaborer des paramètres de prévision des plus optimaux. Ils permettent de résoudre des problèmes divers, notamment d'optimisation, d'affectation ou de prédiction.
- **Les règles d'association :** Ce sont les méthodes les plus répandues dans le domaine de marketing et de la distribution. Elles peuvent être appliquées dans différents secteurs d'activité pour lesquels, il est intéressant de trouver des groupements d'articles qui apparaissent le plus fréquemment ensembles, et de générer des règles d'associations. Le but principal de cette technique est descriptif ou prédictif.
- **Les plus proches voisins (CBR, Case BasedReasoning) :** C'est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation. De façon

similaire, le « raisonnement basé sur la mémoire », technique du Data-Mining dirigé, permet de classer ou de prédire des données inconnues à partir d'instances connues.

- **Le Clustering :** C'est une classification non supervisée, les classes possibles et leur nombre ne sont pas connus à l'avance et les exemples disponibles sont non étiquetés. [4]

6 Concepts et définitions pour les règle d'association :

Nous rappelons brièvement les notions de base dans ce cadre théorique :

6.1 Pattern Mining (L'extraction des motifs) :

L'exploration fréquente de motifs donne lieu à une tâche de niveau supérieur de l'exploration de données, à savoir l'extraction de règles. Les deux types d'extraction de règles qui nous intéressent sont la règle d'association et la règle séquentielle. Du champ de l'extraction des Itemset (Itemset Mining), alors que ce dernier est contenu dans le champ l'extraction de motif séquentielle. Les deux sont des sous-champs de l'extraction des motifs. La hiérarchie complète est illustrée à la Figure 1.3. Nous allons commencer par explorer ces deux sous-champs de Pattern Mining. [view]

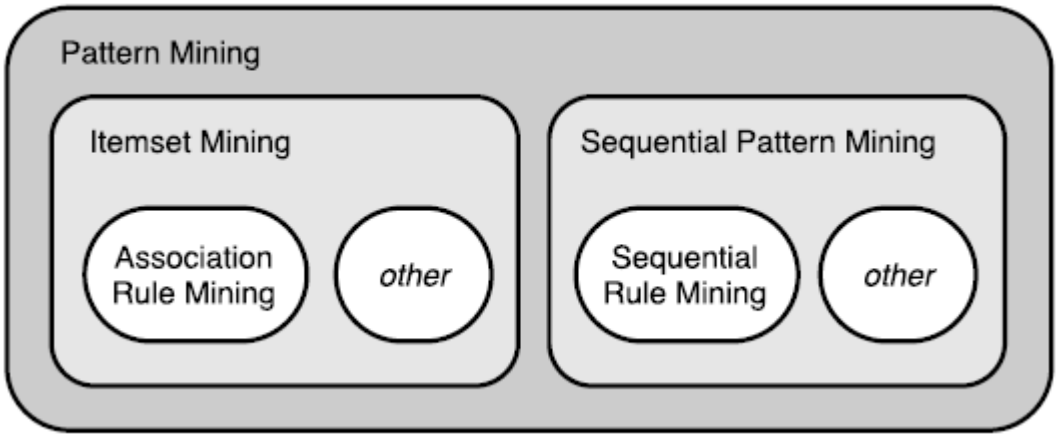

Figure 1.3 : L'extraction des motifs et ses sous-domaines.

6.2 Contexte de fouille

Un contexte de fouille est un triplet $\beta = (O, A, R)$ décrivant un ensemble fini **O** d'objets (ou de transactions), un ensemble fini **A** d'attributs (ou items) et une relation (d'incidence) binaire **R** ($R \subset O \times A$). Chaque couple $(o, a) \in R$, désigne le fait que l'objet $o \in O$, possède l'attribut $a \in A$.

Pour instance, nous prenons l'exemple de contexte de fouille cité dans [SALL 03] : Un complexe cinématographique a décidé de fidéliser son public en lançant la carte d'abonnement au cinéma dit "illimité". Les films vus par chaque cinéphile sont enregistrés dans une base de données à chaque fois que le client se présente au guichet.

Elle est exploitée par la suite pour comprendre les attitudes de "consommation" du cinéma, les types de films les plus prisés par le public, les heures auxquelles les gens préfèrent venir voir un film, etc.

La table β ci-dessous est un extrait (fictif) et donne pour chaque cinéphile identifié par un numéro tid, l'ensemble des films qu'il a vus durant le mois courant. Les films concernés sont donnés dans la table *I*.

Par exemple la ligne d'identificateur tid=1 de β concerne un client ayant vu dans le mois les trois films suivants : « Harry Potter », « Attrape-moi si tu peux » et « Un homme d'exception ». [5]

I		
Item	**Titre**	**Réalisateur**
A	Harry Potter	Chris Columbus
B	Star Wars II	George Lucas
C	Attrape-moi si tu peux	Steven Spielberg
D	Un homme d'exception	Ron Howard
E	Taken	Steven Spielberg

β	
Identifiant	**Transaction**
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Tableau 1.1 : Exemple de contexte de fouille

6.3 Item et itemset :

Un item peut être défini comme un article et un itemset un ensemble d'articles.

6.4 Transactions :

Une transaction est un ensemble d'items achetés par un client C à une date précise. Dans une base de données une transaction est représentée par trois attributs : idClient (identifiant d'un client), idDate (un identifiant pour une date), itemset (un ensemble d'items non vide).

Clients	Dates	Itemsets
Client1	02/03/2008 03/04/2008	Pain, TV beurre
Client2	10/02/2008 11/02/2008	Lecteur DVD Dattes, pain
Client3	12/02/2008 13/02/2008	Pain Beurre

Tableau 1.2 : Un exemple de base de données contenant 4 transactions**.**

6.5 Support d'une règle d'association :

Le support d'une règle d'association s'exprime par le nombre de transactions qui contiennent les éléments de X et les éléments de Y divisé par le nombre total des transactions de la base des transactions.

Dans une base de données D, le support d'une règle d'association X →Y est le nombre de transactions qui contiennent X et Y divisé par le nombre total des transactions.

$$Support(X \rightarrow Y) = \frac{Card(X \cup Y)}{Card(D)}$$

Exemple, la règle d'association " Lait ⇒ Pain ", littéralement, Si Lait Alors Pain.

Le support représente le nombre de transactions dans lesquelles on trouve les Items Lait et Pain, divisé par le nombre total des transactions

6.6 Support minimal

Le support minimal est le nombre minimum d'occurrence d'un motif séquentiel pour être considéré comme fréquent. L'occurrence n'est prise en compte qu'une fois dans la séquence. C'est un seuil choisi par l'utilisateur.

6.7 La force d'une règle d'association

Est mesurée par son indice de support et son indice de confiance.

6.8 L'indice de support

L'indice de support d'une règle **X→→Y** est défini par la proportion de transactions de T qui contiennent **X ∪ Y** (à la fois X et Y, et non X ou Y), soit **Supp (X ∪ Y).** Il s'agit donc d'une estimation de la probabilité **Pr(X∪Y)**.

6.9 L'indice de confiance

L'indice de confiance d'une règle X →→ Y est défini par la proportion de transactions de T contenant X qui contiennent aussi Y, soit

$\frac{(\mathbf{X \cup Y}).\mathbf{count}}{\mathbf{X}.\mathbf{count}}$ **. Pr (Y / X).**

Remarque : **Conf (X→→ Y) =** $\frac{\mathbf{SUPP}\ (\mathbf{X} \cup \mathbf{Y})}{\mathbf{SUPP}\ (\mathbf{X})}$

Exemple, la confiance d'une règle d'association " Lait ⇒ Pain", est égale au support de la règle " Lait ⇒ Pain " divisé par le support de l'Item " Lait ".

6.10 Le « lift »

Le lift d'une règle **X→→Y** mesure l'amélioration apportée par la règle d'association par rapport **Supp** (**X** ∪ **Y**) à un jeu de transactions aléatoire (où X et Y seraient indépendants). Il est défini par $\frac{\mathbf{Supp}(\mathbf{X} \cup \mathbf{Y})}{\mathbf{Supp}(\mathbf{X}).\mathbf{Supp}(\mathbf{Y})}$

Un lift supérieur à 1 traduit une corrélation positive de X et Y, et donc le caractère significatif de l'association. [5]

7 Les règles d'association

7.1 Recherche de règles d'association :

Une règle d'association est une relation d'implication X→Y entre deux ensembles d'articles X et Y. Cette règle indique que les transactions qui contiennent les articles de l'ensemble X ont tendance à contenir les articles de l'ensemble Y. X est appelé condition ou prémisse et Y résultat ou conclusion.

L'extraction des règles d'association est l'un des principaux problèmes de l'ECD (Extraction de Connaissances à partir de Données). Ce problème fut développé à l'origine pour l'analyse de base de données de transactions de ventes. Chaque transaction est constituée d'une liste d'articles achetés, afin d'identifier les groupes d'articles vendus le plus fréquemment ensemble.

Ces règles sont intuitivement faciles à interpréter car elles montrent comment des produits ou des services se situent les uns par rapport aux autres. Ces règles sont particulièrement utiles en marketing. Les règles d'association produites par la méthode peuvent être facilement utilisées dans le système d'information de l'entreprise. Cependant, il faut noter que la méthode, si elle peut

produire des règles intéressantes, peut aussi produire des règles triviales (déjà bien connues des intervenants du domaine) ou inutiles (provenant de particularités de l'ensemble d'apprentissage).

La recherche de règles d'association est une méthode non supervisée car on ne dispose en entrée que de la description des achats. On peut dire donc qu'une règle d'association est une règle de la forme : Si **condition** alors **résultat**.

Dans la pratique, on se limite, en général, à des règles où la condition est une conjonction d'apparition d'articles et le résultat est constitué d'un seul article. Par exemple, une règle à trois articles sera de la forme : Si X et Y alors Z ; règle dont la sémantique peut être énoncée : Si les articles X et Y apparaissent simultanément dans un achat alors l'article Z apparaît. [6]

7.2 L'extraction des règles d'associations

L'extraction des règles associatives est une méthode qui a vu le jour avec la recherche en bases de données, initialement introduite par Agrawal [7] pour l'analyse du panier de la ménagère pour retrouver les relations entre produits. Elle permet d'analyser les tickets de caisse des clients afin de comprendre leurs habitudes de consommations, agencer les rayons du magasin, organiser les promotions, gérer les stocks, etc.

Dans les bases de données de ventes, un enregistrement consiste en une transaction regroupant l'ensemble des articles achetés appelées items. Ainsi une base de données est un ensemble de transactions, qu'on appelle aussi base transactionnelle ou base de transactions.

Par exemple on sera capable de dire que 70% des clients qui achètent du lait achètent en même temps des œufs (lait → œuf : 0.70), une telle constatation est très intéressante puisqu'elle aide le gestionnaire d'un supermarché à ranger ses rayons de telle sorte que le lait et les œufs soient à proximité.

L'extraction des règles d'association peut être décomposée en quatre étapes qu'illustre la Figure 1.4

Les étapes d'extraction de règles d'association suivante :

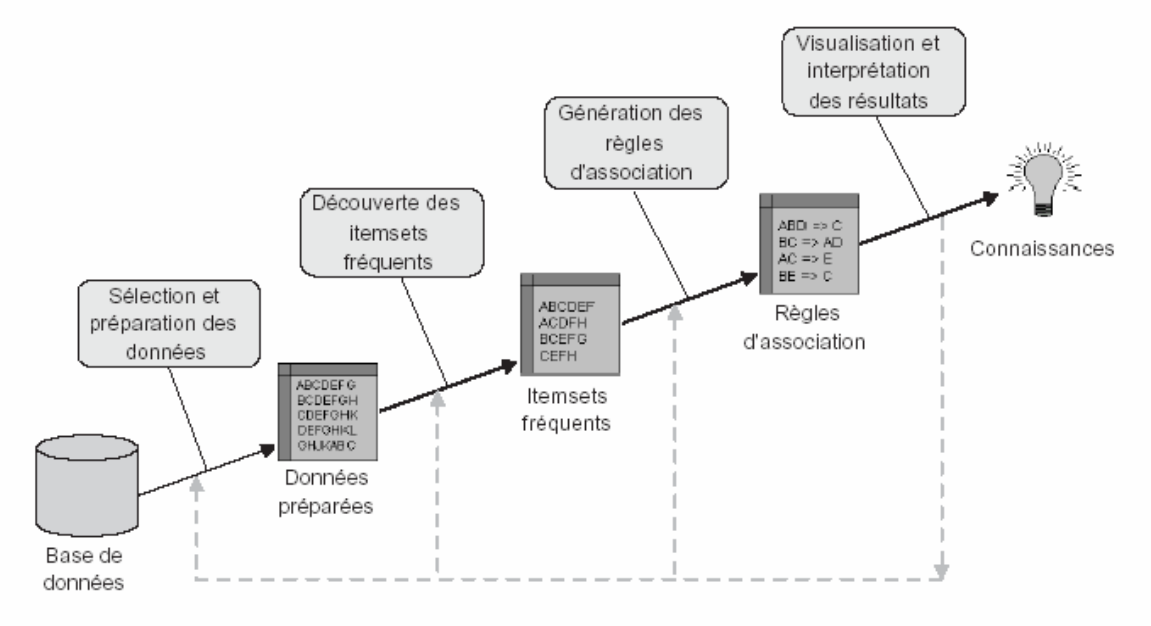

Figure 1.4 : Les étapes d'extraction de règles d'association

Décrivons ces quatre étapes :

7.2.1 Sélection et préparation des données :

Cette étape permet de préparer les données afin de leur appliquer les algorithmes d'extraction des règles d'association. Elle est constituée de deux phases :

- La sélection des données de la base qui permettront d'extraire les informations intéressant l'utilisateur. Ainsi la taille des données traitées est réduite ce qui assure une meilleure efficacité de l'extraction.

- La transformation de ces données en un contexte d'extraction (il s'agit d'un triplet constitué d'un ensemble d'objets, d'un ensemble d'itemsets et d'une relation binaire entre les deux).

La transformation des données sélectionnées en données binaires améliore l'efficacité de l'extraction et la pertinence des règles d'association extraites.

7.2.2 Découverte des itemsets fréquents :

C'est l'étape la plus coûteuse en termes de temps d'exécution car, le nombre d'itemsets fréquents dépend exponentiellement du nombre d'items manipulés (pour n items, on a 2^n itemsets potentiellement fréquents).

Dans le Tableau 1.2 Un exemple de base de données contenant 4 transactions, en fixant le support minimal à 50% c'est-à-dire : pour qu'un item soit considéré fréquent, il faut qu'il soit acheté par au moins deux clients. On aura :

Le pain a été acheté par les trois clients et le beurre par les clients *client1* et *client3.*

Donc les itemsets fréquents sont : {pain, beurre}.

7.2.3 Génération des règles d'association :

A partir de l'ensemble des itemsets fréquents pour un seuil minimal de support *minsup*, la génération des règles d'association est un problème qui dépend exponentiellement de la taille de l'ensemble des itemsets fréquents.

Du Tableau 2 Un exemple de base de données contenant 4 transactions1, on a la règle suivante : Pain → Beurre.

7.2.4 Visualisation et interprétation des règles d'associations :

Elle met entre les mains de l'utilisateur un ensemble de déductions fiables qui peuvent l'aider à prendre une décision. [6]

7.3 Les règles d'association maximales

7.3.1 Définition

Les règles d'association maximale parue dans le début des années 60, est un type de règles d'association représentant une méthode très pertinente pour extraire des relations existantes dans un ensemble de données. [8]

Une règle d'association maximale est notée comme suit : $\mathbf{X} \overset{\mathbf{max}}{\Longrightarrow} \mathbf{Y}$

7.3.2 Catégorie et taxonomie d'item

Soit un ensemble d'items I = {Item1, Item2, Item3 …Itemn}.

Une taxonomie T de I représente chaque sous-ensemble disjoint d'items de I. avec. T = {T_1 T_2, …, T_n}. Un élément de T représente une catégorie d'items de I.

On notera T(i) la catégorie de i, pour tout Item de I, si X est un ensemble d'itemsets qui sont tous d'une seule catégorie, alors on notera cette catégorie par T(X).

7.3.3 M-Support

On note le support maximal d'un item X par Smax (X).

Soit Ti une catégorie de I, avec X un itemset tel que X ⊆Ti, On dit que X apparait seul dans une transaction t, si et seulement si Ti =X. Autrement dit, X est le plus grand sous-ensemble de Ti qui est dans la transaction t.

On dit qu'une règle X ⟹ Y M-supporte un item X, si X apparaît seul dans une transaction t alors l'item Y apparaît également.

Le M-support représente le support maximal de la règle $\mathbf{X} \overset{\mathbf{max}}{\Longrightarrow} \mathbf{Y}$.

Le M-support d'une règle $\mathbf{X} \overset{\mathbf{max}}{\Longrightarrow} \mathbf{Y}$ représente toutes les transactions de la base des transactions qui M-Supporte X et Supporte Y.

$$\text{M-Support } (X \overset{MAX}{\Longrightarrow} Y) = \{ t \in T \text{ tel que } t : M - \text{supporte X et supporte Y} \}$$

Figure 1.5 : M-Support d'une règle d'association maximale**.**

7.3.4 M-Confiance

Le M-confiance représente la confiance maximale de la règle$\mathbf{X} \overset{\mathbf{max}}{\Longrightarrow} \mathbf{Y}$.

Elle est notée par Cmax ($\mathbf{X} \overset{\mathbf{max}}{\Longrightarrow} \mathbf{Y}$).

Soit T(Y) la catégorie de Y, on note D (X, T(Y)) le sous-ensemble de la base de données D constitué de toutes les transactions qui M-supportent X et qui contiennent au moins un élément de T(Y).

La formule de la M-Confiance sera la suivante :

$$\text{M-Confiance } (X \overset{MAX}{\Longrightarrow} Y) = \frac{\text{M-Support } (X \overset{MAX}{\Longrightarrow} Y)}{D(X,T(Y))}$$

Figure 1.6. M-Confiance d'une règle d'association maximale**.** [8]

7.4 Avantages et inconvénients des règles d'association maximales

Les règles d'association maximales permettent de déceler des règles d'association dont la qualité d'information est jugée plus au moins importante, mais qui peut être importante pour

comprendre un corpus donné. Quant à leurs inconvénients, ils résident dans le problème de la complexité computationnelle et l'explosion combinatoire.

La facilité d'utilisation et d'interprétation des résultats en sortie rend l'utilisation des règles d'association maximales plus avantageuse que celle des règles d'association régulières.

7.5 Domaine d'applications

Le fait d'identifier des relations significatives entre les données contenues dans une base peut s'avérer utile à différentes domaines commerciaux, scientifiques est industriels dans leur but de rentabiliser leur profit.

Plusieurs travaux basés sur la recherche des règles d'association ont été appliqués dans des applications réelles comme :

- La planification commerciale.
- Les réseaux de télécommunication.
- La recherche médicale.
- Les données de web.
- Le marketing bancaire.
- Le multimédia.
- Le panier de ménagère.

8 Algorithme d'extraction des règles d'association

Il existe plusieurs façons d'explorer les règles d'association, l'une de ces méthodes est la méthode naïve, on utilise alors toutes les combinaisons possibles des attributs et de leurs valeurs pour créer toutes les règles d'association possibles. Ce qui pose problème sur le plan complexité computationnelle du fait de l'explosion combinatoire. En effet, le nombre de règles générées est énorme. On peut optimiser cette méthode en gardant juste les règles avec un support et une confiance minimum. Cela reste insuffisant et les résultats sont insatisfaisants.

L'algorithme Apriori représente une approche révolutionnaire dans l'apprentissage et l'exploration des règles d'association.

Nous allons présenter deux différents algorithmes d'extraction des règles d'association : l'algorithme Apriori et l'algorithme Fp-Growth.

Ces deux algorithmes représentent une solution efficace. Le premier est très coûteux en termes d'accès à la base de transactions, le deuxième quant à lui optimise le coût d'accès. [8]

8.1 Algorithme Apriori

L'algorithme Apriori créé par Agrawal et Srikant en 1994, procède en deux temps. Il est basé sur le principe lié à l'approche de support et de confiance.

L'algorithme parcourt le treillis des itemsets pour rechercher les itemsets fréquents et en déduire les règles d'association dont la confiance dépasse le seuil de confiance min conf. Le treillis des itemsets permet d'utiliser plus efficacement cet algorithme d'extraction en admettant les propriétés suivantes :

- Tout sous-ensemble d'un Itemset fréquent est fréquent.
- Tout sur-ensemble d'un itemset non fréquent est non fréquent.

Exemple de treillis des itemsets

Le nombre d'itemsets fréquents qui peuvent être générés de n items est de 2n, la génération des Itemsets fréquents est de complexité exponentielle, il est alors essentiel de trouver la méthode de recherche la plus optimale. Ces Itemsets représentent un treillis d'Itemsets représenté sous la forme d'un diagramme de Hasse présenté à la figure 1.7.

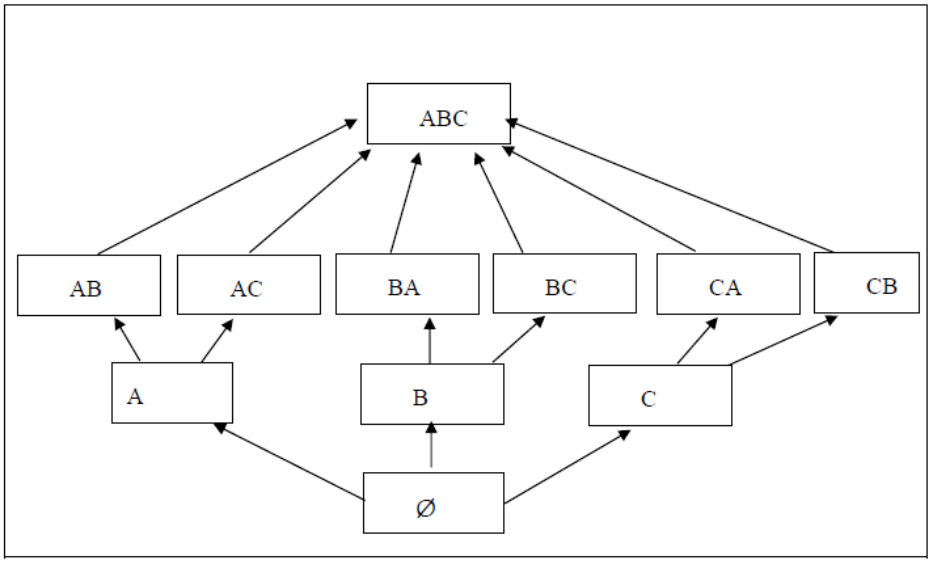

Figure 1.7. Exemple de treillis d'Itemsets ou diagramme de Hasse.

Fonctionnement de l'algorithme Apriori :

D'une manière plus concise, le déploiement de l'algorithme Apriori se fait comme suit :

1. Générer les Règles candidates.
2. Calculer le support pour chaque règle candidate.
3. Apparier les règles dont on a calculé le support avec le support choisi.
4. On rejette les candidats dont le support est inférieur au suppmin.

5. On termine en sortie avec toutes les règles dont le support est supérieur au support minimal.

En résumé le déroulement se fait comme suit :

L'algorithme Apriori fonctionne en deux phases (voir la figure 1.8). La première consiste en la recherche des ensembles d'items fréquents notée EIF et la seconde utilise ces ensembles pour trouver les règles d'association dont la confiance est supérieure à un seuil prédéfini.

Le processus de découverte des ensembles d'items fréquents est itératif, on commence par la construction de EIF avec un seul item, ensuite on réitère pour construire des EIF avec deux items. Ceci va déterminer la taille de chaque ensemble d'items fréquents qui sera noté L_n.

L'ensemble fréquent avec un seul item fréquent sera noté L_1.

Les ensembles d'items candidats sont construits à partir des ensembles d'items fréquents de taille L_{n-1} et seront notés C_n. Par exemple C_2, l'ensemble d'items fréquents candidats de taille 2, sera produit par les ensembles fréquents de taille 1.

L_n est obtenue en ne gardant que les éléments de $C_n$dont le support est supérieur au seuil. On continue les itérations jusqu'à ce que les $L_n$soient vides.

Apriori (T, t)

Calcul de L_1

N ← 2

Tant que L_{N-1} ensemble vide

  C_N ← **Apriori** (L_{N-1})

Pour chaque transaction t ∈ **T**

  C_t ← **Sous-ensemble** (L_K , t)

  Faire

Pour chaque candidate c ∈ C

  Faire

    L_N ← count ≥ $Supp_{Min}$

    N ←N+1

Retourner L_N

Figure 1.8. Algorithme Apriori**.**

Dans cet algorithme les ensembles $\boldsymbol{L_n}$et $\boldsymbol{C_n}$sont des enregistrements dans lesquels on stocke les informations et les valeurs des variables.

La procédure **count** permet de calculer et de stocker la fréquence de chaque item de la base de données. [8]

8.2 Algorithme Fp-Growth

L'algorithme Fp-growth permet la découverte des itemsets fréquents sans génération des itemsets candidats. Le processus se déroule en deux étapes, une étape de construction des arbres FP-tree et une étape d'extraction des itemsets fréquents directement de ces arbres.

La construction de l'arbre FP-tree s'effectue suivant les étapes ci-dessous :

1. Calculer le support minimal.
2. Calculer chacune des occurrences d'un item constituant la base de transactions.
3. Établir un critère de priorité pour ces items.
4. Faire le tri des items en fonction de leur priorité.
5. Établir le noeud racine.
6. À partir de chaque noeud père insérer les enfants en partant du noeud racine
7. Valider la structure de l'arbre FP-Growth. [8]

9 Conclusion

Nous venons de présenter dans ce chapitre les concepts fondamentaux de la fouille de donnée (Data-Mining), Ainsi, les concepts fondamentaux relatives aux règles d'association, comment faire pour générer ces règles et le processus d'extraction des règles d'associations.

Chapitre 2

Les règles séquentielles

Chapitre 2

Les règles séquentielles

1 Introduction

L'extraction des règles séquentielle est un domaine de recherche important dans de le data-mining. C'est une extension de l'extraction des règles d'association. L'extraction de séquences à un large éventail de situations réelles applications. Les algorithmes d'exploration séquentielle résolvent le problème de découvrir la présence de séquences fréquentes dans séquentielle base de données. La base de données consiste en un ensemble de séquences appelé en tant que séquences de données. Chaque séquence de données est une liste de clients transaction, et chaque transaction est un ensemble d'éléments. Le temps de transaction est associé à chaque transaction dans le base de données de séquences. L'extraction des règles séquentielle est presque semblable à l'extraction de règles d'association, mais la différence est que, en séquence, les événements sont liés au temps. L'extraction des règles séquentielle découvre la corrélation entre les différentes transactions, mais en cas de règle d'association il découvre la relation des objets dans la même transaction. L'extraction de règle d'association est utilisée pour découvrir différents articles fréquents rassemblés par divers clients alors l'extraction des règles séquentielle est utilisée pour découvrir les éléments sont apportés dans un ordre particulier par un seul client.

2 Les règles d'associations séquentielles

Les règles d 'associations séquentielles ont été introduites par Agrawal et Srikant en 1995 et cherchent à identifier des associations parmi toutes les séquences ayant un support minimum prédéterminé (Lu, Feng et Han 2000). Les règles d'associations séquentielles sont un prolongement des règles d'associations traditionnelles à deux niveaux. Premièrement, les bases de données utilisées par les règles d'associations séquentielles contiennent en plus de l'identifiant unique de la transaction ou du client et des articles achetés, un identifiant temporel. De ce fait, les règles d 'associations séquentielles permettent d'ordonner les items selon leur date de transaction. Ainsi, une règle séquentielle telle que définie par Agrawal et Srikant (1995) permet d'identifier une

association ayant la forme suivante : 80% des consommateurs ont acheté des souliers après avoir acheté des chemises.

Cette forme de règle st simpliste, puisqu'elle ne permet point d'identifier l'intervalle dans lequel se trouvent les deux items ou événements. Pour résoudre ce problème, d'autres approches ont été proposées (Mannila et al. 1995, Han et al. 2000, Lu et al. 1998, Lu et al. 2000) dont l'algorithme GSP (Generalized Sequential Pattern) (Srikant et Agrawal 1996), qui sera étudié en détail dans ce chapitre. Bien que les autres approches plus complexes offrent une flexibilité accrue à l'analyste, le problème central à résoudre demeure le même, c'est-à-dire découvrir les associations fréquentes parmi des séquences d'items ou d'événements. Avant d'aborder l'algorithme GSP nous nous attarderons sur la notation proposée par les auteurs pour traiter des problèmes d'associations séquentielles. [7]

2.1 Exemple d'une règle d'association séquentielle :

Considérons l'exemple suivant : Un client d'une banque qui a eu un prêt hypothécaire depuis 6 mois est prêt à souscrire à un prêt personnel à présent.

Soit A = souscrire à un prêt hypothécaire et soit B = souscrire à un prêt personnel. La séquence (AB) est différente de (BA) qui stipule qu'un client qui a eu un prêt personnel depuis 6 mois est prêt à souscrire à un prêt hypothécaire maintenant. L'ordre d'apparition des deux items AB ou BA est très important dans une règle séquentielle et l'interprétation qui en découle est totalement différente.

Dans ce qui suit nous allons reproduire les notions et notations utilisées par Agrawal et Srikant (1995) et utilisées par la suite dans les articles qui ont traité ce sujet. D'autres recherches qu'ont développé par la suite des méthodes plus flexibles pour la recherche des associations séquentielles, seront également brièvement exposées plus loin. [9]

2.2 Les paramètres associés aux règles d'associations séquentiel :

Plusieurs paramètres sont associés aux règles d'associations séquentielles (Han et Kamber 2001). Parmi ceux-ci se retrouvent la durée, la fenêtre d'événement et l'intervalle.

- **La durée** d'une séquence temporelle peut être l'ensemble des périodes couvertes par la base de données ou être une période définie par l'usager, comme par exemple les données concernant le deuxième trimestre de la base de données contenant les transactions de l'année 2002. Plus la durée est étendue, plus la complexité de l'analyse de données est

importante. La durée peut également être définie comme un ensemble de séquences fractionnées en fonction d'une échelle prédéterminée. Des exemples de ce type de durée sont les trimestres d 'une année, ou l'intervalle séparant les moments où 500 voitures sont vendues. Dans ce cas, la durée est utile dans la découverte de patterns cycliques.

- **La fenêtre d'événement** (event folding window) détermine le degré de distance événementiel requis pour que deux événements soient considérés comme faisant partie de deux transactions distinctes. Par exemple, lorsque la fenêtre d'événements est de sept jours, un événement se déroulant le lundi et un autre se déroulant le jeudi de la même semaine sont considérés comme simultanés, car ils se sont produits à l'intérieur de la même fenêtre d 'événement. Pour que deux événements soient considérés contigus, le premier événement doit se dérouler dans une fenêtre au temps t et le second dans la fenêtre au temps t+1.
- **L'intervalle**, (int) est le temps ou l'espace écoulé entre deux itemsets reliés dans une séquence-client. Si int = 0, seules les associations entre items survenus dans la même fenêtre d'événement sont examinés, comme c'est les cas dans les règles d'associations de type market basket analysis. Un usager peut également choisir un intervalle plus ou moins défini comme c'est le cas pour l'intervalle suivant : 0jour < int < 30 jours, ou un intervalle fixe pour connaître ce qui se passe après une période précise. Par exemple, int = 2 jours, identifie les événements qui se sont déroulés deux jours plus tard que l'événement initial. Certains auteurs (Lu, Feng et Han 2000) utilisent aussi le « maxspan » pour noter la borne supérieure de l 'intervalle, laissant ainsi tomber la borne inférieure qui est généralement 0. [9]

3 Recherche de règles séquentielles :

Une règle séquentielle est une règle d'association à laquelle on rajoute le facteur temps. La recherche de règles séquentielles est un processus complexe et passe par différentes étapes notamment, la recherche de motifs séquentiels, c'est pourquoi dans cette section nous commencerons par la description d'une séquence, d'une séquence fréquente et d'un motif séquentiel.

3.1 Définition d'une séquence

Une séquence est une liste ordonnée d 'itemsets ou de transactions. Une séquence s est notée ($\{s_1\}$ $\{s_2\}$ …. $\{s_n\}$) où s_j est un itemset et j indique l'ordre d'apparition de l'itemset en question.

Ainsi, l'itemset s_{j-1} précède l'itemset s_j, qui lui est suivi de l'itemset s_{j+1}.Il est à noter que l'indice j donne uniquement l'ordre d'apparition des itemsets et n'est donc pas nécessairement une mesure de temps absolu. Par exemple, la séquence formée par les quatre transactions effectuées par le client 1001 à la figure 2.1 est ({1,5} {2}{3}{4}) où s_1= {1,5}, s_2= {2}, s_3= {3} et s_4= {4} même si deux jours séparent l'itemset $s_2$et l'itemset s_3 alors que quatre jours séparent l'itemset $s_3$et s_4.

Une séquence ($\{a_1\}$ $\{a_2\}$ ... $\{a_n\}$) est contenue dans une autre ($\{b_1\}$ $\{b_2\}$ ... $\{b_m\}$) s'il existe des entier $\boldsymbol{i_1 < i_2 < \ldots < i_n}$ de tel sorte que $\boldsymbol{a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}}$ par exemple, la séquence ({3} {4,5} {8}) est contenue dans la séquence ({7} {3,8} {9} {4,5,6} {8}) car a_1 = {3} $\subseteq \boldsymbol{b_2}$ = {3,8}, a_2 = {4,5} } $\subseteq \boldsymbol{b_4}$ = {4,5,6} et a_2 = {8} $\subseteq \boldsymbol{b_5}$= {8}.Toutefois la séquence ({3}{5}) n'est pas contenue dans la séquence ({3,5}) et vice et versa car dans le premier cas, les items 3 et 5 ont été achetés l'un après l'autre, tandis que dans le cas suivant, les items 3 et 5 ont été achetés lors de la même transaction. Toutes les transactions effectuées par un même client peuvent être vues comme une séquence, où chacune des transactions est un itemset et la liste des transactions ordonnée selon une hiérarchie temporelle est la séquence. Ce type de séquence est nommé séquence-client. Un exemple d'une base de données transformée en séquence- client est illustré à la figure 2.2. Les transactions $\boldsymbol{T_1, T_2, \ldots, T_n}$ effectuées par un client sont notées ($\{\boldsymbol{T_1}\}$, $\{\boldsymbol{T_2}\}$, ... $\{\boldsymbol{T_n}\}$). Un client supporte la séquence s si et seulement si s est contenue dans la séquence-client de ce client en particulier. Le support d'une séquence représente la fraction des séquences-clients qui supportent cette séquence vis-à-vis de l'ensemble des clients de la base de données. Dans l'exemple de la figure 2.2, la séquence ({3}{4}) se retrouve parmi les séquences-clients des clients 1001, 1002 et 1003 et son support est donc de 60 % (3/5). La longueur d'une séquence se définit comme étant le nombre d'items contenu dans la séquence, et non pas le nombre d'itemsets ou de transactions. Par exemple, la séquence ({3}{4} {3,5}) est une 4-séquence, même si cette dernière contient trois itemsets, car l'item 3 est contenu dans deux transactions et est donc compté deux fois. Une séquence contenant k items est donc une k-séquence. [10]

No_client	Date de transaction	Items achetés
1001	11 janvier 2003	1, 5
	13 janvier 2003	2
	15 janvier 2003	3
	19 janvier 2003	4
1002	11 janvier 2003	1
	12 janvier 2003	3
	13 janvier 2003	4
	15 janvier 2003	3, 5
1003	11 janvier 2003	1
	12 janvier 2003	2
	14 janvier 2003	3
	15 janvier 2003	4
1004	11 janvier 2003	1
	12 janvier 2003	3
	14 janvier 2003	5
1005	11 janvier 2003	4
	15 janvier 2003	5

Figure 2.1. Base de données triée par client et par date de transaction.

No_client	Séquences-clients
1001	⟨(1,5)(2)(3)(4)⟩
1002	⟨(1)(3)(4)(3, 5)⟩
1003	⟨(1)(2)(3)(4)⟩
1004	⟨(1)(3)(5)⟩
1005	⟨(4)(5)⟩

Figure 2.2. Version séquence-client de la base de données de la figure 2.1.

3.2 Fréquence d'une séquence :

Une séquence est considérée fréquente, si le support de cette séquence respect le support minimum, en d'autres termes, si le support de cette séquence est supérieur ou égal au support minimum. Celui-ci est introduit par le client afin de mesurer la pertinence d'une séquence.

Exemple : Le MinSupp est fixé à 66% (c'est-`a-dire deux clients minimums, sur les trois, doivent supporter la séquence). Du Tableau 2.1 Un exemple de base de données contenant 4 transactions, en considérant l'item « pain », il est supporté par les clients client1 et client3.

Clients	Dates	Itemsets
Client1	02/03/2008 03/04/2008	Pain, TV beurre
Client2	10/02/2008 11/02/2008	Lecteur DVD Dattes, pain
Client3	12/02/2008 13/02/2008	Pain Beurre

Tableau 2.1 : Une de base de données contenant 4 transactions.

Remarque : Pour chaque client, le support d'une séquence ne peut être incrémenté que d'une unité, même si la séquence apparaît plusieurs fois.

3.3 Séquences fréquentes maximales ou motifs séquentiels :

Une fois toutes les séquences fréquentes trouvées, on procède à la recherche de celles qui ont une fréquence maximale, c'est-à-dire celles qui ne sont incluses dans aucune autre séquence, on les appellera des motifs séquentiels.

Les motifs séquentiels peuvent être vus comme une extension de la notion de règles d'association, intégrant diverses contraintes temporelles. Aussi, la recherche de tels motifs consiste à extraire des ensembles d'items, couramment associés sur une période de temps bien spécifiée. En fait, cette recherche met en évidence des associations inter-transactions, contrairement à celle des règles d'association qui extrait des combinaisons intra-transaction. Dans ce contexte, et contrairement aux règles d'association, l'identification des individus ou objets est indispensable, afin de pouvoir suivre leur comportement au cours du temps.

On trouve dans la littérature actuelle de nombreux travaux sur l'extraction des motifs séquentiels à partir de grandes quantités de données stockées dans des bases de données ou des fchiers .La découverte de ces relations temporelles a été largement utilisée dans des domaines variés comme le marketing sélectif, l'aide à la décision, le management, la bio-informatique, l'analyse des performances des systèmes, l'analyse des réseaux de télé- communication etc. Un secteur important d'applications pour l'extraction de relations temporelles est l'analyse du panier de la ménagère, qui étudie les comportements des clients en cherchant des séries d'articles qui sont fréquemment achetés dans un intervalle de temps donné. [7]

3.4 Exemples d'applications réelles possibles

3.4.1 Exemples 1

L'historique des visites des utilisateurs Web est gardé dans les fichiers Log. Le fait que chaque transaction gardée dans le fichier Log soit associée à une estampille temporelle ouvre largement la porte aux chercheurs de méthodes d'extraction de motifs séquentiels et les avantages sont nombreux : la possibilité de prédire les comportements des utilisateurs, de proposer aux utilisateurs des liens appropriés, etc.

Un système de fouille de Web peut découvrir des comportements comme :

10% des personnes qui ont visité "/company/toys" ont fait dans un délai d'une semaine une recherche google d'après le mot "Disney" ;

15% des personnes qui ont acheté le livre ”/amazon/Theorie_de_la_relativite” ont acheté dans les 10 jours d'après les livres ”/amazon/Bibliographie_de_Einstein” et ”/amazon/Bibliographie_de_Stephen_Hawking”.

On peut aussi être intéressé par des inter-corrélations entre des données dans un certain intervalle de temps. Par exemple, on pourrait savoir :

Quelles caractéristiques communes ont les personnes qui ont continué d'acheter des voitures ou d'investir en bourse dans un intervalle d'une semaine après l'annonce de la crise mondiale ;

Les points communs des personnes qui achètent un jour avant les soldes.

3.4.2 Exemples 2

Des motifs séquentiels peuvent montrer que :

”60% des gens qui achètent une télévision achètent un magnétoscope plus tard”.

”75%des gens qui achètent du pain achètent du beurre ”.

Ce problème, posé à l’origine dans un contexte de marketing, intéresse à présent des domaines aussi variés que les télécommunications (détection de fraudes), la finance, ou encore la médecine (identification des symptômes précédant les maladies), ou la biologie (reconstitution de séquences d’ADN manquantes). [11]

3.5 Extraction des motifs séquentiels

L’extraction des motifs séquentiels est un problème difficile, si plusieurs techniques algorithmiques ont permis l’extraction efficace des ensembles fréquents dans les données

transactionnelles, ces résultats ne se transposent pas facilement à l'extraction des motifs séquentiels. L'ordre des éléments doit être conservé dans une séquence, ce qui demande des efforts supplémentaires par rapport aux ensembles fréquents.

Face à cette problématique, plusieurs auteurs ont proposé des structures de données efficaces mais, la complexité intrinsèque reste non polynomiale.

La problématique de l'extraction des motifs séquentiels dans une base de données, est une sorte d'extension de celle des règles d'association. Une notion est ajoutée par rapport aux règles d'association qui est la prise en compte de la temporalité dans les enregistrements. Cela permet une plus grande précision dans les résultats, mais implique aussi une plus grande difficulté d'implémentation.

La prise en compte de la temporalité permet d'organiser les éléments traités avec une notion d'ordre (chronologique).

La recherche de motifs séquentiels se base sur un format de données bien précis, qui relate des événements liés à différents acteurs. L'exemple du supermarché est le plus utilisé pour illustrer ce concept. Cela est, en partie, dû au fait que ce sont des besoins de type marketing qui ont apporté cette problématique.

Ce format de données est illustré par le Tableau 2.2 Format de données utilisé ci-dessous, où l'on peut distinguer les achats de trois clients. Le but d'un algorithme d'extraction de motifs séquentiels sera alors de trouver des comportements fréquents, dans les achats des clients. Le résultat attendu est donc une ou plusieurs séquences d'achats, représentant un comportement récurrent.

Nous pouvons alors définir le comportement fréquent comme étant un comportement respecté par au moins n clients (n étant considéré comme un minimum de clients qui respectent ce comportement afin que celui-ci soit estimé fréquent).

Client	01/04/2008	01/05/2008
Client1	Pain, beurre	huile
Client2	Pc	Clé USB
Client3	Tv	Lecteur DVD

Tableau 2.2 : Format de données utilisé.

3.6 Propriétés des séquences fréquentes

Il existe trois propriétés principales qui sont des éléments déterminants pour l'extraction. Il faut noter que toutes ces propriétés sont déjà appliquées sur les règles d'association.

- **L'anti-monotonie :** L'anti-monotonie signifie que si une séquence quelconque S viole une contrainte C, alors toute sur-séquence S' de S la viole aussi. L'exemple le plus connu de contrainte anti monotone est la contrainte de fréquence minimale, si une séquence n'est pas fréquente, aucune de ces sur-séquences ne le sera. Ce type de contrainte permet un élagage dans l'espace de recherche et la diminution du nombre de séquences à considérer Les contraintes succinctes permettent d'énumérer les candidats sans nécessairement explorer tout l'espace de recherche en générant toutes les séquences possibles.
- **La monotonie :** La monotonie est une propriété qui stipule que si une sous-séquence S' d'une séquence quelconque S satisfait une contrainte C, alors S le satisfait aussi. Ce type de propriété permet de génération de candidats efficace.
- **Support des sous-séquences :** Le support de S est supérieur ou égal au support de S' « (supp(S)>=supp(S') » car, toutes les transactions dans la base de données qui supportent S' supportent aussi nécessairement S.

3.7 Support d'une séquence :

Le support d'une séquence quelconque S est le pourcentage de clients qui supportent cette séquence S. c'est une mesure dite d'utilité.

Supp({ae} → {bc}) = supp({ae} U {bc})

3.8 Confiance d'une règle :

La confiance d'une règle est une mesure dite de précision, c'est la probabilité qu'on achète un certain nombre d'articles A sachant qu'on a déjà acheté B, soit la probabilité conditionnelle : p(A/B).

Conf ({ae → {/bc}) = supp({abce}) / supp({ae})

On voit immédiatement que la confiance se traduit par un rapport de support. La notion de confiance serra bien détaillée dans la partie mise en œuvre.

Il existe d'autres critères d'évaluation utilisant différentes formules comme, L'Intérêt, la Conviction, et le Cosinus.

4 Applications de l'extraction de règles séquentielle

Quelques exemples d'applications de l'extraction de règles séquentielle sont :

- E-learning
- La simulation de fabrication,
- Le contrôle de qualité,
- La pré-lecture de pages web,
- La détection anti-pattern dans les systèmes basés sur les services,
- Les systèmes embarqués,
- L'analyse de séquence d'alarme.

Par exemple, voici quelques articles décrivant de telles applications [12]

5 Algorithme d'extraction des règles séquentielle

Il existe une grande diversité d'algorithmes pour l'extraction séquentielle.

Dans cette partie, nous introduisons d'abord quelques algorithmes généraux et de base pour l'extraction des règles séquentielle, extensions de ces algorithmes à des fins spéciales, telles que l'exploitation séquentielle multidimensionnelle et l'exploitation séquentielle incrémentielle sont abordées plus loin.

L'extraction de motifs périodique est également élaborée comme une extension du modèle séquentiel exploitation minière.

5.1 L'algorithme de CMRules :

Sur la base des observations précédentes de la relation entre les règles séquentielles et les règles d'association, nous proposons l'algorithme CMRules pour l'extraction des règles séquentielles. L'algorithme est présenté à la figure 2.3.

La figure 2.4 montre un exemple d'exécution avec une base de données de séquences contenant quatre séquences.

L'algorithme prend en entrée une base de données de séquences et les seuils $minSeqSup$ et $minSeqConf$ et délivre l'ensemble de toutes les règles séquentielles dans la base de données en

respectant ces seuils. CMRules commence en ignorant les informations séquentielles de la base de données de séquences pour obtenir une base de données de transaction (figures 2.3 et 2.4. Étape 1). Il applique ensuite un algorithme d'exploration de règles d'association pour découvrir toutes les règles d'association de cette base de données de transaction avec $minsup = minSeqSup$ et $minconf = minSeqConf$ (figures 2.3 et 2.4 Étape 2). L'algorithme calcule ensuite le support séquentiel et la confiance séquentielle de chaque règle d'association en analysant la base de données de séquences, puis élimine les règles qui ne respectent pas les seuils minimum $minSeqSup$ et $minSeqConf$ (figures 2.3 et 2.4 Étape 3). L'ensemble de règles conservé est l'ensemble de toutes les règles séquentielles. [13]

INPUT: a sequence database, ***minSeqSup, minSeqConf***
OUTPUT: the set of all sequential rules
PROCEDURE:

1. Consider the sequence database as a transaction database.
2. Find all association rules from the transaction database by applying an association rule mining algorithm such as Apriori [1]. Select *minsup* = *minSeqSup* and *minconf* = *minSeqConf*.
3. Scan the original sequence database to calculate the sequential support and sequential confidence of each association rule found in the previous step. Eliminate each rule *r* such that:
   a. seqSup(r) < *minSeqSup*
   b. seqConf(r) < *minSeqConf*
4. Return the set of rules.

Figure 2.3. L'algorithme CMRules**.**

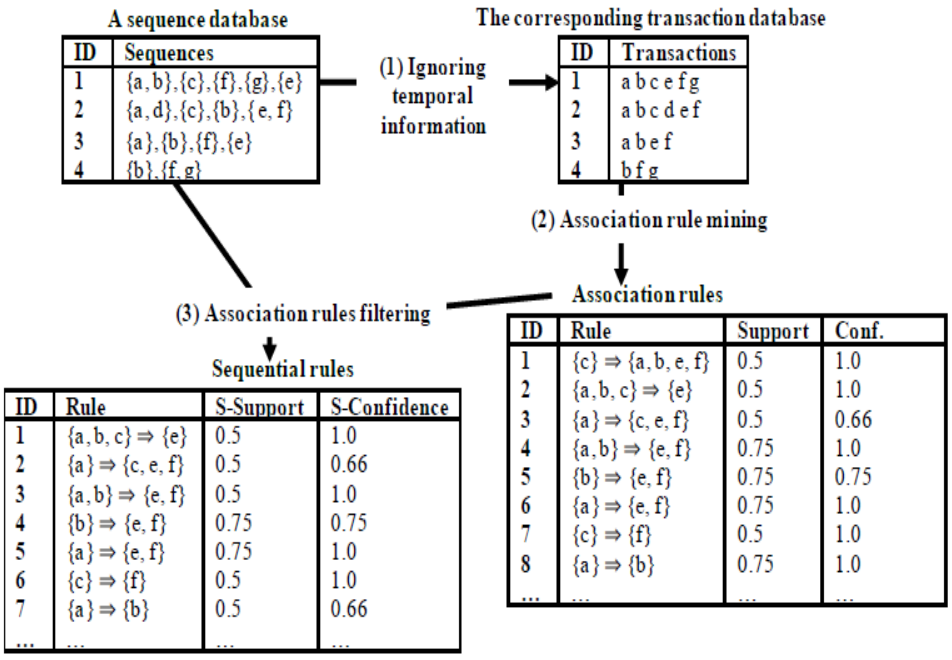

Figure 2.4. Exemple d'exécution de l'algorithme CMRules**.**

5.2 L'algorithme de CMDeo :

Bien que l'algorithme Deogun-et-al. A été proposé pour l'extraction de règles séquentielles dans une seule séquence d'événements, il est facile de l'adapter pour les règles d'exploration de données communes à plusieurs séquences, comme défini dans cet article. Nous nommons cette adaptation CMDeo et l'avons implémentée en Java. Puisque ce n'est pas le sujet principal de cet article, nous allons décrire brièvement l'idée principale de cet algorithme. Avant cette explication, nous donnons deux définitions. Nous disons que la taille d'une règle séquentielle est $k * n$, si la partie gauche et la partie droite contiennent respectivement k et n éléments. Par exemple, les règles $\{a, b, c\} \Rightarrow \{e, f, g\}$ et $\{a\} \Rightarrow \{e, f\}$ sont de taille $3 * 3$ et $1 * 2$, respectivement. De plus, une règle de taille $f * g$ est dite plus grande qu'une autre règle de taille $h * i$ si $f > h$ et $g \geq i$, ou bien si $f \geq h$ et $g > i$.

L'algorithme CMDeo fonctionne comme suit. Il trouve d'abord toutes les règles valides de taille $1 * 1$. Ceci est fait en trois étapes. Premièrement, le CMDeo compte la prise en charge des éléments uniques en analysant une fois la base de données de séquences. Ensuite, pour chaque paire d'éléments fréquents x, y apparaissant dans au moins les mêmes séquences $minSeqSup$, l'algorithme génère des règles candidates $\{x\} \Rightarrow \{y\}$ et $\{y\} \Rightarrow \{x\}$. Ensuite, le support séquentiel et

la confiance séquentielle de chaque règle sont calculés comme à l'étape 3 de CMRules. Chaque règle respectant $minSeqSup$ et $minSeqConf$ est notée comme étant une règle valide de taille 1*1.

Ensuite, CMDeo découvre des règles valides plus importantes par niveau (similaire à Apriori [1]) en appliquant récursivement deux processus :

a) expansion gauche.

b) expansion droite.

Leur rôle est de combiner une paire de règles de la même taille pour obtenir de plus grandes règles candidates.

L'expansion côté gauche est le processus consistant à prendre deux règles $X \Rightarrow Y$ et $Z \Rightarrow Y$, où X et Z sont des ensembles d'items de taille n partageant $n-1$ éléments, afin de générer une règle plus grande $X \cup Z \Rightarrow Y$. L'expansion du côté droit est le processus consistant à prendre deux règles $Y \Rightarrow X$ et $Y \Rightarrow Z$, où X et Z sont des itemsets de taille n partageant $n-1$ éléments, pour générer une règle plus grande $Y \Rightarrow X \cup Z$. Ces deux processus peuvent être appliqués récursivement pour trouver toutes les règles à partir de la règle de taille $1*1$ (par exemple, les règles de taille $1*1$ permettent de trouver des règles de taille $1*2$ et de règle de taille $2*1$). [13]

5.3 L'algorithme de RuleGrowth (règle croissance) :

Nous présentons RuleGrowth, un nouvel algorithme que nous avons conçu pour extraire séquentielle partiellement ordonnée règles plus efficacement. [14]

5.3.1 Caractéristiques principales :

Une caractéristique commune de CMRules et CMDeo est que les deux utilisent une approche génératrice de candidats et de tests, qui consiste à générer des règles de candidats, puis à numériser la base de données pour déterminer leur soutien et leur confiance.

Le problème avec cette approche est qu'elle produit souvent une grande quantité de règles candidates et qu'une grande proportion sont invalides ou n'apparaissent pas dans la base de données. Donc, ces algorithmes passent beaucoup de temps à distinguer règles valides des invalides.

L'algorithme RuleGrowth que nous proposons dans cet article évite ce problème de génération de candidats à la place s'appuyant sur une approche de croissance de modèle en partie inspirée par celui utilisé dans l'algorithme PrefixSpan pour l'extraction séquentielle. RuleGrowth trouve d'abord des règles de taille $1*1$ puis les augmente récursivement en numériser les séquences les contenant pour trouver unique élément qui peuvent étendre leur côté gauche ou droit. Cette stratégie veille à

ce que seules les règles figurant dans la base de considérées comme des règles valides potentielles par l'algorithme. Nous nommer les deux processus d'expansion des règles dans RuleGrowth a quitté l'expansion et l'expansion droite s'apparente aux processus homonymes de CMDeo. Notez cependant que ces les processus sont différents ; RuleGrowth trouve des règles plus importantes en ajouter des éléments aux règles en analysant des séquences contenantes les règles (une recherche en profondeur), alors que CMDeo combine paires de règles pour générer des candidats (une rechercher largeur). Une autre caractéristique de RuleGrowth est qu'il garde la trace de la première et dernière occurrence de chaque élément, ainsi que des antécédents et conséquents à éviter balayage des séquences complètement, comme il sera expliqué.

5.3.2 Définitions et propriétés :

Avant de présenter l'algorithme, nous introduisons définitions et propriétés. Une expansion à gauche est le processus d'ajouter un élément i à gauche d'une règle X⇒Y pour obtenir une règle plus grande X∪ {i} ⇒Y. Une expansion correcte est définie comme processus d'ajout d'un élément i sur le côté droit d'une règle X⇒Y pour obtenir une règle plus grande X⇒Y∪ {i}. Expansions gauche et droite avoir les quatre propriétés importantes suivantes.

5.3.3 L'algorithme

Nous présentons maintenant l'algorithme RuleGrowth. Sa procédure principale est nommée RULEGROWTH et est représentée sur la Figure 2.5. Elle prend comme paramètres une base de données de séquences et les seuils $minsup$ et $minconf$. Cette procédure génère d'abord toutes les règles r de taille $1 * 1$ telles que $sup\,(r) \geq minsup$ puis appelle deux procédures récursives pour développer chaque règle. Ces procédures, nommées $EXPANDLEFT$ et $EXPANDRIGHT$.

RuleGrowth(S, *minsup, minconf*)
1. Scan the database one time. For each item *c* found, record the *sids* of the sequences that contains *c* in a variable *sids(c)*.
2. Scan the database S a second time and remove each item *c* such that $|sids(c)| / |S| \leq minsup$.
3. FOR each pair of items *i, j* :
4. $sids(i \Rightarrow j) := \emptyset$. $sids(j \Rightarrow i) := \emptyset$.
5. FOR each sid $s \in (sids(i) \cap sids(j))$
6. IF *i* occurs before *j* in *s*, $sids(i \Rightarrow j) := sids(i \Rightarrow j) \cup \{s\}$.
7. IF *j* occurs before *i* in *s*, $sids(j \Rightarrow i) := sids(j \Rightarrow i) \cup \{s\}$.
8. IF $(|sids(i \Rightarrow j)| / |S|) \geq minsup$ THEN
9. **EXPANDLEFT**($\{i\} \Rightarrow \{j\}$, *sids(i)*, $sids(i \Rightarrow j)$).
10. **EXPANDRIGHT**($\{i\} \Rightarrow \{j\}$,*sids(i)*, *sids(j)*, $sids(i \Rightarrow j)$).
11. IF $(|sids(i \Rightarrow j)| / |sids(i)|) \geq minconf$ THEN OUTPUT rule $\{i\} \Rightarrow \{j\}$ with its conf. and support.
12. IF $(|sids(j \Rightarrow i)| / |S|) \geq minsup$ THEN
13. **EXPANDLEFT**($\{j\} \Rightarrow \{i\}$, *sids(j)*, $sids(j \Rightarrow i)$).
14. **EXPANDRIGHT**($\{j\} \Rightarrow \{i\}$,*sids(j)*, *sids(i)*, $sids(j \Rightarrow i)$).
15. IF $(|sids(j \Rightarrow i)| / |sids(j)|) \geq minconf$ THEN OUTPUT rule $\{j\} \Rightarrow \{i\}$ with its conf. and support.

Figure 2.5. L'algorithme de RuleGrowth.

6 Conclusion

Dans Ce chapitre, Nous venons présenter les concepts fondamentaux relatives aux règles séquentielles, comment faire pour générer ces règles, et les algorithme existant pour l'extraction de ces réglé.

Chapitre 3

Etude sur ERMiner

Chapitre 3

Etude sur ERMiner

1 Introduction :

ERMiner est un algorithme d'extraction de règles séquentielles partiellement ordonnées développé par Philippe Fournier-Viger et al. ERMiner utilise une notion proche des classes d'équivalence pour garder une trace des règles qui doivent être inspectées plus avant. ERMiner est différent de l'ancien algorithme de Fournier, RuleGrowth, en ce qu'il ne se développe pas récursivement les côtés gauche ou droit des règles, mais procède plutôt par des fusions à gauche et à droite.

Pour comprendre comment fonctionne ERMiner, il est nécessaire de comprendre la notion de fusion à gauche et à droite. Cependant, avant cela, nous devons couvrir certains concepts de base et la notation.

2 Définition du problem

2.1 Définition 1 (base de données de séquences)

Soit $I = \{i_1, i_2, \dots, i_i\}$ un ensemble d'items (symboles). Un $itemset\ I_x = \{i_1, i_2, \dots, i_m\} \subseteq I$ est un ensemble non ordonné d'items. L'ordre l$exicographique \succ Lex$ est défini comme n'importe quel ordre total sur I. Sans pour autant perte de généralité, il est supposé dans ce qui suit que tous les itemsets sont commandés selon $\succ Lex$. Une séquence est une liste ordonnée d'itemset $s = [I_1, I_2, \dots, I_n]$ tel que $I_k \subseteq I (1 \leq k \leq n)$. Une base de données de séquences *SDB* est une liste de séquences $SDB = [s_1, s_2, \dots, s_p]$ ayant des identifiants de séquence (*SIDs*) 1,2 ... p.

Exemple 1. Une base de données de séquences est représentée sur la figure 3.1 (à gauche). Il contient quatre séquences ayant les SIDs 1, 2, 3 et 4. Chaque lettre représente un item. Items entre accolades représentent un itemset. La première séquence $[\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}]$contient cinq itemsets. Il indique que les items *a* et *b* sont apparus en même temps, suivis de *c*, puis *f* et enfin *e*.

ID	Sequences
seq1	⟨{*a*, *b*},{*c*},{*f*},{*g*},{*e*}⟩
seq2	⟨{*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*}⟩
seq3	⟨{*a*},{*b*},{*f*},{*e*}⟩
seq4	⟨{*b*},{*f*, *g*, *h*}⟩

ID	Rule	Support	Confidence
r1	{*a*, *b*, *c*}→{*e*}	0.5	1.0
r2	{*a*}→{*c*, *e*, *f*}	0.5	0.66
r3	{*a*, *b*}→{*e*, *f*}	0.75	1.0
r4	{*b*}→{*e*, *f*}	0.75	0.75
r5	{*a*}→{*e*, *f*}	0.75	1.0
r6	{*c*}→{*f*}	0.5	1.0
r7	{*a*}→{*b*}	0.5	0.66

Figure 3.1. Une base de données de séquences (à gauche) et quelques règles séquentielles (à droite)

2.2 Définition 2 (règle séquentielle)

Une règle séquentielle $X \rightarrow Y$ est une relation entre deux ensembles d'items non ordonnés $X, Y \subseteq I$ tel que $X \cap Y = \emptyset$ et $X, Y \neq \emptyset$.

L'interprétation d'une règle $X \rightarrow Y$ est que si les éléments de *X* se produisent dans une séquence, les items de *Y* se produiront ensuite dans la même séquence.

2.3 Définition 3 (occurrence d'itemset/ règles)

Soit $s: [I_1, I_2, \dots, I_n]$ une séquence.

Un itemset I se produit ou est contenu dans s (écrit comme $I \sqsubseteq s$) si $I \subseteq \bigcup_{i=1}^{n} I_i$.

Une règle $r: X \rightarrow Y$ se produit ou est contenu dans s (écrit comme $r \sqsubseteq s$) s'il existe un nombre entier k tel que $1 \leq k < n, X \subseteq \bigcup_{i=1}^{k} I_i$ et $Y \subseteq \bigcup_{i=k+1}^{n} I_i$.

Exemple 2. 'itemset $\{a, b, f\}$ est contenu dans la séquence $[\{a\}, \{b\}, \{f\}, \{e\}]$.

La règle $\{a, b, f\} \rightarrow \{e, f, g\}$ se produit dans $[\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}]$, alors que le règle $\{a, b, f\} \rightarrow \{c\}$ ne le fait pas, car item c ne se produit pas après f.

2.4 Définition 4 (taille d'une règle séquentielle)

Une règle $X \rightarrow Y$ est dit être de taille *k*m* si $|X| = k$ et $|Y| = m$. De plus, une règle de taille *f*g* est dit être plus grand qu'une autre règle de taille *h*j* si $f > h$ et $g \geq i$, ou alternativement si $f \geq h$ et $g > i$.

Exemple 3. Les règles $r: \{a, b, f\} \rightarrow \{e, f, g\}$ et $s: \{a\} \rightarrow \{e, f\}$ sont respectivement de taille *3*3* et *1*2*. Ainsi, *r* est plus grand que *s*.

2.5 Définition 5 (support)

Le support d'une règle *r* dans une base de données de séquences *SDB* est défini comme :

$$supSDB\ (r) = \frac{|\{s | s\ \epsilon\ SDB \wedge r \sqsubseteq s\}|}{|SDB|}.$$

2.6 Définition 6 (confiance)

La confiance d'une règle $r: X \rightarrow Y$ dans une base de données de séquence *SDB* est définie comme : $confSDB\ (r) = \frac{|\{s|s\ \epsilon\ SDB \wedge r \sqsubseteq s\}|}{|s|s \epsilon SDB \wedge X \sqsubseteq s|}$

2.7 Définition 7 (extraction d'une règle séquentielle)

Laissez $minsup, minconf\ \epsilon[0,1]$ être seuils définis par l'utilisateur et *SDB* être une base de données de séquences. Une règle r est une règle séquentielle fréquente si $supSDB\ (r) \geq\ minsup$. Une règle r est une règle séquentielle valide s'il est fréquent et $confSDB\ (r) \geq\ minconf$. Le problème de l'extraction de règles séquentielles à partir d'une base de données de séquences est de découvrir toutes les règles séquentielles valides [6].

Exemple 4. La figure 1 (à droite) montre 7 règles valides trouvées dans la base de données illustrée. Tableau 1 pour $minsup = 0.5$ et $minconf = 0.5$. Par exemple, la règle $\{a, b, c\} \rightarrow \{e\}$ a un support de $2/4 = 0.5$ et une confiance de $2/2 = 1$. Parce que ces valeurs sont respectivement non moins de $minsup$ et $minconf$, la règle est réputée valide. [15]

3 L'algorithme ERMiner

Dans cette section, nous présentons l'algorithme ERMiner avec des exemples pour une bonne compréhension de l'algorithme. Il s'appuie sur le nouveau concept des classes d'équivalence de règles séquentielles, définies comme suit.

3.1 Définition 8 (classes d'équivalence de règle).

Pour une base de données d'ordre, laissez R soyez l'ensemble de toutes les règles séquentielles fréquentes et I soit l'ensemble de tous les articles. Une gauche classe d'équivalence $LE_{w,i}$ est l'ensemble derègles fréquentes

$LE_{w,i} = \{W \rightarrow Y \,|\, Y \subseteq I \wedge |Y| = i\}$ tels que $W \subseteq I$ et I est un nombre entier.

De même, une droite classe d'équivalence $RE_{w,i}$ est l'ensemble de règles fréquentes

$RE_{w,i} = \{X \rightarrow W \,|\, X \subseteq I \wedge |X| = i\}$, où $W \subseteq I$, et i est un nombre entier.

Exemple 5 : Pour $minsup = 2$ et notre exemple courant,

$LE_{\{c\},1} = \{\{c\} \rightarrow \{f\}, \{c\} \rightarrow \{e\}\}$,

$RE_{\{e,f\},1} = \{\{a\} \rightarrow \{e,f\}, \{b\} \rightarrow \{e,f\}, \{c\} \rightarrow \{e,f\}\}$ et

$RE_{\{e,f\},2} = \{\{a,b\} \rightarrow \{e,f\}, \{a,c\} \rightarrow \{e,f\}, \{b,c\} \rightarrow \{e,f\}\}$.

Deux opérations appelées fusions à gauche et à droite (left and right merges) sont employées par ERMiner pour les explorer l'espace de recherche des règles séquentielles fréquentes.

Elle laisse se produire directement une classe d'équivalence utilisant une plus petite classe d'équivalence.

3.2 Définition 9 (Fusions gauche/droite).

Let soit une classe d'équivalence gauche $LE_{\{w\},i}$ et deux règles $r: W \to X$ et $s: W \to Y$ tels que $r, s \in LE_{\{w\},i}$ et $|X \cap Y| = |X - 1|$, c-à-d, X et Y sont identique excepté pour un item simple. Une fusion gauche de r, s est le processus de fusionner r, s pour obtenir $W \to X \cup Y$. De même, laissez être une droite classe d'équivalence $RE_{\{x\},i}$ et deux règles $r: X \to W$ et $r: Y \to W$ tels que $r, s \in RE_{\{w\},i}$ et $|X \cap Y| = |X - 1|$.

Une droite fusion de r, s est le processus de fusionnement de r, s pour obtenir la règle $X \cup Y \to W$.

Propriété 1 (générant une classe d'équivalence gauche). Soit une classe d'équivalence $LE_{w,i}$. $LE_{w,i+1}$ peut être obtenu en effectuant toutes les fusions à gauche sur les paires des règles de $LE_{w,i}$.

Preuve. Soit une règle $r: W \to \{a_1, a_2 \ldots, a_{i+1}\}$ dans $LE_{w,i+1}$.

Par définition 8, règles $r: W \to \{a_1, a_2 \ldots, a_{i-1}, a_i\}$ et $r: W \to \{a_1, a_2 \ldots, a_{i-1}, a_{i+1}\}$ sont membres de $LE_{w,i}$ et une fusion à gauche de ces règles générera r.

Propriété 2 (générant une classe d'équivalence droite). Soit une classe d'équivalence droite $RE_{w,i}$.$RE_{w,i+1}$ peut être obtenu en effectuant toutes les bonnes fusions sur paires de règles de $RE_{w,i}$.

Preuve. La preuve est similaire à la propriété 1 et est donc omis.

Pour explorer l'espace de recherche des règles séquentielles fréquentes en utilisant la fusion ci-dessus, ERMiner analyse d'abord la base de données pour construire toutes les classes d'équivalence pour les règles fréquentes de taille $1 * 1$. Ensuite, il effectue de manière récursive les fusions gauche / droite à partir de ces classes d'équivalence pour générer les autres classes d'équivalence. Pour s'assurer qu'aucune règle n'est générée deux fois, les idées suivantes ont été utilisées.

Tout d'abord, une observation importante est qu'une règle peut être obtenue par déférente combinaisons de fusion à gauche et à droite. Par exemple, considérez la règle $\{a, b\} \rightarrow \{c, d\}$. Il peut être obtenu en effectuant des fusions à gauche pour $LE_{\{a\},1}$ et $LE_{\{b\},1}$ suivi par des fusions à droite sur $RE_{\{c,d\},1}$. Mais il peut également être obtenu en effectuant right fusionne sur $RE_{\{c\},1}$ et $RE_{\{d\},1}$ suivi des fusions à gauche en utilisant $LE_{\{a,b\},1}$. Une solution simple pour éviter ce problème est de ne pas permettre d'effectuer une fusion à gauche après une fusion à droite mais pour autoriser une fusion à droite après une fusion à gauche. Cette solution est illustrée sur la figure 3.2.

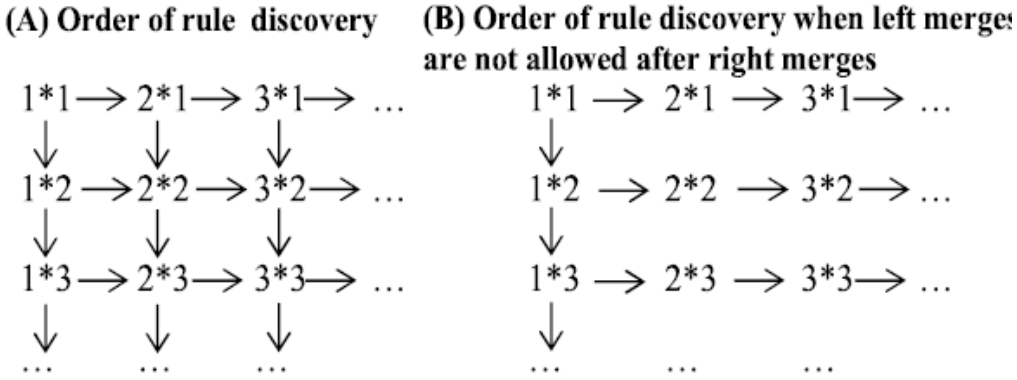

Figure 3.2. L'ordre de découverte des règles par les opérations de fusion gauche / droite.

Deuxièmement, une autre observation clé est qu'une même règle peut être obtenue par fusionner déférentes paires de règles de la même classe d'équivalence. Par exemple, une règle $\{a, b, c\} \rightarrow \{e\}$ peut être obtenu en effectuant une fusion à droite de $\{a, b\} \rightarrow \{e\}$ avec $\{a, c\} \rightarrow \{e\}$ ou avec $\{b, c\} \rightarrow \{e\}$. Pour éviter de générer deux fois la même règle, une solution simple consiste à imposer un ordre total sur les éléments des antécédents (conséquents) et pour effectuer uniquement une fusion à gauche (fusion à droite) si la règle (règle antécédent) partage tous sauf le dernier article en fonction de la commande totale. Dans l'exemple précédent, cela signifie $\{a, c\} \rightarrow \{e\}$ ne serait pas fusionné avec

$\{b, c\} \rightarrow \{e\}$.

En utilisant les solutions ci-dessus, il est facile de voir que toutes les règles sont générées juste une fois. Cependant, pour être efficace, un algorithme d'exploration de règles séquentiel devrait être capable d'élaguer l'espace de recherche. Ceci est fait en utilisant les propriétés suivantes pour fusionner les opérations.

Propriété 3 (antimonotonicité avec des fusions gauche / droite). Soit une séquence base de données SDB et deux règles fréquentes r, s. Soit t une règle obtenue par une fusion gauche ou droite de r, s . Le support de t est inférieur ou égal au support de r et celui de s.

Preuve. Puisque t contient exactement un article de plus que r et s, il peut n'apparaissent que dans les mêmes séquences numériques ou moins.

Propriété 4 (taille). Si le support d'une règle est inférieur à $minsup$, alors il ne devrait pas être fusionné avec d'autres règles car toutes ces règles sont infréquentées.

Preuve. Cela vient directement de la propriété 3.

Car il n'existe pas de propriétés d'élagage similaires pour la confiance, il est nécessaire d'explorer l'espace de recherche des règles fréquentes pour obtenir les règles valides.

La figure 3.3 montre le pseudo-code principal d'ERMiner, qui intègre toutes les idées vicieuses. ERMiner prend en entrée une base de données de séquences SDB, et le $minsup$ et les seuils $mincon\ f$. Il scanne d'abord la base de données pour construire toute l'équivalence classes de règles de taille 1*1, c'est-à-dire contenant un seul objet dans l'antécédent et un seul article dans le conséquent. Ensuite, pour découvrir des règles plus grandes, les fusions à gauche sont effectuée avec toutes les classes d'équivalence de gauche en appelant la procédure $leftSearch$.

De même, les fusions à droite sont effectuées pour toutes les classes d'équivalence de droite en appelant la procédure $rightSearch$. Notez que la procédure $rightSearch$ peut générer de nouvelles classes d'équivalence gauche parce que les fusions à gauche sont autorisées après le droit fusionne. Ces classes d'équivalence sont stockées dans une structure nommée $leftSearch$. À traiter ces classes d'équivalence, une boucle supplémentaire est effectuée. Finalement, L'algorithme renvoie l'ensemble des règles trouvées.

Algorithm 1: The ERMiner algorithm

input : SDB: a sequence database, $minsup$ and $minconf$: the two user-specified thresholds
output: the set of valid sequential rules

```
1  leftStore ← ∅ ;
2  rules ← ∅ ;
3  Scan SDB once to calculate EQ, the set of all equivalence classes of rules of
   size 1*1;
4  foreach left equivalence class H ∈ EQ do
5  |  leftSearch (H, rules);
6  end
7  foreach right equivalence class J ∈ EQ do
8  |  rightSearch (J, rules, leftStore);
9  end
10 foreach left equivalence class K ∈ leftStore do
11 |  leftSearch (K, rules);
12 end
13 return rules;
```

Figure 3.3. L'algorithme de ERMiner.

La figure 3.4 montre le pseudo-code de la procédure leftSearch. Il prend comme paramètre une classe d'équivalence LE. Ensuite, pour chaque règle r de cette classe d'équivalence, une gauche la fusion est effectuée avec toutes les autres règles pour générer une nouvelle classe d'équivalence.

Seules les règles fréquentes sont conservées. De plus, si une règle est valide, elle est sortie. Alors, leftSearch est appelé récursivement pour explorer chaque nouvelle classe d'équivalence générée de cette façon. Le rightSearch (voir Figure 3.5) est similaire. La déférence principale est cette nouvelle les équivalences à gauche sont stockées dans la structure du magasin de gauche parce que leur exploration est retardée, comme expliqué précédemment dans la procédure principale d'ERMiner.

Maintenant, il est important d'expliquer comment le soutien et la confiance de chaque règle est calculé par ERMiner (nous avions précédemment délibérément omis cette explication nation). En raison de la limitation de l'espace et parce que ce calcul est fait de la même manière Comme dans l'algorithme RuleGrowth [4], nous ne donnons ici que l'idée principale. Initialement, une analyse de base de données est effectuée pour enregistrer les premières et dernières occurrences de chaque élément dans chaque séquence où il apparaît. Par la suite, le support de chaque règle de taille 1 * 1 est directement généré en comparant les premières et dernières occurrences, avec sur l'analyse de

la base de données. De même, les premières et dernières occurrences de chaque règle antécédent et conséquent sont mis à jour pour les règles plus grandes sans scanner la base de données. Cela permet de calculer efficacement la confiance et le support (voir [4] pour plus de détails sur la façon dont ce calcul peut être fait).

Algorithm 2: The leftSearch procedure

input : LE: a left equivalence class, *rules*: the set of valid rules found until now, *minsup* and $minconf$: the two user-specified thresholds

```
1  foreach rule r ∈ LE do
2      LE' ← ∅ ;
3      foreach rule s ∈ LE such that r ≠ s and the pair r, s have not been
       processed do
4          Let c, d be the items respectively in r, s that do not appear in s, r ;
5          if countPruning(c, d) = false then
6              t ← leftMerge(r, s) ;
7              calculateSupport(t, r, s);
8              if sup(t) ≥ minsup then
9                  calculateConfidence(t, r, s);
10                 if conf(t) ≥ minconf then
11                     rules ← rules ∪ {t};
12                 end
13                 LE' ← LE' ∪ {t};
14             end
15         end
16     end
17     leftSearch (LE', rules);
18 end
```

Figure 3.4. Le pseudo-code de la procédure leftSearch.

En outre, une optimisation consiste à utiliser une structure que nous nommons le nombre clairsemé Matrice($Sparse\ Count\ Matrix$)(SCM) (aka CMAP [7]). Cette structure est construite lors de la première base de données numériser et enregistrer dans combien de séquences chaque élément apparaît avec les autres éléments.

Par exemple, la figure 3.6 montre la structure construite pour la base de données de la figure 3.1 (à gauche), représenté comme une matrice triangulaire. Considérez la deuxième rangée. Cela indique que l'item b apparaît avec les items b, c, d, e, f, g et h respectivement en 2, 1, 3, 4, 2 et 1 séquences. La structure SCM est utilisée pour élaguer l'espace de recherche comme suit (implémenté en tant que fonction $countPruning$ sur les Fig. 3.2 et 3.6). Soit une paire de règles r, squi est considéré pour une fusion gauche ou droite et c, d être les éléments de r et s qui

n'apparaissent pas respectivement dans s et r. Si le nombre de c, d est inférieur à $minsup$ dans le SCM, alors la fusion n'a pas besoin d'être effectuée et le support de la règle n'est pas calculé.

Enfin, une autre optimisation importante est de savoir comment implémenter le magasin de gauche structure pour stocker efficacement les classes d'équivalence de gauche des règles générées par des fusions à droite. Dans notre implémentation, nous utilisons une hashmap de hashmaps, où la première fonction de hachage est appliquée à la taille d'une règle et la seconde fonction de hachage est appliqué à l'ensemble d'éléments de gauche de la règle. Cela permet de trouver rapidement à quelle classe d'équivalence appartient à une règle générée par une fusion à droite.

Algorithm 3: The rightSearch procedure

input : RE: a right equivalence class, $rules$: the set of valid rules found until now, $minsup$ and $minconf$: the two user-specified thresholds, $leftStore$: the structure to store left-equivalence classes of rules generated by right-merges

```
1  foreach rule r ∈ RE do
2      RE' ← ∅ ;
3      foreach rule s ∈ RE such that r ≠ s and the pair r, s have not been
       processed do
4          Let c, d be the items respectively in r, s that do not appear in s, r ;
5          if countPruning(c, d) = false then
6              t ← rightMerge(r, s) ;
7              calculateSupport(t, r, s);
8              if sup(t) ≥ minsup then
9                  calculateConfidence(t, r, s);
10                 if conf(t) ≥ minconf then
11                     rules ← rules ∪ {t};
12                 end
13                 RE' ← RE' ∪ {t};
14                 addToLeftStore(t)
15             end
16         end
17     end
18     rightSearch (RE', rules);
19 end
```

Figure 3.3. Le pseudo-code de la procédure rightSearch.

Item	a	b	c	d	e	f
b	3					
c	2	2				
d	1	1	1			
e	3	3	2	1		
f	3	4	2	1	3	
g	1	2	1	0	1	2
h	0	1	0	0	0	1

Figure 3.6 La matrice du comptage clairsemé (SCM).

4 Évaluation expérimentale

Nous avons effectué des expériences pour évaluer la performance de l'algorithme proposé.

Les expériences ont été effectuées sur un ordinateur avec un processeur Core i5 de troisième génération fonctionnant sous Windows 7 et 5 Go de RAM libre. Nous avons comparé les formance d'ERMiner avec les algorithmes de pointe pour la règle séquentielle extraction de règles minière RuleGrowth [4]. Tous les algorithmes ont été implémentés en Java.

Toutes les mesures de mémoire ont été effectuées en utilisant l'API Java. Les expériences étaient portées sur cinq ensembles de données réelles ayant des caractéristiques variées et représentant quatre différents types de données (flux de clics Web, énoncés en langue des signes et séquences protéiques). Ces ensembles de données sont Sign, Snake, FIFA, BMS et Kosarak10k.

Le tableau 3.1 résume leurs caractéristiques.

dataset	sequence count	distinct item count	avg. seq. length (items)	type of data
Sign	730	267	51.99 (std = 12.3)	language utterances
Snake	163	20	60 (std = 0.59)	protein sequences
FIFA	20450	2990	34.74 (std = 24.08)	web click stream
BMS	59601	497	2.51 (std = 4.85)	web click stream
Kosarak10k	10000	10094	8.14 (std = 22)	web click stream

Tableau 3.1. Caractéristiques du itemsets.

Nous avons couru tous les algorithmes sur chaque ensemble de données tout en diminuant le minsup seuil jusqu'à ce que les algorithmes deviennent trop longs à exécuter, manquent de mémoire ou clair gagnant a été observé. Pour ces expériences, nous avons fixé le seuil minconf seuil à 0,75. Cependant, notez que les résultats sont similaires pour les autres valeurs de la minconf paramètre puisque la confiance n'est pas utilisée pour élaguer l'espace de recherche les algorithmes comparés.

Pour chaque ensemble de données, nous avons enregistré le temps d'exécution, le pourcentage du candidat élagué par la structure SCM et la taille totale des SCMs.

Execution times (Les temps d'exécution). La comparaison des temps d'exécution est montrée sur la figure 3.7. Elle on peut voir que ERMiner est plus rapide que RuleGrowth sur tous les jeux de données et que l'écart de performance augmente pour les valeurs minsup inférieures. ERMiner est à environ cinq fois plus vite que RuleGrowth. C'est parce que RuleGrowth doit effectuer opérations coûteuses de projection de bases de données.

Memory overhead of using SCM (Mémoire de l'utilisation de SCM). Nous avons mesuré les frais généraux produits en utilisant la structure SCM par ERMiner. La taille de SCM est généralement assez petit (moins de 35 Mo). La raison en est que nous l'avons implémenté en tant que sparse matrice (une hashmap de hashmaps) plutôt qu'une matrice complète (un tableau $n * n$ pour n articles). Si une matrice complète est utilisée, la taille de SCM augmente jusqu'à environ 300 Mo.

Overall memory of usage SCM (Utilisation générale de la mémoire SCM). L'utilisation maximale de la mémoire de RuleGrowth / ERMiner pour les jeux de données Snake, FIFA, Sign, BMS et Kosarak était respectivement 300 Mo / 1950 Mo, 478 Mo / 2030 Mo, 347 Mo / 1881 Mo, 1328 Mo / 2193 Mo et 669 Mo / 1441 Mo Nous remarquons donc qu'il y a un compromis entre avoir des temps d'exécution plus courts avec ERMiner et avoir moins de mémoire consommation avec RuleGrowth. La plus grande consommation de mémoire pour ERMiner est en grande partie due à l'utilisation de la structure de magasin de gauche qui nécessite de maintenir plusieurs classes d'équivalence en mémoire en même temps.

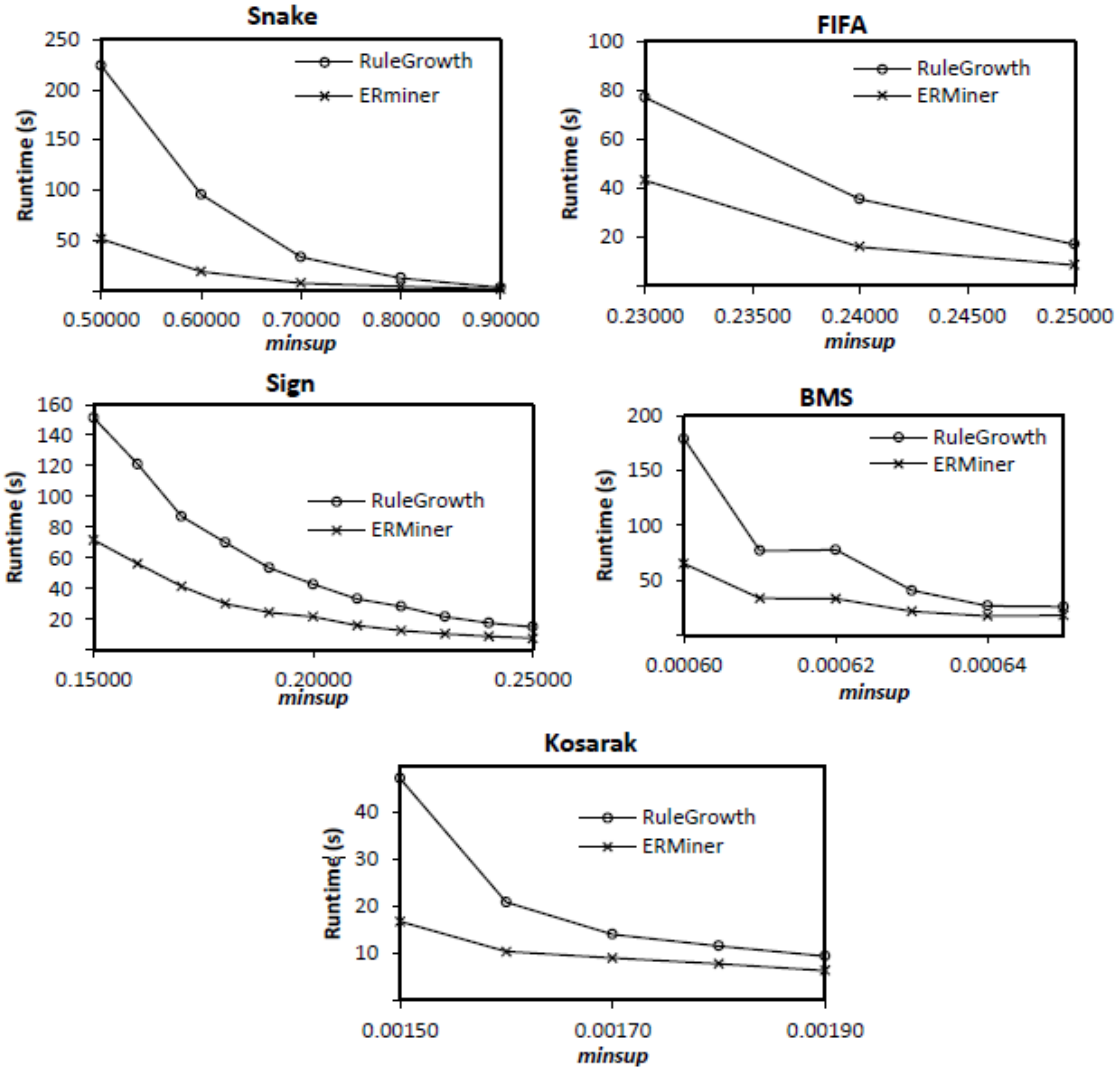

Figure 3.7. Temps d'exécution avec ERMiner et RuleGrowth.

Effectiveness of candidate pruning (L'efficacité de l'élagage des candidats). Le pourcentage de règles candidates élagué en utilisant la structure de données SCM dans ERMiner pour Snake, FIFA, Sign, BMS et Kosarak étaient respectivement de 1%, 0,2%, 3,9%, 3% et 51%. Ça peut conclure que l'élagage basé sur SCM est moins ef fi cace pour les ensembles de données contenant séquences denses ou longues (par exemple Snake, FIFA, Sign) où chaque élément co-produit de nombreuse fois avec tous les autres articles. Il pourrait donc être désactivé sur de tels ensembles de données. [15]

5 Exemple de ERMiner

Ici, nous allons fournir un exemple d'un algorithme ERMiner en action. Chaque règle R dans l'exemple est de la forme :

$$X \rightarrow Y\ [R_s]\ \langle R_c \rangle$$

Où R_s, appelé support, est un ensemble d'identificateurs de séquence indiquant des séquences contenir la règle. Le R_c, appelé confiant, est un ensemble d'identificateurs de séquence dénotant séquences qui contiennent au moins X. Notez qu'il est toujours vrai que $R_s \subseteq R_c$. Nous utilisons cette notation augmentée pour mieux illustrer le processus de fusion.

Compte tenu $minsup$ de 0,5 et $minconf$ de 0,5, et les séquences suivantes :

1. $(a, b), (c), (f), (g), (e)$
2. $(a, d), (c), (b), (a, b, e, f)$
3. $(a), (b), (f), (e)$
4. $(b), (f, g, h)$

La première étape consiste à générer une règle initiale 1*1 :

$a \rightarrow b$ [2,3] ⟨1,2,3⟩

$a \rightarrow c$ [1,2] ⟨1,2,3⟩

$a \rightarrow e$ [1,2,3] ⟨1,2,3⟩

$a \rightarrow f$ [1,2,3] ⟨1,2,3⟩

$b \rightarrow e$ [1,2,3] ⟨1,2,3,4⟩

$b \rightarrow f$ [1,2,3,4]⟨1,2,3,4⟩

$b \rightarrow g$ [1,4] ⟨1,2,3,4⟩

$c \rightarrow e$ [1,2] ⟨1,2⟩

$c \rightarrow f$ [1,2]⟨1,2⟩

$f \rightarrow e$ [1,3] ⟨1,2,3,4⟩

A partir de ces règles, nous pouvons générer les classes d'équivalence. L'équivalence gauche Les classes LE_1 de taille 1 se présentent comme suit :

$LE_{a,1}$ $a \rightarrow b$ [2,3]⟨1,2,3⟩,
$a \rightarrow c$ [1,2]⟨1,2,3⟩,
$a \rightarrow e$ [1,2,3]⟨1,2,3⟩,
$a \rightarrow f$ [1,2,3]⟨1,2,3⟩

$LE_{b,1}$ $b \rightarrow e$ [1,2,3]⟨1,2,3,4⟩,
$b \rightarrow f$ [1,2,3,4]⟨1,2,3,4⟩,
$b \rightarrow g$ [1,4] ⟨1,2,3,4⟩

$LE_{c,1}$ $c \rightarrow e$ [1,2] ⟨1,2⟩;
$c \rightarrow f$ [1,2] ⟨1,2⟩

$LE_{f,1}$ $f \rightarrow e$ [1,3] ⟨1,2,3,4⟩

A partir de LE_1 , nous pouvons produire des classes d'équivalence de gauche de taille deux LE_2 en effectuant une fusion à gauche :

$LE_{a,2}$ $a \rightarrow (b, c)$ [2]⟨1,2,3⟩,
$a \rightarrow (c, f)$[1,2]⟨1,2,3⟩,
$a \rightarrow (b, f)$[2,3]⟨1,2,3⟩,
$a \rightarrow (e, f)$[1,2,3]⟨1,2,3⟩,
$a \rightarrow (c, e)$[1,2]⟨1,2,3⟩,
$a \rightarrow (b, e)$[2,3]⟨1,2,3⟩

$LE_{b,2}$ $b \rightarrow (e, f)$[1,2,3]⟨1,2,3,4⟩,
$b \rightarrow (e, g)$[1]⟨1,2,3,4⟩,
$b \rightarrow (g, f)$[1,4] ⟨1,2,3,4⟩

$LE_{c,2}$ $c \rightarrow (e, f)$[1,2] ⟨1,2⟩;

Nous les nettoyons en rejetant toutes les règles qui ne respectent pas le minimum besoin de soutien de 0,5, comme $a \rightarrow (b, c)[2]\langle 1,2,3\rangle$, qui se produit seulement dans une séquence *#2* sur le total de quatre, faisant son support $\frac{1}{4} = 0.25$.

$LE_{a,2}$ $a \rightarrow (c, f)$ $[1,2]\langle 1,2,3\rangle$,
$a \rightarrow (b, f)[2,3]\langle 1,2,3\rangle$,
$a \rightarrow (e, f)[1,2,3]\langle 1,2,3\rangle$,
$a \rightarrow (c, e)[1,2]\langle 1,2,3\rangle$,
$a \rightarrow (b, e)[2,3]\langle 1,2,3\rangle$

$LE_{b,2}$ $b \rightarrow (e, f)[1,2,3]\langle 1,2,3,4\rangle$,
$b \rightarrow (g, f)[1,4]$ $\langle 1,2,3,4\rangle$

$LE_{c,2}$ $c \rightarrow (e, f)[1,2]$ $\langle 1,2\rangle$;

Nous pouvons faire la fusion à gauche sur $LE_{a,2}$ et $LE_{b,2}$:
$LE_{a,3}$ $a \rightarrow (b, c, f)$ $[2]\langle 1,2,3\rangle$,
$a \rightarrow (c, e, f)[1,2]\langle 1,2,3\rangle$,
$a \rightarrow (b, f, e)[2,3]\langle 1,2,3\rangle$

$LE_{b,3}$ $b \rightarrow (e, g, f)[1]\langle 1,2,3,4\rangle$,

Filtrer les règles $a \rightarrow (b, c, f)[2]\langle 1,2,3\rangle$ et $b \rightarrow (e, g, f)[1]\langle 1,2,3,4\rangle$, qui ne répondent pas à l'exigence minimale de support, le $LE_{b,3}$ est complètement éliminé et il nous reste :
$LE_{a,3}$ $a \rightarrow (c, e, f)$ $[1,2]\langle 1,2,3\rangle$,
$a \rightarrow (b, f, e)[2,3]\langle 1,2,3\rangle$,

Les classes d'équivalence à droite de taille 1 sont :
$RE_{g,1}$ $b \rightarrow g$ $[1,4]\langle 1,2,3,4\rangle$

$RE_{b,1}$ $a \rightarrow b[2,3]\langle 1,2,3\rangle$

$RE_{e,1}$ $a \rightarrow e[1,2,3]\langle 1,2,3\rangle$,
$b \rightarrow e[1,2,3]\langle 1,2,3,4\rangle$,
$c \rightarrow e[1,2]\langle 1,2\rangle$,
$f \rightarrow e[1,3]\langle 1,2,3,4\rangle$

$RE_{c,1}$ $a \rightarrow c[1,2]\ \langle 1,2,3\rangle$

$RE_{f,1}$ $a \rightarrow f[1,2,3]\langle 1,2,3\rangle$,
$b \rightarrow f[1,2,3]\langle 1,2,3,4\rangle$,
$c \rightarrow f[1,2]\langle 1,2\rangle$

En effectuant une fusion droite, nous avons les classes d'équivalence droite de taille deux :

$RE_{e,2}$ $(a,b) \rightarrow e[1,2,3]\langle 1,2,3\rangle$,
$(a,c) \rightarrow e[1,2]\langle 1,2\rangle$,
$(b,c) \rightarrow e[1,2]\langle 1,2\rangle$,
$(a,f) \rightarrow e[1,3]\langle 1,2,3\rangle$,
$(b,f) \rightarrow e[1,3]\langle 1,2,3,4\rangle$

$RE_{f,2}$ $(a,b) \rightarrow f[1,2,3]\langle 1,2,3\rangle$,
$(a,c) \rightarrow f[1,2]\langle 1,2\rangle$,
$(b,c) \rightarrow f[1,2]\langle 1,2\rangle$

La dernière fusion droite produit des classes d'équivalence de taille trois :

$RE_{e,2}$ $(a,b,c) \rightarrow e[1,2]\langle 1,2\rangle$,
$(a,b,f) \rightarrow e[1,3]\langle 1,2,2\rangle$

$RE_{f,3}$ $(a,b,c) \rightarrow f[1,2]\langle 1,2\rangle$

Notez que pour toutes les règles, par ex. $(a,b,c) \rightarrow e[1,2]\langle 1,2\rangle$, nous pouvons fusionner tout deux des règles de la classe d'équivalence inférieure, c'est-à-dire $(a,b) \rightarrow e[1,2,3]\langle 1,2,3\rangle$,

$(a,c) \rightarrow e[1,2]\langle 1,2\rangle$, ou $(b,c) \rightarrow e[1,2]\langle 1,2\rangle$ .Le résultat sera le même. C'est juste une belle propriété du processus de fusion.

L'étape suivante consiste à prendre toutes les règles dans les classes d'équivalence droite *RE* supérieures à 1*1 et les mettre dans leurs classes d'équivalence gauche *LE* respectives.

À partir des classes d'équivalence droite de la taille deux RE_2 nous obtenons des classes d'équivalence à gauche de taille un. Notons-les par LE^{α}:

$LE^{\alpha}_{(a,b),1}$ $(a,b) \rightarrow e[1,2,3]\langle 1,2,3\rangle$,

$(a,b) \rightarrow f[1,2,3]\langle 1,2,3\rangle$

$LE^{\alpha}_{(a,e),1}$ $(a,c) \rightarrow e[1,2]\langle 1,2\rangle$,

$(a,c) \rightarrow f[1,2]\langle 1,2\rangle$

$LE^{\alpha}_{(b,c),1}$ $(b,c) \rightarrow e[1,2]\langle 1,2\rangle$,

$(b,c) \rightarrow f[1,2]\langle 1,2\rangle$

À partir des classes d'équivalence de droite de taille trois, nous obtenons ces équivalences à gauche classes que nous dénotons avec un exposant α aussi :

$LE^{\alpha}_{(a,b,c),1}$ $(a,b,c) \rightarrow e[1,2]\langle 1,2\rangle$,

$(a,b,c) \rightarrow f[1,2]\langle 1,2\rangle$

$LE^{\alpha}_{(a,b,f),1}$ $(a,b,f) \rightarrow e[1,3]\langle 1,2,3\rangle$

À ce point, puisque tous les LE^{α} les classes d'équivalence sont de nouvelles classes d'équivalence à gauche, nous effectuons une fusion à gauche sur eux :

$LE_{(a,b),2}$ $(a,b) \rightarrow (e,f)[1,2,3]\langle 1,2,3\rangle$

$LE_{(a,c),2}$ $(a,c) \rightarrow (e,f)[1,2]\langle 1,2\rangle$

$LE_{(b,c),2}$ $(b,c) \rightarrow (e,f)[1,2]\langle 1,2\rangle$

$LE_{(a,b,c),2}$ $(a,b,c) \rightarrow (e,f)[1,2]\langle 1,2\rangle$

À ce point, nous avons fait tout le travail nécessaire pour produire l'ensemble des règles valides. Ce que nous devons faire est :

a) collecter toutes les règles fréquentes ;

b) rejeter toute règle qui ne respecte pas l'exigence de confidentialité minimale.

La collecte des règles fréquentes est facile, car toutes les règles peuvent être trouvées en scannant classes d'équivalence de gauche (LE etLE^{α}). Parce que nous avons pris soin de ne jamais mettre de rares règles dans le LE ouLE^{α} (puisque c'est juste une reformulation des règles dans tous les ER qui aussi ne pas avoir des règles peu fréquentes) toutes les règles que nous recevons seront des règles fréquentes.

Voici donc maintenant la liste de toutes les règles fréquentes, mais pas forcément valides :

$a \rightarrow b$ [2,3] ⟨1,2,3⟩

$a \rightarrow c$ [1,2] ⟨1,2,3⟩

$a \rightarrow e$ [1,2,3] ⟨1,2,3⟩

$a \rightarrow f$ [1,2,3] ⟨1,2,3⟩

$b \rightarrow e$ [1,2,3] ⟨1,2,3,4⟩

$b \rightarrow f$ [1,2,3,4]⟨1,2,3,4⟩

$b \rightarrow g$ [1,4] ⟨1,2,3,4⟩

$c \rightarrow e$ [1,2] ⟨1,2⟩

$c \rightarrow f$ [1,2]⟨1,2⟩

$f \rightarrow e$ [1,3] ⟨1,2,3,4⟩

$a \rightarrow (c, f)$ [1,2] ⟨1,2,3,4⟩

$a \rightarrow (b, f)$[2,3] ⟨1,2,3⟩

$a \rightarrow (e, f)$[1,2,3] ⟨1,2,3⟩

$a \rightarrow (c, e)$[1,2] ⟨1,2,3⟩

$a \rightarrow (b, e)$[2,3] ⟨1,2,3⟩

$b \rightarrow (e, f)$[1,2,3] ⟨1,2,3,4⟩

$b \rightarrow (g, f)$[1,4]⟨1,2,3,4⟩

$c \rightarrow (e, f)$[1,2] ⟨1,2⟩

$a \rightarrow (c, e, f)$[1,2] ⟨1,2,3⟩

$a \rightarrow (b, f, e)$[2,3]⟨1,2,3⟩

$(a, b) \rightarrow e$ [1,2,3] ⟨1,2,3⟩

$(a, b) \rightarrow f$ [1,2,3]⟨1,2,3⟩

$(a, c) \rightarrow e$ [1,2]⟨1,2⟩

$(a, c) \rightarrow f$ [1,2]⟨1,2⟩

$(b, c) \rightarrow e$ [1,2]⟨1,2⟩

$(b, c) \rightarrow f$ [1,2]⟨1,2⟩

$(a, b, c) \rightarrow e$ [1,2]⟨1,2⟩

$(a, b, c) \rightarrow f$ [1,2]⟨1,2⟩

$(a, b, f) \rightarrow e$ [1,3]⟨1,2,3⟩

$(a, b) \rightarrow (e, f)$ [1,2,3]⟨1,2,3⟩

$(a, c) \rightarrow (e, f)$ [1,2]⟨1,2⟩

$(b, c) \rightarrow (e, f)$ [1,2]⟨1,2⟩

$(a, b, c) \rightarrow (e, f)$ [1,2]⟨1,2⟩

Obtenir seulement des règles valides est simple. Puisque chaque règle a le $support$ et la $confiance$ composants, il s'agit juste de diviser la longueur de l'un par l'autre.

En d'autres termes, la confiance de la règle $R: X \rightarrow Y$ $[R_s]$ $\langle R_c \rangle$ est :

$$\boldsymbol{C(R) = \frac{|R_s|}{|R_c|}}$$

Par exemple : pour une règle comme $(a, c) \rightarrow (e, f)$ [1,2]⟨1,2⟩ le $\boldsymbol{R_s} = [1,2]$ et $\boldsymbol{R_c} = [1,2]$donc :$\boldsymbol{C((a, c) \rightarrow (e, f)\ [1;\ 2]\ \langle 1;\ 2 \rangle) = \frac{2}{2} = 1.0}$

Qui est clairement supérieur à 0.5, notre exigence de confiance minimale.

En l'occurrence, toutes nos règles fréquentes sont également valables.

6 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme d'exploration de règles séquentielles nommé ERMiner (Equivalence class based sequential Rule Miner). Il repose sur l'idée originale de recherche utilisant des classes d'équivalence de règles ayant le même antécédent ou conséquent. En outre, il comprend une structure de données appelée SCM (Sparse Count Matrix) pour élaguer l'espace de recherche. Une vaste étude expérimentale avec les jeux de données réels montre que ERMiner est jusqu'à cinq fois plus rapide que l'algorithme de pointe (the-state-of-the-art), mais

consomme plus de mémoire. Il peut donc être vu comme un compromis intéressant lorsque la vitesse est plus importante que la mémoire

Chapitre 4

Implémentation

Chapitre 4

Implémentation

1 Introduction

Ce chapitre est essentiellement axé sur les grandes lignes qui nous ont permis de réaliser et de mettre en œuvre ce projet de recherche, et les outils exploités pour le développement du logiciel tels que l'environnement de programmation, le matériel utilisé et les principales fonctions de traitement à utiliser.

Aussi nous mettrons en évidence les différentes propriétés de notre implantation, et donnerons des captures-écrans, des principales interfaces et montrerons les spécifications et fonctionnalités qu'offre à l'utilisateur notre algorithme, mais aussi une présentation du mode de fonctionnement.

Nous allons présenter les étapes de réalisation de notre système d'extraction dans l'algorithme ERMiner, et nous examinerons les détails spécifiques de l'implémentation de l'algorithme.

En particulier, nous examinerons comment les différents types de règles sont modélisés, comment ils sont comparés et comment le composant générateur de règles est implémenté.

2 Implémentation :

Cette partie constitue le dernier volet de ce projet, après avoir terminé la phase de spécification et conception, la solution étant déjà choisie et étudiée, il nous reste que de se décider dans quel environnement nous allons travailler, exposer les choix techniques utilisés et le langage adopté, et présenter l'implémentation et les tests réalisés ainsi les résultats obtenus.

2.1 Choix du langage de programmation :

2.1.1 Java

Java est un langage de programmation orienté objet, développé par Sun Microsystems et destiné à fonctionner dans une machine virtuelle, il permet de créer des logiciels compatibles avec des nombreux systèmes d'exploitation.

Java et non seulement un langage de programmation puissant conçu pour être sûr, inter plateformes et international, mais aussi un environnement de développement qui est continuellement étendu pour fournir des nouvelles caractéristiques et des bibliothèques permettant

de gérer de manière élégante des problèmes traditionnellement complexes dans les langages de programmation classiques, tels que le multithreading, les accès aux bases des données, la programmation réseau, l'informatique répartie.

En plus java est considéré comme un langage adaptable aux plusieurs domaines puisqu'une application web implémentée par celle-ci peut avoir des extensions ou des modifications dans le futur. De plus, java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé. [17]

2.1.2 JDK

Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java.

Il existe plusieurs éditions de JDK, selon la plate-forme Java considérée (et bien évidemment la version de Java ciblée) :

- JSE pour la Java 2 Standard Edition également désignée J2SE ;
- JEE, sigle de Java Enterprise Edition également désignée J2EE ;
- JME 'Micro Edition', destinée au marché mobile ; etc.

À chacune de ces plateformes correspond une base commune de Development Kits, plus des bibliothèques additionnelles spécifiques selon la plate-forme Java que le JDK cible, mais le terme de JDK est appliqué indistinctement à n'importe laquelle de ces plates-formes.

La version que nous avons utilisée est **JDK 8** (java developpement kit).

2.2 Outils de développement NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine

virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

L'IDE NetBeans s'enrichit à l'aide de greffons. [18]

L'environnement de base comprend les fonctions générales suivantes :

- Configuration et gestion de l'interface graphique des utilisateurs,
- Support de différents langages de programmation,
- Traitement du code source (édition, navigation, formatage, inspection),
- Fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- Accès et gestion de bases de données, serveurs Web, ressources partagées,
- Gestion de tâches (à faire, suivi…),
- Documentation intégrée.

2.3 Choix de l'architecture de l'application :

L'architecture à 1-tiers consiste à mettre tous les composants nécessaires à une application logicielle ou une technologie sur un seul poste ou plate-forme. Ce type d'architecture est souvent opposé à l'architecture multi-niveaux ou l'architecture à trois tiers qui est utilisé pour certaines applications Web et d'autres technologies où différentes couches de présentation, ou d'accès aux données sont logés séparément.

Fondamentalement, une architecture 1-tiers conserve tous les éléments d'une application, y compris l'interface, exécutable et les données, en un seul endroit. Les développeurs voient ces types de systèmes comme le plus simple et le plus direct. Certains experts décrivent comme des applications qui pourraient être installées et exécutées sur un seul ordinateur.

La nécessité pour les modèles distribués pour les applications Web et des solutions d'hébergement Cloud a créé de nombreuses situations où les architectures à un seul niveau ne sont pas suffisantes. Cela fait à trois niveaux ou de l'architecture multi-niveaux pour devenir plus populaire. Les avantages d'une solution multi-niveaux sont souvent évidents. Ils peuvent fournir une meilleure sécurité, une meilleure performance et plus d'évolutivité. Toutefois, l'appel d'une

architecture 1-tiers peut se rapporter aux coûts qui sont impliqués, où il serait plus logique de garder des applications plus simples contenues dans une plate-forme facile.

L'architecture 1-tiers peut être bénéfique lorsque nous traitons avec des données qui se rapportent à un seul utilisateur (ou un petit nombre d'utilisateurs) et nous avons une quantité relativement faible de données. Ils sont peu coûteux à déployer et à maintenir [18].

A cet effet, notre application sera utilisée sur un seul poste ou une seule plateforme, et on va opter pour cette architecture pour les raisons suivantes :

- Notre application est basé sur un petit nombre d'utilisateurs
- Notre cas n'est pas coûteux en terme de matériels et sécurité
- On peut gérer notre application avec un seul poste performant en terme de mémoire et vitesse.

2.4 SPMF : une bibliothèque d'exploration de modèles Open Source Java

SPMF (Sequential Pattern Mining Framework), une bibliothèque d'exploration de données spécialisée dans l'exploration fréquente de motifs, un important sous-domaine de data mining qui vise à découvrir des modèles intéressants et des associations dans les bases de données. SPMF est un projet open-source, lancé en 2009 pour pallier le manque de grande bibliothèque de data mining open-source spécialisée dans l'exploration fréquente de motifs. Il existe plusieurs bibliothèques d'exploration de données open-source à usage général telles que Weka (Witten et al., 2005), Mahout (Mahout, 2013) et Knime (Knime, 2013), qui fournissent une large gamme de techniques d'exploration de données. Cependant, ils offrent un ensemble très limité d'algorithmes pour l'exploration fréquente de motifs. Weka, Knime et Mahout n'offrent que quelques algorithmes d'extraction de motifs populaires tels qu'Apriori (Agrawal et Srikant, 1994), GSP (Srikant et al., 1996) et FPGrowth (Han et al., 2004). Certaines plateformes spécialisées comme Coron (Coron, 2013), LUCS-KDD (LUCS-KDD, 2013) et Illimine (Illimine, 2013) offrent un choix un peu plus large d'algorithmes d'exploration de motifs. Cependant, le code source de Coron n'est pas public, Illimine fournit le code source d'un seul de ses algorithmes d'exploration de modèles et le code source LUCS-KDD ne peut pas être utilisé à des fins commerciales.

SPMF fournit plus de 55 algorithmes pour l'exploration de motifs. Les implémentations de la plupart de ces algorithmes ne peuvent être trouvées que dans SPMF. [19]

Depuis sa première édition majeure en 2010, SPMF a été utilisé dans plus de 70 recherches projets dans différents domaines tels que l'exploration de l'utilisation du Web, l'analyse du comportement des apprenants en apprentissage électronique, la récupération de texte clinique, la prévision des ventes, la recommandation de restaurants, l'analyse des séquences d'acides nucléiques, la détection d'anomalies dans les traitements médicaux.

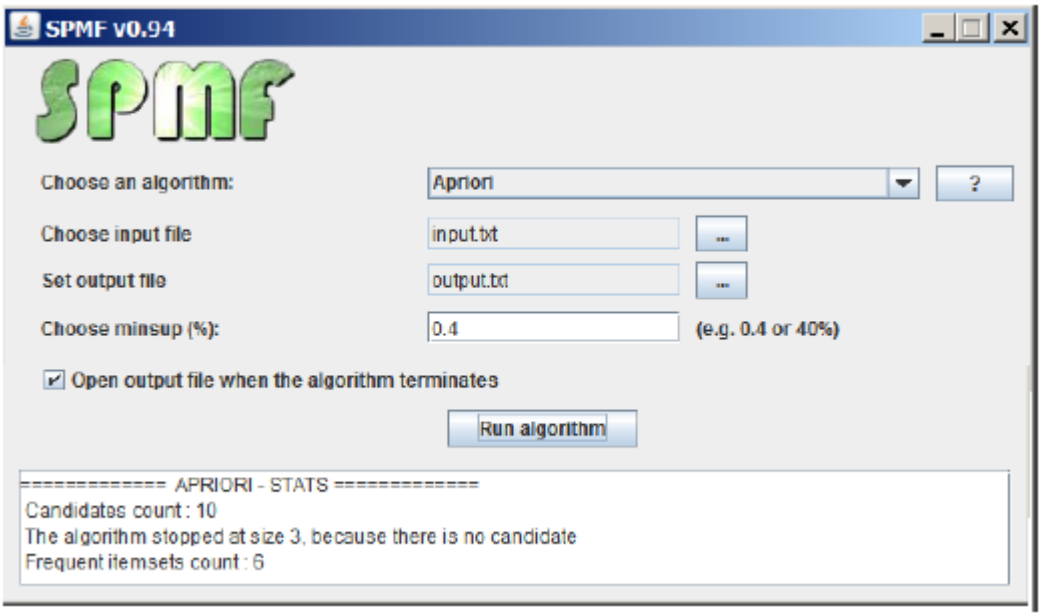

Figure 4.1 : Interface graphique SPMF.

Cet exemple explique comment exécuter l'algorithme ERMiner en utilisant notre application.

- Sélectionnez le fichier d'entrée " INPUTE.txt",
- Définissez le nom du fichier de sortie (par exemple "output.txt")
- Déterminé $minsup = 75\%, minconf = 50\%$
- Cliquez sur " Run algorithm ".

2.4.1 Quelle est l'entrée d'ERMiner ?

L'entrée de ERMiner est une base de données de séquences et deux seuils spécifiés par l'utilisateur nommés $minsup$ (une valeur dans [0, 1] représentant un pourcentage) et $minconf$ (une valeur dans [0, 1] représentant un pourcentage).

Une base de données de séquences est un ensemble de séquences où chaque séquence est une liste des itemsets. Un itemset est un ensemble non ordonné d'item. Par exemple, le tableau ci-dessous contient quatre séquences. La première séquence, appelée S1, contient 5 ensembles

d'éléments. Cela signifie que l'item 1 a été suivi par les items 1 2 et 3 en même temps, suivis de 1 et 3, suivis de 4 et suivis de 3 et 6. Il est supposé que les items d'un itemset sont triés dans l'ordre lexicographique. Cette base de données est fournie dans le fichier "INPUTE.txt" de la distribution SPMF.

ID	Sequences
S1	(1), (1 2 3), (1 3), (4), (3 6)
S2	(1 4), (3), (2 3), (1 5)
S3	(5 6), (1 2), (4 6), (3), (2)
S4	(5), (7), (1 6), (3), (2), (3)

Tableau 4.1. Une base de données de séquences.

2.4.2 Format de fichier d'entrée

Le format du fichier d'entrée est défini comme suit. C'est un fichier texte où chaque ligne représente une séquence d'une base de données de séquences. Chaque élément d'une séquence est un entier positif et les éléments du même ensemble d'éléments dans une séquence sont séparés par des espaces uniques. Notez qu'il est supposé que les éléments d'un même ensemble d'éléments soient triés en fonction d'un ordre total et qu'aucun élément ne peut apparaître deux fois dans le même jeu d'éléments. La valeur "-1" indique la fin d'un ensemble d'éléments. La valeur "-2" indique la fin d'une séquence (elle apparaît à la fin de chaque ligne). Par exemple, l'exemple de fichier d'entrée "INPUTE.txt" contient les quatre lignes suivantes (quatre séquences).

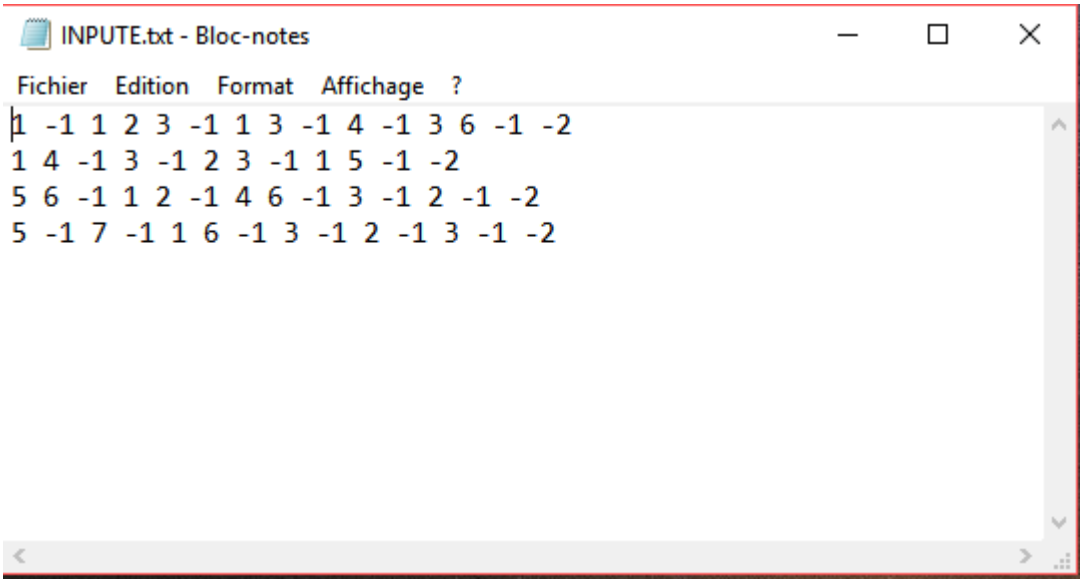

Figure 4.2 Fichier de base de données de séquences au format texte.

La première ligne représente une séquence où l'itemset {1} est suivi par l'itemset {1, 2, 3}, suivi du l'itemset {1, 3}, suivi l'itemset {4}, suivi l'itemset {3, 6}. Les lignes suivantes suivent le même format.

Notez qu'il est également possible d'utiliser un fichier texte contenant un texte (plusieurs phrases) si le fichier texte a l'extension ".text", comme alternative au format d'entrée par défaut. Si l'algorithme est appliqué sur un fichier texte à partir de l'interface graphique ou de l'interface de ligne de commande, le fichier texte sera automatiquement converti au format SPMF, en divisant le texte en phrases séparées par ".", "?" et "!", où chaque mot est considéré comme un élément. Notez que lorsqu'un fichier texte est utilisé comme entrée d'un algorithme d'exploration de données, les performances seront légèrement inférieures à celles du format de fichier SPMF natif car une conversion du fichier d'entrée sera automatiquement effectuée avant le lancement de l'algorithme et le résultat sera doivent également être convertis. Ce coût devrait cependant être faible.

2.4.3 Quelle est la sortie d'ERMiner ?

Étant donné une base de données de séquences, et des paramètres nommés $minsup$ et $minconf$, ERMiner produit toutes les règles séquentielles ayant un support et une confiance respectivement plus élevés que $minsup$ et $minconf$.

Dans cet exemple, nous appliquons ERMiner avec $minsup = 75\%$,$minconf = 50\%$. Nous obtenons 9 règles séquentielles :

Rule	Support	Confidence
1 ==> 2	100 %	100 %
1 ==>3	100 %	100 %
2 ==> 3	75 %	75 %
3 ==> 2	75 %	75 %
4 ==> 3	75 %	100 %
1 3 ==> 2	75 %	75 %
1 2 ==> 3	75 %	75 %
1 4 ==> 3	75 %	100 %
1 ==> 2 3	100 %	100 %

Tableau 4.2. Le résultat de ERMiner avec minsup = 75%, minconf = 50% .

Par exemple, la règle 1 4 →3 signifie que si 1 et 4 apparaissent dans n'importe quel ordre, ils seront suivis de 3 avec une confiance de 100%. De plus, cette règle a un support de 75% car elle apparaît dans trois séquences (S1, S2 et S3) sur quatre séquences.

2.4.4 Format de fichier de sortie

Le format du fichier de sortie est défini comme suit. C'est un fichier texte. Chaque ligne est une règle séquentielle. Chaque élément d'une règle séquentielle est un nombre entier positif. Sur chaque ligne, les éléments de l'antécédent de la règle sont d'abord listés, séparés par des espaces uniques. Ensuite, le mot-clé "==>" apparaît, suivi des éléments de la règle conséquente, séparés par des espaces uniques. Ensuite, le mot clé **"#SUP:"** apparaît suivi d'un entier indiquant le support de la règle sous la forme d'un nombre de séquences. Ensuite, le mot-clé **"#CONF:"** apparaît suivi d'une double valeur dans l'intervalle [0, 1] indiquant la confiance de la règle. Par exemple, quelques lignes du fichier de sortie de l'exemple précédent sont affichées ci-dessous :

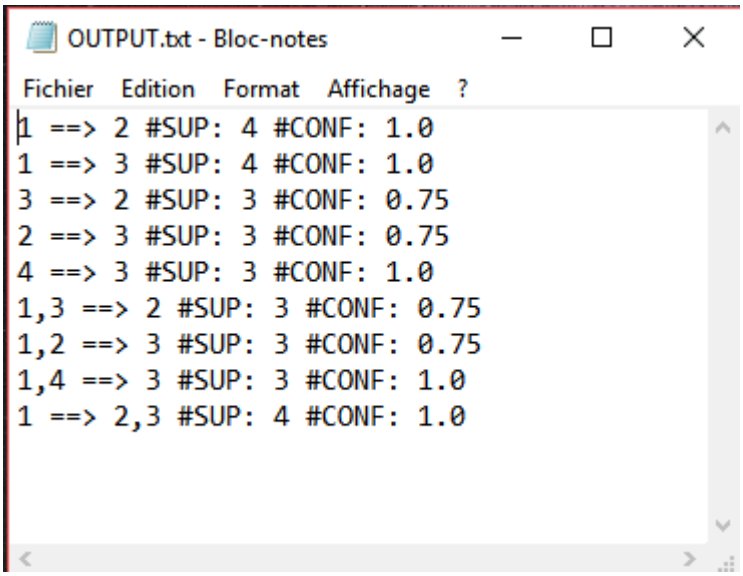

Figure 4.3. Le fichier de sortie au format texte.

Considérez la première ligne. Il indique que la règle {1} ==> {2} a un support de 4 séquences et une confiance de 100%. Les lignes suivantes suivent le même format.[]

3 Réalisation

Dans cette partie, on décrit le coté réalisation et implémentation de l'application conçue, on décrit l'interface utilisateur développée à l'aide de Java sous environnement NetBeans, et les résultats de tests obtenus.

3.1 Description de l'application

L'interface suivante représenter l'interface principale qui contient l'algorithmes à exécuter ERMiner, cette interface est représentée ci-après :

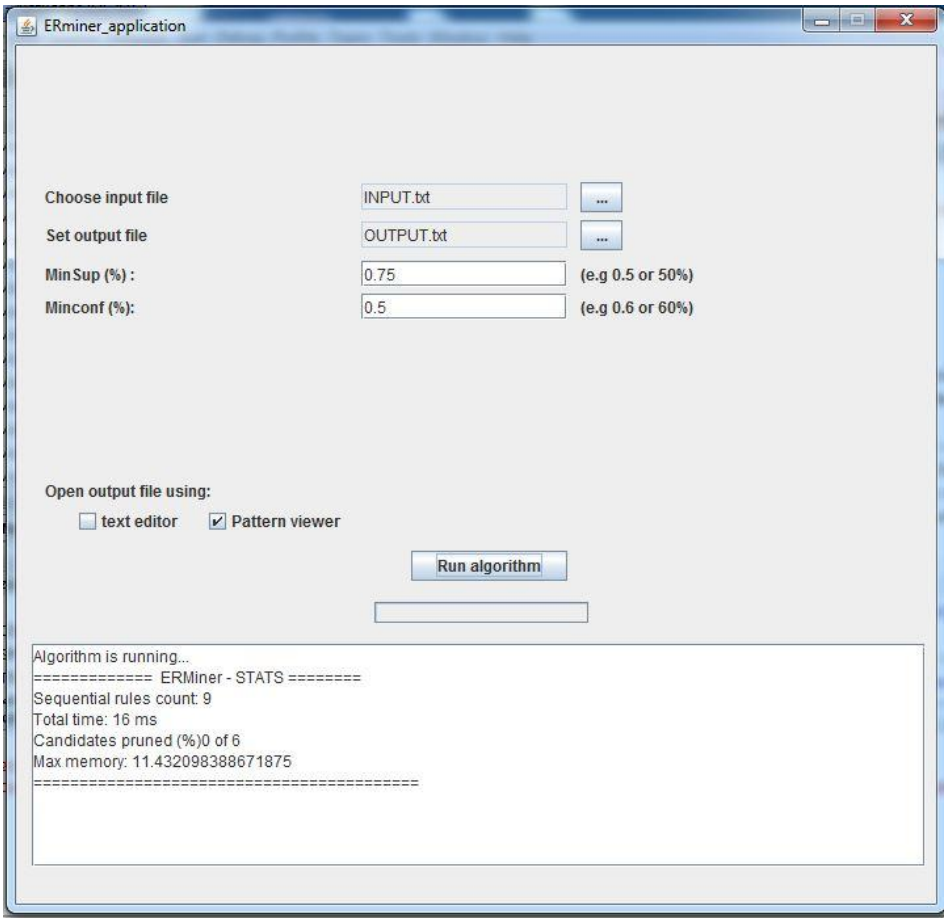

Figure 4.4. Interface principale.

Pour exécuter m'algorithme en passant plusieurs paramètres, parmi ces paramètres en cite :

- **Choose input file** : c'est un fichier de plusieurs transactions qui contient des items en extension .txt.
- **Set output file** : c'est un fichier texte qui contient les différentes règles générées après l'exécution de l'algorithme, ce dernier contient aussi le support de la règle ainsi la confiance.
- **MinSup** : c'est le support minimal saisi par l'utilisateur, pour définir le nombre d'occurrence de chaque partie de la règle.
- **MinConf** : c'est la confiance minimale saisi par l'utilisateur pour dire que cette règle est valide.

- **Run algorithm** : pour exécuter l'algorithme.
- **Open output file using** : pour choisir la méthode d'ouverture du fichier d'exécution.

Si vous choisissez **text editor** le résultat s'ouvre dans le fichier **OUTPUT.txt** Sur cette interface

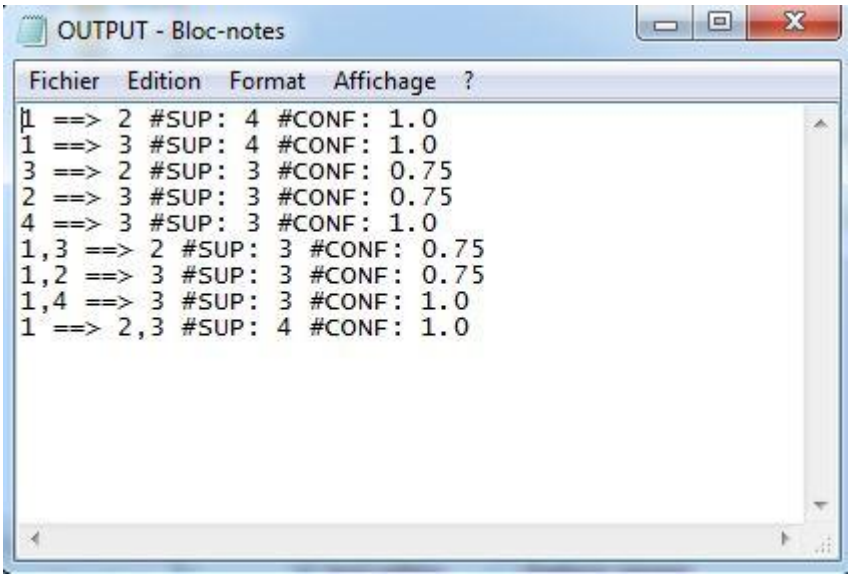

Figure 4.5. Le résultat de la sélection **text editor**.

Et si vous choisissez **Pattern viewer** l'interface suivante s'ouvre :

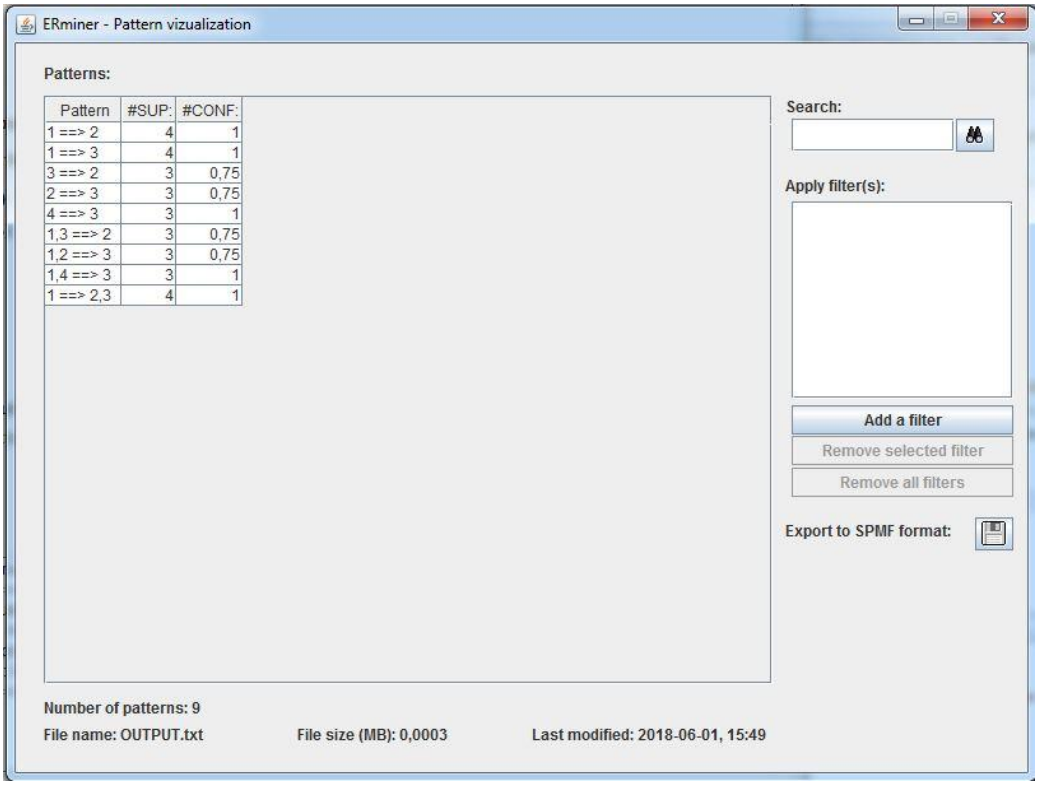

Figure 4.6. L'interface de l'exécution de l'algorithme ERMiner.

Cette interface contient :

- **Patterns** : Le résultat de l'exécution de l'algorithme.
- **Search** : pour chercher dans le résultat.
- **Apply filter** : pour le filtrage de résultat.
- **Export to SPMF format :** le bouton de l'enregistrement de résultat sous la forme de ficher text.

Si vous voulez ajouter un filtre, cette fenêtre va afficher :

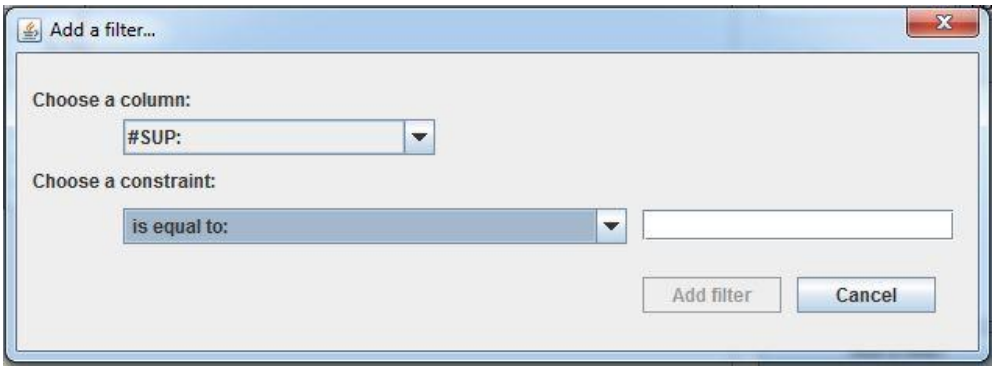

Figure 4.7. L'interface d'ajouter un filtre.

Elle contient :

Le choix de colonne (support minimal, la confiance ou bien la règle).

La contrainte de filtrage (égale, moine que ou bien plus que).

Par exemple si vous choisissez le filtrage : « le support minimal égale à 4 » (#SUP is equal to : 4) Ce résultat sera affiché :

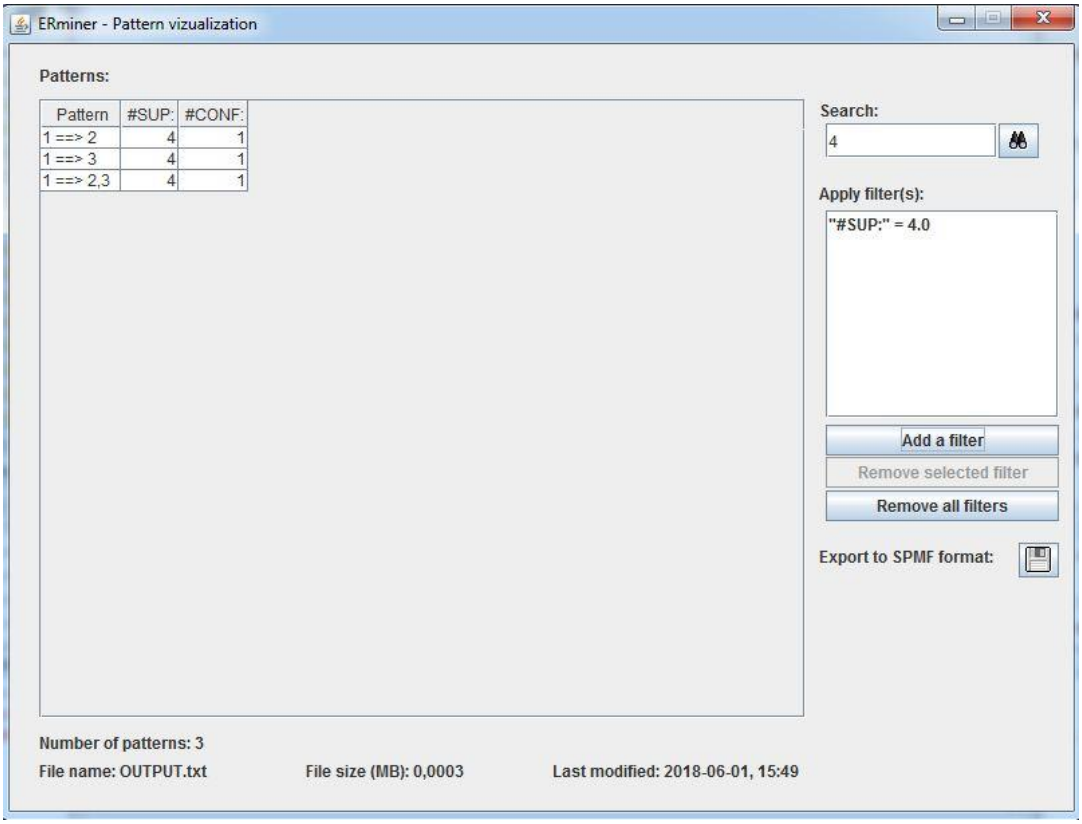

Figure 4.8. Le résultat de filtre **(#SUP is equal to : 4).**

4 Conclusion

Dans ce chapitre, nous avons vu les différents outils utilisés pour implémenter notre application, ainsi les interfaces de l'application et l'utilisation de l'application pour l'extraction d'une base de données de séquences.

Conclusion générale

L'extraction de règles séquentielles à partir de bases de données volumineuses est un sujet important dans les champs d'exploration de données avec des applications étendues. La plupart des études pertinentes se sont concentrées sur la recherche de règles séquentielles apparaissant dans une seule séquence d'événements et la tâche d'exploration de plusieurs séquences a été beaucoup moins explorée.

En effet, les méthodes d'extraction des règles séquentielles nécessitent une optimisation pour diminuer le temps d'exécution ainsi la consommation de l'espace mémoire, Alors, il existe plusieurs facteurs qui contrôlent les résultats obtenus :

- Le choix des paramètres par l'utilisateur comme le support minimal et confiance minimale.
- Le choix de nombre k des règles d'association qui doit être rechercher.

Dans notre travail, nous avons basés sur l'étude de l'algorithme ERMiner, un algorithme correct et complet pour l'extraction de règles séquentielles communes à plusieurs séquences dans des bases de données de séquences.

Et si nous comparons l'algorithme de ERMiner avec les autres algorithmes de l'extraction des règles séquentielles on trouve qu'il est un algorithme très efficace. C'est plus rapide que CMDeo et CMRules. De plus, ERMiner est aussi plus rapide que RuleGrowth (jusqu'à 5 fois plus rapide que RuleGrowth). Cependant, ERMiner consomme généralement plus de mémoire que RuleGrowth, donc il y a un compromis.

À long terme, ERMiner a permis de réaliser des économies substantielles en termes d'exécution. Il nous semble que dans le monde des données en continu, un tel algorithme pourrait fournir une certaine valeur et peut être utilisé comme un bloc de construction pour d'autres recherches.

Enfin, du côté de la programmation nous avons fait des modifications a l'application original SPMF pour répondre aux exigences de notre travail, et faire l'extraction des régles séquentielles par l'algorithme ERMiner seulement.

Pour conclure, ce projet nous a été grandement bénéfique et représente une expérience très enrichissante, il nous a permis de découvrir des domaines complètement nouveaux telle la fouille de données et le monde de la recherche.

Bibliographe

Articles :

[1] Data mining temporel et prédiction de PV Dans les SMART GRIDS ; OUALI Rachid ; université d'Orléans.

[2] Data Mining: Concepts and Techniques Jiawei Han and Micheline Kamber Simon Fraser University.

[3] Fouille de données Notes de cours Ph. PREUX Université de Lille 3, philippe.preux@univ-lille3.fr 26 mai 2011.

[7] AGRAWAL R., S. R. (March 1995). "Mining Sequential Patterns", Procee-dings of the 11th International Conference on Data Engineering (IC-DE'95). Tapei, Taiwan.

[9] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining," Proc. 2000 ACM SIGKDD Int'l Conf. Knowledge Discovery in Databases (KDD '00), pp. 355-359, Aug. 2000.

[10] Jian Pei, Behzad Mortazavi-Asl, Umeshwar Dayal, "Mining Sequential Patterns by Pattern-Growth: The Prefix Span Approach", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 16, NO. 10, OCTOBER 2004

[11] Alice Marascu. Extraction de motifs séquentiels dans les flux de données. Computer Science [cs]. University Nice Sophia Antipolis, 2009. French.

[13] Fournier-Viger, P., Faghihi, U., Nkambou, R., Mephu Nguifo, E. (2012). CMRules: Mining Sequential Rules Common to Several Sequences. Knowledge-based Systems, Elsevier, 25(1): 63-76.

[14] Fournier-Vinger, P., Nkambou, R., Tseng, V. S: "RuleGrowth: Mining Sequential rules common to several sequence by Pattern growth", Proc. of the 26th Symposium on applied computing, Taiwan, pp. 954-959, ACM Press,2011, ISBN: 978-1-4503-0113-8.

[15] Philippe Fournier-Viger et al. "ERMiner: sequential rule mining using equivalence classes". In: Advances in Intelligent Data Analysis XIII. Springer, 2014, pp. 108–119.

Mémoires :

[4] : Mr. MOUNA Azzedine, Mémoire de magister en Informatique, titre : datamining distribue dans les grilles : approche règles d'association, Université des sciences et technologie d'Oran, 2012/2013.

[5] Salleb (A.). Thèse doctorat : "Recherche de motifs fréquents pour l'extraction de règles d'association et de caractérisation". Université d'Orléans, décembre 2003.

[8] ABDERRAOUF NOUASRIA, EXTRACTION D'ASSOCIATIONS LEXICALES FORTES DANS LES COMMENTAIRES, L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES, JUIN 2016.

Site web:

[12] http://data-mining.philippe-fournier-viger.com/introduction-to-sequential-rule-mining/

[16] https://fr.wikipedia.org/wiki/Java

[17] https://fr.wikipedia.org/wiki/NetBeans

[18] https://www.techopedia.com/definition/17374/one-tier-architecture

[19] http://www.philippe-fournier-viger.com/spmf/index.php

[20] http://www.philippe-fournier-viger.com/spmf/ERMiner.php