الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي –جامعة محمد البشير الابراهيمي-برج بوعريريج-

Ministère De L'enseignement Supérieur et de la Recherche Scientifique

Université Mohamed El-Bachir El-Ibrahimi-Bordj Bou Arreridj-

Faculté des Mathématiques et d'informatique

Département de L'informatique

Thèse de Doctorat

Présentée par :

SEDDIKI Imane

En vue de l'obtention du grade de Docteur en :

Domaine : Mathématiques et informatique

Filière : Informatique

Contribution à la fouille de données imparfaites

Soutenue le: 00/00/0000, devant le jury composé de :

AKHROUF Samir	Professeur	Université de M'Sila	Président
NOUIOUA Farid	Professeur	Université de B.B.A	Rapporteur
BELHADJ Foudil	M.C.A	Université de B.B.A	Examinateur
BRAHIMI Mahmoud	M.C.A	Université de M'Sila	Examinateur
LEKHFIF Abdelaziz	M.C.A	Université de Sétif 1	Examinateur

Année universitaire 2024-2025

À mes tendres parents, piliers de ma vie.

À mon cher mari, source de bonheur quotidien.

À mon frère et à mes sœurs précieuses, reflets de ma joie.

Aux joyaux de mon cœur, Aya et Amira, éclats de mon existence.

En écho à leur amour, je dédie cet éclat d'effort

Remerciements

« Après avoir remercié ALLAH le tout puissant »

Je tiens à exprimer ma profonde reconnaissance à mon directeur de thèse Pr. NOUIOUA Farid, professeur à l'université de Bordj Bou Arreridj, pour la confiance qu'il m'a témoignée en acceptant d'encadrer ce travail, pour sa disponibilité, pour son suivi et ses encouragements. Vos conseils tout au long de ce parcours de recherche ont été inestimables

Je tiens à remercier les membres du jury qui m'ont fait un grand honneur en examinant et en participant à l'évaluation de cette thèse. L'intérêt qu'ils manifesteront pour notre recherche avec leurs propositions et commentaires conduira certainement à son enrichissement.

Les mots ne suffisent pas pour exprimer ma profonde gratitude à tous les membres de ma famille, mes parents les plus chers pour leurs sacrifices, leur disponibilité et leur soutien tout au long de mes années de travail. Mon mari, qui a été la force qui m'a soutenu et m'a motivé Je remercie aussi mes sœurs et mon frère, qui m'ont soutenue moralement. Si un jour je réussirai, c'est parce que ma famille a toujours été à mes côtés, qu'Allah la protège.

Un merci à toute ma famille, à tous mes proches, et à tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail

SEDDIKI Imane

Abstract

Sequential rule mining is a technique in the field of data mining that allows one for the discovery of temporal relationships between events and for making predictions. The discovery of temporal relationships between events stored in large databases is important in many areas such as stock market analysis, e-learning, etc. In many real-world applications, such as wireless sensor networks, databases contain uncertain data due to factors such as missing, incomplete, or inaccurate information. Data uncertainty is modeled by an existential probability that is associated with each element in each sequence in the database. In this thesis, we focus on the extraction of uncertain sequential rules. We propose a new approach for extracting sequential rules from uncertain sequence databases, by evaluating the performance of the developed methods on synthetic and real data.

Keywords. Association rule, sequential database, sequential rule, uncertain data, probabilistic database, probabilistic sequence database.

مـــلخـــص

استخراج القواعد التسلسلية، المعروف أيضًا باسم استخراج الأنماط التسلسلية، هو تقنية في مجال استكشاف البيانات تسمح باكتشاف العلاقات الزمنية بين الأحداث والتنبؤ بها. يعد اكتشاف العلاقات الزمنية بين الأحداث المخزنة في قواعد البيانات الكبيرة مهمًا في العديد من المجالات مثل تحليل سوق الأسهم والتعلم الإلكتروني وما إلى ذلك. في العديد من التطبيقات الواقعية، مثل شبكات المستشعرات اللاسلكية، تحتوي قواعد البيانات على بيانات غير مؤكدة بسبب عوامل مثل المعلومات المفقودة أو غير المكتملة أو غير الدقيقة. يتم نمذجة عدم اليقين في البيانات من خلال احتمال وجودي مرتبط بكل عنصر في كل تسلسل في قاعدة البيانات. تركز هذه الأطروحة على استخراج القواعد التسلسلية غير المؤكدة. نقترح نهجًا جديدًا لاستخراج القواعد التسلسلية من قواعد بيانات التسلسلات غير المؤكدة، من خلال تقييم أداء الطرق المتطورة على البيانات الاصطناعية والواقعية.

الكلمات المفتاحية. قاعدة الارتباط، قاعدة البيانات التسلسلية، القاعدة التسلسلية، البيانات غير المؤكدة، قاعدة البيانات الاحتمالية، قاعدة بيانات التسلسلات الاحتمالية.

Résumé

La fouille des règles séquentielles est une technique dans le domaine de la fouille de données, qui permet de découvrir des relations temporelles entre des événements et d'en faire des prédictions. La découverte des relations temporelles entre les événements stockés dans de grandes bases de données est importante dans de nombreux domaines tels que l'analyse du marché boursier, l'apprentissage en ligne, etc. Dans de nombreuses applications réelles comme dans le cas des réseaux de capteurs sans fil, les bases de données contiennent des données incertaines en raison de facteurs tels que des informations manquantes, incomplètes ou inexactes. L'incertitude des données est modélisée par une probabilité existentielle qui est associée à chaque élément dans chaque séquence de la base de données. Dans Cette thèse notre attention se porte sur l'extraction des règles séquentielles incertaines. Nous proposons une nouvelle approche pour extraire des règles séquentielles à partir de bases de données de séquences incertaines, en évaluant les performances des méthodes développées sur des données synthétique et réelles.

Mots-clés. Règles d'association, base de donnée séquentielle, Règle séquentielle, Données incertaines, Base de données probabiliste, Base de données de séquences probabiliste.

Table des matières

Remerciements iii

Abstract iv

ملخص v

Résumé vi

Table des matières vii

Liste des figures xi

Liste des tableaux xii

Introduction générale 12

Chapitre 1: Etat de l'art sur la fouille de motifs fréquents et de règles d'association 16

1.1 Introduction 17

1.2 Exploration de règles d'association 17

1.2.1 Exploration d'itemsets fréquents (FIM) 17

1.2.1.1 Définition de l'extraction de motifs fréquents 18

1.2.2 Découverte de règles d'association 19

1.2.2.1 Définition du problème de découverte de règles d'association 19

1.3 Applications de l'exploration de règles d'association 20

1.4 Différentes méthodes d'exploration de règles d'association 20

1.5 Extraction de motifs séquentiels 23

1.5.1 Modèle original d'extraction de motifs séquentiels 24

1.5.1.1 Définition de problème d'extraction de motifs séquentiels 25

1.5.2 Les algorithmes d'extraction de motifs séquentiels 26

1.5.3 Applications de l'extraction de motifs séquentiels 28

1.6 Conclusion 28

Chapitre 2 :Bases de données incertaines 29
2.1 Introduction 30
2.2 Les bases de données incertaines ou probabilistes 30
2.3 Modèles de données probabilistes 31
2.3.1 Incertitude au niveau du tuple (tuple-level uncertainty) 31
2.3.2 Incertitude au niveau des attributs (attribute-level uncertainty) 31
2.4 Extraction d'itemsets fréquents incertains (uFIM) 32
2.4.1 Itemsets fréquents basés sur le support attendu 33
2.4.2 Itemsets fréquents basés sur la probabilité de fréquence 34
2.5 Algorithmes d'extraction d'itemsets fréquents incertains 34
2.5.1 Algorithmes basés sur le support attendu 35
2.5.2 Algorithmes basés sur la probabilité exacte de fréquence 36
2.5.3 Algorithmes basés sur l'approximation de la probabilité de fréquence 37
2.6 Extraction de motifs séquentiels incertains 38
2.6.1 Extraction de motifs séquentiels fréquents basés sur le support attendu 39
2.6.2 Extraction de motifs séquentiels basée sur la probabilité de fréquence 39
2.7 Conclusion 40
Chapitre 3: Extraction des règles séquentielles 40
3.1 Introduction 41
3.2 L'extraction de règles séquentielles certaines 42
3.3 Les types de règles séquentielles 43
3.3.1 Règles séquentielles standards 43
3.3.2 Règles séquentielles partiellement ordonnées 43
3.4 Algorithmes d'extraction de règles séquentielles à partir de données certaines 45
3.4.1 L'algorithme CMRules (Class-based Mining Rules) 46
3.4.2 L'algorithme RuleGrowth 47
3.4.3 L'algorithme ERMiner (Equivalence Class Based Sequential Rule Miner) 48

3.5. Comparaison entre les algorithmes 49

3.6 Domaines d'application 49

3.7 Conclusion 51

Chapitre 4: Contributions 52

4.1 Introduction 53

4.2 Contribution 1. Extraction d'itemsets séquentiels fréquents à partir de bases de séquences probabilistes (Extracting sequential frequent itemsets from probabilistic sequences database) 53

4.2.1 Résumé 53

4.2.2 L'approche proposée 54

4.2.3 Expérimentations et discussions de Résultats 56

4.2.4 Etude comparative 57

4.3 Contribution2: Extraction de Séquences Incertaines : Une Nouvelle Approche pour l'Extraction de Règles Séquentielles (Mining Uncertain Sequences : A Novel Approach to Sequential Rule Extraction) 58

4.3.1 Résumé 59

4.3.2 Définition du problème 59

4.3.3 Hypothèses et idées clés de notre approche proposée 61

4.3.4 L'approche proposée 62

4.3.4.1 Transformer la base séquentielle probabiliste en une base transactionnelle probabiliste 63

4.3.4.2 Extraire l'ensemble des motifs fréquents singletons 63

4.3.4.3 Construire l'arbre des motifs fréquents probabilistes 64

4.3.4.4 Extraction de l'ensemble des itemsets fréquents probabilistes 66

4.3.4.5 Génération de règles probabilistes 66

4.3.4.6 Filtrage des règles d'associations probabilistes 67

4.3.5 Expérimentations et discussions des résultats 68

4.3.5.1 Plateforme 68

4.3.5.2 Ensembles de données (datasets)..... 68
4.3.5.3. Tester l'algorithme sur les données synthétiques..... 69
4.3.5.4. Test de l'algorithme sur des données réelles..... 70
4.4 Conclusion 71
Conclusion générale et Perspectives..... 72
Bibliographie 74

Liste des figures

Figure 1. L'algorithme Apriori 22
Figure 2.L'algorithme FP-growth pour découvrir des itemsets fréquents sans génération de candidats. 23
Figure 3. Quelques règles trouvées avec l'algorithme CMRules 43
Figure 4.Règles séquentielles standard vs Règles séquentielles partiellement ordonnées . 44
Figure 5. L'algorithme CMRules 47
Figure 6. Un exemple d'exécution de l'algorithme CMRules 47
Figure 7. Complexité temporelle et spatiale de l'algorithme proposé comparée à l'algorithme naïf 58
Figure 8. L'approche proposée de six étapes 62
Figure 9. Évolution (de gauche à droite) de l'arbre PFP après l'insertion des trois premières transactions (Etape 3). 66
Figure 10. L'arbre PFP après avoir parcouru toute la base de données transactionnelle probabiliste 67
Figure 11. Complexité temporelle et spatiale de l'algorithme proposé sur des données synthétiques. 69
Figure 12. Complexité temporelle et spatiale de l'algorithme proposé testé sur des données réelles 70

Liste des tableaux

Tableau 1. Un exemple de base de données transactionnelle..... 19
Tableau 2. Base de données de séquences..... 24
Tableau 3. Base de données avec l'incertitude au niveau de tuple 31
Tableau 4. Base de données avec l'incertitude au niveau d'attribut 32
Tableau 5. Un exemple d'une base de données transactionnelle incertaine..... 33
Tableau 6. La distribution de probabilité du support de A du Tableau 5 34
Tableau 7.Un résumé comparatif entre les algorithmes d'extraction de règles séquentielles 49
Tableau 8. Un exemple d'une base de séquences probabilistes 60
Tableau 9. La transformation de base de données séquentielle probabiliste en base transactionnelle probabiliste 63
Tableau 10. Le support attendu pour chaque élément unique 64

Liste des abréviations

RFID: Radio Frequency Identification/ Identification par Radiofréquence

FIM: Frequent Itemset Mining/ Extraction d'itemsets fréquents

SPM: Sequential Pattern Mining/ Extraction de motifs séquentiels

SPADE: Sequential Pattern Discovery using Equivalence classes/ Découverte de motifs séquentiels en utilisant des classes d'équivalence

SPAM: Sequential Pattern Mining/ Extraction de motifs séquentiels

UFIM: Uncertain Frequent Itemset Mining/ Extraction d'itemsets fréquents incertains

USPM: Uncertain Sequential Pattern Mining/ Extraction de motifs séquentiels incertains

GSP: Generalized Sequential Pattern/ Extraction de motifs séquentiels généralisés

U-Apriori: Uncertain Apriori

FP-Growth: Frequent Pattern Growth

UH-Mine: Uncertain High-utility pattern Mine/ Extraction de motifs à haute utilité incertains

Min-sup : Minimum Support/ Support minimum

Min-conf :Minimum Confidence / Confiance minimum

Introduction générale

La fouille de données (en anglais : Data Mining) est le processus d'extraction de modèles, d'informations pertinentes et de connaissances exploitables à partir de vastes ensembles de données. L'importance de la fouille de données est de découvrir des schémas utiles et de prendre des décisions éclairées en exploitant les données disponibles. La fouille de données est un domaine de recherche important à l'intersection des bases de données, de l'apprentissage automatique et de l'intelligence artificielle.

Il convient également de noter que la fouille de données n'est pas limitée aux bases de données relationnelles ou transactionnelles traditionnelles, mais peut être appliquée à une variété de données telles que les données en continu, les données des capteurs, les bases de données textuelles, les séries temporelles ou les données séquentielles.

L'extraction de règles d'association est une technique puissante dans le domaine de la fouille de données. Cette technique consiste à découvrir des associations entre des ensembles d'éléments dans de grandes bases de données.

Les règles d'association peuvent être utilisées dans le marketing, bio-informatique, les promotions des ventes, etc. Par exemple, dans l'analyse de panier de marché, une règle d'association $\{lait, café \rightarrow sucre\}$, peut être expliquée comme suit : si un client achète du lait et du café, alors il y a de fortes chances qu'il achète également du sucre. Les règles d'association permettent de comprendre le comportement des clients et de prendre des décisions stratégiques en vue d'augmenter les ventes, telles que la promotion conjointe de produits et l'offre de réductions. L'exploration des règles d'association a été étendue aux bases de données séquentielles contenant un ensemble de séquences où les événements doivent respecter un ordre partiel, contrairement aux bases de données transactionnelles classiques où l'ordre des événements n'a pas d'importance. L'approche traditionnelle de l'exploration des règles d'association repose sur l'hypothèse selon laquelle les données à extraire sont parfaites ou sont entièrement déterminées. Cette hypothèse est fondée sur l'idée que tout bruit ou incohérence dans les données aurait été éliminé lors du processus de prétraitement des données.

Récemment les nouvelles technologies émergentes telles que les réseaux de capteurs, la technologie d'identification par radiofréquence (RFID) et les modèles d'accès au Web

produisent en permanence un grand volume de données. Ces données disponibles dans ces applications du monde réel sont pour la plupart incertaines. Par exemple, la technologie RFID génère une énorme quantité de données et le système RFID comporte de nombreux lecteurs qui capturent et collectent en permanence les lectures de différentes étiquettes. Les appareils RFID étant sensibles aux facteurs environnementaux, ils peuvent générer des données incertaines à cause des limitations du monde réel telles qu'une lecture manquante due à l'incapacité de détecter les étiquettes dans la plage de lecture, ou à des données incohérentes provoquées par la lecture de la même étiquette par plusieurs lecteurs RFID ou d'autres erreurs. Ce type de données peut complexifier le processus traditionnel d'extraction d'associations ou de règles séquentielles et révèle le besoin de nouvelles méthodes efficaces de modélisation pour traiter les données incertaines et obtenir des informations précises et utiles. L'incertitude de ces données est souvent modélisée comme une probabilité existentielle qui est soit associée à chaque séquence de la base de données ou associée à chaque élément de chaque séquence de la base de données. Par conséquent, si la probabilité d'une séquence (resp. élément) est égale à un, cela signifie que l'occurrence de cette séquence (rep. élément) est certaine.

Agrawal et Srikant [1] ont introduit l'idée des motifs séquentiels, qui est un sujet de recherche très actif dans le domaine de la fouille de données. En effet, il s'agit d'un ensemble de techniques parmi les plus populaires pour découvrir les relations temporelles entre les événements dans des séquences d'événements.

L'exploration de modèles séquentiels permet de découvrir des sous-séquences fréquentes communes à plusieurs séquences dans une base de données séquentielle. Cependant, savoir qu'une séquence d'événements apparaît fréquemment dans une base de données n'est pas suffisant pour prédire des événements. Par exemple, il est possible qu'un événement 'y' apparaisse fréquemment après un événement ''x' mais qu'il existe également de nombreux cas où 'x' n'est pas suivi de 'y', pour résoudre le problème de la prédiction, les chercheurs ont proposées l'exploration de règles séquentielles comme alternative à l'exploration de modèles séquentiels.

Une règle séquentielle énonce que si un ou plusieurs événements se produisent, il est probable que d'autres événements se produisent également, avec une confiance ou une probabilité assez élevée. La fouille de règles séquentielles peut être utilisée dans plusieurs domaines tels que l'apprentissage en ligne, l'observation météorologique, etc.

Dans ce travail, notre attention porte sur la fouille des règles séquentielles au sein d'une base de données temporelle incertaines. Nos principales contributions sont les suivantes :

Nous proposons une nouvelle approche pour extraire des motifs séquentiels et des règles séquentielles à partir de bases de données de séquences incertaines. Cette approche se compose de deux étapes principales : la première consiste à extraire un ensemble de motifs ou de règles probabilistes candidates, et la deuxième consiste à filtrer cet ensemble en utilisant les informations temporelles codés dans les séquences de données.

Le plan de cette thèse est organisé comme suit :

Le premier chapitre présente un aperçu des problèmes de L'extraction des itemsets (FIM) et L'exploration de motifs séquentiels (SPM). Il commence par définir ces deux problèmes en présentant quelques algorithmes bien connus et en discutant de nombreuses applications de la FIM et de la SPM à la fin du chapitre.

Dans le **deuxième chapitre**, nous abordons le sujet des bases de données incertaines, et nous discutons des modèles de données probabilistes utilisés pour représenter des données incertaines. Ensuite, nous définissons les problèmes de uFIM (Fouille d'Itemsets Fréquents Incertains) et uSPM (Fouille de Motifs Séquentiels Incertains), ainsi que leurs domaines d'applications respectifs. De plus, nous présentons des algorithmes bien connus pour ces problèmes afin de fournir un aperçu approfondi de leur fonctionnement et de leur utilité.

Dans le **troisième chapitre**, nous commençons par présenter les différents types de règles séquentielles qui peuvent être identifiées. Cette section explique les diverses catégories de règles séquentielles et comment elles peuvent être utilisées pour extraire des informations significatives à partir de données séquentielles. Ensuite, nous présentons une analyse approfondie des méthodes existantes pour l'extraction de règles séquentielles, en détaillant les principes de fonctionnement. Nous fournissons également un résumé comparatif de ces algorithmes. Enfin, nous explorerons divers domaines d'application de règles séquentielles : cette partie illustre comment les règles séquentielles peuvent être appliquées dans des contextes réels.

Le **quatrième chapitre** est consacré à la description des principales contributions réalisées au cours de nos recherches sur la fouille de motifs et de règles séquentielles incertaines. Nous présentons une définition de notre problème, ainsi qu'une nouvelle

approche pour extraire des règles séquentielles à partir de bases de données de séquences incertaines, Ensuite, nous décrivons les expérimentations réalisées et nous exposons l'ensemble des résultats obtenus sur des données synthétiques et réelles, tout en évaluant les performances de ces résultats en termes de complexité spatiale et temporelle.

Enfin, dans la **conclusion et les perspectives**, nous récapitulons les résultats obtenus au cours de cette recherche et proposons des pistes pour des travaux futurs.

Chapitre 1

Etat de l'art sur la fouille de motifs fréquents et de règles d'association

1.1 Introduction

L'exploration de règles d'association a été proposée pour la première fois en 1993 par Agrawal et ses collègues dans le contexte de l'analyse du panier de marché [2]. Ensuite elle a été utilisée dans de nombreux domaines d'application. Un aperçu détaillé peut être trouvé dans [3]. Ce domaine est un sujet bien étudié par de nombreux chercheurs dans l'exploration de données [4]. Par la suite, en 1995 Agrawal et Srikant [1] ont introduit des modèles séquentiels, des règles d'association avec un ordre temporel.

Dans ce chapitre, nous présentons un aperçu de l'exploration de règles d'association en deux phases : L'extraction de l'ensemble des itemsets fréquents (FIM) et la découverte de règles d'association sur la base de deux mesures : support et confiance. Ensuite, nous présentons certaines applications des règles d'association, et nous discutons quelques algorithmes bien connus. Ainsi, nous présentons le problème de l'exploration de motifs séquentiels (SPM), et nous discutons quelques algorithmes et donnons quelques applications récentes. A la fin nous terminons le chapitre par une conclusion.

1.2 Exploration de règles d'association

L'exploration de règles d'association, ou "Association rule mining", est l'une des techniques les plus populaires pour découvrir des associations intéressantes entre des éléments dans une base de données transactionnelle [2].

Le problème de l'exploration des règles d'association peut être divisé en deux étapes ou sous-problèmes. Premièrement, l'étape d'exploration d'itemsets fréquents s'intéresse à trouver tous les ensembles d'éléments (items) qui figurent fréquemment dans la base de donnée, et deuxièmement, la découverte de règles d'association qui consiste à trouver les associations entre les itemsets fréquents découverts à l'issue de la première phase, nous expliquons en détail chacun de ces points ci-dessous.

1.2.1 Exploration d'itemsets fréquents (FIM)

Le problème d'extraction d'itemsets fréquents dans une base de données transactionnelle est défini par *R*. Agrawal et ses collègues [2] [5] comme suit : soit *I* un ensemble d'éléments (items), un itemset est un sous-ensemble de *I*. Nous supposons que les items de *I* sont ordonnés.

Une base de données transactionnelle est un ensemble de transactions où chaque transaction est représentée par un tuple composé d'un identifiant unique TID et un sous-ensemble d'éléments distincts X tel que $X \subseteq I$. $|X|$ est le nombre d'items dans l' itemset X.

Le but de l'extraction d'itemsets fréquents est de trouver des motifs qui se répètent de façon fréquente dans la base. Ces motifs peuvent révéler des informations implicites utiles et servent ensuite de base pour détecter des associations intéressantes entre les itemsets fréquents. Il existe plusieurs mesures qui peuvent être utilisées pour évaluer l'intérêt des motifs. La mesure la plus utilisée dans ce contexte est le support d'un itemset. Le support (absolu) d'un itemset X dans une base T, noté par $Support(X)$, est le nombre de transactions où X est présent, ce qui est formellement représenté par :

$$Support(X) = |\{t \in T \mid X \subseteq t\}| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

où t est une transaction dans la base de transactions T.

Notons que certains auteurs définissent le support d'un ensemble d'éléments X comme un rapport (support relatif) $relSup(X) = sup(X)/|D|$ où D est le nombre de transactions dans la base de données. Un itemset est considéré comme fréquent si son support ($Support(X)$) n'est pas inférieur à un seuil de support minimal ($minsup$) défini par l'utilisateur. La tâche de découverte des itemsets fréquents consiste donc à détecter l'ensemble de tels itemsets dans l'espace de tous les itemsets possibles.

1.2.1.1 Définition de l'extraction de motifs fréquents

Étant donné une base de données D comportant n transactions, et un seuil de support minimum θ défini par l'utilisateur ($0 \leq \theta \leq n$). L'extraction des itemsets fréquents est le processus qui consiste à trouver tous les ensembles possibles qui ont un support non inférieur à θ [6].

Exemple 1. Considérons un échantillon de la base de données D représenté dans le Tableau 1 et supposons que le seuil minimal de support est : $\theta = 2$.

L'ensemble $\{a\}$ est fréquent car $Support(\{a\}) = 2$ (l'item a est présent dans t_2 et t_3) alors que $\{c\}$ ne l'est pas car le $Support(\{c\}) = 1$ (l'item c est présent seulement dans t_2).

Tableau 1. Un exemple de base de données transactionnelle

TID	Contenu de la transaction
T_1	*{B, D}*
T_2	*{A, C, D}*
T_3	*{A, B, D}*

1.2.2 Découverte de règles d'association

Une règle d'association est une implication de la forme $A \Rightarrow B$ où $A, B \subseteq I$ et $A \cap B = \emptyset$. Une telle règle signifie que si une transaction supporte l'itemset A, alors il y a de fortes chances qu'elle supportera également l'itemset B. La confiance d'une règle d'association $A \Rightarrow B$, notée $Confiance\ (A \Rightarrow B)$ est définie comme suit :

$$Confiance(A \Rightarrow B) = Support\ (A \cup B)/Support\ (A) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Notons que le support d'une règle d'association $A \Rightarrow B$ est défini simplement comme étant le support de $A \cup B$: $Support(A \Rightarrow B) = Support\ (A \cup B)$.

La tâche de découverte de règles d'association est formellement défini comme suit [6].

1.2.2.1 Définition du problème de découverte de règles d'association

Étant donné une base de données D comportant n transactions, et deux seuils définis par l'utilisateur, un seuil de support θ, $0 \leq \theta \leq n$, et un seuil de confiance τ. $0 \leq \tau \leq 1$. Le problème consiste à trouver toutes les règles d'association qui ont un support et une confiance non inférieurs à θ et τ, respectivement.

Exemple 2. Dans l'exemple de base de données D qui est présentée dans le Tableau 1 considérons que le seuil de confiance est : $\tau = 90\%$. $\{b, d\}$ est un itemset fréquent puisque $Support\ (\{b: d\}) = 2$. La confiance de la règle $\{b\} \Rightarrow \{d\}$ est de 100% car pour deux transactions sur deux dans D où $\{b\}$ apparaît, $\{d\}$ apparaît aussi. On déduit alors que la règle $\{b\} \Rightarrow \{d\}$ est une règle d'association valide.

Le problème de l'extraction de règles d'association dépend fortement de celui de l'extraction de itemsets fréquents car la deuxième étape de l'exploration des règles d'association est beaucoup moins coûteuse que la première est c'est pourquoi la plupart des chercheurs se concentrent sur la première étape.

1.3 Applications de l'exploration de règles d'association

L'exploration de règles d'association peut être appliquée à des ensembles de données de différents domaines d'application. Comme nous l'avons évoqué précédemment, l'application classique est celle de l'analyse du panier du marché, mais la technique peut être appliquée à de nombreux autres domaines dont on présente ci-dessous quelques exemples.

- **Diagnostic médical**

Le diagnostic médical représente un domaine d'application prometteur pour les règles d'association. Dans ce contexte, les règles d'association sont utilisées pour découvrir des relations intéressantes entre les symptômes, les diagnostics, les traitements et les résultats des patients pour connaître la probabilité d'apparition d'une maladie et permettre ainsi au médecin de prendre une meilleure décision pour le patient [7].

- **Exploration de texte (Text mining)**

La fouille de texte (text mining) est un autre domaine d'application intéressant pour les règles d'association. Par exemple, le système développé par Don et al. [8], nommé FeatureLens, permet une exploration visuelle des motifs textuels fréquents dans les phrases pour découvrir des insights cachés dans de vastes collections de documents textuels. Les règles d'association, dans ce contexte, sont utilisées pour identifier des associations fréquentes entre des mots, des phrases, ou des concepts au sein d'un corpus de texte.

- **Bio-informatique (Séquences de protéines)**

Les séquences de protéines désignent l'ordre spécifique dans lequel les acides aminés sont arrangés pour former une protéine. Il existe une très forte dépendance du fonctionnement des protéines vis-à-vis des acides aminés. Les règles d'association générées entre les différents acides aminés d'une protéine peuvent aider à comprendre la composition des protéine [7].

1.4 Différentes méthodes d'exploration de règles d'association

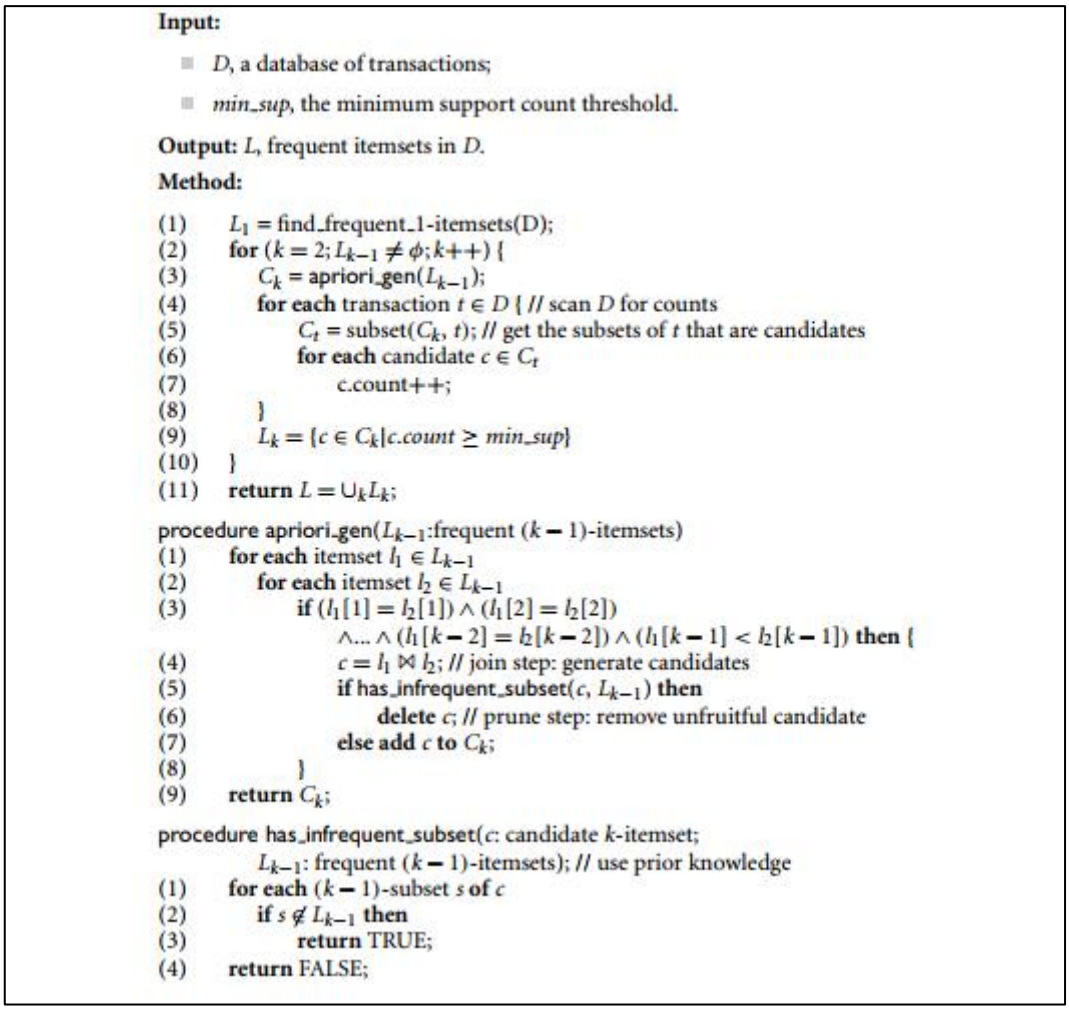
Les algorithmes fondamentaux, notamment *Apriori* et *FP-Growth* sont utilisés pour identifier d'abord les ensembles d'éléments fréquents. Une fois que ces ensembles sont

identifiés, les règles d'association peuvent être extraites en évaluant les relations de ces ensembles fréquents en fonction de leurs mesures de support et de confiance**.**

L'algorithme Apriori. C'est un algorithme populaire utilisé pour l'extraction de motifs fréquents proposé par Agrawal et Srikant [9]. C'est le premier algorithme proposé pour la découverte des itemsets fréquents et des règles d'association. Il utilise une méthode itérative pour découvrir les motifs fréquents dans un ensemble de transactions. Il est basé sur une approche en largeur d'abord [10].

Comme le montre la Figure 1 qui représente le pseudo code d'Apriori, l'algorithme commence par analyser la base de données pour trouver tous les ensembles d'éléments fréquents de longueur un et capturer tous les ensembles d'éléments qui ont un support supérieur au seuil minimum de support. Ensuite, l'algorithme génère tous les ensembles d'items candidats à partir des ensembles fréquents de l'étape précédente et capture ceux qui ont un support supérieur au seuil de minimum support. Cela nécessite une analyse complète de la base de données. Le processus se répète jusqu'à ce qu'aucun autre ensemble d'item fréquent ne puisse être identifié [10].

Pour améliorer l'efficacité de la génération de motifs fréquents de manière progressive, l'algorithme *Apriori* utilise une propriété importante appelée "propriété d'anti-monotonie" ou "propriété de fermeture descendante". Selon cette propriété, tous les sous-ensembles non vides d'un ensemble fréquent sont également fréquents. Inversement, si un itemset n'est pas fréquent, alors tous ses sur-ensembles ne seront pas fréquents non plus. En utilisant cette propriété, l'algorithme réduit l'espace de recherche et augmente l'efficacité de la recherche de motifs fréquents [10].



Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)     Ck = apriori_gen(Lk−1);
(4)     for each transaction t ∈ D { // scan D for counts
(5)       Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)       for each candidate c ∈ Ct
(7)         c.count++;
(8)     }
(9)     Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)     for each itemset l2 ∈ Lk−1
(3)       if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
            ∧... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)         c = l1 ⋈ l2; // join step: generate candidates
(5)         if has_infrequent_subset(c, Lk−1) then
(6)           delete c; // prune step: remove unfruitful candidate
(7)         else add c to Ck;
(8)       }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
        Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)     if s ∉ Lk−1 then
(3)       return TRUE;
(4)   return FALSE;
```

Figure 1. L'algorithme *Apriori[11]*

L'algorithme FP-Growth. Cet algorithme a été proposé pour surmonter certains inconvénients de l'algorithme Apriori. Notamment la nécessité de multiples parcours de la base de données et l'augmentation de l'espace de recherche requis. L'algorithme de croissance *FP-Growth* découvre les itemsets d'éléments fréquents sans génération d'itemsets candidats. Dans cet algorithme, il est nécessaire de construire un arbre *FP*, appelé arbre de fréquence (FP-tree), qui contient des ensembles d'éléments fréquents. L'arbre *FP* est construit à partir des transactions de la base de données et il est utilisé pour représenter les ensembles d'éléments fréquents de manière compacte, ce qui permet une recherche efficace des motifs fréquents [12].

Algorithm: FP_growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

Input:

- D, a transaction database;
- *min_sup*, the minimum support count threshold.

Output: The complete set of frequent patterns.

Method:

1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database D once. Collect F, the set of frequent items, and their support counts. Sort F in support count descending order as L, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in D do the following. Select and sort the frequent items in *Trans* according to the order of L. Let the sorted frequent item list in *Trans* be $[p|P]$, where p is the first element and P is the remaining list. Call insert_tree($[p|P]$, T), which is performed as follows. If T has a child N such that $N.item\text{-}name = p.item\text{-}name$, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link to the nodes with the same *item-name* via the node-link structure. If P is nonempty, call insert_tree(P, N) recursively.
2. The FP-tree is mined by calling FP_growth(*FP_tree*, *null*), which is implemented as follows.

```
procedure FP_growth(Tree, α)
(1)  if Tree contains a single path P then
(2)     for each combination (denoted as β) of the nodes in the path P
(3)        generate pattern β∪α with support_count = minimum support count of nodes in β;
(4)  else for each a_i in the header of Tree {
(5)     generate pattern β = a_i∪α with support_count = a_i.support_count;
(6)     construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)        call FP_growth(Tree_β, β); }
```

Figure 2.L'algorithme FP-growth pour découvrir des itemsets fréquents sans génération de candidats*[11]*.

1.5 Extraction de motifs séquentiels

La fouille de motifs (pattern mining) est un domaine très populaire car elle a des applications dans de nombreux domaines. Elle consiste à découvrir des motifs intéressants dans des bases de données. Ce domaine de recherche a émergé dans les années 1990 avec l'article fondateur d'Agrawal et Srikant [9] avec l'algorithme Apriori. La tâche d'exploration de motifs séquentiels a été ensuite proposée car les techniques classiques de fouille d'itemsets fréquents ne permettent pas de traiter convenablement les données contenant des informations temporelles. Dans de nombreux domaines, l'ordre des événements ou des éléments est important pour trouver des modèles utiles. Par exemple, pour analyser des textes, il est souvent pertinent de considérer l'ordre des mots dans les phrases [13]. Dans la détection d'intrusion réseau, l'ordre des événements est également important[14]. Plus généralement, l'extraction de motifs et de règles séquentiels à partir de bases de séquence a de nombreux domaines d'application tels que la bio-informatique, l'apprentissage en ligne, l'analyse du panier de la ménagère, les réseaux complexes, l'analyse des visites des pages Web, etc.

Par exemple, considérons une base de données d'accès au Web d'un site populaire, où un utilisateur Web et une page Web sont considérés comme un objet et un attribut respectivement. Les modèles découverts sont les séquences des pages les plus fréquemment consultées sur ce site. Ce type d'information peut aider à améliorer la conception d'un système, comme une meilleure structure d'hyperliens entre les pages corrélées, et conduire à de meilleures décisions de marketing, comme le placement stratégique de la publicité [15].

1.5.1 Modèle original d'extraction de motifs séquentiels

Le modèle original d'extraction de motifs séquentiels est formellement défini comme suit [1]. Soit $I = \{i_1, i_2, \dots, i_k\}$ l'ensemble de tous les items. Un sous-ensemble de I est appelé un ensemble d'items (itemset ou événement). Une séquence $s = s_1, s_1, \dots, s_m$ $(s_i \subseteq I)$ est une liste ordonnée. La taille $|s|$, d'une séquence s est le nombre d'itemsets dans la séquence. La longueur $l(s)$, est le nombre total d'items dans la séquence.

Une séquence $\alpha = (\alpha_1, \dots, \alpha_m)$ est une sous-séquence d'une autre séquence $\beta = (\beta_1, \dots, \beta_r)$, notée $\alpha \sqsubseteq \beta$ si et seulement si $\exists\, k_1, k_2, \dots, k_m$, tels que $1 \leq k_1 < k_2 < \cdots < k_m \leq r$ et $\alpha_1 \subseteq \beta_{k_1}$, $\alpha_2 \subseteq \beta_{k_2}$,…, $\alpha_m \subseteq \beta_{k_m}$. Nous appelons également β une super-séquence de α et on dit aussi β contient α.

Soit $I = \{i_1, i_2, \dots, i_n\}$ l'ensemble de tous les items. Une base de données de séquences, $D = \{d_1, d_2, \dots, d_k\}$, est un ensemble de séquences où chaque séquence d_x est une liste ordonnée de transactions : $d_x = \{X_1, X_2, \dots, X_h\}$ tel que $X_1, X_2, \dots, X_h \subseteq I$. Chaque séquence est associée à un identificateur (TID). Pour simplifier, disons que l'identifiant de d_x est simplement x. $|D|$ représente le nombre de séquences dans la base de données D. Par exemple, le Tableau 2 est un exemple d'une base de séquences qui contient quatre séquences.

Tableau 2. Base de données de séquences

Id (identifiant)	Séquence
1	$\langle\{a\},\{c,d\},\{g\}\rangle$
2	$\langle\{a\},\{b\},\{f,g\}\rangle$
3	$\langle\{c\},\{f,e\}\rangle$
4	$\langle\{a,b\},\{c\},\{f,g\},\{g\}\rangle$

Le but de l'extraction de motifs séquentiels est de découvrir des sous-séquences fréquentes ou intéressantes dans une base de données de séquences. C'est-à-dire des relations séquentielles entre des éléments intéressants pour l'utilisateur. Il existe plusieurs mesures qui peuvent être utilisées pour évaluer l'intérêt d'une sous-séquence. La plus utilisée est celle du support.

Le support d'une séquence α dans une base de séquences D est le nombre de séquences dans D qui contiennent α :

$$Support(\alpha) = |\{d \mid d \in D \; et \; \alpha \sqsubseteq d\}|. \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Par exemple, le support de la séquence 3 est 1 dans la base du Tableau 2 car seule la séquence 3 dans la base de données contient ⟨{c},{f,e}⟩. Notez que certains auteurs définissent le support d'une séquence X comme un rapport. Ce rapport appelé le support relatif est donné par : $relSup(\alpha) = Support(\alpha)/|SDB|$, i.e. le nombre de séquences contenant α divisé par le nombre total de séquences présentes dans la base de données [1].

Étant donné un seuil de support minimal *minsup*. L'ensemble des motifs séquentiels fréquents (*FS*), comprend toutes les séquences dont le support n'est pas inférieur à *minsup*. Étant donné une base de données de séquences et un seuil *minsup* spécifié par l'utilisateur, le problème de l'extraction de motifs séquentiels consiste à trouver toutes les sous-séquences fréquentes dans la base de données. L'hypothèse étant que les sous-séquences fréquentes sont intéressantes à l'utilisateur car elles peuvent révéler des informations implicites utiles.

1.5.1.1 Définition de problème d'extraction de motifs séquentiels

Étant donné une base de séquences D comportant n séquences et un seuil minimum de support défini par l'utilisateur θ, $0 \leq \theta \leq m$. Le problème consiste à trouver toutes les séquences fréquentes dans D.

Exemple 3. Prenons la base de séquences donnée dans le Tableau 2. Prenons comme seuil minimum de support $\theta = 2$. On peut vérifier que la sous-séquence $\langle\{a\}\rangle$ est fréquente car $Support(\langle\{a\}\rangle, D) = 2$ alors que $\langle\{a, e\}\rangle$ ne l'est pas car $Support(\langle\{a, e\}\rangle, D) = 1$.

Afin de résoudre le problème de la découverte de modèles séquentiels, une approche naïve consiste à calculer le support de toutes les sous-séquences possibles dans une base de données de séquences pour ensuite ne sortir que celles qui respectent la contrainte de support minimum spécifiée par l'utilisateur. Mais l'approche naïve est inefficace car le nombre de sous-séquences peut-être très grand. Une séquence contenant q éléments dans une base de séquences peut avoir jusqu'à $2^q - 1$ sous-séquences distinctes[16]. Il existe de nombreux algorithmes conçus pour découvrir des motifs séquentiels en évitant d'explorer l'intégralité de l'espace de recherche. Ils seront discutés dans la suite de ce chapitre.

1.5.2 Les algorithmes d'extraction de motifs séquentiels

Les algorithmes d'extraction de motifs séquentiels sont divisés en plusieurs catégories, chaque catégorie contient de nombreux algorithmes qui ont été développés pour assurer un bon niveau d'efficacité. Les principaux types d'algorithmes d'extraction de motifs séquentiels sont [16]:

- ***Méthode basée sur la recherche en largeur d'abord (BFS : Breadth-first Search)***

  Ces méthodes comportent une phase de génération de candidats basée sur la propriété d'anti-monotonie qui s'énonce comme suit : Pour deux séquences quelconques s et t :

  si $s \sqsubseteq t$ alors $Support\ (s) \geq Support\ (t)$… ……(4)

  AprioriAll, AprioriSome et *DynamicSome*, sont trois algorithmes ont été proposés pour la première fois par Agrawal et Srikant dans leurs travaux pionniers sur l'extraction de motifs séquentiels [1]. Le premier algorithme, *AprioriAll* a été inspiré de l'algorithme *Apriori* et les deux derniers algorithmes ont été développés pour découvrir uniquement les motifs séquentiels maximaux. La performance *d'AprioriAll* s'est montrée meilleur que les deux autres algorithmes.

  En 2000 zaki et ses collègues [15] ont proposé l'algorithme *GSP* qui fonctionne de façon proche de celui d'*AprioriAll* mais qu'il est plus rapide. Il commence par scanner la base de données pour identifier les séquences fréquentes de taille 1 (motifs

séquentiels contenant un seul item). Ensuite, il crée des séquences de 2 items en ajoutant des items à la suite ou à l'intérieur des séquences de taille 1, puis génèrent des séquences de 3 items à partir des séquences de taille 2, et ainsi de suite jusqu'à ce qu'aucune nouvelle séquence ne puisse être générée. Cet algorithme a plusieurs limites importantes :

- Analyses multiples de la base de données.
- Génération de candidats inexistants.
- Maintenir les candidats en mémoire [16] .
- Cet algorithme explorera toutes les séquences possibles contenant x items ou moins. Si une base de données contient m items, ce nombre peut être supérieur à 2^n.

- ***Méthode basée sur la recherche en profondeur d'abord (DFS : Depth First Search) :***

Cette méthode effectue la génération de candidats en utilisant la structure *IDList* et sa variation SPADE [17].

La méthode SPAM commence par les séquences contenant des items uniques (par exemple, $\{a\}$, $\{b\}$ et $\{c\}$), puis effectue de manière récursive des extensions internes (*i-extensions*) et des extensions séquentielles (*s-extensions*) avec l'une de ces séquences pour générer des séquences plus grandes. Lorsqu'un motif ne peut plus être étendu, l'algorithme revient en arrière pour générer d'autres motifs en utilisant d'autres séquences.

La méthode basée sur la croissance de modèles *FreeSpan* est un algorithme qui vise à réduire la génération de sous-séquences candidates. Il a été proposé par han et al [18]. Son idée générale est d'utiliser des éléments fréquents pour projeter de manière récursive des bases de séquences dans un ensemble de bases de données projetées plus petites basées sur les ensembles fréquents actuellement exploités, et de développer des fragments de sous-séquence dans chaque base de données projetée respectivement.

Un autre algorithme basé sur la projection nommé *PrefixSpan* [19] s'inspire de l'algorithme *FPGrowth* pour l'exploration d'itemset fréquents. Son idée générale est d'examiner uniquement les sous-séquences de préfixes et de projeter uniquement leurs sous-séquences de postfixes correspondantes dans des bases de données projetées.

1.5.3 Applications de l'extraction de motifs séquentiels

L'extraction de motifs séquentiels intéressants à partir des données séquentielles peut être utile pour la prédiction ou l'interprétation, Les méthodes d'extraction de modèles séquentiels ont été appliquées avec succès dans de nombreux domaines comme [20]:

- Bourses et marchés : tendance des cours boursiers.
- Traitement médical : découvrir des modèles dans les historiques des dossiers médicaux pour améliorer le niveau de diagnostic.
- Télécommunication : trouver des schémas d'alarme de réseau, des schémas d'appels téléphoniques.
- Bio-Informatique : prédiction de la fonction des protéines et la reconnaissance du pli des protéines.
- Administration : identification des échecs du plan.
- Service Web : découverte des modèles d'accès des utilisateurs.
- Sécurité de l'information : analyse du comportement de l'utilisateur.
- Catastrophes naturelles : prévision des tremblements de terre.*
- etc.

1.6 Conclusion

Dans ce chapitre nous avons donné un aperçu sur l'exploration de règles d'association et nous avons présenté le problème de la fouille de motifs séquentiels (SPM). Le chapitre a commencé avec le sujet de l'extraction d'itemsets fréquents et de règles d'association à partir de bases de données transactionnelles classiques. Nous en avons discuté les principaux algorithmes et applications. Le chapitre s'est ensuite poursuivi en examinant la tâche de l'extraction de motifs séquentiels. Nous avons donné une définition formelle de cette tâche avant de discuter les principaux algorithmes utilisés ainsi que les champs d'applications potentielles où l'exploration de motifs séquentiels peut être utile.

Chapitre 2

Bases de données incertaines

2.1 Introduction

Les nouvelles applications du monde réel apparues récemment telles que les réseaux de capteurs contiennent un grand volume de données. Ces données sont généralement incertaines c.-à-d. sont souvent incomplètes/ imprécises /incorrectes /incohérentes pour diverses raisons. La propagation de données incertaines nécessite des applications adaptées de traitement de données incertaines.

Il existe deux classes de méthodes d'exploration de données : les méthodes prédictives et les méthodes descriptives. Parmi les tâches d'exploration de données les plus fondamentales, on trouve la classification, l'analyse des valeurs aberrantes (exploration de données prédictive), le clustering et l'extraction des motifs fréquents et des règles d'association [11]. Ce dernier domaine de recherche a émergé dans les années 1990 avec l'article fondateur d'Agrawal et Srikant [9]. Tout d'abord, nous avons besoin d'un modèle pour représenter des données incertaines. Nous présentons ensuite les problèmes d'extraction d'itemsets fréquents (resp. de motifs séquentiels fréquents) à partir de données incertaines (uFIM, resp. uSPM) ainsi que les algorithmes associés à l'extraction d'itemsets fréquents.

2.2 Les bases de données incertaines ou probabilistes

Les informations de base de données incertaines sont modélisées par un espace probabiliste dont les résultats possibles sont toutes les instances certaines traditionnelles. L'interprétation d'une base de données incertaine se fait à l'aide de modèle de mondes possibles appelé modèle complet où chaque monde possible est une instance de données déterministe. Cependant, le nombre d'instances de base de données peut être exponentiel par rapport à la taille de la base de données initiale car ce modèle permet de générer toutes les instances de données possibles, Ce modèle est donc extrêmement coûteux [21]. Pour surmonter ce problème il faudrait proposer d'autres techniques alternatives pour atteindre l'objectif (ici trouver les itemsets ou les motifs séquentiels fréquents), sans être obligé d'énumérer tous les mondes possibles. Il existe deux façons d'exprimer l'incertitude dans les bases de données probabilistes. Nous les présentons dans ce qui suit.

2.3 Modèles de données probabilistes

Selon [22], il y a deux façons d'exprimer l'incertitude dans les bases de données probabilistes :

2.3.1 Incertitude au niveau du tuple (tuple-level uncertainty)

Dans une base de données incertaine, à chaque tuple (ou transaction) est associée une probabilité existentielle. Le Tableau 3 illustre un exemple de ces bases de données.

Tableau 3. Base de données avec l'incertitude au niveau de tuple

ID	*Event*	*Probabilité*
T_1	*{1}*	*1.0*
T_2	*{2,3}*	*0.6*
T_3	*{3,4}*	*0.8*

2.3.2 Incertitude au niveau des attributs (attribute-level uncertainty)

L'incertitude concerne la valeur d'un attribut d'une transaction. C'est-à-dire que chaque attribut possède sa propre valeur de probabilité existentielle.

Le Tableau 4 montre un exemple de bases de données d'événements avec une incertitude au niveau des attributs.

Tableau 4. Base de données avec l'incertitude au niveau d'attribut

ID	*valeurs d'attribut*
T_1	$(x_1, 0.2)$ $(x_2, 0.3)$ $(x_3, 0.5)$
T_2	$(x_4, 0.6)$ $(x_2, 0.4)$
T_3	$(x_5, 1.0)$

2.4 Extraction d'itemsets fréquents incertains (uFIM)

L'extraction d'itemsets fréquents à partir de données incertaines est une extension de la tâche classique d'extraction d'itemsets fréquents (FIM) pour prendre en compte l'incertitude concernant les données. Dans les bases de données incertaines, la définition des ensembles d'items fréquents repose sur l'objectif de découvrir des combinaisons d'items qui se produisent fréquemment ensemble au sein de l'ensemble de données incertain. Cela dépend de deux explications sémantiques différentes : (1) une explication basée sur la notion de support attendu (expected support), visant à filtrer les itemsets ayant un support attendu au moins égal à un seuil de support minimum spécifié par l'utilisateur [2], et (2) une explication basée sur la probabilité de fréquence d'un itemset. C'est une approche qui fait souvent intervenir des modèles probabilistes exacts ou approximatifs pour quantifier la probabilité de fréquence d'itemsets. Cela peut inclure des distributions de probabilités associées à la présence ou à l'absence d'items. Les deux définitions prennent en compte le support comme une variable aléatoire discrète. L'étude de l'exploration d'itemsets à partir de données incertaines peuvent trouver des applications dans divers domaines tels que l'exploration de texte, l'analyse client et la bio-informatique. Dans ce contexte nous expliquons les deux définitions existantes des itemsts fréquents dans une base de données incertaine.

Une base de données transactionnelle incertaine D contient un ensemble de transactions $\{t_1, t_1, \dots, t_n\}$, où chaque transaction se compose d'un ensemble d'items (itemset) X, Chaque élément de l'ensemble X est formé d'un item x_i et une valeur de probabilité existentielle de cet item noté p_i qui est comprise entre 0 et 1. Il est noté par $x_i(p_i) : X = \{x_1(p_1), x_2(p_2), \dots, x_s(p_s))\}$.

Le Tableau 5 donne un exemple d'une base transactionnelle probabiliste. Notons que nous utilisons ici l'approche où l'incertitude est représentée au niveau des attributs.

Tableau 5. Un exemple d'une base de données transactionnelle incertaine

Tid	*Transaction*
T_1	$\{A\ (0.3), B\ (0.8), C\ (0.9)\}$
T_2	$\{(B\ (0.1), (C\ (0.5)\}$
T_3	$\{A\ (0.3), D\ (0.8), B(0.9)\}$

2.4.1 Itemsets fréquents basés sur le support attendu

Le support attendu (en Anglais : expected support) d'un itemset X, noté $expSup(X)$ dans une base de données incertaine D est la somme des probabilités existentielles de l'itemset X dans chaque transaction t de D, noté $P(X, D)$ comme indiqué dans l'équation suivante :

$$expSup(X) = \sum_{t \in D} P(X, t) \quad \text{.....(5)}$$

Notons que la probabilité existentielle de l'itemset $X = \{x_1(p_1), x_2(p_2), \dots, x_s(p_s))\}$ dans une transaction t est simplement le produit des probabilités existentielles de chaque item dans t :

$$P(X, t) = \prod_{x_i(p_i) \in X} p_i \quad \text{..... (6)}$$

Notons que $P(X, t)$ est strictement positive si et seulement si chaque item de X possède une probabilité existentielle strictement positive dans t.

Un itemset X est dit fréquent dans une base de données probabiliste si et seulement si son support attendu $expSup(X)$ est supérieur ou égal à un seuil minimum de support θ défini par l'utilisateur [23], i.e., $expSup(X, D) \geq \theta$.

Exemple

Étant donné la base de données incertaine D présentée dans le Tableau 5 et un seuil minimum de support : $\theta = 0.9$. $\{B\}(1.8)$ et $\{C\}(1.4)$ sont deux itemsets fréquents basés sur le support attendu où le nombre entre parenthèses est le support attendu de l'itemset en question.

2.4.2 Itemsets fréquents basés sur la probabilité de fréquence

Etant donné une base de transactions incertaine et un seuil minimum de support θ. La probabilité de fréquence d'un itemset X dans D, notée $Pr(X)$ indique la probabilité que X soit fréquent dans D, autrement dit, la probabilité que le support de X soit supérieur ou égal au seil minimum de support θ. Ceci est exprimé par la formule suivante :

$$Pr(X) = P(Sup(X) \geq \theta) \text{..... (7)}$$

Maintenant, étant donné en plus un seuil probabiliste δ, un itemset X est dit fréquent probabiliste si la probabilité de fréquente de X est n'est pas inférieure au seuil minimum de probabilité δ [24].

$$X \text{ est fréquent ssi } Pr\,(X) \geq \delta\ c.-\text{à}-d.\ P(Sup(X)\ \geq\ \theta\ ) \geq \delta \ldots\ldots\ldots\ldots\ldots\ldots(8)$$

Exemple. Considérons à nouveau la base de données incertaine D donnée dans le Tableau 5 et considérons comme seuil minimum de support $\theta = 1$ et comme seuil de probabiliste $\delta = 0{,}6$ la distribution de probabilité du support de l'itemset $\{A\}$ est donnée dans le Tableau 6. ainsi, la probabilité de fréquence de $\{A\}$ est : $Pr(\{A\}) = P(Sup(A) \geq 1) = P(Sup(A) = 1) + P(Sup(A) = 2) = 0.42 + 0.09 > 0.6 = \delta$, $\{A\}$ est un itemset fréquent probabiliste

Tableau 6. La distribution de probabilité du support de $\{A\}$ du Tableau 5

Sup(A)	*0*	*1*	*2*
Probabilité	*0.49*	*0.42*	*0.09*

2.5 Algorithmes d'extraction d'itemsets fréquents incertains

Les algorithmes d'extraction d'ensembles d'éléments fréquents à partir de données probabilistes peuvent être classés en trois catégories : La première catégorie est constituée des algorithmes fréquents basés sur le support attendu. Pour chaque itemset, ces algorithmes considèrent uniquement le support attendu pour mesurer sa fréquence. La complexité du calcul du support attendu d'un ensemble d'éléments est O(N), où N est le nombre de transactions[25]. La deuxième catégorie concerne les algorithmes fréquents probabilistes exacts. Les algorithmes de cette catégorie découvrent tous les ensembles d'éléments fréquents probabilistes et rapportent la probabilité fréquente exacte pour chaque ensemble d'éléments ces algorithmes doivent dépenser au moins un coût de calcul O (NlogN) pour chaque ensemble d'éléments[25]. La troisième catégorie est celle des algorithmes fréquents probabilistes approximatifs qui construit en fait par les deux définitions différentes d'ensembles d'éléments fréquents sur des bases de données incertaines[25]. Nous explorons ensuite quelques algorithmes bien connus de chaque catégorie

2.5.1 Algorithmes basés sur le support attendu

Parmi les algorithmes d'extraction d'itemsets fréquents basés sur le support attendu, les plus connus sont : U-Apriori [23], UFP- growth [26] et UH-Mine [27].

- ***Algorithme U-Apriori***

  L'algorithme U-Apriori a été initialement proposé par Chui et ses collègues en 2007 [23]. Il fonctionne de la manière générer-et-tester et utilise la recherche en largeur d'abord. C'est une version d'Apriori adaptée aux données incertaines. L'algorithme trouve tous les itemsets fréquents basés sur le support attendu de taiile 1 puis il joint de manière récursive les itemsets fréquents basés sur le support attendu de taiile *i* (i-itemset) pour créer des itemsets candidats de taille $i + 1$. Ces candidats sont ensuite testés pour détecter ceux d'entre eux qui sont des itemsets fréquents basés sur le support attendu de taille $i + 1$. Le processus prend fin dès qu'il n'est plus possible de générer d'itemsets fréquents basés sur le support attendu. Semblable à l'algorithme Apriori, l'algorithme U-Apriori ne donne pas de bonnes performances lorsqu'il est appliqué aux grandes bases de données et que les probabilités existentielles des itemsets candidats prennent de faibles valeurs.

- ***Algorithme UFP- Growth [26]***

  C'est une extension de l'algorithme FP-Growth [28] basé sur une stratégie de diviser pour régner et qui applique la recherche en profondeur d'abord. Tout comme l'algorithme FP-Growth, l'algorithme UFP-Growth construit d'abord un arbre d'index appelé arbre UFP. Ensuite, sur la base de l'arbre UFP, l'algorithme construit de manière récursive des sous arbres conditionnels et trouve les itemsets fréquents basés sur le support attendu.

- ***L'algorithme UH-Mine [27]***

  C'est une extension de l'algorithme H-Mine [29] au cas de données incertaines. Il est aussi basé sur une approche de diviser pour régner en utilisant une recherche en profondeur d'abord. D'abord, il analyse la base de données incertaine et trouve tous les itemsets fréquents de taille 1. Ensuite, l'algorithme crée et maintient une table principale incluant tous les itemsets fréquents basés sur le support attendu. Toutes les transactions sont insérées dans la structure de données UH-Struct après la construction de la table principale. Par la suite, l'algorithme construit les tables

principales de manière répétitive où les itemsets distincts sont des préfixes et génère les itemsets fréquemment rencontrés. De façon générale, UH-mine fonctionne mieux que UFP-Growth dans le cas de bases de données qui ont de faibles valeurs de probabilité attribuées aux items et l'algorithme U-Apriori est plus rapide que l'algorithme UF-Growth et UH-Mine dans les bases de données incertains denses.

2.5.2 Algorithmes basés sur la probabilité exacte de fréquence

Deux principaux algorithmes représentatifs existent dans la littérature pour l'extraction d'itemsets fréquents basé sur le calcul exact de la probabilité de fréquence : l'algorithme basé sur la programmation dynamique (DP) et l'algorithme basé sur la stratégie diviser-pour-régner (DC). Les deux algorithmes sont du style d'Apriori. Nous les présentons brièvement dans ce qui suit :

- ***Algorithme basé sur la programmation dynamique***

  Pour trouver les itemsets fréquents probabilistes, Bernecker et ses collègues ont proposé un premier algorithme qui utilise une approche basée sur la programmation dynamique. L'algorithme propose un schéma de programmation dynamique basé sur des équations récursives pour le calcul incrémental de la probabilité de fréquence d'un itemset. Ce schéma est basée sur les équations suivantes [24] :

$$Pr_{\geq i,j}(X) = Pr_{\geq i-1,j-1}(X) \times Pr\big(X \subseteq T_j\big) + Pr_{\geq i,j-1}(X) \times \Big(1 - Pr_j\big(X \subseteq T_j\big)\Big) \ldots (9)$$

  Cas limite :

$$Pr_{\geq 0,j}(X) \quad (0 \leq j \leq N)$$
$$Pr_{\geq i,j}(X) = 0 \quad pour \quad i > j$$

  Où N est le nombre total de transactions contenues dans la base de données probabiliste et $Pr_{\geq i,j}(X)$ est la probabilité que l'itemset X apparaisse au moins i fois parmi les premières j transactions de la base de données probabiliste. La complexité temporelle du calcul en programmation dynamique pour chaque itemset est $O(N^2 \times \theta)$ avec N est le nombre total de transactions et θ est le seuil minimum de support [24].

- ***Algorithmes basés sur la stratégie diviser-pour-régner :***

  Un autre algorithme pour l'extraction d'itemsets fréquents probabilistes en se basant sur la valeur exacte de la probabilité de fréquence a été proposé par Sun et *al*. dans [30]. Il est basé sur la méthode diviser-pour-régner. Cet algorithme que l'on note ici *DC* divise une base de données incertaine D, en deux sous-bases de données : D_1 et D_2. Puis, ces deux sous-bases de données sont divisées de manière récursive jusqu'à ce qu'il ne reste qu'une transaction. L'algorithme s'arrête pour enregistrer la distribution de probabilité du support de l'ensemble d'éléments dans cette transaction. Finalement, la distribution de probabilité complète du support de l'ensemble d'éléments est progressivement obtenue. La complexité temporelle pour calculer la probabilité de fréquence d'un itemset est de $O(N^2)$ où N est le nombre total de transactions dans la base de données incertaine [30].

2.5.3 Algorithmes basés sur l'approximation de la probabilité de fréquence

En 2013 bernecker et ses collègues [31] ont proposé une nouvelle approche qui se base sur l'idée d'approximer la distribution de probabilité de fréquence d'un itemset à l'aide d'une loi de distribution de probabilité. Deux algorithmes ont été proposés en utilisant respectivement, une loi de distribution de Poisson et une loi de distribution normale. Le processus commence par trouver des itemsets dont les probabilités de fréquence sont supérieures à un certain seuil puis extraire des itemsets ayant les k probabilités de fréquence les plus élevées [31].

En conclusion, les résultats affirment que les modèles de support attendu et d'approximation de Poisson sont tous deux faciles à implémenter et à calculer efficacement parce que l'élagage peut être effectué facilement. Par contre dans le modèle d'approximation normale, l'élagage ne peut pas être effectué et sa mise en œuvre nécessite plus d'efforts.

Les techniques d'exploration discutées précédemment visent à analyser des données, où l'ordre séquentiel des événements n'est pas pris en compte. Si ces techniques d'exploration sont utilisées sur des données incluant des informations temporelles ou séquentielles, ces informations seront tout simplement ignorées [16]. Cependant, dans de nombreux domaines, l'ordre des événements ou des éléments est important. Nous allons donc aborder la question de la fouille de motifs séquentiels dans le cas de données incertaine.

2.6 Extraction de motifs séquentiels incertains

L'extraction de motifs séquentiels à partir de bases de données probabilistes a été introduite pour la première fois en 2011 dans [32] . C'est l'un des domaines de recherche les plus importants. Il consiste à découvrir toutes les sous-séquences intéressantes dans un ensemble de séquences. L'extraction de motifs séquentiels a des applications dans divers domaines, notamment la finance, la santé, la bio-informatique, l'apprentissage en ligne, l'analyse de texte, etc. De façon similaire à la recherche d'itemsets fréquent, une séquence est considérée comme fréquente si son support n'est pas inférieur à un seuil minimum de support [10], bien entendu avec une définition adaptée de la notion de support pour intégrer l'information temporelle. En présence de données incertaine, on distingue également deux approches pour la recherche de séquences fréquentes.

La première approche est basée sur le concept de support attendu (expected support) d'une séquence. En définissant le support attendu d'une séquence, le but est de déterminer toutes les séquences ayant un support attendu non inférieur à un seuil minimum défini par l'utilisateur [32].

La seconde approche se fonde sur l'estimation de la probabilité de fréquence d'une séquence et le but est de trouver toutes les sous-séquences dont la probabilité de fréquence (d'avoir une fréquence au moins égale à un seuil minimum de fréquence spécifié par l'utilisateur) est au moins égale à un seuil de probabilité minimum défini également par l'utilisateur [33].

Une base de séquences incertaines est une collection de séquences incertaines $S = \{s_1, s_2, \dots, s_n\}$, où chaque $s \in S$ est représentée par $e = (sid, s_e, p_e)$ où sid est l'identifiant de s , s_e est une séquence (classique) et p_e est la probabilité existentielle de e.

Il existe deux types de base de séquences incertaines selon que les probabilités soient associées au niveau des événements ou alors au niveau des séquences. Pour interpréter une base de séquences incertaines S, on utilise la sémantique des mondes possibles. Pour chaque événement e_i dans une séquence, il existe deux mondes possibles : un où l'événement e_i se produit et un autre où il ne se produit pas. Le nombre de mondes possibles augmente de manière exponentielle avec le nombre d'événements et de séquences. La valeur de probabilité de chaque monde possible W est calculée comme suit [32]:

$$Pr(W) = \prod_{c_i \in W} \prod_{e_{i,j} \in c_i} Pr(e_{i,j}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (10)$$

Où $c_i \in W$ est une séquence dans W, $e_{i,j} \in c_i$ est un événement dans c_i et $Pr(e_{i,j})$ est la probabilité existentielle de $e_{i,j}$.

2.6.1 Extraction de motifs séquentiels fréquents basés sur le support attendu

En 2011 Muzammal et Raman [32] ont proposée pour la première fois une méthode d'extraction de motifs séquentiels basés sur le support attendu à partir de bases de séquences incertaines. C'est une méthode utilisée pour identifier des séquences d'éléments qui apparaissent fréquemment au sein d'une base de données séquentielle incertaine où chaque élément d'une séquence a une certaine probabilité existentielle. L'objectif de cette méthode est de découvrir tous les motifs séquentiels qui ont un support attendu supérieur à un seuil spécifié. Le support attendu dans une séquence particulière est calculé en multipliant les probabilités existentielles des éléments du motif, à condition que le motif apparaisse dans la séquence. Si le motif n'apparaît pas dans la séquence, son support attendu est considéré comme nul. Le support attendu d'un motif séquentiel dans l'ensemble de la base de données est la somme de ses supports attendus dans toutes les séquences où le motif apparaît.

2.6.2 Extraction de motifs séquentiels basée sur la probabilité de fréquence

En 2014 Zhao et ses collègues [33] ont proposé le modèle d' extraction de motifs séquentiels probabilistes. Comme pour l'extraction d'itemsets fréquents à partir d'une base de données transactionnelle probabiliste, dans une base de séquences probabiliste, un motif séquentiel X est considéré comme un motif fréquent si la probabilité qu'il apparaisse au moins θ fois est supérieure ou égale à δ où θ est le seuil minimum de support δ le seuil minimum de probabilité.

2.7 Conclusion

Dans ce chapitre, nous avons présenté un aperçu des bases de données probabilistes et introduit certains des modèles les plus connus pour ce type de données. Ensuite, nous avons abordé le problème de l'extraction des itemsets fréquents à partir de données incertaines (uFIM) et discuté des algorithmes couramment utilisés pour résoudre ce problème. Enfin, nous avons abordé le problème de l'extraction de motifs séquentiels à partir de données incertaines (uSPM).

Chapitre 3

Extraction des règles séquentielles

3.1 Introduction

Comme nous l'avons mentionné précédemment la fouille de motifs séquentiels fréquents est l'une des taches de fouille de motifs les plus populaires pour découvrir des sous-séquences communes à plusieurs séquences dans une base de séquences, i.e., toutes les sous-séquences ayant un support supérieur ou égal à un seuil prédéfini *minsup*. Mais il se peut que la fouille de motifs séquentiels fréquents ne soit pas suffisante pour la tâche de prédiction. Par exemple, il est possible que le fait que quelqu'un achète du « lait » ensuite du « café » soit courant mais qu'il existe également de nombreux cas où l'achat du lait n'est pas suivi de l'achat du café. Pour cela nous avons besoin d'une mesure de la confiance selon laquelle si l'achat du lait se produit, alors l'achat du café se produira ensuite. Ce modèle alternatif bien utile pour la prédiction prend la forme de règles appelées **règles séquentielles**. Dans ce chapitre nous aborderons le problème de l'extraction de règles séquentielles à partir de bases de séquences qui représente une méthode essentielle pour découvrir des schémas temporels ou séquentiels significatifs dans les données. Tout d'abord, nous présentons les différents types de règles séquentielles qui peuvent être identifiées, mettant en lumière leur importance dans les base de séquence Ensuite, nous plongerons dans les détails des algorithmes connus d'extraction des règles séquentielles et nous donnons un résumé comparatif entre ces algorithmes Enfin, nous explorerons divers domaines d'application avant de conclure le chapitre.

3.2 L'extraction de règles séquentielles certaines

L'idée d'extraction de règles séquentielles à partir de bases de séquence consiste à trouver toutes les règles avec un support séquentiel et une confiance séquentielle supérieure à certains seuils *minSeqSup* et *minSeqConf* Définis par l'utilisateur.

Une base de données séquentielles ou une base de séquences (notée SD) est définie comme un ensemble de séquences : $SD = \{s_1, s_2, \dots, s_n\}$ et un ensemble d'items $I = \{i_1, i_2, \dots, i_m\}$, où chaque séquence s_x est une liste ordonnée d'ensembles d'items (itemsets) $s_x = \{I_1, I_2, \dots, I_k\}$ telle que $I_1, I_2, \dots, I_k \subseteq I$, et où chaque séquence se voit attribuer un identifiant unique *sid* (identificateur de séquence). La Figure 2 représente un exemple d'une base de données séquentielles contenant quatre séquences. Par exemple, la séquence seq_1dans la Figure 2 signifie que les items a et b sont apparus simultanément, et ont été suivis successivement par c, f, g et enfin e. [34]

Une règle séquentielle $X \Rightarrow Y$ est une relation entre deux itemsets X, Y tels que $X, Y \subseteq I$ et que l'intersection de X et Y soit vide ($X \cap Y = \emptyset$) [34]. L'interprétation d'une règle $X \Rightarrow Y$ est que si les items de X apparaissent dans une séquence, les items de Y apparaîtront ensuite dans la même séquence. Par exemple, la règle $\{a, b, c\} \Rightarrow \{e, f, g\}$ est vérifiée dans la séquence $\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}$. Mais la règle $\{a, b, f\} \Rightarrow \{c\}$ ne l'est pas dans cette même séquence.

Deux mesures d'intérêt sont définies pour les règles séquentielles :

- ***Support séquentiel :***

$$seqSup(X \Rightarrow Y) = Sup(X \blacksquare Y)/|SD| \quad \text{(11)}$$

- ***Confiance séquentielle :***

$$seqConf(X \Rightarrow Y) = Sup(X \blacksquare Y)/Sup(X) \quad \text{(12)}$$

La notation $sup(X \blacksquare Y)$ désigne le nombre de séquences dans une base de données de séquences où tous les éléments de X apparaissent avant tous les éléments de Y (notez que les éléments de X ou les éléments de Y n'ont pas besoin d'être dans la même transaction). La notation $sup(X)$ désigne le nombre de séquences qui contiennent X.

Exemple. La Figure 2 montre un exemple sur la fouille de quelques règles dans une base de données séquentielle. On prend ici $minsup = 0{,}5$ et $minconf = 0{,}5$. Comme seuils de support et de confiance, respectivement.

ID	***Séquences***
1	$(a\ b), (c), (f), (g), (e)$
2	$(a\ d), (c), (b), (e\ f)$
3	$(a), (b), (f), (e)$
4	$(b), (f\ g)$

$\longrightarrow$

ID	***Rule***	***seqSup***	***seqConf***
1	$a\ b\ c \Rightarrow e$	*0.5*	*1.0*
2	$a \Rightarrow c\ e\ f$	*0.5*	*0.66*
3	$a\ b \Rightarrow e\ f$	*0.5*	*1.0*
4	$b \Rightarrow e\ f$	*0.75*	*0.75*
5	$a \Rightarrow e\ f$	*0.75*	*1.0*
6	$c \Rightarrow f$	*0.5*	*1.0*
7	$a \Rightarrow b$	*0.5*	*0.66*
...	...	...	...

Figure 3. Quelques règles trouvées avec l'algorithme CMRules

3.3 Les types de règles séquentielles

Les principaux types de règles séquentielles comprennent :

3.3.1 Règles séquentielles standards

C'est une relation de la forme: $X \Rightarrow Y$, où X et Y sont des séquences. L'interprétation d'une règle $X \Rightarrow Y$ est que si X se produit dans une séquence, il est probable qu'il soit suivi par Y dans la même séquence [35]. Par exemple, la règle $\langle\{a\},\{b\}\rangle \rightarrow \langle\{f\}\rangle$ est une règle séquentielle se produisant dans les séquences s_1, s_2 et s_3 de la base de séquences représentée dans la Figure 2, c'est-à-dire que si a est suivi de b, alors cette séquence sera suivie de f. L'idée principale étant d'extraire des règles séquentielles ayant un support et une confiance qui soient supérieurs à leurs seuils respectifs.

Exemple. Considérons la règle $\langle\{a\},\{b\}\rangle \Rightarrow \langle\{f\}\rangle$. Cette règle est vérifiée dans trois séquences. Son support est donc de *75%*. C'est uniquement dans ces trois séquences que l'on trouve la sous-séquence $\langle\{a\},\{b\}\rangle$. La confiance de cette règle est donc de *100%*.

3.3.2 Règles séquentielles partiellement ordonnées

Dans les règles séquentielles standard nous pouvons trouver certaines règles séquentielles qui sont très similaires et représentent la même situation avec de petites variations d'ordre. Par exemple les deux règles $\langle\{a\},\{b\}\rangle \Rightarrow \langle\{f\}\rangle$ et $\langle\{b\},\{a\}\rangle \Rightarrow \langle\{f\}\rangle$ expriment pratiquement le même contenu.

Une règle séquentielle partiellement ordonnée est une relation de la forme $X \Rightarrow Y$, où X et Y sont des ensembles d'items non ordonnés. L'interprétation d'une règle $X \Rightarrow Y$ est que si les éléments de X sont présents dans une séquence (dans n'importe quel ordre), les éléments de Y seront présents dans la même séquence (dans n'importe quel ordre également) [35]. La particularité intéressante des règles séquentielles partiellement ordonnées est qu'elles sont plus générales que les règles séquentielles standards [36]. Il est possible de représenter plusieurs règles séquentielles standard par une seule règle séquentielle partiellement ordonnée. Voici un exemple qui illustre cela dans la Figure 3.

Les algorithmes renvoient toutes les règles valides, les règles ayant un support et une confiance supérieurs aux seuils respectifs *minSup* et *minConf* définis par l'utilisateur.

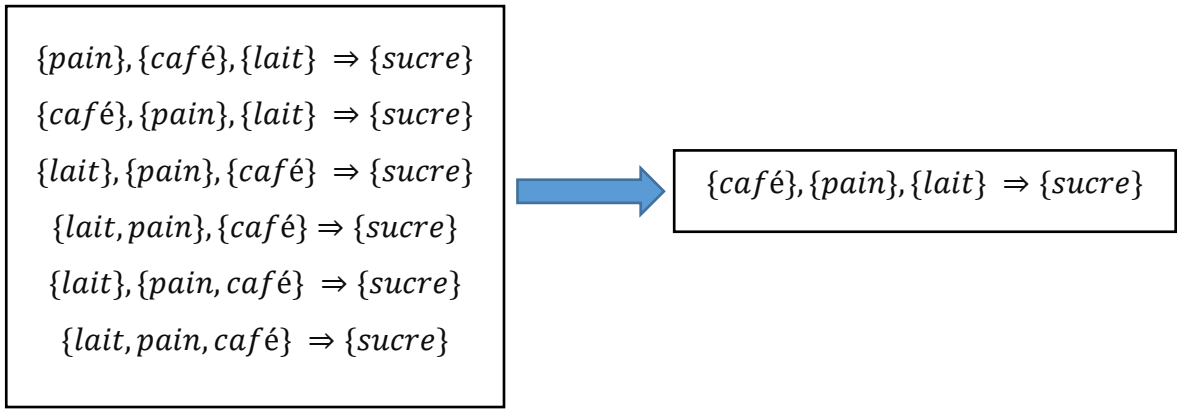


Figure 4.Règles séquentielles standard *vs* Règles séquentielles partiellement ordonnées

Exemple. Considérons la règle $\langle\{a\},\{b\},\{f\}\rangle \Rightarrow \langle\{e\}\rangle$. Cette règle est vérifiée dans deux séquences (s_1 et s_3). Son support est donc de *50 %*. On trouve la séquence $\langle\{a\},\{b\},\{f\}\rangle$ dans trois séquences de la base, d'où une confiance de *66 %*.

3.4 Algorithmes d'extraction de règles séquentielles à partir de données certaines

Plusieurs algorithmes pour l'extraction de règles séquentielles à partir de bases de séquences certaines ont été développés. Ils peuvent être regroupés en deux catégories principales.

Le premier type concerne les algorithmes pour l'extraction de règles séquentielles apparaissant dans une seule séquence d'événements, L'approche la plus célèbre de ce groupe est celle de Mannila et ses collègues [37] en 1997. Cette approche ne peut découvrir que des règles dans une seule séquence d'événements. Les travaux alternatifs qui extraient des règles séquentielles à partir d'une seule séquence d'événements sont les algorithmes de Hamilton & Karimi (2005) [38], Hsieh (2006) [39], Deogun (2005) [40], etc. qui découvrent respectivement des règles entre plusieurs événements et un seul événement, entre deux événements et entre plusieurs événements.

La deuxième catégorie comprend les algorithmes qui découvrent des règles séquentielles dans plusieurs séquences. L'objectif de ces algorithmes est de découvrir des règles qui apparaissent fréquemment dans plusieurs séquences. Par exemple, l'algorithme de Das et al[41] découvre des règles telles que la partie gauche contienne plusieurs événements tandis que la partie droite ne peut contenir qu'un un seul événement. C'est une

limitation importante car dans les applications réelles, les relations séquentielles peuvent entraîner la présence de plusieurs événements

Il y a un autre algorithme qui extraire des règles séquentielles à partir de bases de séquences sans mettre de limile pour le nombre d'événements présents dans chaque règle : c'est celui de Harms & Tadesse (2002) [42].

Seuls Lo et ses collègues [43], Pitman et Zanker [44]et Zhao et ses collègues [45] ont développé des algorithmes permettant d'extraire des règles communes à plusieurs séquences. Leurs algorithmes découvrent des règles telles que les parties gauche et droite des règles sont des séquences d'ensembles d'éléments par exemple dans un supermarché les clients qui ont acheté du café, du lait et du sucre ensemble, et ensuite du pain, ont ensuite acheté du beurre, selon les définitions de Lo et ses collègues, Pitman et Zanker et Zhao et ses collègues, cette règle est distincte de toutes ses variations telles que:

- $\{Café\}, \{Lait, Sucre\}, \{Pain\} \Rightarrow \{Beurre\}$
- $\{Café, Lait, Sucre\}, \{Pain\} \Rightarrow \{Beurre\}$
- $\{Pain\}, \{Café\}, \{Lait\}, \{Sucre\} \Rightarrow \{Beurre\}$
- $\{sucre, pain, café, lait\} \Rightarrow \{ Beurre \}$

Ces algorithmes sont restrictifs, car que les règles sont très spécifiques, elles ont moins de chances de correspondre à une séquence d'événements nouvelle pour faire des prédictions. Par exemple, considérons la règle $\{Pain\}, \{Café\}, \{Lait\}, \{Sucre\} \Rightarrow \{Beurre\}$. Cette règle peut être utilisée pour prédire qu'un client achètera le beurre s'il a acheté les produits : pain, café, lait et sucre, dans cet ordre. Cependant, si le client a acheté les éléments dans un ordre différent, la règle ne correspond pas et ne peut donc pas être utilisée pour faire des prédictions précises. En alternative, il est préférable d'avoir une forme de règles séquentielles plus générale, de manière à ce que les événements dans la partie gauche et la partie droite de chaque règle ne soient pas ordonnés. Afin d'extraire ces règles, plusieurs algorithmes ont été proposés, nous allons expliquer certaines de ces algorithmes.

3.4.1 L'algorithme CMRules (Class-based Mining Rules)

En 2010 Fournier-Viger et ses collègues [34] ont proposé un algorithme de découverte de règles séquentielles dans des bases de données de séquences. La Figure 4 représente le pseudo-code de l'algorithme CMRules qui commence par transformer la base de données séquentielle en une base de données transactionnelle pour réduire l'espace de recherche aux éléments qui apparaissent conjointement dans plusieurs séquences Par la suite, il extrait toutes les règles d'association qui respectent les seuils minimaux de support et de confiance. Ensuite, ces règles d'association sont examinées pour ne conserver que celles qui respectent les seuils minimaux de confiance et de support en fonction de l'ordre temporel (voir Figure 5). CMRules a une meilleure scalabilité pour les faibles seuils de support. Le principal avantage de CMRules réside dans sa facilité d'implémentation, car de nombreux les algorithmes de fouille de règles d'association à partir de bases transactionnelles sont déjà existants. Le principal inconvénient de CMRules qui le rend parfois inefficace est que ses performances dépendent du nombre de règles d'association trouvées. Si cet ensemble est grand, CMRules devient moins efficace.

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 (Agrawal et al., 1993). 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 5. L'algorithme CMRules

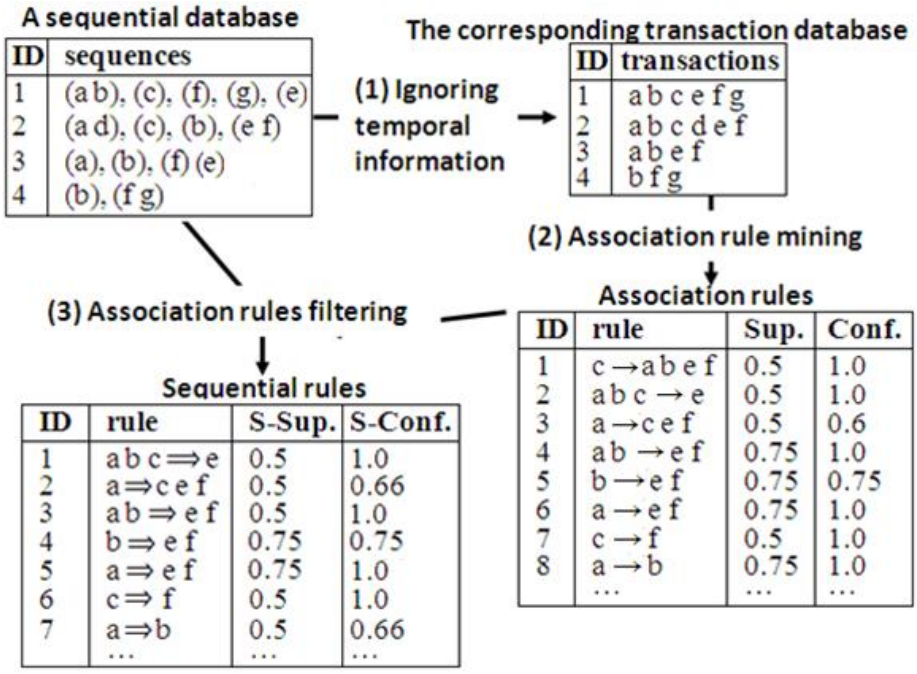


Figure 6. Un exemple d'exécution de l'algorithme CMRules

3.4.2 L'algorithme RuleGrowth

C'est un algorithme efficace qui utilise une approche différente de CMRules pour l'exploration de règles séquentielles. Cette approche basée sur le principe de FP-Growth pour éviter la génération de candidats redondants. Il a été présenté par Fournier-Viger et ses collègues [36] en 2011. RuleGrowth comprend également quelques idées pour éviter d'analyser toute la base de données à chaque fois. Le fonctionnement de cet algorithme est comme suit :

- ***Étape 1.*** Analyser la base de données pour calculer le support de chaque élément. Ne conserver que les éléments fréquents.

- ***Etape 2.*** Pour chaque paire d'éléments fréquents, essayez de créer une règle avec seulement deux éléments. Seules les règles dont le support et la confiance soient supérieurs ou égaux aux seuils minimaux définis sont affichées en sortie.

- ***Etape 3.*** Cette étape est appelée expansion à gauche et à droite. Elle recherche des règles plus larges en explorant de manière récursive la base de données pour ajouter un seul élément à la fois à la partie gauche ou droite de chaque règle. Elle calcule ensuite la confiance et le support pour chaque nouvelle règle. Si le support et la confiance sont respectivement supérieurs ou égaux à *minconf* et *minsup,* afficher la règle.

RuleGrowth est plus performant que CMRules en termes de temps d'exécution et d'utilisation de la mémoire.

3.4.3 L'algorithme ERMiner (Equivalence Class Based Sequential Rule Miner)

ERMiner est un algorithme d'extraction de règles séquentielles qui utilise des classes d'équivalence pour découvrir des règles fréquentes à partir de grands ensembles de données. Il a été développé en 2014 par Philippe Fournier-Viger et ses collègues [46]. L'algorithme ERMiner fonctionne en deux phases principales :

- ***Génération de classes d'équivalence.*** Tout d'abord, parcourir séquentiellement la base de séquences afin d'identifier tous les antécédents et conséquents uniques présents. Ensuite, pour chaque antécédent et conséquent identifié, une classe d'équivalence est formée, regroupant toutes les règles qui partagent cet antécédent ou conséquent spécifique.

- ***Extraction de règles séquentielles.*** On explore de façon itérative les classes d'équivalence, en commençant par les plus petites. A l'intérieur de chaque classe d'équivalence, on génère des règles candidates en combinant des antécédents et des conséquents appartenant à la même classe Ensuite, On applique l'élagage pour éliminer les candidates redondantes et on évalue les règles candidates restantes en fonction des seuils de support et de confiance prédéfinis. A la fin, les règles candidates qui satisfont ces seuils sont conservées.

ERMiner offre plusieurs avantages par rapport aux algorithmes CMRules et ERMiner car L'utilisation de classes d'équivalence réduit le nombre de calculs redondants et l'utilisation de structure de données nommée SCM (Sparse Count Matrix) réduit l'espace de recherche.

3.5. Comparaison entre les algorithmes

Le Tableau 7 représente une étude comparative entre les algorithmes liés à l'extraction de règles séquentielles. Nous avons choisi dans ce tableau les algorithmes les plus utilisées dans la littérature. On a remarqué que les algorithmes CMRules, RuleGrowth et ERMiner visent à trouver toutes les règles séquentielles qui nous intéressent dans notre travail.

Tableau 7.Un résumé comparatif entre les algorithmes d'extraction de règles séquentielles

Algorithme	**Année**	**Algorithme de base**	**Objectif**
CMDeo	2010	Apriori	l'extraction de règles séquentielles à partir de données d'événements naturels
CMRules	2010	Apriori	Trouver toutes les règles séquentielles
RuleGrowth	2011	pattern growth	Trouver toutes les règles séquentielles
TRuleGrowth	2011	pattern growth	Trouver des règles séquentielles avec une contrainte de fenêtre
ERMiner	2014	Equivalence Classes	Trouver toutes les règles séquentielles
HUSRM	2014	l'apriori / FP-Growth	Trouver des règles séquentielles à haute utilité

3.6 Domaines d'application

La fouille de règles séquentielles a été appliquée dans de nombreux domaines tels que :

- ***Apprentissage en ligne*** [34], [47]

  Dans le domaine de l'apprentissage en ligne, l'extraction des règles séquentielles se révèle être une méthode d'analyse et d'amélioration des expériences d'apprentissage des apprenants. Un exemple dans ce contexte est l'analyse des parcours d'apprentissage en examinant les séquences d'activités d'apprentissage des apprenants afin de concevoir des trajets plus efficaces par les éducateurs dans la façon dont les apprenants progressent à travers les cours ou les modules. De plus l'extraction des règles séquentielles alimente les systèmes d'apprentissage adaptatif, qui adaptent le contenu et la méthode d'enseignement en fonction des interactions passées de l'apprenant. Ces systèmes recommandent des ressources et des activités spécifiques, ce qui permet d'offrir une expérience d'apprentissage plus personnalisée à chaque apprenant.

- ***Analyse de la séquence d'alarmes*** [48]

L'analyse de séquences d'alarmes est une méthode utilisée dans les systèmes de surveillance et de contrôle pour détecter les schémas temporels anormaux qui peuvent indiquer des problèmes ou des pannes, par exemple dans la surveillance des réseaux de télécommunications. Dans un centre de contrôle des réseaux, les opérateurs reçoivent des alarmes provenant de divers équipements réseau tels que des routeurs, des commutateurs et des serveurs. Les opérateurs peuvent améliorer la détection des anomalies telle qu'un pic soudain de congestion de trafic à l'aide de techniques d'extraction de règles séquentielles. Ils peuvent anticiper les pics de trafic et prendre des mesures préventives, telles que la répartition du trafic vers d'autres équipements avant que la panne ne se produise.

- ***la Simulation de fabrication*** [49]

La fouille de règles séquentielles peut être appliquée pour la simulation de fabrication est une méthode utilisée dans le domaine industriel afin de modéliser et simuler les processus de fabrication d'un produit avant sa mise en œuvre réelle. Cette méthode consiste à simuler le comportement du système de production en fonction des paramètres spécifiés, tels que les délais de production, les capacités des machines, les flux de matériaux, etc. Cette approche permet aux fabricants de prendre des décisions afin d'obtenir des produits de meilleure qualité.

- ***Prélecture de pages Web*** [35]

La Prélecture de pages Web également connue sous le nom de "Web page prefetching", peut envisager l'utilisation des règles séquentielles comme une technique utilisée dans les navigateurs web afin de prédire les actions futures des utilisateurs lorsqu'ils naviguent sur un site web. Cela permet de réduire le délai de chargement perçu par l'utilisateur lorsqu'il clique sur un lien ou une référence vers une autre page, car la page est déjà en mémoire cache et prête à être affichée.

A titre d'exemple, une règle séquentielle pourrait être formulée comme suit : *"Si un utilisateur visite la page d'accueil, puis la page de produits, il est probable qu'il visite ensuite la page de paiement"*. En préchargeant la page de paiement en arrière-plan dès que l'utilisateur accède à la page de produits, le site web peut réduire le temps de chargement perçu par l'utilisateur lorsque celui-ci décide de passer à l'étape suivante de l'achat.

- ***Analyse des comportements des clients*** [50]

  Un autre champ d'application des règles séquentielles est l'analyse des comportements des clients, un processus qui consiste à comprendre les actions des clients afin d'obtenir des informations sur leurs comportements et leurs besoins. Cette analyse aide les entreprises à mieux comprendre leurs clients et de prédire leurs besoins futurs, et permet ainsi les entreprises à prendre des décisions éclairées pour améliorer leurs stratégies de marketing.

3.7 Conclusion

Dans ce chapitre, nous avons exploré la problématique de la fouille de règles séquentielles, une méthode puissante pour découvrir des motifs temporels significatifs contenus dans les données. Nous avons présenté dans ce chapitre les différents types de règles séquentielles ainsi que les algorithmes utilisés pour les extraire avec fiabilité. De plus, nous avons discuté certaines domaines d'application de la fouille de règles séquentielles. Finalement dans le dernier chapitre, nous allons nous focaliser sur notre apport dans cette thèse : trouver des règles séquentielles à partir de bases de séquences incertaines.

Chapitre 4

Contributions

4.1 Introduction

Dans ce dernier chapitre, nous exposons les grandes lignes nécessaires pour atteindre l'objectif de la thèse. Ce chapitre présente les contributions significatives réalisées tout au long de ce travail de recherche. Il explore les approches et les techniques développées pour relever les défis liés à l'exploration de motifs et de règles séquentielles incertaines. Les contributions s'articulent sur deux principaux points :

- Nouvelles approches pour l'exploration de motifs séquentiels incertaines
- Une nouvelle méthode pour extraire des règles séquentielles incertaines

Pour les deux contributions, nous présentons dans ce qui suit les idées clés, les algorithmes développés et nous discutons les différentes expérimentations et les résultats obtenus.

4.2 Contribution 1. Extraction d'itemsets séquentiels fréquents à partir de bases de séquences probabilistes (Extracting sequential frequent itemsets from probabilistic sequences database)[1]

4.2.1 Résumé

L'extraction d'itemsets fréquents est un sous-domaine important de la fouille de données. Elle prouve son utilité en trouvant les motifs les plus pertinents dans de grandes bases de données selon leur fréquence. L'extraction des itemsets fréquents a d'abord été appliquée aux bases de données transactionnelles mais l'émergence de nouvelles technologies a conduit à de nouveaux types de bases de données, comme les bases de données séquentielles. De plus, dans des scénarios réels, les données peuvent être incomplètes ou incertaines, posant un défi pour l'extraction des itemsets fréquents. Dans ce chapitre nous nous concentrons sur l'extraction des motifs séquentiels fréquents dans des bases de séquences probabilistes. Nous proposons une nouvelle approche pour identifier des itemsets fréquents en tenant compte des probabilités d'occurrence et en filtrant les itemsets en utilisant des informations séquentielles.

[1] Ce travail a fait l'objet d'un papier publié par le journal Springer : « International Journal of Information Technology » en mai 2023 [51].

Il existe plusieurs façons de modéliser l'incertitude dans les bases de données, ce travail se concentre sur le modèle probabiliste, qui représente l'incertitude en utilisant une probabilité existentielle. Comme nous l'avons discuté dans le chapitre 2, cette probabilité peut être associée à des éléments individuels au sein d'une transaction, ce qui est appelé l'incertitude au niveau de l'attribut.

Dans le contexte des bases de données transactionnelles incertaines, les mesures classiques de support et de confiance ne peuvent pas être utilisées pour évaluer la signification d'un itemset en raison de l'incertitude des données. L'utilisation du concept de mondes possibles est très coûteuse en termes de temps de calcul. Pour remédier à cela, une autre mesure a été proposée, "support attendu" (en anglais : expected support). Le support attendu d'un itemset X dans une base transactionnelle D est donné par :

$$ExpSup(X) = \sum_{j=1}^{|D|} \prod_{x \in X} p(x, T_j) \quad \text{..... (13)}$$

où $p(x, T_j)$ est la probabilité existentielle de l'item x dans la transaction T_j .

Dans le cadre des bases de séquences incertaines, nous utilisons une autre forme adaptée du concept de support, appelé support séquentiel, afin de tenir compte de l'ordre des éléments, au lieu de simplement compter le nombre de transactions dans lesquelles un itemset apparaît, comme dans la définition classique du support. Le support séquentiel est donné par :

$$SeqSup(\{A, B\}) = Supp(A > B)/|D| \quad \text{..... (14)}$$

où $Supp(A > B)$ représente le nombre de séquences dans la base D où A apparaît avant B.

4.2.2 L'approche proposée

Tout d'abord, nous nous basons sur l'hypothèse que dans la base de séquences probabiliste, chaque séquence est composée d'un ensemble ordonné de singletons.

La première étape de notre approche proposée consiste à transformer la base de séquences probabilistes en une base de données transactionnelle probabiliste en ignorant les informations séquentielles présentes dans l'ensemble des séquences et en traitant les données comme une base de données transactionnelle probabiliste traditionnelle. Ensuite,

nous appliquons l'algorithme *U-Apriori*[2] qui prend en entrée une base de données probabiliste ainsi qu'un seuil minimum de support attendu, afin de trouver les itemsets fréquents probabilistes valides. On utilise pour cela la définition du support attendu donnée par l'équation 13. En fait, l'algorithme *Apriori* classique appliquée aux données transactionnelles certaines utilise le support traditionnel défini par le nombre de transactions dans lesquelles l'ensemble d'éléments apparaît. Par contre, comme nos données sont probabilistes (voir Algorithme 1), nous utilisons une approche différente basée sur la notion de support attendu donné par l'équation 13.

Algorithm 1 An Apriori algorithm using expected support

```
Input: probabilistic database (PDB), minExpectedSupp
C_k ← candidate itemset of size k.
L_k ← frequent itemset of size k.
L_1 ← frequent items.
K ← 1
while L_k ≠ ∅ do
    K ← k + 1
    C_{k+1} ← candidates generated from L_K
    for all t ∈ PDB do
        C_t ← Subset(C_{k+1}, t)
        for all c ∈ C_t do
            c.count ← c.count + 1
        end for
    end for
    for all c ∈ C_t do
        if c.count ≥ min then
            L_{k+1}.add(c)
        end if
    end for
end while
return L ← ⋃_i L_i
```

Dans la deuxième étape, une fois que les itemsets probabilistes fréquents (itemsets ayant un support attendu non inférieur au seuil prédéfini) sont identifiés, nous filtrons ces itemsets trouvé dans l'étape précédente afin de ne garder que ceux qui respectent l'ordre des itemsets imposé par les contraintes de précédence présentes dans la base originale de séquences. Nous utilisons pour cela la mesure de support séquentiel donnée dans l'équation 14 afin de trouver tous les itemsets séquentiels fréquents probabilistes valides (voir Algorithme 2).

[2] On peut également utiliser ici n'importe quel autre algorithme de fouille d'itemsets fréquents à partir de base transactionnelle probabiliste.

Algorithm 2 Algorithm for Filtering the set of probabilistic frequent itemset

```
Input: a probabilistic sequences database (PSDB), minSeqSupp
L ← set of probabilistic frequent itemset.
for all f_itemset ∈ L do
    possible_f_itemset ← Permuation(f_itemset)
    for all S ∈ PSDB do
        for all c ∈ possible_f_itemset do
            if c ∈ S then
                c.count ← c.count + 1
            end if
        end for
    end for
    for all c ∈ possible_f_itemset do
        if c.count ≥ minSeqSupp then
            F.add(c)
        end if
    end for
end for
return F
```

4.2.3 Expérimentations et discussions de Résultats

Nous avons utilisé Jupyter Notebook et le langage de programmation Python pour mettre en œuvre notre approche. Ces outils ont été installés sur un ordinateur Windows 10 équipé d'un processeur Intel Core i5. Nous avons utilisé des données synthétiques pour évaluer notre implémentation car il n'existe pas des données réelles séquentielles et incertaines en même temps. Les données synthétiques se composent de 12 fichiers, avec le nombre de séquences augmentant d'un fichier à l'autre d'environ 1 000 séquences. Chaque séquence contient au max sept items, et chaque item se voit attribuer une probabilité existentielle aléatoire comprise entre zéro et un.

La performance de l'approche proposée est évaluée en fonction de sa complexité spatiale et temporelle Pour chaque fichier de notre ensemble de données synthétiques, nous réalisons dix expériences et nous calculons le temps moyen et l'espace mémoire consommés à chaque exécution.

Les résultats finaux sont présentés dans la Figure 6, où le graphique (a) montre que le temps d'exécution augmente avec le nombre de séquences dans le fichier. Malgré cela, le graphique montre une bonne performance de l'approche proposée, que nous pouvons exprimer par la notation Big-O dans l'ordre de (n^2) où où n représente le nombre de séquences. Le graphique (b) représente la complexité spatiale et montre que l'espace mémoire consommé augmente avec le nombre de séquences dans le fichier. Malgré cela, le

graphique montre une très bonne performance de l'approche proposée que nous pouvons exprimer dans l'ordre $O(n)$, où n représente le nombre de séquences.

Il n'est pas tout à fait exact d'attribuer la réduction de la consommation de mémoire et de temps uniquement à la valeur des probabilités existentielles. Les techniques employées lors de la conception et la mise en œuvre de l'algorithme proposé, telles que l'utilisation de structures de données optimisées, une gestion efficace de la mémoire et la sélection soigneuse des algorithmes et des techniques de traitement des données, jouent également un rôle important dans la réduction de la consommation de mémoire et de temps. Cependant, il est vrai que la valeur des probabilités existentielles peut affecter le nombre d'ensembles d'éléments dépassant le support attendu minimum, ce qui peut à son tour affecter la consommation de temps et de mémoire. Plus précisément, si la valeur des probabilités existentielles est élevée, alors il peut y avoir moins d'itemsets dépassant le support attendu minimum, ce qui entraîne une consommation moindre de temps et de mémoire. En revanche, si la valeur des probabilités existentielles est faible, alors il peut y avoir plus d'itemsets dépassant le support attendu minimum, ce qui entraîne une consommation plus importante de temps et de mémoire.

4.2.4 Etude comparative

Comparer un travail scientifique avec des recherches existantes est un aspect fondamental de toute investigation scientifique. Cependant, il n'est pas toujours possible de trouver des travaux précédents directement comparables aux siens. Dans ces cas, une approche utile est de comparer avec une méthode naïve qui sert de référence simple pour l'évaluation. L'algorithme naïf utilisé comme référence pour la comparaison dans ce travail est composé de trois étapes :

1. Extraction de l'ensemble des itemsets fréquents probabilistes en utilisant le support attendu.

2. Extraction de l'ensemble des itemsets séquentiels fréquents en utilisant le support séquentiel.

3. Calcul de l'intersection entre les deux ensembles, pour trouver l'ensemble des itemsets séquentiels probabilistes.

En termes de complexité temporelle, comme le montre la Figure 6a, l'algorithme proposé est plus efficace que l'algorithme naïf. Le temps pris par l'algorithme proposé augmente beaucoup plus lentement que celui de l'algorithme naïf lorsque la taille de l'entrée augmente. Par conséquent, pour de grandes entrées, notre algorithme sera plus rapide que l'algorithme naïf.

Cependant, il est important de noter que la complexité temporelle n'est pas le seul facteur à considérer lors de la comparaison des algorithmes. D'autres facteurs tels que la complexité spatiale, comme on peut le remarquer lors d'un examen approfondi du graphique dans la figure 6b, montrent que l'algorithme proposé a une complexité spatiale inférieure à celle de l'algorithme naïf, ce qui signifie qu'il nécessite moins de mémoire pour s'exécuter à mesure que la taille de l'entrée augmente. Par conséquent, pour des entrées plus grandes, notre algorithme proposé sera plus efficace en termes d'utilisation de la mémoire que l'algorithme naïf.

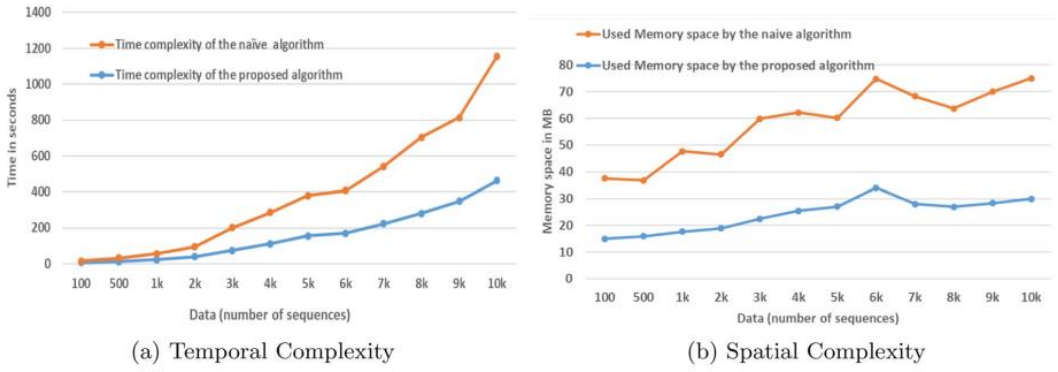


Figure 7. Complexité temporelle et spatiale de l'algorithme proposé comparée à l'algorithme naïf

4.3 Contribution2: Extraction de Séquences Incertaines : Une Nouvelle Approche pour l'Extraction de Règles Séquentielles (Mining Uncertain Sequences : A Novel Approach to Sequential Rule Extraction)[3]

4.3.1 Résumé

L'extraction de règles séquentielles est une méthode puissante de fouille de données qui vise à découvrir des modèles significatifs dans des données séquentielles. Cette approche est utilisée dans divers domaines tels que le commerce électronique, la santé, la surveillance des réseaux, etc.

[3] Ce travail a fait l'objet d'un papier en cours d'évaluation dans la revue : International Journal of Advanced Technology and Engineering Exploration IJATEE.

Dans de nombreux scénarios réels, les données peuvent être incomplètes, imprécises ou sujettes à des erreurs. Par exemple, dans un environnement où les conditions changent rapidement, les données séquentielles peuvent être influencées par des changements soudains ou imprévus, ce qui entraîne une incertitude dans les valeurs attendues.

L'objectif de l'extraction de règles séquentielles qui prend en compte l'incertitude dans les données est de découvrir des dépendances significatives même en présence d'incertitude, ce qui permet de prendre des décisions éclairées et d'améliorer les processus et les performances dans une variété de domaines et d'applications.

Dans ce travail, nous nous intéressons à trouver des règles séquentielles à partir de bases de séquences contenant des données incertaines. L'incertitude est prise en compte au niveau des attributs et elle est modélisée par les valeurs de probabilité existentielle associées aux items.

Nous proposons une nouvelle approche pour extraire des règles séquentielles à partir de bases de séquences incertaines, qui se compose de deux étapes principales : la première consiste à extraire l'ensemble de règles probabilistes, et la seconde consiste à filtrer cet ensemble en utilisant les informations séquentielles dans les données.

Étant donné que les règles séquentielles standards peuvent parfois être très similaires et représenter essentiellement la même situation, nous proposons l'utilisation de règles séquentielles partiellement ordonnées. Ces règles permettent une représentation plus souple et générale des séquences, Ce qui peut être particulièrement bénéfique pour représenter plusieurs règles séquentielles standards par une seule règle séquentielle partiellement ordonnée.

4.3.2 Définition du problème

Étant donné un ensemble d'items $I = \{i_1, i_2, \dots, i_n\}$. Une base de séquences incertaines est définie comme un ensemble de séquences $S = \{s_1, s_2, \dots, s_N\}$ où chaque séquence s_x est une liste ordonnée d'itemsets : $s_x = X_1, X_2, \dots, X_m$ tel que $X_i \subseteq I$, pour $1 \leq i \leq m$. Ces itemsets peuvent contenir des éléments qui ne sont pas certains, et la présence d'un item i incertain dans une séquence est définie par une probabilité existentielle p_i avec $0 < p_i \leq 1$). L'élément est considéré comme certain si sa probabilité existentielle est égale à 1.

Dans le tableau suivant (Tableau 8), nous considérons une base de séquences probabilistes qui comprend huit séquences. Chacune contient des éléments certains et incertains. Cette base de séquences probabilistes sera utilisée comme exemple tout au long du reste de ce chapitre.

Tableau 8. Un exemple d'une base de séquences probabilistes

Sid	***Séquences (itemset)***
$\boldsymbol{S_1}$	$(Z, 0.2), (W, 0.6)$
$\boldsymbol{S_2}$	$(X, 1.0), (Y, 1.0), (Z, 1.0), (W, 0.4)$
$\boldsymbol{S_3}$	$(X, 1.0), (Y, 0.3), (Z, 0.8)$
$\boldsymbol{S_4}$	$(X, 0.4), (Y, 0.7)$
$\boldsymbol{S_5}$	$(Y, 0.4), (Z, 1.0)$
$\boldsymbol{S_6}$	$(X, 0.1), (Y, 1.0), (Z, 1.0)$
$\boldsymbol{S_7}$	$(X, 1.0), (Y, 1.0), (W, 1.0)$
$\boldsymbol{S_8}$	$(Y, 0.3), (Z, 1.0)$

L'objectif ici est d'extraire l'ensemble des règles séquentielles utiles et suffisamment solides et qui respectent l'ordre des événements imposé dans les séquences de la base de données. Il existe des méthodes établies, telles que : CMRules, RuleGrowth, IRMiner, etc., pour extraire des règles pertinentes à partir de données séquentielles certaines, appelées règles séquentielles. Il existe également méthodes, telles que U-Apriori, UFP-Growth, etc., pour extraire des règles pertinentes à partir de données incertaines classiques (à partir de bases de données transactionnelle incertaines), appelées règles d'association probabilistes (c-à-d : règles d'association à partir de base de données incertaines).

Cependant, l'objectif de notre travail est de combiner ces approches afin d'extraire des règles pertinentes à partir de données incertaines et séquentielles, que nous appelons : "règles séquentielles probabilistes ou incertaines".

Pour cela, nous proposons une nouvelle méthode pour extraire les règles séquentielles probabilistes à partir de bases de données de séquences incertaines qui est expliquée dans la partie suivante.

4.3.3 Hypothèses et idées clés de notre approche proposée

Nous supposons quelques hypothèses qui nous permettent de spécifier clairement et précisément nos objectifs :

- Les données sont stockées dans une base de données séquentielle probabiliste (PSDB).
- Chaque élément dans la base de données est un élément singleton (chaque itemset d'une séquence est formé d'un seul item).
- Chaque item dans chaque séquence a une probabilité existentielle dans l'intervalle]0,1].

Les idées clés de notre proposition sont les suivantes :

- Pour tenir compte des contraintes d'ordre présentes dans les séquences, notre approche fonctionne en deux phases :
    a) **Première phase.** Les informations temporelles sont ignorées et un ensemble de règles probabilistes classiques est généré (règles d'association à partir d'une base de données transactionnelle)
    b) **Deuxième phase.** Les règles générées dans la première phase sont filtrées afin de ne conserver que celles qui respectent l'information temporelle.
- Pour traiter l'incertitude des données, nous utilisons la mesure de support attendu pour évaluer la fréquence probabiliste des sous-séquences et des règles séquentielles.
- Notre proposition est un algorithme de style FP-Growth pour extraire des règles séquentielles.

4.3.4 L'approche proposée

Pour aborder le problème défini et en tenant compte des hypothèses données, nous proposons une approche en six étapes illustrée dans la Figure 7.

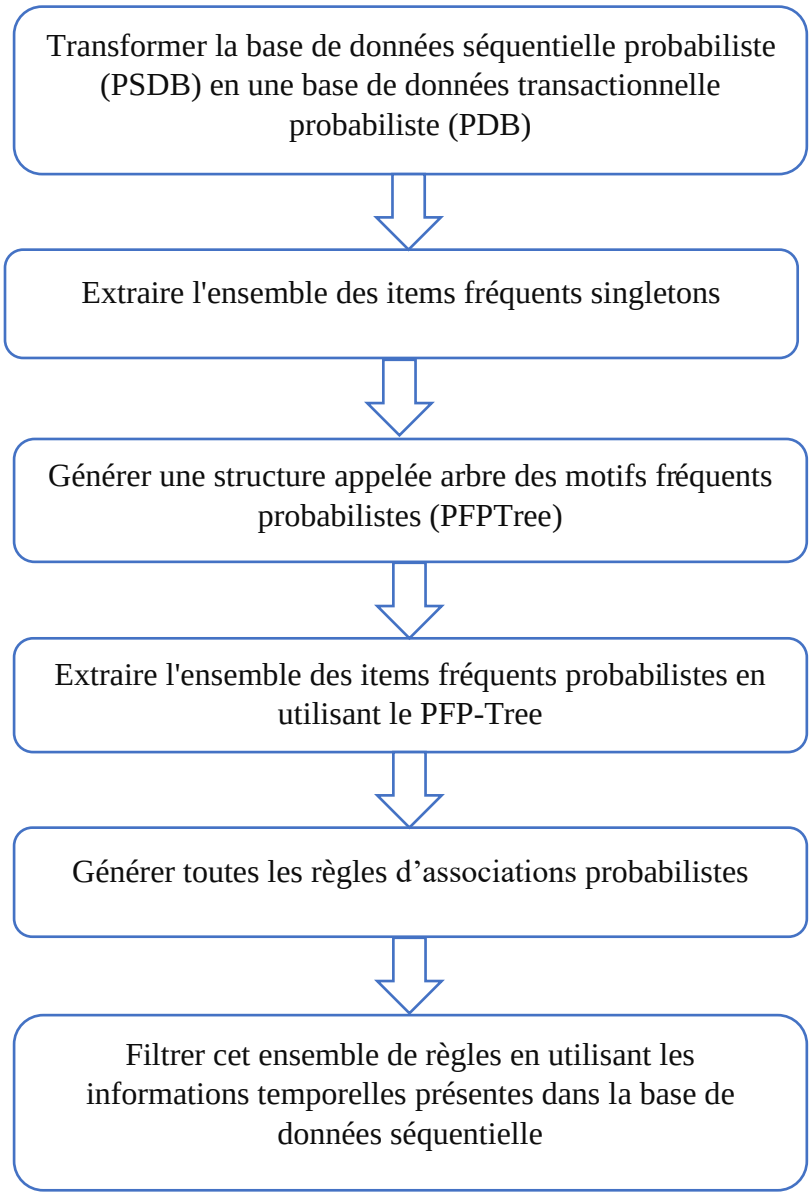


Figure 8. L'approche proposée de six étapes

Dans ce qui suit, nous détaillerons chacune des six étapes mentionnées précédemment.

4.3.4.1 Transformer la base séquentielle probabiliste en une base transactionnelle probabiliste

Nous appliquerons un principe simple utilisé dans [34] pour transformer une base de données séquentielle probabiliste en une base de données transactionnelle probabiliste, ce qui consiste à ignorer l'ordre des éléments imposé par la nature séquentielle des données afin d'obtenir une base de données probabiliste classique. Cela signifie que les séquences sont traitées simplement comme des transactions.

La transformation de base de données séquentielle probabiliste en base transactionnelle probabiliste permet d'utiliser des algorithmes d'exploration de règles d'association probabiliste existants comme *U-Apriori* par exemple. Notons que les données transactionnelles sont généralement plus simples à traiter et à analyser que les données séquentielles.

Tableau 9. La transformation de base de données séquentielle probabiliste en base transactionnelle probabiliste

Une base de données séquentielle

SID	***Séquences***
S_1	$\{Z, 0.2\}, \{W, 0.6\}$
S_2	$\{X, 1.0\}, \{Y, 1.0\}, \{Z, 1.0\}, \{W, 0.4\}$
S_3	$\{X, 1.0\}, \{Y, 0.3\}, \{Z, 0.8\}$
S_4	$\{X, 0.4\}, \{Y, 0.7\}$
S_5	$\{Y, 0.4\}, \{Z, 1.0\}$
S_6	$\{X, 0.1\}, \{Y, 1.0\}, \{Z, 1.0\}$
S_7	$\{X, 1.0\}, \{Y, 1.0\}, \{W, 1.0\}$
S_8	$\{Y, 0.3\}, \{Z, 1.0\}$

En ignorant l'information temporelle

La base de données transactionnelle Correspondante

TID	***éléments***	***Probabilités***
T_1	Z, W	*0.2 ,0.6*
T_2	X, Y, Z, W	*1.0, 1.0, 1.0, 0.4*
T_3	X, Y, Z	*1.0, 0.3, 0.8*
T_4	X, Y	*0.4, 0.7*
T_5	Y, Z	*0.4, 1.0*
T_6	X, Y, Z	*0.1, 1.0, 1.0*
T_7	X, Y, W	*1.0, 1.0, 1.0*
T_8	Y, Z	*0.3, 1.0*

4.3.4.2 Extraire l'ensemble des motifs fréquents singletons

Pour identifier l'ensemble des motifs fréquents singletons à partir de la base de données transactionnelle probabiliste, nous devons considérer l'ensemble W de tous les mondes possibles générés à partir de cette base de donnée. Chaque monde possible est obtenu en prenant une décision pour chaque élément incertain quant à sa présence effective ou non. Ainsi, un monde possible w_i (le monde possible) représente une réalisation possible (certaine) de la PDB avec une probabilité notée $P(w_i)$. En utilisant les mondes possibles, le support attendu d'un élément *X* dans la *PDB* est déterminé par l'équation suivante :

$$ExpSup(X) = \sum_{i=1}^{|W|} P(w_i) \times X_{w_i} \quad \text{.....(15)}$$

Le support attendu de X est égal à la somme des multiplications de la probabilité de chaque monde possible w_i et du nombre d'occurrences de X dans ce monde (noté dans

l'équation 15 par X_{w_i}). L'utilisation de cette équation est très coûteuse en termes de temps d'exécution, car le nombre de mondes possibles générés à partir d'une base de donnée probabiliste augmente de manière exponentielle en fonction du nombre d'éléments incertains.

Dans [23], Chui et al ont démontré que le support attendu d'un itemset X peut être calculé sans avoir à générer tous les mondes possibles, comme suit :

$$ExpSup(X) = \sum_{j=1}^{|PDB|} \prod_{x \in X} P(x, T_j) \quad \text{.....(16)}$$

où $P(x, T_j)$ est la probabilité existentielle de l'item x dans la transaction T_j de la base PDB.

Par application de l'équation 16 à notre exemple d'exécution, nous avons calculé le support attendu pour chaque élément singleton (voir Tableau 10) :

Tableau 10. Le support attendu pour chaque élément unique

Elément	***Support attendu (Expected Support)***
X	*3,5*
Y	*4,7*
Z	*5*
W	*2*

4.3.4.3 Construire l'arbre des motifs fréquents probabilistes

La méthode de construction de l'arbre des motifs fréquents probabilistes (PFP-tree) s'inspire de [52] où nous adaptons l'algorithme utilisé par bernecker et ses collègues pour construire notre version du PFP-tree. Dans notre travail, la construction du PFP-tree nécessite une seule passe de la base de données transactionnelle probabiliste. L'algorithme proposé prend les transactions de la base de données une par une et les insère dans le PFP-tree, de sorte que chaque transaction soit représentée par un chemin dans le PFP-tree. La transaction commence à partir de la racine de l'arbre (un nœud vide), et chaque nœud représente un élément unique dans une transaction. L'arbre est structuré comme suit (on considère un nœud n dans le PFP-Tree) :

$n.item$: une étiquette utilisée pour désigner le nœud. L'étiquette de l'élément est utilisée ici parce que chaque nœud représente un élément.

$n.list$: une liste de transactions où l'élément de ce nœud est présent. Cette liste est composée d'un ensemble de paires : chaque paire contient l'étiquette de la transaction et la probabilité existentielle de l'élément dans cette transaction (si l'élément est certain, nous mettons la probabilité existentielle égale à 1).

$n.next$: un pointeur vers le nœud suivant dans l'arbre. Ce champ doit être nul si le nœud actuel est une feuille.

Label	*list*	*next*
A	$t_1 = 1, \quad t_2 = 0.6$	t_j

L'algorithme commence avec un arbre qui ne contient qu'un seul élément ; la racine (élément nul). Ensuite, il parcourt les transactions une par une, en commençant par la transaction t_1 qui sera insérée directement comme un chemin dans l'arbre (chemin = racine $\rightarrow t_1.élément_1 \rightarrow t_1.élément_2 \ldots \rightarrow t_1.élément_n$) comme indiqué dans la Figure 8. Les transactions suivantes présentent deux possibilités :

- Soit le chemin correspondant à la transaction courante existe déjà dans l'arbre. Dans ce cas, une mise à jour est effectuée en ajoutant l'étiquette de cette transaction et la probabilité d'existence de l'élément courant à l'étiquette du nœud correspondant dans la liste des transactions.

- Soit le chemin correspondant à la transaction courante n'existe pas dans l'arbre. On procède alors à sa création complète.

La Figure 9 représente le *PFP-Tree* complet généré après l'exécution de l'algorithme sur notre exemple.

4.3.4.4 Extraction de l'ensemble des itemsets fréquents probabilistes

Dans cette étape on extrait l'ensemble des itemsets fréquents probabilistes à partir des items fréquents unitaires identifiés précédemment. On utilise le concept de support attendu défini par l'équation 16 pour les items fréquents unitaires car les données ne sont pas toutes certaines. Par exemple, en appliquant l'équation 16 à notre exemple (voir Tableau 8) pour calculer le support attendu de l'itemset $\{Z, W\}$, on obtient $ExpSup(\{Z, W\}) = 1{,}9$.

4.3.4.5 Génération de règles probabilistes

Les règles probabilistes sont générées à partir de l'ensemble des itemsets fréquents probabilistes. Pour chaque itemset fréquent *X*, tous ses sous-ensembles non vides sont générés. Par exemple, si *Y* est un sous-ensemble de *X*, nous considérons la règle $R : Y \rightarrow X - s$ si sa confiance (voir Equation 17) est supérieure ou égale à *minConf*, où *minConf* est le seuil de confiance minimum. Bien sûr, lorsque nous voulons calculer la confiance d'une règle générée, nous utilisons simplement le PFP-tree généré plutôt que d'accéder à la base de données.

$$Conf(R) = ExpSup(X)/ExpSup(Y) \quad \text{.....(17)}$$

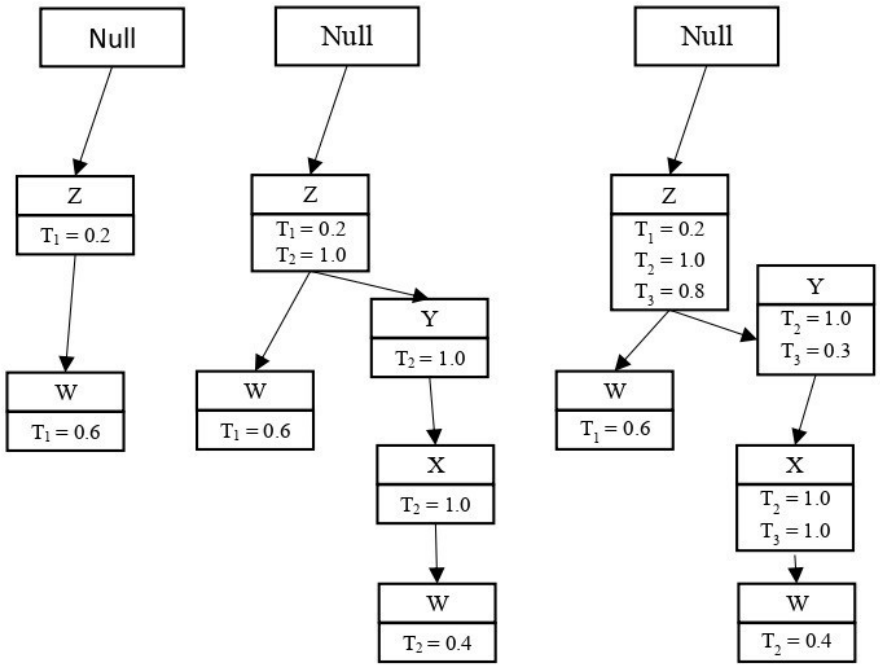


Figure 9. Évolution (de gauche à droite) de l'arbre PFP après l'insertion des trois premières transactions (Etape 3).

4.3.4.6 Filtrage des règles d'associations probabilistes

Après avoir extrait l'ensemble des règles d'associations probabilistes, nous passons à l'étape de filtrage séquentiel, de sorte que pour chaque règle trouvée à l'étape précédente, nous allons calculer le support et la confiance séquentiels tels qu'ils sont définis dans le chapitre précédent. Pour ne conserver finalement que les règles dont la confiance séquentielle est supérieure ou égale à une valeur prédéfinie nommée "confiance séquentielle minimale" et notée *SeqConf*.

Si $X \rightarrow Y$ est une règle parmi les règles identifiées à l'étape précédente, on définit son support séquentiel (*SeqSupp*) par la formule :

$$SeqSup(X \rightarrow Y) = Sup(X \bullet Y)/Card(S) \text{.....(18)}$$

où $Card(S)$ est le nombre total de séquences dans la base de données et $Sup(X \bullet Y)$ représente le nombre de séquences dans la base de données où X apparaît avant Y.

La confiance séquentielle de la règle est calculée en divisant le nombre de séquences où le côté gauche apparaît avant le côté droit de la règle, par le nombre de séquences qui contiennent le côté gauche de la règle :

$$SeqConf(X \rightarrow Y) = Sup(X \bullet Y)/Sup(X) \text{.....(19)}$$

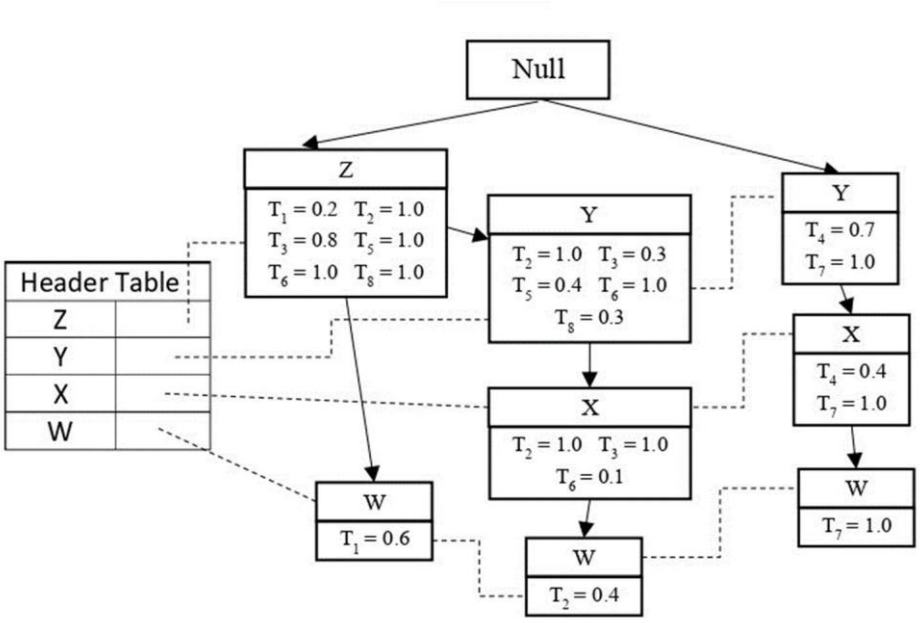


Figure 10. L'arbre PFP après avoir parcouru toute la base de données transactionnelle probabiliste

À la fin de cette étape, seules les règles respectant les contraintes temporelles et ayant une confiance séquentielle supérieure ou égale à un seuil prédéfini sont sélectionnées.

4.3.5 Expérimentations et discussions des résultats

4.3.5.1 Plateforme

L'algorithme proposé a été réalisée sur un ordinateur équipé d'un processeur Intel Core i5 avec 16 Go de mémoire et fonctionnant sous Windows, nous avons utilisé Python comme langage de programmation.

4.3.5.2 Ensembles de données (datasets)

Pour tester notre algorithme, nous avons utilisé à la fois un dataset réel (Mushroom [53]) et des datasets synthétiques :

- Les datasets synthétiques.

  Les données sont générées aléatoirement. Nous avons simulé la définition d'une base de données de séquences probabilistes (PSDB) et généré des données contenant un ensemble de séquences. Chaque séquence est composée d'un ensemble d'éléments, dont chaque élément possède sa propre probabilité existentielle comprise entre zéro et un. Ensuite, nous avons créé un ensemble de fichiers (10 fichiers) contenant un certain nombre de ces séquences. Le premier fichier comporte 100 séquences tandis que le dernier en comporte 8 k séquences.

- L'ensemble de données Mushroom

  Le dataset Mushroom utilisé dans ce travail [53] comprend des descriptions d'échantillons académiques appartenant à 23 espèces de champignons à lamelles de la famille des Agaricus et des Lepiota. Chaque espèce est classée comme comestible, vénéneuse ou comestible mais inconnue et déconseillée. Cette dernière catégorie est regroupée avec les espèces toxiques. Le guide précise qu'il n'existe pas de règle simple pour déterminer la comestibilité d'un champignon. Dans cette phase de recherche, nous n'avons pas trouvé de base de données séquentielles probabilistes utilisée comme référence pour évaluer ce type de travaux. Cependant, une base de données séquentielles peut être obtenue à partir d'une base de données transactionnelle en considérant simplement l'ordre des éléments dans chaque transaction. Une base de données probabiliste est quant à elle une base de données ordinaire où chaque élément possède sa propre probabilité existentielle.

Par conséquent, pour obtenir une base de donnée séquentielle probabiliste nous avons utilisé la base de données Mushroom, considéré chaque transaction comme une séquence, et ajouté à chaque élément une valeur aléatoire comprise entre zéro et un représentant une probabilité existentielle.

4.3.5.3. Tester l'algorithme sur les données synthétiques

La figure 10-(a) présente la complexité spatiale de l'algorithme proposé. L'axe des *x* représente le nombre de séquences dans nos données synthétiques, et l'axe des *y* la quantité de mémoire nécessaire à l'exécution de notre algorithme en mégaoctets. Ce test révèle que l'espace mémoire consommé par l'algorithme augmente en fonction du volume des données d'entrée. Cependant, l'algorithme montre ses meilleures performances lorsque les données d'entrée deviennent plus volumineuses, où l'espace mémoire utilisé dans cette phase reste stable et continue d'augmenter mais à un taux négligeable

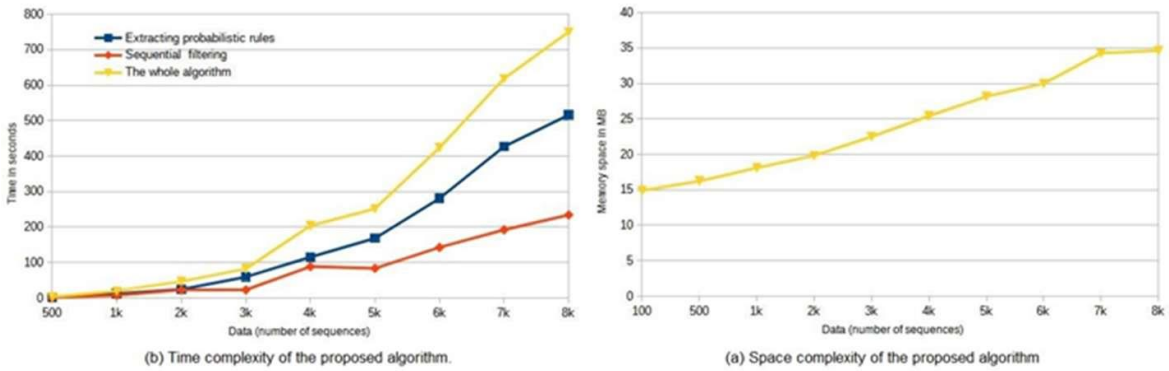


Figure 11. Complexité temporelle et spatiale de l'algorithme proposé sur des données synthétiques.

Le deuxième graphique (Figure 10-(b)) représente le temps d'exécution par l'algorithme proposé pour générer l'ensemble des règles séquentielles, mesuré en secondes. Le temps dépend du nombre de séquences dans les données. On peut observer trois courbes sur la figure 10-(b) : la première représente le temps d'exécution total de l'algorithme, la deuxième le temps nécessaire pour générer l'ensemble de règles probabilistes et la dernière pour le temps consacré au filtrage séquentiel, où la valeur du premier temps correspond à la somme des deux temps restants.

Il est évident que le temps d'exécution augmente en fonction du nombre de séquences. Cependant, en termes de complexité temporelle, la figure montre que la complexité de

l'algorithme proposé est tout à fait acceptable et se situe en dessous de $O(nlog(n))$ ou de $O(n^2)$ dans le pire des cas.

4.3.5.4. Test de l'algorithme sur des données réelles

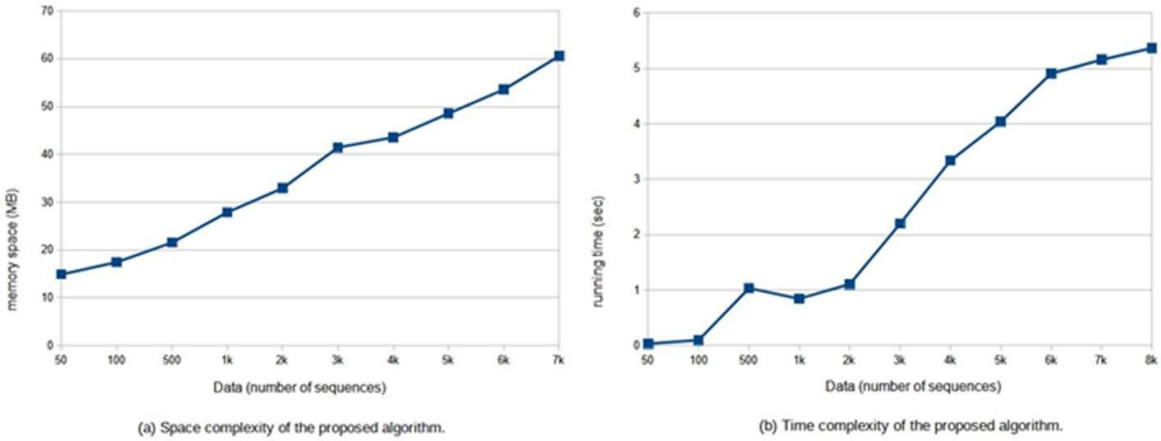


Figure 12. Complexité temporelle et spatiale de l'algorithme proposé testé sur des données réelles

La figure 11 montre la complexité spatiale et temporelle de l'algorithme proposé appliqué au dataset mushroom. Cette série d'expériences est réalisée avec la valeur de (support minimum / nombre de séquences = 0,5). Cette figure contient deux graphiques. Le premier, 11-(a), montre la complexité spatiale de notre algorithme. L'axe des *x* représente la taille de l'ensemble de données mesurée en termes de nombre de séquences, tandis que la quantité d'espace mémoire utilisée dans chaque expérience apparaît sur l'axe des *y*. Le graphique confirme que le comportement de l'algorithme proposé peut être classé comme une complexité spatiale quadratique par rapport à la taille des données.

Le deuxième graphique (11-b) représente le temps d'exécution de notre algorithme en fonction du nombre de séquences dans l'ensemble de données. Ici aussi, l'algorithme présente un comportement quadratique, le temps d'exécution augmentant avec le nombre de séquences. On peut observer que dans certains cas, le temps d'exécution diminue entre deux expériences ; Bien que les données de la première expérience soient plus volumineuses que celles de la deuxième, cela s'explique par le fait que le nombre de règles trouvées dans la deuxième expérience est supérieur à celui de la première, ce qui réduit le temps nécessaire pour traiter ce nombre de règles, ce qui se reflète dans le graphique.

4.4 Conclusion

Ce chapitre a présenté nos deux principales contributions. La première contribution est une nouvelle méthode pour l'extraction de motifs séquentiels fréquents à partir de bases de séquences probabilistes. La deuxième contribution s'est intéressée à l'extraction de règles séquentielles à partir des bases de données incertaines, l'incertitude est modélisée pour chaque item appartenant à chaque séquence est associé à une probabilité existentielle. Notre méthode proposée a contribué à découvrir une nouvelle approche pour extraire des règles séquentielles à partir de bases de données incertaines, où nous expliquons en détail les six étapes de notre proposition, illustrées par un exemple d'exécution. Nous avons présenté l'ensemble des résultats obtenus sur des données synthétiques et réelles. La méthode proposée a montré une performance remarquable en termes de complexité spatiale et temporelle.

Conclusion générale et Perspectives

De nos jours, de grandes quantités d'informations séquentielles sont stockées dans les bases de données. Ces grands ensembles de données contiennent à la fois des données utiles et inutiles. La découverte de relations séquentielles dans les bases de données temporelles est importante dans de nombreux domaines tels que la bio-informatique pour stocker les séquences d'ADN (acide désoxyribonucléique) ou de protéines pour faire des prédictions.

Dans le domaine de la fouille de données, l'une des techniques les plus importantes est l'exploration de motifs séquentiels (SPM) utiles à partir d'un large ensemble de séquences qui apparaissent dans une base de données séquentielle. On s'intéresse souvent à des motifs séquentiels dont la fréquence est au moins égale à un seuil donné *minsup* défini par l'utilisateur. Parmi les algorithmes déjà introduits dans ce contexte, on peut citer PrefixSpan, Spade, SPAM, GSP, etc. Cependant, savoir qu'une séquence apparaît fréquemment dans une base de séquences n'est pas toujours suffisant pour faire des prédictions. Il a été constaté que l'extraction de motifs séquentiels fréquents pourrait être trompeuse, c'est pourquoi est apparue la tâche d'extraction de règles séquentielles. Les règles séquentielles sont des motifs qui soulignent des liens importants entre des sous-séquences d'événements dans une base de données séquentielle.

Plusieurs recherches ont étudié l'extraction de règles séquentielles déterministes, mais dans de nombreux domaines d'application, les données disponibles ne sont pas parfaites et sont entachées de différentes formes d'incertitude. A titre d'exemple, les données collectées à partir de réseaux de capteurs sans fil peuvent être incertaines en raison de facteurs d'interférence externes ou du vieillissement des capteurs. Proposer des solutions qui gèrent correctement l'incertitude des données afin de continuer à pouvoir extraire des conclusions utiles devient ainsi une nécessité de recherche d'une grande importance. Des recherches se sont penchées sur l'incertitude des bases de données afin d'améliorer les performances de la fouille de données où l'incertitude est représentée en attribuant des probabilités existentielles aux données.

Notre principale contribution dans cette thèse est de généraliser les méthodes d'extraction de motifs séquentiels fréquents et de règles de séquences à partir de bases séquentielles déterministes d'un côté et de bases transactionnelles probabilistes d'un autre côté pour pouvoir gérer à la fois l'aspect séquentiel et l'aspect incertain dans un même cadre. L'objectif est donc de de proposer une nouvelle approche pour extraire des motifs séquentiels et des règles séquentielles probabilistes à partir d'une base de données séquentielle probabiliste. Pour cela, nous avons suivi une approche divisée en deux étapes principales : la première consiste à ignorer dans un premier temps l'information temporelle et d'extraire l'ensemble des itemsets fréquents ou de règles d'association probabilistes. La seconde étape consiste ensuite à filtrer l'ensemble de motifs ou de règles obtenus par l'étape précédente en utilisant l'information séquentielle présente dans les données afin de générer des règles séquentielles probabiliste valides. Les études expérimentales menées montrent clairement que l'approche établie possède de bonnes performances en termes de complexité spatiale et temporelle.

A notre connaissance, aucun travail antérieur n'a considéré le problème d'extraction des règles séquentielles dans une base de données qui soit à la fois probabiliste et incertaines. Par conséquent, nous n'avons pas pu faire une comparaison de nos résultats avec des recherches similaires dans le même domaine.

Il existe plusieurs orientations à envisager pour l'avenir. Nous souhaitons étendre notre travail pour couvrir le cas du modèle d'incertitude des tuples expliqué précédemment. Nous visons aussi à construire une (des) base(s) de données séquentielle(s) probabiliste(s) à partir de données réelles issues d'applications pratiques (réseaux de communications, domaine médical, domaine de la finance, bio-informatique, etc.). Ces bases de données seraient utiles pour tester les nouveaux algorithmes de fouille de données proposés et d'apporter des outils intelligents d'aide à la décision dans divers domaines pratiques.

Bibliographie

[1] R. Agrawal et R. Srikant, « Mining sequential patterns », in *Proceedings of the Eleventh International Conference on Data Engineering*, mars 1995, p. 3-14. doi: 10.1109/ICDE.1995.380415.

[2] R. Agrawal, T. Imieliński, et A. Swami, « Mining association rules between sets of items in large databases », in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, in SIGMOD '93. New York, NY, USA: Association for Computing Machinery, juin 1993, p. 207-216. doi: 10.1145/170035.170072.

[3] B. Goethals, « Survey on Frequent Pattern Mining », 2003. Consulté le: 17 février 2023. [En ligne]. Disponible sur: https://www.semanticscholar.org/paper/Survey-on-Frequent-Pattern-Mining-Goethals/0426f2081a6d08bb0e8388d1135eb99de33744b3

[4] J. Hipp, U. Güntzer, et G. Nakhaeizadeh, « Algorithms for association rule mining — a general survey and comparison », *ACM SIGKDD Explor. Newsl.*, vol. 2, n° 1, p. 58-64, juin 2000, doi: 10.1145/360402.360421.

[5] Y.-X. Tong, L. Chen, et J. She, « Mining Frequent Itemsets in Correlated Uncertain Databases », *J. Comput. Sci. Technol.*, vol. 30, n° 4, p. 696-712, juill. 2015, doi: 10.1007/s11390-015-1555-9.

[6] M. Muzammal, « Mining sequential patterns from probabilistic data », Ph.D., University of Leicester, 2012. Consulté le: 17 février 2023. [En ligne]. Disponible sur: https://figshare.com/articles/Mining_Sequential_Patterns_from_Probabilistic_Data/10142480

[7] A. Rajak et M. Gupta, « Association Rule Mining: Applications in Various Areas », in *International Conference on Data Management.*, 2009.

[8] A. Don *et al.*, « Discovering interesting usage patterns in text collections: integrating text mining with visualization », in *Proceedings of the sixteenth ACM conference on Conference*

on information and knowledge management, Lisbon Portugal: ACM, nov. 2007, p. 213-222. doi: 10.1145/1321440.1321473.

[9] R. Agrawal et R. Srikant, « Fast Algorithms for Mining Association Rules in Large Databases », in *Proceedings of the 20th International Conference on Very Large Data Bases*, in VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., sept. 1994, p. 487-499.

[10] R. Srikant et R. Agrawal, « Mining sequential patterns: Generalizations and performance improvements », P. Apers, M. Bouzeghoub, et G. Gardarin, Éd., in Lecture Notes in Computer Science, vol. 1057. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, p. 1-17. doi: 10.1007/BFb0014140.

[11] j Han, j Pei, et M. Kamber, *Data mining: concepts and techniques*. Amsterdam: Elsevier ; Morgan Kaufmann, 2011.

[12] A. Meenakshi, « SURVEY OF FREQUENT PATTERN MINING ALGORITHMS IN HORIZONTAL AND VERTICAL DATA LAYOUTS », 2015.

[13] Y. J. M. Pokou, P. Fournier-Viger, et C. Moghrabi, « Authorship Attribution Using Small Sets of Frequent Part-of-Speech Skip-grams », présenté à The Florida AI Research Society, 2016. Consulté le: 26 janvier 2023. [En ligne]. Disponible sur: https://www.semanticscholar.org/paper/Authorship-Attribution-Using-Small-Sets-of-Frequent-Pokou-Fournier-Viger/7a92e77b078a2d9bfe9abeb907cce23eb5633575

[14] Y. W. Trio Pramono et Suhardi, « Anomaly-based intrusion detection and prevention system on website usage using rule-growth sequential pattern analysis: Case study: Statistics of Indonesia (BPS) website », *2014 Int. Conf. Adv. Inform. Concept Theory Appl. ICAICTA*, p. 203-208, août 2014, doi: 10.1109/ICAICTA.2014.7005941.

[15] M. Zaki, N. Lesh, et M. Ogihara, « PLANMINE: Predicting plan failures using sequence mining », *Artif Intell Rev*, vol. 14, p. 421-446, déc. 2000, doi: 10.1023/A:1006612804250.

[16] P. Fournier Viger, C.-W. Lin, U. Rage, Y. S. Koh, et R. Thomas, « A Survey of Sequential Pattern Mining », *Data Sci. Pattern Recognit.*, vol. 1, p. 54-77, févr. 2017.

[17] M. J. Zaki, « SPADE: An Efficient Algorithm for Mining Frequent Sequences », *Mach. Learn.*, vol. 42, n° 1, p. 31-60, janv. 2001, doi: 10.1023/A:1007652502315.

[18] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, et M. C. Hsu, « FreeSpan: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001) », *Proceeding Sixth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, p. 355-359, 2000.

[19] Jian Pei *et al.*, « PrefixSpan,: mining sequential patterns efficiently by prefix-projected pattern growth », in *Proceedings 17th International Conference on Data Engineering*, Heidelberg, Germany: IEEE Comput. Soc, 2001, p. 215-224. doi: 10.1109/ICDE.2001.914830.

[20] T.-R. Li, Y. Xu, D. Ruan, et W. Pan, « Sequential Pattern Mining* », in *Intelligent Data Mining: Techniques and Applications*, D. Ruan, G. Chen, E. E. Kerre, et G. Wets, Éd., in Studies in Computational Intelligence. , Berlin, Heidelberg: Springer, 2005, p. 103-122. doi: 10.1007/11004011_5.

[21] A. D. Sarma, O. Benjelloun, A. Halevy, et J. Widom, « Working Models for Uncertain Data », in *22nd International Conference on Data Engineering (ICDE'06)*, avr. 2006, p. 7-7. doi: 10.1109/ICDE.2006.174.

[22] P. Sen, A. Deshpande, et L. Getoor, « Representing Tuple and Attribute Uncertainty in Probabilistic Databases », in *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, oct. 2007, p. 507-512. doi: 10.1109/ICDMW.2007.11.

[23] C.-K. Chui, B. Kao, et E. Hung, « Mining Frequent Itemsets from Uncertain Data », in *Advances in Knowledge Discovery and Data Mining*, Z.-H. Zhou, H. Li, et Q. Yang, Éd., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, p. 47-58. doi: 10.1007/978-3-540-71701-0_8.

[24] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, et A. Zuefle, « Probabilistic frequent itemset mining in uncertain databases », in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, in KDD '09. New York, NY, USA: Association for Computing Machinery, juin 2009, p. 119-128. doi: 10.1145/1557019.1557039.

[25] Y. Tong, L. Chen, Y. Cheng, et P. S. Yu, « Mining Frequent Itemsets over Uncertain Databases », 1 août 2012, *arXiv*: arXiv:1208.0292. doi: 10.48550/arXiv.1208.0292.

[26] C. K.-S. Leung, M. A. F. Mateo, et D. A. Brajczuk, « A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data », in *Advances in Knowledge Discovery and Data Mining*, T. Washio, E. Suzuki, K. M. Ting, et A. Inokuchi, Éd., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, p. 653-661. doi: 10.1007/978-3-540-68125-0_61.

[27] C. C. Aggarwal, Y. Li, J. Wang, et J. Wang, « Frequent pattern mining with uncertain data », in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, in KDD '09. New York, NY, USA: Association for Computing Machinery, juin 2009, p. 29-38. doi: 10.1145/1557019.1557030.

[28] J. Han, J. Pei, et Y. Yin, « Mining frequent patterns without candidate generation », *ACM SIGMOD Rec.*, vol. 29, n° 2, p. 1-12, mai 2000, doi: 10.1145/335191.335372.

[29] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, et D. Yang, « H-mine: hyper-structure mining of frequent patterns in large databases », in *Proceedings 2001 IEEE International Conference on Data Mining*, nov. 2001, p. 441-448. doi: 10.1109/ICDM.2001.989550.

[30] L. Sun, R. Cheng, D. W. Cheung, et J. Cheng, « Mining uncertain data with probabilistic guarantees », in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, in KDD '10. New York, NY, USA: Association for Computing Machinery, juill. 2010, p. 273-282. doi: 10.1145/1835804.1835841.

[31] T. Bernecker *et al.*, « Model-based probabilistic frequent itemset mining », *Knowl. Inf. Syst.*, vol. 37, n° 1, p. 181-217, oct. 2013, doi: 10.1007/s10115-012-0561-2.

[32] M. Muzammal et R. Raman, « Mining Sequential Patterns from Probabilistic Databases », présenté à Knowledge and Information Systems, mai 2011, p. 210-221. doi: 10.1007/978-3-642-20847-8_18.

[33] Z. Zhao, D. Yan, et W. K. Ng, « Mining Probabilistically Frequent Sequential Patterns in Large Uncertain Databases », *IEEE Trans. Knowl. Data Eng.*, vol. 26, p. 1171-1184, mai 2014, doi: 10.1109/TKDE.2013.124.

[34] P. Fournier-Viger, U. Faghihi, R. Nkambou, et E. M. Nguifo, « CMRules: Mining sequential rules common to several sequences », *Knowl.-Based Syst.*, vol. 25, n° 1, p. 63-76, févr. 2012, doi: 10.1016/j.knosys.2011.07.005.

[35] P. Fournier-Viger, T. Gueniche, et V. S. Tseng, « Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction », in *Advanced Data Mining and Applications*, vol. 7713, S. Zhou, S. Zhang, et G. Karypis, Éd., in Lecture Notes in Computer Science, vol. 7713. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 431-442. doi: 10.1007/978-3-642-35527-1_36.

[36] P. Fournier-Viger, R. Nkambou, et V. S.-M. Tseng, « RuleGrowth: mining sequential rules common to several sequences by pattern-growth », in *Proceedings of the 2011 ACM Symposium on Applied Computing*, in SAC '11. New York, NY, USA: Association for Computing Machinery, mars 2011, p. 956-961. doi: 10.1145/1982185.1982394.

[37] H. Mannila, H. Toivonen, et A. Inkeri Verkamo, « Discovery of Frequent Episodes in Event Sequences », *Data Min. Knowl. Discov.*, vol. 1, n° 3, p. 259-289, sept. 1997, doi: 10.1023/A:1009748302351.

[38] H. J. Hamilton et K. Karimi, « The TIMERS II Algorithm for the Discovery of Causality », in *Advances in Knowledge Discovery and Data Mining*, T. B. Ho, D. Cheung, et H. Liu, Éd., Berlin, Heidelberg: Springer, 2005, p. 744-750. doi: 10.1007/11430919_86.

[39] D.-L. Yang, Y. L. Hsieh, et J. Wu, « Using Data Mining to Study Upstream and Downstream Causal Relationship in Stock Market », présenté à 9th Joint International Conference on Information Sciences (JCIS-06), Atlantis Press, oct. 2006, p. 528-531. doi: 10.2991/jcis.2006.191.

[40] J. Deogun et L. Jiang, « Prediction Mining – An Approach to Mining Association Rules for Prediction », in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, D. Ślęzak, J. Yao, J. F. Peters, W. Ziarko, et X. Hu, Éd., Berlin, Heidelberg: Springer, 2005, p. 98-108. doi: 10.1007/11548706_11.

[41] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, et P. Smyth, « Rule Discovery from Time Series ».

[42] S. K. Harms, J. Deogun, et T. Tadesse, « Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences », in *Foundations of Intelligent Systems*, M.-S. Hacid, Z. W. Raś, D. A. Zighed, et Y. Kodratoff, Éd., Berlin, Heidelberg: Springer, 2002, p. 432-441. doi: 10.1007/3-540-48050-1_47.

[43] D. Lo, S.-C. Khoo, et L. Wong, « Non-redundant sequential rules—Theory and algorithm », *Inf. Syst.*, vol. 34, n° 4, p. 438-453, juin 2009, doi: 10.1016/j.is.2009.01.002.

[44] A. Pitman et M. Zanker, « An Empirical Study of Extracting Multidimensional Sequential Rules for Personalization and Recommendation in Online Commerce ».

[45] Y. Zhao, H. Zhang, L. Cao, C. Zhang, et H. Bohlscheid, « Efficient Mining of Event-Oriented Negative Sequential Rules », in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, déc. 2008, p. 336-342. doi: 10.1109/WIIAT.2008.60.

[46] P. Fournier-Viger, T. Gueniche, S. Zida, et V. S. Tseng, « ERMiner: Sequential Rule Mining Using Equivalence Classes », in *Advances in Intelligent Data Analysis XIII*, H. Blockeel, M. van Leeuwen, et V. Vinciotti, Éd., Cham: Springer International Publishing, 2014, p. 108-119. doi: 10.1007/978-3-319-12571-8_10.

[47] U. Faghihi, P. Fournier-Viger, et R. Nkambou, « CELTS: A Cognitive Tutoring Agent with Human-Like Learning Capabilities and Emotions », in *Intelligent and Adaptive Educational-Learning Systems: Achievements and Trends*, A. Peña-Ayala, Éd., Berlin, Heidelberg: Springer, 2013, p. 339-365. doi: 10.1007/978-3-642-30171-1_14.

[48] Ö. F. Çelebi, E. Zeydan, İ. Arı, Ö. İleri, et S. Ergüt, « Alarm sequence rule mining extended with a time confidence parameter », 2014. Consulté le: 11 mai 2024. [En ligne]. Disponible sur: https://eresearch.ozyegin.edu.tr/handle/10679/1957

[49] B. Kamsu-Foguem, F. Rigal, et F. Mauget, « Mining association rules for the quality improvement of the production process », *Expert Syst. Appl.*, vol. 40, n° 4, p. 1034-1045, mars 2013, doi: 10.1016/j.eswa.2012.08.039.

[50] E. A. Z. Noughabi, A. Albadvi, et B. H. Far, « How Can We Explore Patterns of Customer Segments' Structural Changes? A Sequential Rule Mining Approach », in *2015 IEEE International Conference on Information Reuse and Integration*, août 2015, p. 273-280. doi: 10.1109/IRI.2015.52.

[51] I. Seddiki, F. Nouioua, et A. Barkat, « Extracting sequential frequent itemsets from probabilistic sequences database », *Int. J. Inf. Technol.*, vol. 15, n° 5, p. 2509-2515, juin 2023, doi: 10.1007/s41870-023-01292-w.

[52] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, et A. Züfle, « Probabilistic Frequent Pattern Growth for Itemset Mining in Uncertain Databases », in *Scientific and Statistical Database Management*, A. Ailamaki et S. Bowers, Éd., Berlin, Heidelberg: Springer, 2012, p. 38-55. doi: 10.1007/978-3-642-31235-9_3.

[53] D. Mushroom, « UCI Machine Learning Repository », 1987.