REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

N° :……..

DOMAINE : Mathématiques et Informatique

FILIERE : Informatique

OPTION : Réseaux et Technologie de l'Information et de la Communication

Mémoire présenté pour l'obtention
Du diplôme de Master Académique

Par : OUAIL Fatiha

Intitulé

Extraction des règles séquentielles avec la contrainte de taille de fenêtre (L'algorithme TRuleGrowth)

Soutenu devant le jury composé de :

M. BOUNIF Mohamed	Université de M'sila	Président
M. GUESMIA Salah	Université de M'sila	Rapporteur
M. LOUCIF Hamza	Université de M'sila	Examinateur

Année universitaire : 2017 /2018

Dédicaces

A mes très chers parents, Pour leurs sacrifices, leur amour, leurs prières et leur soutien.

A ma chère famille, Petite ou grande, proche ou lointaine.

A tous nos enseignants avec notre profonde considération, qui n'ont épargné aucun effort pour nous offrir un bon enseignement.

A tous ceux qui nous ont assistés, dans la réalisation et le bon déroulement de ce travail.

OUAIL.F

Remerciements

Je tiens à remercier, Monsieur ***Guesmia.Salah****, pour ses conseils judicieux, et les précieuses discussions que nous avons eues ensemble. Je lui exprime ma profonde gratitude pour m'avoir fait profiter de ses connaissances.*

J'exprime toute ma gratitude aux des membres de jury, monsieur ***Bounif mohamed*** *et monsieur* ***Loucif Hamza*** *pour avoir accepté de juger ce travail et pour leurs remarques constructives qui m'ont permis d'améliorer grandement mon document de thèse.*

Enfin un grand merci à mes parents et à mes frères et sœurs, qui mon toujours soutenu, et je leur dédie ce travail.

Table Des Matières

Table Des Matières

Liste des Tableaux

Liste des Figures

Liste des Figures

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Durant ces dernières années, on assiste à une forte augmentation tant dans le nombre que dans le volume des informations mémorisées par des bases des données scientifiques, informatique, économiques, financières, administratives, médicales etc.

Le stockage en lui-même ne pose pas de réelles difficultés du point de vue informatique, mais le besoin d'interpréter ou de trouver des nouvelles relations entre les éléments stockés dans ces bases des données a suscité beaucoup d'intérêt. Ainsi, la mise au point des nouvelles techniques informatiques est devenu un thème important pour un bon nombre de chercheurs. Donc la " Fouille de données " est alors devenue rapidement un domaine de recherche émergeant essayant de répondre à ces objectifs.

Dans ce domaine, la recherche des règles séquentielles est une méthode populaire étudiée d'une manière approfondite, dont le but est de découvrir des relations temporelles entre l'occurrence de deux ou plusieurs événements stockés dans des très importantes bases des données.

Une règle séquentielle (également appelée règle d'épisode, règle temporelle ou règle de prédiction) indique que si des événements se produisent, d'autres événements sont susceptibles de suivre avec une confiance ou une probabilité donnée. L'extraction de ce type de règles a été appliquée dans plusieurs domaines tels que l'analyse du marché boursier, l'observation météorologique, la gestion de la sécheresse et le commerce électronique.

L'objectif de ce travail est d'étudier quelques algorithmes spécifiques d'extraction des règles séquentielles, afin d'évaluer et de comparer leurs performances en fonction de plusieurs paramètres et différentes bases des séquences.

Le reste de ce mémoire est organisé de la façon suivante :

❖ Nous consacrerons le premier chapitre à la présentation des principes fondamentaux du processus d'extraction des connaissances, ainsi que les taches et les

techniques de la fouille de données comme la classification, la segmentation et les règles d'associations.

- Le second chapitre portera sur la présentation des différents concepts nécessaires pour mieux introduire les motifs et les règles séquentielles.

- Le troisième chapitre fait le point sur l'extraction des règles séquentielles sans et avec la contrainte de taille de fenêtre en utilisant les algorithmes *RuleGrowth* et *TRuleGrowth*.

- Le quatrième chapitre est consacré à la présentation des outils utilisés pour développer notre application et à la discussion sur les résultats obtenus après l'exécution des algorithmes étudiés sur des données réels.

Enfin, nous concluons notre mémoire par une conclusion générale et des perspectives.

CHAPITER I: FOUILLE DE DONNÉES

Chapitre I: Fouille de Donnée (Data Mining)

1.1 /INTRODUCTION :

L'accroissement du volume des données stockées dans les bases des données a fait qu'il est devenu urgent et vital de recourir à des techniques et des méthodes pour extraire l'information à partir de cette masse volumineuse des données. La **" Fouille de données "** est alors devenue rapidement un thème de recherche suscitant l'intérêt de la communauté scientifique. La **"Fouille de données "** connue aussi sous l'expression de Datamining, Forage de données, prospection de données, ou encore extraction de connaissances à partir des données, a pour objet l'extraction d'un savoir ou d'une connaissance à partir de grandes quantités des données, par des méthodes automatiques ou semi-automatiques.

Dans ce chapitre, nous allons d'abord faire un tour d'horizon sur l'extraction des connaissances, ensuite nous verrons les concepts principaux de Fouille de données, et plus particulièrement les taches et les techniques de Fouille de données comme la classification, segmentation et les règles d'associations….

1.2/ Pourquoi et d'où vient l'extraction des connaissances?

On avait besoin d'une discipline pour résumer automatiquement les données, pour extraire l'essence de l'information stockée et pour découvrir des motifs (modèles) dans les ensembles des données. En 1989, **Gregory Piatetsky-Shapiro** a donné un nom à cette discipline, qu'est : **Extraction des Connaissances à partir de Données (ECD)**, ou en Anglais : **Knowledge Discovery in Data bases (KDD).**

Il ne s'agissait pas d'une discipline toute nouvelle, car elle avait des racines bien ancrées, et des intersections avec plusieurs domaine, tel que les statistiques, l'intelligence artificielle, L'apprentissage automatique, les bases de données… **[1]**

La première définition de l'ECD a été donnée par **Fayyad, Piatetsky-Shaphiro** et **Smith** en 1996 :

« Un processus non-trivial d'identification de structures compréhensibles, potentiellement utiles, valides et nouvelles (inconnues) dans les bases de données ». **[2]**

1.3/ Etapes du processus de l'ECD:

Le processus d'extraction des connaissances dans les bases de données (ECD), présenté sur la Figure 1.1, désigne l'ensemble des opérations qui permettent d'exploiter rapidement des données stockées en grande quantités. **[3]**

Les deux premières étapes, la sélection et le prétraitement, regroupent les différentes transformations que les données doivent subir avant d'être disponibles pour l'analyse. Une fois mises en forme, les données sont traitées par l'algorithme d'extraction, c'est l'étape de Fouille de données. Cette phase consiste à utiliser des méthodes d'analyse de données et des algorithmes de recherche qui permettent d'obtenir, dans temps acceptable, un certain nombre de schémas et de motifs caractéristiques des données. Il s'agit d'un processus interactif d'analyse d'un grand ensemble de données, destiné à extraire des connaissances exploitables. Après la phase de Fouille de données, les connaissances extraites ne sont pas directement utilisables par l'utilisateur. Les schémas extraits sont à la fois très nombreux et difficilement compréhensibles pour les non experts du domaine de l'ECD. De ce fait, dans le processus ECD, une phase de post-traitement est proposée pour interpréter des connaissances extraites et les rendre utilisables et compréhensibles. **[3]**

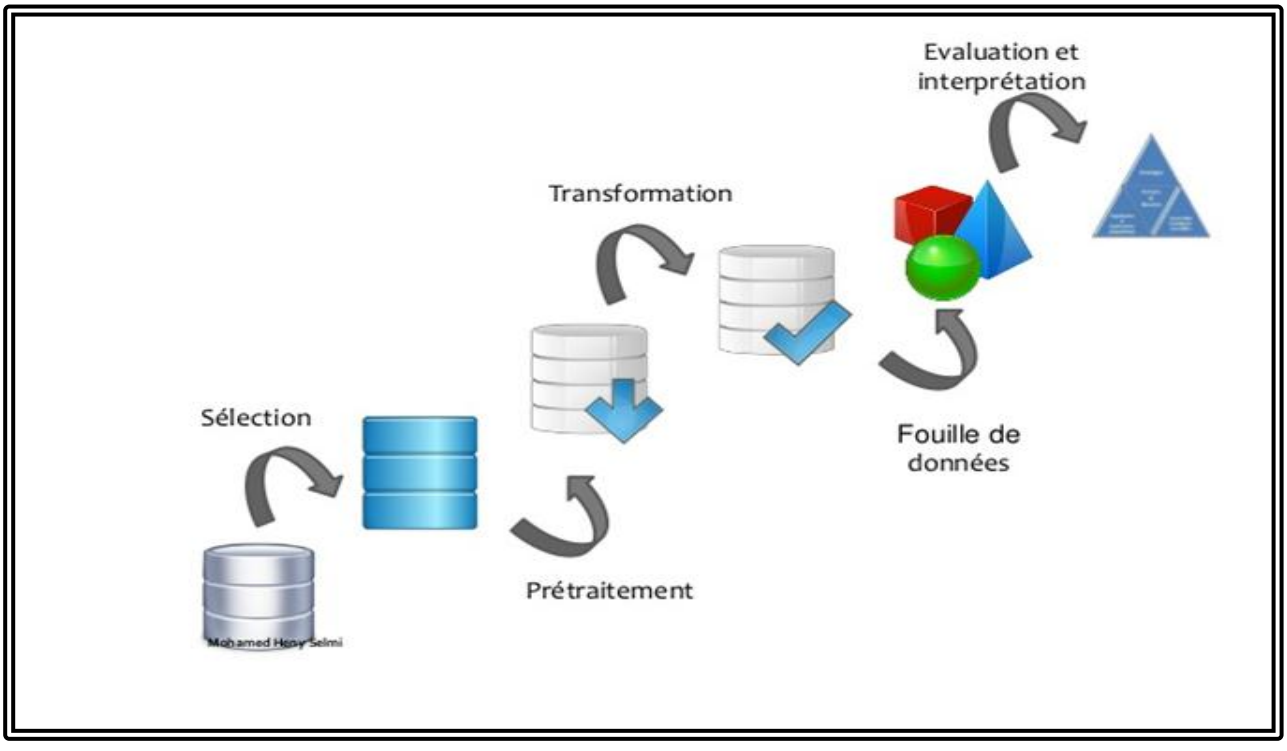

Figure 1.1 / Etapes de l'ECD

1.4/ Définition de la Fouille de données (Datamining) :

Le terme Datamining signifie littéralement « Fouille de données » ou « forage de données». En théorie, " la Fouille de données est une étape dans le processus d'extraction de connaissances, qui consiste dans l'application des algorithmes de découverte et d'analyse de données qu'est avec des limites computationnelles acceptables, produit une certaine énumération de motifs (ou modèles) à partir des données". **[2]**

Mais en pratique, on peut citez quelques définitions :

❖ La Fouille de données est un processus d'extraction d'informations recevables, compréhensibles, préalablement inconnues et valides et leurs utilisations pour prendre des décisions cruciales. **[4]**

❖ La Fouille de données est un ensemble des méthodes utilisées dans le processus d'extraction de connaissances pour distinguer des relations et des motifs préalablement inconnus dans les données. **[5]**

Autrement dit,

❖ La Fouille de données consiste à rechercher et extraire de l'information (utile et inconnue) de gros volumes de données stockées dans des bases ou des entrepôts de données.

❖ La Fouille de données est l'ensemble des techniques qui permettent de transformer les données en connaissances.

Le développement de la Fouille de données est lié en fait à plusieurs facteurs:

- ❖ une puissance de calcul importante.
- ❖ un volume sans cesse croissant des bases de données.
- ❖ un accès plus simple aux réseaux et un accroissement du débit.

Ces facteurs rendent possible le calcul distribué et la distribution d'information sur un réseau d'échelle mondiale. La Fouille de données a aujourd'hui une grande importance économique du fait qu'elle permet d'optimiser la gestion des ressources humaines et matérielles.

En tant que processus, il est pertinent de souligner ici que la Fouille de données ne se réfère pas seulement à des outils et à une technologie informatique très développée.

Effectivement, il faut également relever le rôle fondamental de l'humain dont l'implication se doit d'être totale dans chaque phase du processus, alors Il est erroné de penser que la Fouille de données est une entité qui fonctionne de manière autonome

1.5/ A quel type de données s'applique la Fouille de données ?

En principe, la Fouille de données peut s'appliquer à tous les types de données.

Toutefois, selon chaque type de données, les algorithmes de la Fouille de données sont différents. Quelques exemples de types de données. [6] auxquels peut s'appliquer la Fouille de données sont :

❖ **"Flat File" :** Ce sont des fichiers en format texte ou binaire, contenant un enregistrement par ligne, avec des champs séparés par des délimiteurs, tels que les virgules ou les tabulations. Dans ce type de fichiers, les données peuvent être des transactions, des séries temporelles, des mesures scientifiques etc.

❖**Base de données relationnelle :** Une base de données relationnelle est une base de données consistant dans des tableaux séparés, avec des liaisons explicitement définies et dont les éléments peuvent être combinés sélectivement comme des résultats à des interrogations. Chaque tableau contient des colonnes (correspondantes à des tuples) et des lignes (correspondantes à des attributs) ;

❖**Entrepôt de données :** Ce terme désigne une base de données utilisée pour collecter et stocker des informations provenant de multiples bases de données (souvent hétérogènes) et qui les traite comme un tout-unitaire (comme une seule base de données).

❖**Base de données transactionnelle :** Une base de données transactionnelle est un ensemble d'enregistrements représentant des transactions, chaque enregistrement ayant une marque temporelle, un identificateur et un ensemble d'articles.

❖**Base de données multimédia :** Ce type de base de données inclut des vidéos, des images, des audio et des textes média.

❖**Base de données temporelles :** Elles contiennent des données organisées dans le temps, comme par exemple des activités log.

❖ **Bases de données orientées objet et relationnelle objet :** Il s'agit d'un type spécial de base de données (ou base de données relationnelle) où les données sont des objets.

❖**Bases de données spatiales :** Une telle base de données est optimisée pour garder des données spatiales et de pouvoir en être interroger.

❖**World Wide Web :** Le WWW est le dépôt le plus hétérogène et dynamique possible.

1.6 / Tâches de la Fouille de données :

1.6.1/ Description :

C'est souvent l'une des premières tâches demandées à un outil de Fouille de données**.** On lui demande de décrire les données d'une base complexe. Cela engendre souvent une exploitation supplémentaire en vue de fournir des explications. L'importance de cette tâche est de permettre à l'analyste d'interpréter les résultats d'un modèle de Fouille de données ou d'un algorithme, de manière la plus transparente et efficace possible.

1.6.2/ Estimation :

La variable cible est numérique. L'estimation permet par exemple d'estimer la somme d'argent qu'un couple va dépenser pour des produits scolaires cet été ou encore d'estimer le nombre de buts que va marquer l'équipe de Suisse lors de l'Euro. On pourra également estimer le revenu d'une famille en fonction de divers critères (profession, catégorie de voiture...).

L'intérêt principal de cette tâche est de pouvoir ordonner les résultats afin d'avoir la possibilité de retenir seulement les meilleurs valeurs, technique qui sera surtout utilisée en marketing dans le but de pouvoir proposer des offres aux meilleurs clients potentiels de l'entreprise.

1.6. 3/ Classification :

La classification va s'intéresser au regroupement de données ou d'observations en groupes d'objets similaires. En d'autres termes, elle va segmenter l'ensemble des données afin de former des sous-groupes homogènes. Ceux-ci s'appellent des classes, qui sont des groupes dans lesquels les données sont similaires entre elles et, par définition différente des autres groupes.

1.6.4/ Prévision :

Les résultats de la prévision portent, comme son nom l'indique, sur le futur, ce qui la différencie de l'estimation. La prévision permet par exemple de prévoir quels vont être les gagnants du championnat de basket-ball en tenant compte de la comparaison des résultats de chaque équipe ou encore de prévoir quel va être le taux de décroissance des décès sur la route l'année suivante en prenant compte de l'augmentation des limitations de vitesse . **[7]**

1.7/ Techniques de la Fouille de données :

1.7.1/ Arbres de décision :

Ce sont des outils très puissants principalement utilisés pour la classification, la description ou l'estimation. Il s'agit d'une représentation graphique sous forme d'arbre, représentant un enchaînement hiérarchique de règles logiques qui permet de diviser la base d'exemples en sous-groupes.

1.7.2/ Réseaux de neurones :

Inspirés de la biologie, les réseaux de neurones représentent une transposition simplifiée des neurones du cerveau humain. Ils sont utilisés dans la prédiction et la classification.

1.7.3/ Algorithmes génétiques :

Ce sont des méthodes d'optimisation de fonctions basée sur les mécanismes génétiques pour élaborer des paramètres de prévision des plus optimaux. Ils permettent de résoudre des problèmes divers, notamment d'optimisation, d'affectation ou de prédiction.

1.7.4/ Règles d'association :

Ce sont les méthodes les plus répandues dans le domaine de marketing et de la distribution. Elles peuvent être appliquées dans différents secteurs d'activité pour lesquels, il est intéressant de trouver des groupements d'articles qui apparaissent le plus fréquemment ensembles, et de générer des règles d'associations. Le but principal de cette technique est descriptif ou prédictif.

1.7.5/ Plus proches voisins (CBR, Case Based Reasoning):

C'est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation, et est une technique du Fouille de données dirigé, permet de classer ou de prédire des données inconnues à partir d'instances connues.

1.7.6/ Segmentation :

La segmentation est une méthode de regroupement des éléments. Contrairement à la classification, il n'y a pas de connaissances a priori sur les classes prédéfinies des éléments. La séparation des objets dans les différents groupes (clusters) s'effectue en se basant sur le calcul de similarité entre les éléments.

L'objectif des techniques du segmentation est de grouper des éléments proches dans un même groupe de manière à ce que deux éléments d'un même groupe soient le plus similaires possible et que deux éléments de deux groupes différents soient le plus dissemblables possible.

Cette manière de définir intuitivement l'objectif de la segmentation cache la difficulté de formaliser la notion de ressemblance entre deux éléments. Une partie importante de la définition une méthode de segmentation consiste en effet à définir et formaliser une mesure de similarité adaptée aux caractéristiques des données**. [8]**

1 .8/ Application de la Fouille de données :

- **Médecine :** biomédecine, drogue, séquence génétique, gestion hôpitaux, ...
- **Finance :** assurance, crédit, prédiction du marché, détection de fraudes, …
- **Social :** données démographiques, votes, résultats d'élections,
- **Militaire :** fusion de données , Secret défense
- **Astrophysique :** astronomie

- **Marketing et ventes :** comportement des utilisateurs, prédiction des ventes,
- **Informatique :** agents, règles actives, IHM, réseau, Data Warehouse, Internet …..

1.9/ Les règles d'association :

1.9 .1 / Définition et concepts généraux:

- **Item et itemset:**

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

❖ **Transaction :**

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

❖ **Règle d'association :**

est une relation d'implication **X→Y** entre deux ensembles disjoints d'articles X et Y. Cette règle indique que les transactions qui contiennent les articles de l'ensemble X ont tendance à contenir les articles de l'ensemble Y.
X est appelé condition ou prémisse et **Y** résultat ou conclusion.
La règle d'association peut être quantifiée par un ensemble de mesures**. [9]**

Les deux mesures les plus classiques sont le support et la confiance :

❖ **Le Support :**

Le support d'un itemset est le pourcentage d'instances de la base de données qui contiennent l'itemset. (Le support permet de mesurer l'intérêt de l'itemset).

La règle **X => Y** a un support « **sup** » dans l'ensemble des transactions D si **sup**%

des transactions de D contiennent **X ∪Y. [10]**

Support (X => Y) = |(X∪ Y)|/ |D|

❖ **La Confiance :**

La confiance représenté la proportion de transactions couvrant X et qui couvrent aussi Y. La règle **X => Y** se reporte à l'ensemble de transactions avec une confiance (confidence) « **conf** » si **conf** % des transactions D qui contiennent l'ensemble des objets X contiennent aussi l'ensemble des objets Y**. [10]**

Confiance(X =>Y) = |(X ∪Y)| / Sup |X|

❖ **Itemset fréquent :** si son support est supérieur ou égal à un seuil minimale de support **(**note **: Minsup)** spécifié par l'utilisateur.

Exemple :

« Céréales ∧ Sucre → Lait (support 70%, confiance 50%) »

Cette règle d'association extraite d'une base de données de ventes de supermarché et indique que les clients qui achètent des céréales et du sucre ont également tendance à acheter du lait.

La mesure de ***support*** définit la portée de la règle, c'est à dire la proportion de clients qui ont acheté les trois articles.

La mesure de ***confiance*** définit la précision de la règle, c'est à dire la proportion de clients qui ont acheté du lait parmi ceux qui ont acheté des céréales et du sucre.

1.9.2/ L'extraction des règles d'association :

L'extraction des règles d'association est un processus itératif et interactif consiste à extraire les règles dont le support et la confiance sont au moins égaux à des seuils minimaux de support et de confiance prédéfini par l'utilisateur. [11]

Ce processus peut être décomposé en deux étapes :

- **Découverte des itemsets fréquents :**

L'extraction des itemsets fréquents permet d'isoler, depuis une base de données, des ensembles des itemset, qui satisfont un seuil minimal de support **(Minsup)** spécifié par l'utilisateur.

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

- **Génération des règles d'association :**

Sur la base des itemsets fréquents extraits à l'étape précédente, il est possible de construire des règles d'associations, de forme générales **X** =>**Y** qui associent un sous ensemble d'itemsets **X** avec un second sous-ensemble d'itemsets **Y**. [12]

1.9.3 / Domaine d'applications :

Les règles d'association ont été étudiées et employées dans de nombreux contextes et pour des domaines d'applications variés, On peut citer notamment:

- La planification commerciale
- L'aide au diagnostic et la recherche médicale
- L'amélioration des processus de télécommunications
- L'organisation et l'accès aux sites Internet
- Le marketing bancaire
- Le multimédia
- L'analyse des données spatiales et des données statistiques. **[13]**

1.10/ Conclusion :

Nous venons de présenter dans ce chapitre le processus d'extraction des connaissances à partir des données (ECD), ainsi les concepts fondamentaux de la Fouille de donnée (Datamining), et en fin nous venons présenter des visions générales sur les règles d'association, et ses domaines d'applications.

Le chapitre suivant entre dans les détails en se focalisant sur un axe de travail particulier, qu'est l'extraction des règles séquentielles.

CHAPITRE II : LES REGLES SEQUENTIELLES

2-1/ Introduction :

Dans des plusieurs domaines, les informations temporelles sont stockées dans des bases de données (par exemple des données boursières, des données biologiques, dossiers hospitaliers patients et données client). La découverte des relations temporelles dans de telles bases de données est important, car il permet une meilleure compréhension des données et établit une base pour faire des prédictions.

Par exemple, dans le commerce international, on pourrait s'intéresser à la découverte des relations entre les appréciations des monnaies pour prendre les décisions commerciales.

Divers méthodes ont été proposées pour extraire ces relations temporelles, dans le domaine de Fouille de données, l'une des techniques les plus populaires pour découvrir les relations temporelles est l'extraction des règles séquentielles.

Une règle séquentielle indique que si un ou plusieurs événements se sont produits, certains autres événements sont susceptibles de se produire.

Dans ce chapitre, nous allons expliquer les règles séquentielles et les différentes méthodes de les extraire.

2-2/ Concepts de Base :

- **« Item »** : Un item peut être défini comme un article.
- **« Itemset »** : Un itemset est un ensemble non vide d'items (d'articles),

  note **(I_1, I_2, I_3…….$I_{k)}$.**
- **« Sequence »:**

Une séquence est une liste ordonnée d'itemsets non vides. Contrairement à la théorie ensemble liste des règles d'association dont les éléments ne sont pas ordonnés, une séquence utilise le principe de précédence c'est-à-dire chaque élément de la liste est précédé des éléments qui l'ont précédé dans les transactions d'un client donné. [9]

❖ **« Base de données des séquences D » :** est un ensemble des séquences.

Par exemple, considérez la Base de données des séquences suivante:

SID	**Séquences**
seq1	{a, b},{c},{f},{g},{e}
seq2	{a, d},{c},{b},{a, b, e, f}
seq3	{a},{b},{f},{e}
seq4	{b},{f, g, h}

Tableau 2.1: *Une Base de données des séquences contient quatre séquences.* **[14]**

Cette Base des séquences contient quatre séquences nommées : seq1, seq2, seq3 et seq4.

Pour notre exemple:

Considérons que les symboles "a", "b", "c", d "," e "," f "," g "et" h " représentent respectivement certains articles vendus dans un supermarché. Par exemple, "a" pourrait représenter une "pomme", "b" pourrait être un "pain", et nous supposerons que chaque séquence représente ce qu'un client a acheté dans notre supermarché au fil du temps.

Par exemple :

Considérons la deuxième séquence **"seq2".**

Cette séquence indique que le deuxième client a acheté les articles **«a»** et **«d»** ensemble, que l'article **«c»** acheté, puis acheté **«b»,** puis acheté **«a», «b», «e»** et **«f»** ensemble .

↳ Les séquences sont un type très courant de structures de données qui peuvent être trouvées dans de nombreux domaines tels que la bioinformatique (séquence d'ADN), les séquences de clics sur des sites Web, le comportement des apprenants en e-learning, les séquences de texte..... .[14]

❖ **« Support d'une séquence S » :** Le support d'une séquence quelconque S, est le pourcentage de clients qui supportent cette séquence S. c'est une mesure dite d'utilité.

❖ **« Support Minimal »** : Le support minimal est le nombre minimum d'occurrence d'un motif séquentiel (nous l'expliquerons plus tard), pour être considéré comme fréquent, c'est un seuil choisi par l'utilisateur. **[9]**

2-3/ Les Motifs Séquentiels:

2-3-1 / Définition d'un Motif Séquentiel:

Un motif séquentiel est une sous-séquence qui apparaît de plusieurs séquences d'une base de données des séquences**. [14]**

Par exemple :

Le Motif Séquentiel< **{a} {c} {e}**>apparaît dans les deux premières séquences **Seq1** et **seq2** de notre base de données des séquences (**Tableau 2.1**).
Ce motif est assez intéressant. Il indique que les clients qui ont acheté **{a},** ont souvent acheté **{c}** après, suivi de l'achat de **{e}.**

2-3-2 / Motifs Séquentiels Fréquents *:*

Motif séquentiel est considérée fréquent, si le support de ce motif respecte le support minimum, en d'autres termes, si son support est supérieur ou égal au support minimum. Celui-ci est introduit par l'utilisateur afin de mesurer la pertinence d'une séquence.

2-3-3/ L'extraction des Motifs Séquentiels Fréquents :

Un important problème d'extraction de données consiste à concevoir un algorithme permettant de découvrir des motifs cachés dans des séquences. Il y a eu beaucoup de recherches sur ce sujet dans le domaine d'extraction de données et divers algorithmes ont été proposés.

Soit d'une base de séquences D, étant donné un seuil de support minimal (ou un seuil de fréquence minimale) note **Minsup**.

Le problème d'extraction des motifs séquentiels est l'énumération de toutes les sous séquences S dans la base des séquences D , telles que leur support est supérieur ou égal au support minimum (**Minsup**).

Plusieurs algorithmes ont été proposés pour trouver tous les motifs séquentiels dans une base de séquences telle que : **CM-SPADE**, **PrefixSpan** et **GSP** ….

Ces algorithmes prennent en entrée une base des séquences et un seuil de support minimum **(Minsup)**. Ensuite, ils vont extrait tous les motifs séquentiels ayant un support pas moins de Minsup. Ces motifs sont réputés être les motifs séquentiels fréquents**. [14]**

Exemple :

Considérons la base des séquences de (**Tableau 2.1)** ci-dessus,

Si nous prendre **Minsup = 3,** Nous trouverons les motifs séquentiels fréquents suivants:

< {a}> avec un support de 3 séquences

< {a}, {e}> avec un support de 3 séquences

< {a}, {f}> avec un support de 3 séquences

< {b}, {e}> avec un support de 3 séquences

< {b}, {f}> avec un support de 4 séquences

Les Motifs séquentiels peuvent être très intéressants.

Dans l'exemple ci-dessus, le motif **< {b}, {e}>** indique que l'achat de l'article **"b"** est suivi par l'achat de l'article **"e"** en 3 séquences.

2.3.4/ Limitation des Motifs Séquentielles :

- La recherche des motifs séquentiels est généralement considérée comme plus difficile que l'extraction des itemsets. Ceci est dû à l'espace de recherche exponentiel que le problème peut engendrer et qui est souvent inévitable lorsque les bases de données sont très denses.
- Les motifs séquentiels peuvent être trompeurs.
- Une limitation importante des motifs séquentiels est qu'il n'y a pas d'évaluation de la probabilité qu'un motif soit suivi,

Par exemple :

Si l'on considère le motif **< {b}, {e}>.** Ce motif est censé apparaître dans 3 séquences.

Il semble donc probable que si quelqu'un achète "b", il achètera aussi "e" après.

Mais quelle est la probabilité ?

Nous pouvons observer que l'élément **"b"** apparaît dans quatre séquences. Ainsi, la probabilité que "e" apparaisse après "b" est en réalité 3/4 = 75% (soit P (e | b) = 75%).

Mais les Motifs séquentiels indiquent seulement à quelle fréquence le motif apparaît. Ils ne fournissent aucune indication sur cette probabilité**. [14]**

2.4/Les Règles Séquentielles :

2 .4.1/ Définition d'une règle séquentielle :

Une règle séquentielle est une règle de la forme **X → Y** où **X** et **Y** sont des ensembles d'éléments (itemsets).

Une Règle **X → Y** est interprétée comme si les éléments de **X** se produisent (dans n'importe quel ordre), alors elle sera suivie par les éléments de **Y** (dans n'importe quel ordre). [14]

Exemple :

Considérons la base des séquences de (**Tableau 2.1)** ci-dessus,
On prendre la règle **{a}→{e, f}.** Cela signifie que si un client achète un article **"a",** alors le client achètera plus tard les articles **"e"** et **"f".** Mais l'ordre parmi les éléments de **{e, f}** n'est pas important. Cela signifie qu'un client peut acheter "e" avant "f " ou "f" avant "e".

2-4-2/ Support d'une Règle Séquentielle :

Le support d'une règle **X → Y** est le nombre de séquences contient les articles de **X** suivis par les articles de **Y**, autrement dit :
Le Support d'une règle est le nombre des séquences contenant la règle divisé par le nombre total de séquences. [14]
Par exemple : le support de la règle précèdent **{a}→{e, f}** est 3/4 **(0.75)** car **{a}** apparaît avant les items de **{e, f}** en trois séquences (seq1, seq2 et seq3).

2-4-3/ Confiance d'une règle séquentielle :

La confiance d'une règle **X -> Y** est une mesure dite de précision, c'est la probabilité qu'on achète un certain nombre d'articles Y sachant qu'on a déjà acheté X, autrement dit :
La confiance d'une règle est le nombre de séquences contenant la règle divisé par le nombre des séquences contenant son antécédent.[14]
Par exemple :
La confiance de la règle **{a}→{e, f}** est de **1** (ou 100% si écrit en tant que percent).

2-5 /L'Extraction des Règles Séquentielles :

L'extraction des règles séquentielles est un type de techniques de Fouille de données où l'entrée est un ensemble des séquences et la sortie est un ensemble des règles séquentielles.

Le but de l'extraction des règles séquentielles est de découvrir toutes les règles séquentielles ayant un Support et une Confiance de pas moins de deux seuils donnés par l'utilisateur nommé " **Minsup** " et " **Minconf** ".

L'extraction des règles séquentielle a été proposée comme une alternative à l'extraction des motifs séquentielle pour prendre en compte la probabilité qu'un motif soit suivi et pour faire la prédiction.

2-5 -1/ L'extraction des Règles Séquentielles en une seule séquence :

Nombreuses des travaux ont été effectués pour découvrir des règles séquentielles dans une seule séquence temporelle. Le plus célèbre est celui de Mannila et al**. [15]**

◈ Concepts :

- **Une séquence temporelle :** est une séquence où chaque itemsets est annoté avec un temps
- **Une fenêtre :** est un groupe d'itemsets consécutifs dans une séquence.
- **Une fenêtre coulissante:** est une fenêtre supposée se déplacer du début d'une séquence à sa fin. **[15]**

Dans cette approche, les règles de la forme X⇒Y telles que X et Y sont des ensembles d'items non ordonnés, et qui sont interprétés comme :

« Si le (s) article (s) de X apparaît (s), le (s) article (s) de Y apparaitront dans cette fenêtre avec une confiance »

Pour extraire ces règles, l'utilisateur doit spécifier une taille de fenêtre glissante, un seuil de support minimal (**Minsup**) et un seuil de confiance minimale (**Minconf**).

❖ **Le support** d'une règle est le pourcentage de fenêtres dans lesquelles la règle se produit.

❖ **La confiance** est le nombre des fenêtres contenant la règle divisé par le nombre des fenêtres contenant son antécédent.

Exemple : On considère la séquence temporelle représentée dans la **Figure 2.1**

- Cette séquence contient 13 itemsets,
- Si on prendre une fenêtre coulissante avec une longueur de 3, peut se déplacer dans 15 positions (fenêtres) différentes.
- La règle **{a, b} ⇒ {e}** a un support de **2/15** et une confiance de **2/9**

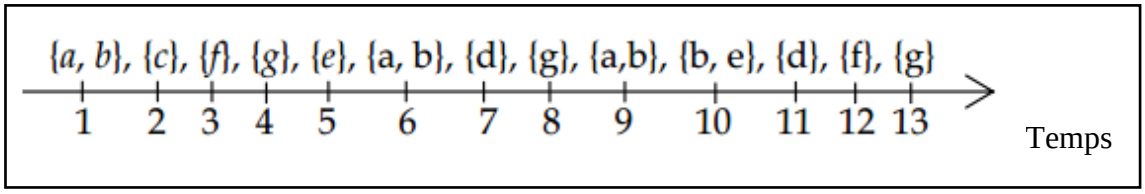

Figure 2.1 : Séquence Temporelle. [15]

↳ Pour découvrir cette forme des règles, deux étapes sont effectuées:

1/ découvrir des itemsets qui apparaissent dans des fenêtres et avec des supports pas moins de Minsup.

2 / générer des règles en utilisant des paires de ces itemsets. **[15]**

Plusieurs algorithmes s'appuyant sur des définitions similaires ont été proposées,

2-5-2/ L'extraction des Règles Séquentielles à travers les séquences :

Pour l'extraction des règles séquentielle dans les bases des séquences, il y plusieurs algorithmes ont été conçus. Qui permettant d'extraire des règles séquentielles à travers des séquences, tel que ils découvrent les règles de la forme **X⇒Y**, où **X** et **Y** sont des ensembles d'items non ordonnés. **[15]**

↳ Pour extraire ces règles, l'utilisateur doit fournir une base des séquences temporelles et spécifier une taille fenêtre coulissante, un seuil de support minimal et un seuil de confiance minimale.

↳ Le support d'une règle **X⇒Y** est défini comme l'approche précède, sauf que X et Y ne sont pas tenus d'apparaître dans la même séquence.

Par exemple :

On dit que la règle **{d} ⇒ {f, g, h}** apparaît une fois dans la base de séquence de (Tableau 2.1)**,** car **{d}** apparaît dans le premier itemset de seq2 et {f, g, h} apparaissent dans le second itemset de seq4.

2-5-3/ L'extraction des Règles Séquentielles communes à plusieurs séquences :

Plusieurs algorithmes ont été proposés pour extraire des règles séquentielles communes à plusieurs séquences d'une base des séquences. La plupart de ces algorithmes consistent à appliquer un algorithme d'extraction des motifs séquentiels, puis d'effectuer une étape de post-traitement pour générer des règles entre des paires de motif séquentiel extraies.

Donc, cette approche, découvre des règles de la forme **X⇒Y** telles que **X** et **Y** sont des motifs séquentiels, et que les règles respectent un support et une confiance minimum. [15]

Exemple : La règle ({**a}, {c} ⇒ {e})** peut être trouvée dans la base des séquences du Tableau 2.1, cela signifie qu'un **a** suivi par **c**, puis suivi de **e**, a un support de **50%** et une confiance de **100 % .**

❖ Une limitation importante pour l'extraction des règles séquentielles communes à plusieurs séquences est que les règles sont très spécifiques et que par conséquent des nombreuses règles similaires peuvent représenter la même situation. Cela peut causer trois problèmes majeurs:

1 / des règles similaires peuvent être notées différemment

2 /des règles peuvent ne pas être trouvées parce qu'elles sont individuellement considérées comme inintéressantes.

3 /des règles trop spécifiques sont moins susceptibles d'être utilisées pour faire prédictions.

Exemple

1. {a} {b} {c} => {d}
2. {a} {c} {b} => {d}
3. {b} {a} {c} => {d}
4. {b} {c} {a} => {d}
5. {c} {a} {b} => {d}

↳ On peut remarquer que chacune des règles séquentielles décrites ci-dessus est considérée aussi différent, malgré ils représentent la même situation.

Pour résoudre ces problèmes, autre approche explorée qu'est l'extraire des règles séquentielles partiellement ordonnées (POSR), une forme plus générale des règles séquentielles, telles que les éléments de l'antécédent et les éléments de conséquent de chaque règle ne sont pas ordonnés.

↳ On peut remarquer que POSR aide à conclure que les règles ci-dessus sont les mêmes, c'est-dire décrire la même information, qui est le client qui a acheté articles a, b, c dans n'importe quel ordre puis acheté l'article d.

Donc, ceux-ci les règles permettent de prendre plus utile et précis les décisions.

2-5-4/L'extraction des Règles Séquentielles partiellement ordonnées (POSR)

Un nouveau type des règles séquentielles différent de ce que nous avons vu récemment, qu'est Règles Séquentielles Partiellement Ordonnées (POSR) **,** parce que les exigences d'un ordre séquentiel à l'intérieur de l'antécédent et à l'intérieur de conséquent de règle sont éliminées ,mais l'exigence d'une relation séquentielle entre l'antécédent et conséquent d'une règle est préservée.

Pour l'extraction des règles POSR, l'approche croissance des règles est utilisée. Cette approche est étendue pour accepter la contrainte de taille de fenêtre.

❖ **Définition le Problème :**

Considérons une base des séquences D contenant un ensemble des séquences S et un ensemble d'items I.

Une règle séquentielle partiellement ordonnée **X⇒Y** est une relation entre deux ensembles d'itemset non ordonnés **X, Y⊆ I** tel que :

$$X \cap Y = \emptyset , \quad X \neq \emptyset , \quad Y \neq \emptyset .$$

L'interprétation d'une règle séquentielle POSR **X⇒Y** est :

« Si des items de **X** se produisent dans une séquence, les items de **Y** se produiront ensuite dans la même séquence **»**

Formellement :

On dit qu'une règle **X⇒Y** se produit dans une séquence $\mathbf{S = I_1, I_2, ……..In}$, s'il existe un entier **k** tel que :

$$1 \leq k < n , \quad X \subseteq \bigcup_{i=1}^{k} I_i \text{ and } Y \subseteq \bigcup_{i=k+1}^{n} I_i .$$

Exemple :

La règle **{a, b, c} ⇒ {e, f, g}** se produit dans la **seq1** (Tableau 2.1).

Mais la règle **{a, b, f} ⇒ {c}** ne le fait pas, parce que l'item **{c}** ne se produit pas après l'item **{f}.** [15]

❖ **La Taille de règle séquentielle :**

Une règle **X⇒Y** est dite de taille **k * m** si | X | = k et | Y | = m.

Notez que la notation (k * m) n'est pas un produit. C'est simplement un symbole d'écriture que les tailles des parties gauche et droite d'une règle sont respectivement **k** et **m**. [15]

Exemple

Les règles {a, b, c} ⇒ {e, f, g} et {a} ⇒ {e, f} sont de taille 3 * 3 et 1 * 2 respectivement.

En autre, une règle de taille **f * g** est dit être plus grande qu'une autre règle de taille **h * i**

Si **f > h et g ≥ i,** ou bien **Si f ≥ h et g> i.**

❖ **Observation :**

1/ Pour une base des séquences donnée et une règle **X⇒Y :**

La notation **sids (X⇒Y)** représente l'ensemble sid (l'ensemble des identifiants de séquence) des séquences où se produit la règle.

2/Pour une Base des séquences donnée et un ensemble d'itemset **X** :

La notation **sids (X)** désigne l'ensemble sid correspondant aux séquences où tous les items de **X** apparaissent.

Exemple :

Pour la base des séquences illustrée dans le Tableau 2.1, on trouve :

sids ({a} ⇒ {b}) = {seq2, seq3}.

sids ({a, b, c}) = {seq1, seq2}.

↳ Nous définissons deux mesures d'intérêt pour des règles séquentielles, qui sont basées sur les mesures utilisées dans l'extraction des règles séquentielles :

Le support d'une règle **X⇒Y** est défini comme : **Sup (X⇒Y) = | sids (X⇒Y) | / | S |.**

La confiance d'une règle **X⇒Y** est défini comme: **Conf (X⇒Y) = | sids (X⇒Y) | / | sids (X) |.**

↳ Considérer un seuil de support minimum note **Minsup** et un seuil de confiance minimum note **Minconf** défini par l'utilisateur dans l'intervalle [0, 1].

❖ Une règle est dite **fréquente** si son support supérieur ou égale que Minsup.

❖ Une règle est considérée comme **règle valide** si c'est une règle fréquente et sa confiance supérieure ou égale que Minconf**. [15]**

Alors Le problème de l'extraction des règles séquentielles communes à plusieurs séquences (POSR) est extraire toute les règles valides dans une Base des séquence D, en tenant compte des seuils Minsup et Minconf définis par l'utilisateur.

Exemple :

Le Tableau 2 .2 montre quelques règles valides trouvées dans la Base des séquences du Tableau 1.1, pour Minsup = 0.5 et Minconf = 0.5.

↳ La règle {a, b, c} ⇒ {e} * À un support de | sids (X⇒Y) | / | S | = 2/4 = 0.5

* Une confiance de| sids (X⇒Y) | / | sids (X) | = 2/2 = 1.

↳ Parce que ces valeurs sont respectivement pas moins de Minsup et Minconf , la règle est valide

ID	Rule	Support	Confiance
r1	**{a, b, c}⇒{e}**	**0.50**	**1.00**
r2	**{a}⇒{c, e, f}**	**0.50**	**0.66**
r3	**{a, b}⇒{e, f}**	**0.75**	**1.00**
r4	**{b}⇒{e, f}**	**0.75**	**0.75**
r5	**{a}⇒{e, f}**	**0.75**	**1.00**
r6	**{c}⇒{f}**	**0.50**	**1.00**
r7	**{a}⇒{b}**	**0.50**	**0.66**

Tableau 2 .2 : *Liste des Règles Séquentielles(POSR) générées* [15]

2-6 / Domaine d'Application des Règles Séquentielles :

L'extraction des règles séquentielle est utilisé pour extraire des données importantes dans divers application comme :

- Analyse de marche
- Gestion de commerce électronique
- Les changements et les relations météorologiques
- L'apprentissage en ligne
- E-Learning
- Analyse de séquence d'alarme
- Simulation de fabrication
- Recommandation
- Analyse du comportement du client
- Bioinformatique (analyse des séquences comme l'ADN ou protéines pour trouve une corrélation entre l'expression des gènes). **[15]**

2.7/ Conclusion

Dans ce chapitre, Nous avons présenté les concepts fondamentaux relatives aux Motifs séquentielles, Règles séquentielles et les déférents approches pour l'extraction de ces règles.

Dans Le prochain chapitre, nous approfondirons notre travail à travers une étude détaillée des algorithmes d'extraction des règles séquentielles.

CHAPITER III: LES ALGORITHMES

RuleGrowth et TRuleGrowth

3.1/Introduction:

L'extraction des règles séquentielles à partir de Bases des séquences volumineuses est un sujet important dans les champs d'exploration des données avec des applications étendues. La plupart des études axées sur la recherche des règles séquentielles apparaissant dans une seule séquence , et la tâche d'extraction des règles traitant de multiples séquences étaient beaucoup moins explorées.

L'un des approches générales de l'extraction des règles séquentielle est l'extraction des règles séquentielles partiellement ordonnées (POSR) dans lesquelles, les éléments répertoriés dans les côtés gauche et droit de la règle n'ont pas besoin d'être ordonnées.
Ces règles sont identifiées à l'aide de plusieurs algorithmes, parmi lesquelles : RuleGrowth et TRuleGrowth.

Dans ce chapitre, nous allons nous concentrer sur ces deux algorithmes.

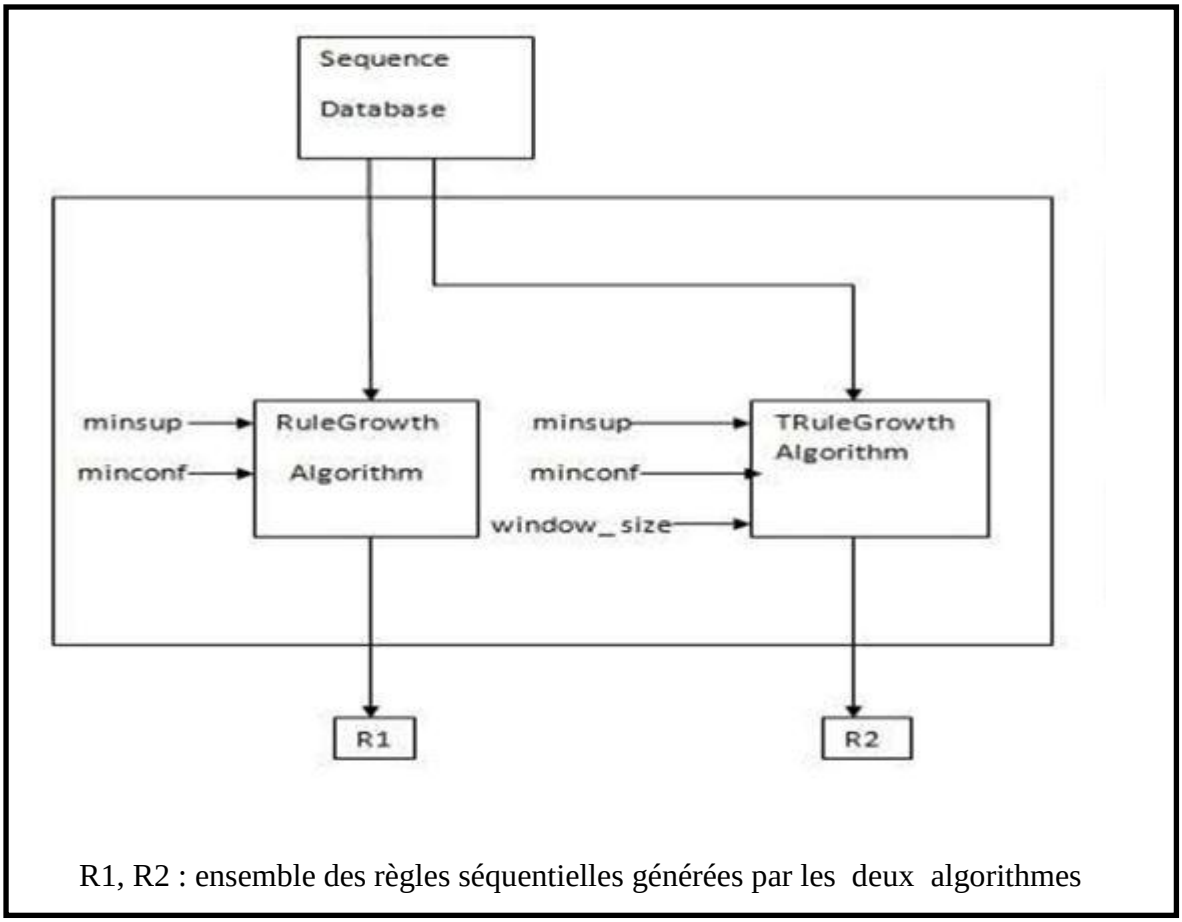

Figure 3.1 : Algorithmes de génération des règles séquentielles POSR. **[16]**

3.2/ L'Algorithme RuleGrowth :

P. Fournier-Viger, R. Nkambou et V. S. Tseng ont présenté l'approche RuleGrowth comme nouvelle algorithme pour l'extraction POSR commun à plusieurs séquences. Cette approche n'utilise pas les techniques de génération candidat et ensuite les tester.
Au contraire, il utilise approche modèle-croissance pour découvrir des règles plus efficaces et valides. [16]

RuleGrowth est le premier algorithme qui extrait les règles POSR, qui prennent des base de séquences en entrée (SD), les valeurs de support minimum (Minsup) et la confiance minimale (Minconf), respectivement.

L'algorithme RuleGrowth utilise l'approche pattern-growth pour découvrir plus règles efficaces et valides. Il trouve des règles de manière incrémentale. Il trouvera d'abord des règle séquentielle de taille 1 * 1, et puis se développe en analysant la base de données des séquences pour développer leur partie gauche ou droite de la règle.

* Deux procédures d'expansion des règles utilisées dans RuleGrowth sont nommées expansion à gauche (EXPANDLEFT) et expansion à droite (EXPANDRIGHT).
* RuleGrowth est un algorithme trouve de plus grandes règles en ajoutant les items dans les règles en balayant la base de données des séquences,

Une autre caractéristique du RuleGrowth est qu'il conserve simultanément la première et les dernières occurrences de chaque élément, ainsi que des antécédents et des conséquences pour éviter le balayage complet des séquences. Il repose sur l'utilisation de jeux d'identifiants de séquence pour le calcul de support et de confiance des règles générées obtenues en élargissant le côté à gauche et à droite des règles. [15]

3.2.1/ Définitions préliminaires et propriétés :

Avant de présenter l'algorithme, nous introduisons des définitions et des propriétés importantes :

❖ **Expansion à gauche (EXPANDLEFT) :** est une procédure consistant à ajouter un item i du côté gauche d'une règle X⇒Y pour obtenir une règle plus grande **X∪ {i} ⇒Y. [15]**

❖ **Expansion à droite (EXPANDRIGHT):** est définie comme la procédure d'ajout d'un item **i** du côté droite d'une règle **X⇒Y** pour obtenir une règle plus grande **X⇒Y∪ {i}. [15]**

Les Expansions à gauche et droite ont les quatre propriétés importantes suivantes:

❖ **Propriété 1 (Expansion à gauche, effet sur le Support) :**

Si un élément **i** est ajouté à gauche d'une règle :**R: X⇒Y**, le support de la règle résultante **R': X∪ {i} ⇒Y** peut seulement être inférieur ou égal au support de **(R). [15]**

Preuve:

Le support de **R** et **R'** sont respectivement **| sids (X⇒Y) | / | S | et | sids (X∪ {i} ⇒Y) | /| S|**

Depuis **| sids (X⇒Y) | ≥ | sids (X∪ {i} ⇒Y) |, Sup (R) ≥ sup (R').**

❖ **Propriété 2 (Expansion à droite, effet sur le support) :**

Si un item **i** est ajouté du côté droit d'une règle **R: X ⇒Y**, le support de la règle résultante **R ' : X⇒Y∪ {i}** ne peut être que inférieur ou égal au support de **(R). [15]**

Preuve:

Le support de **R** et **R'** sont respectivement **|sids (X⇒Y) | / | S | et | sids (X⇒Y∪ {i}) |/ | S |.**

Depuis **| sids (X⇒Y) | ≥ | sids (X⇒Y∪ {i}) |, Sup (R) ≥ sup (R').**

Les deux propriétés précédentes impliquent que le support est monotonie par rapport aux expansions gauche et droite, en d'autres termes, l'exécution des expansions de gauche/droite d'une règle ne peut avoir pour résultat que des règles ayant un support inférieur ou égal au support de la règle d'origine.

❖ **Propriété 3 (Expansion à gauche, effet sur la confiance) :**

Si un item **i** est ajouté à gauche d'une règle **R: X⇒Y**, la confiance de la règle résultante **R': X∪ {i} ⇒Y** peut être inférieur, supérieur ou égale la confiance de règle **R. [15]**

Preuve:

La confiance de **R** et **R'** sont respectivement

| sids (X⇒Y) | / | sids (X) | et | sids (X∪ {i} ⇒Y) | / | sids (X∪ {i}) |

Parce que **| sids (X⇒Y) | ≥ | sids (X∪ {i} ⇒Y) | et | sids (X) | ≥ | sids (X∪ {i}) |,** La confiance de **(R)** peut être inférieur, supérieur ou égal la confiance de règle **(R').**

❖ **Propriété 4 (Expansion à droite, effet sur la confiance) :**

Si l'item **i** est ajouté du côté droit d'une règle **R: X⇒Y**, la confiance de la règle résultante **R' : X⇒Y∪ {i}** est inférieure ou égale à la confiance de de règle **R. [15]**

Preuve :

La confiance de R et R' sont respectivement

| sids (X⇒Y) | / | sids (X) | et | sids (X⇒Y∪ {i}) | / | sids (X) | depuis **| sids (X⇒Y) | ≥ | sids (X⇒Y∪ {i}) |** , Alors **conf (R) ≥ conf (R') .**

Par contre le support, la confiance est pas monotone par rapport aux procédures expansions gauche et droite de règle.

❖ L'algorithme RuleGrowth repose sur l'utilisation de l'ensemble des séquences pour calculer le support et la confiance des règles obtenues par expansions à gauche ou droite de règle.

Les ensembles des Sid ont deux propriétés importantes pour les règles séquentielles.

❖ **Propriété 5 (ensemble sid d'une règle et de ses éléments) :**

Pour toute règle séquentielle **X⇒Y, sids (X⇒Y) ⊆ (sids (X) ∩ sids (Y)). [15]**

❖ **Propriété 6 (ensemble sid d'une règle obtenue par l'expansion à gauche ou droite) :**

Pour toute règle séquentielle **R'** obtenue par une expansion à gauche ou droite d'une règle **R**, la relation **sids (R') ⊆ sids (R)** est réalisé. **[15]**

Observation :

RuleGrowth s'appuie également sur deux définitions supplémentaires :

❖ La première apparition d'un itemset **X** dans une séquence **S = I_1, I_2……. I_n** est le itemset **I_K ∈ S** tels que $X \subseteq \bigcup_{i=1}^{k} I_i$ et il existe pas **g < k** tel que $X \subseteq \bigcup_{i=1}^{g} I_i$

❖ La dernière occurrence d'un itemset **X** dans une séquence **S = I_1, I_2……. I_n** est l'ensemble itemset **I_K ∈ S** tel que $X \subseteq \bigcup_{i=k}^{n} I_i$ et il existe pas **g < k** tel que $X \subseteq \bigcup_{i=g}^{n} I_i$

Par exemple : la première occurrence de **{a, b}** dans la séquence **({a, d}, {b}, {a}, {b}, {e})** est le deuxième ensemble d'itemset, tandis que la dernière occurrence de **{a, b}** est le troisième d'itemset. **[15]**

3.2.2/ La Procédure principal de L'algorithme RuleGrowth:

L'algorithme RuleGrowth prennent en entrée une Base des séquences SD et les paramètres support minimal et confiance minimale (SD , Minsup , Minconf) .

La procédure principale de RuleGrowth est montrée dans la Figure suivant :

ALGORITHME RULEGROWTH (SD, Minsup, Minconf)

1/ Scan the database SD once. For each item c found, record the sids of the sequences that contains c in a variable sids(c).

2/ FOR each pair of items i, j such that | sids(i)|/ | S| ≥ Minsup ,and | sids(j)| / | S| ≥Minsup{

3/ sids (i⇒j) := Ø. sids(j⇒i) := Ø.

4/ FOR each sid s ∈ (sids(i) ∩ sids(j)) {

5/ IF i occurs before j in s, sids(i⇒j) := sids(i⇒j) ∪ {s}.

6/ IF j occurs before i in s, sids(j⇒i) := sids(j⇒i) ∪ {s}. }

7/ IF (|sids(i⇒j)| / | S|) ≥ Minsup & size of left rule ≤ maxleft & size of right rule ≤ maxright THEN {

8/ EXPANDLEFT ({i} ⇒ {j}, sids(i), sids(i⇒j)).

9/ EXPANDRIGHT ({i} ⇒ {j}, sids(i), sids(j), sids(i⇒j)).

10/ IF (| sids(i⇒j)| / |sids(i)|) ≥ Minconf THEN OUTPUT rule {i}⇒{j} with its confidence and support. }

11/ IF (|sids(j⇒i)| / | S|) ≥ Minsup & size of left rule ≤ maxleft & size of right rule ≤ maxright THEN {

12/ EXPANDLEFT({j}⇒{i}, sids(j), sids(j⇒i)).

13/ EXPANDRIGHT({j}⇒{i}, sids(j), sids(i), sids(j⇒i)).

14/ IF (| sids(j⇒i)| / |sids(j)|) ≥ Minconf THEN OUTPUT rule {j}⇒{i} with its confidence and support. }}

Figure 3.2 : La Procédure Principale de L'algorithme RuleGrowth. [15]

3.2.3/ Les étapes de l'algorithme RuleGrowth :

On peut expliquer les différentes étapes de L'algorithme RuleGrowth comme montré ci-dessous :

Etape1 : Scannez la base de séquences **SD** une fois, pour chaque item **c** trouvé, notez-les identificateur (sids) des séquences qui contient **c** dans une variable sids (c).

Etape2: Pour chaque paire d'items i, j tels que le support de i $\geq$ support minimal et le support de j $\geq$ support minimal {

Etape3 : Initialise les identificateurs des séquences sids (i⇒j): = Ø. sids (j⇒i): = Ø.

Etape4 : Pour chaque séquences s qui contient i et j (sid s $\in$ (sids (i) $\cap$ sids (j))) {

Etape 5: SI **i** apparaît avant **j** dans la séquence S, ajouter cette séquence au liste des séquences appartient la règle (i⇒j)

Etape6: Si **j** apparaît avant **i** dans S, ajouter cette séquence au liste des séquences appartient la règle (j⇒i).

Etape7: Si le support de règle (i⇒j) $\geq$ support minimal et la taille de l'antécédent et la conséquent moins de la taille maximal défini par l'utilisateur.

Etape8: La procédure Expansion à gauche

Etape9 : La procédure Expansion à droite

Etape10: Si la confiance de règle (i⇒j) $\geq$ confiance minimal,

Extraire la règle (i⇒j), avec son support et confiance.

Etape11 : Si le support de règle (j⇒i) $\geq$ support minimal et la taille de l'antécédent et la conséquent moins de la taille maximal défini par l'utilisateur.

Etape12 : La procédure Expansion à gauche

Etape13 : La procédure Expansion à droite

Etape14 : Si la confiance de règle (j⇒i) $\geq$ confiance minimal

Extraire la règle (j⇒i) avec son support et confiance.

3.2.4/ La procédure Expansion à gauche de l'algorithme RuleGrowth:

❖ **La procédure Expansion à gauche**

EXPANDLEFT (I⇒J, sids(I), sids(I⇒J))

1/ FOR each sid∈ sids (I⇒J), scan the sequence sid.

For each item c appearing in sequence sid that is lexically larger than all items in I and appears before J, record sid in a variable sids(I∪{c}⇒J).

2/ FOR each item c where | sids(I∪{c}⇒J)/ **| S|)**≥ minsup {

3/ sids(I∪{c}) := Ø.

4/ FOR each sid∈sids(I) such that sid∈sids(c){

5/ sids(I∪{c}):= sids(I∪{c}) ∪{sid}. }

6/ EXPANDLEFT(I∪{c}⇒J, sids(I∪{c}), sids(I∪{c}⇒J))

7/ IF | sids(I∪{c}⇒J)| / | sids(I∪{c})| ≥ minconf

8/ THEN OUTPUT rule I∪{c}⇒J. }

Figure 3.3 : La Procédure Expansion à gauche de L'algorithme RuleGrowth. **[15]**

↳ On peut expliquer les différentes étapes de procédure Expansion à gauche de L'algorithme RuleGrowth ,comme montré ci-dessous :

Étape 1: pour chaque séquence sids où la règle (I⇒J) est présente, scannez la séquence sids. pour chaque item c apparaissant dans sids qui est lexicalement plus grand que tous les items de i et apparait avant j, enregistrer sids dans une variable sids(I∪{*c*}⇒J).

Étape 2: pour chaque item c où le support de la règle (I∪{c}⇒J) ≥ Minsup {

Étape 3: Initialisez la liste des séquences où la règle (I∪{c}⇒J) est présente, (sids (I∪{c}) :=Ø)

Étape 4: pour chaque séquence sids ∈ sids (I) tel que sids ∈ sids (c) {

Etape 5: Calculer les sids (I∪{c}):= sids(I∪{c}) ∪{sids}. }

Étape 6: si la taille de partie gauche de la règle ≤ max antécédent alors

Étape 7: la procédure Expansion à gauche

Etape 8: si la confiance de règle (I∪{c}⇒J) ≥ confiance minimal

Extrait la règle (I∪{c}⇒J)

3.2.5/ La procédure Expansion à droite de l'algorithme RuleGrowth :

❖ **La procédure Expansion à droite:**

EXPANDRIGHT (I⇒J, sids(I), sids(I⇒J))

1/ FOR each sid∈sids(I⇒J), scan the sequence sid. For each item c appearing in sequence sid that is lexically larger than all items in J and appear after I, record sid in a variable sids(I⇒J∪{c}).

2/ FOR each item c such that |sids(I⇒J∪{c})| / | **S**| ≥ minsup {

3/ EXPANDLEFT(I⇒J∪{c}, sids(I), sids(I⇒J∪{c})).

4/ EXPANDRIGHT (I⇒J∪{c}, sids(I), sids(I⇒J∪{c})).

5/ IF | sids(I⇒J∪{c})| / | sids(I)| ≥minconf

6/ THEN OUTPUT rule I⇒J∪{c}. }

Figure 3.4: La Procédure Expansion à droite de L'algorithme RuleGrowth. **[15]**

↳ On peut expliquer les différentes étapes de procédure Expansion à droite de L'algorithme RuleGrowth, comme montré ci-dessous :

Étape 1: pour chaque séquence sids où la règle (I⇒J) est présente, scannez la séquence sids. pour chaque item **c** apparaissant dans la séquence sids qui est lexicalement plus grand que tous les items de **J** et apparaît après **I**, enregistrer sids dans une variable sids(I⇒J∪{c}).

Étape 2: Pour chaque item **c** tel que le support de la règle (I⇒J∪{c})| ≥ Minsup {

Étape 3: si la taille de partie gauche de la règle ≤ Max antécédent alors :

La procédure Expansion à gauche.

Étape 4: si la taille de partie droite de la règle ≤ Max conséquence alors:

La procédure Expansion à droite

Étape 5: si la confiance de la règle (I⇒J∪{c}) ≥ Minconf alors

Etape 6 : Extrait la règle I⇒J∪{c}}.

3.3/ L'extraction des Règles Séquentielles avec la contrainte de Taille de Fenêtre :

L'algorithme RuleGrowth incrémente les règles obtenues par un item à la fois, pour cette raison, les contraintes peuvent être facilement ajoutées à l'algorithme selon les besoins d'applications spécifiques.

Par exemple, il serait facile d'ajouter des contraintes sur le nombre d'items que les règles peuvent contient.

Nous définissons une extension particulière qui consiste à découvrir des règles séquentielles avec une contrainte de Taille de Fenêtre**,** c'est-à-dire dans un nombre maximum d'itemsets consécutifs dans chaque séquence.

3.3.1 / Définition du problème:

Le problème de l'extraction des règles séquentielles communes à plusieurs séquences avec une fenêtre glissante étant le même que le problème de l'extraction des règles séquentielles commun à plusieurs séquences sauf que la définition de l'apparition d'une règle dans une séquence est modifiée de manière que les règles doivent respecter un nombre donnée d'itemsets consécutifs apparaissent dans la fenêtre coulissante.

On dit que la règle **X⇒Y** se produit dans une séquence **$S = I_1, I_2 \ldots\ldots\ldots I_n$** s'il existe des entiers **j, k, m** tels que :

$$1 \leq j \leq k < m \leq n,\ X \subseteq \bigcup_{i=j}^{k} I_i \qquad Y \subseteq \bigcup_{i=k+1}^{m} I_i$$

Et que **m - j + 1 ≤ Taille de Fenêtre**, où Taille de fenêtre est définie par l'utilisateur.

Nous indique que cette extension très importante , car l'application d'une fenêtre coulissante s'est révélée très utile pour la découverte de modèles temporels pour des nombreux applications réelles telles que l'analyse de réseaux de capteurs, données boursières, l'analyser des séquences des clics sur le web, en plus les utilisateurs souhaitent souvent découvrir des règles de la forme X⇒Y où X et Y doivent être proches l'un de l'autre par rapport au temps. [15]

3.3.2 / Exemple illustratif :

Le Tableau suivant montre les règles séquentielles trouvées dans une base des séquences présentée dans le **Tableau 2 .1**, pour Minsup = 0.5, Minconf= 0.5 et Taille de Fenêtre = 3.

Considérons la règle **{a} ⇒ {f}.** Parce qu'il se produit dans les séquences **seq1** et **seq3**, Il a un **:** **le support** de {a} ⇒ {f} est | sids (a⇒f) | / | S | = 2 / 4.

La confiance de {a} ⇒ {f} est | sids (a⇒f) | / | sids (a) | = 2/3.

Notez que **{a} ⇒ {f}** ne se produit pas dans **seq2**, car la règle ne respecte pas la contrainte de la Taille de Fenêtre.

Si la Taille de Fenêtre a été défini sur une valeur supérieure à 3, la règle {a} ⇒ {f} se produirait dans la **seq2** et donc le support et la confiance de la règle ({a} ⇒ {f}) seraient 3/4 et 3/3, respectivement.

ID	Règle	Support	Confiance
R1	**{*a*} ⇒{*b*}**	**0.5**	**0.66**
R2	**{*a*} ⇒{*c*}**	**0.5**	**0.66**
R3	**{*a*} ⇒{*f*}**	**0.5**	**0.66**
R4	**{*b*} ⇒{*e*}**	**0.5**	**0.50**
R5	**{*b*} ⇒{*f*}**	**1.0**	**1.00**
R6	**{*c*} ⇒{*f*}**	**0.5**	**1.00**
R7	**{*f*} ⇒{*e*}**	**0.5**	**0.50**

Tableau 3.1: Règles Séquentielles générées avec la Taille de Fenêtre= 3. [15]

↳ Cette approche a été proposée pour répondre à des besoins spécifiques, en raison de ses nombreux avantages importants:

- Il peut réduire le temps d'exécution de nombreuses applications en réduisant l'espace de recherche.
- Il peut produire un ensemble de règles beaucoup plus petit, ce qui réduit l'espace disque requis pour stocker les règles trouvées et facilite l'analyse des résultats par l'utilisateur.
- La définition de la contrainte de taille de fenêtre peut augmenter la précision prédictive si les règles obtenues sont utilisées pour prédire.

Pour cette raison, de nombreux algorithmes permettent d'extraire des règles séquentielles avec une contrainte de Taille de Fenêtre, parmi elles **TRuleGrowth**, l'extension de RuleGrowth, qu'est génèrent toutes les règles séquentielles se produit dans la Fenêtre glissante.

3.4 /Algorithme TRuleGrowth:

3.4.1/Introduction:

TRuleGrowth est un deuxième algorithme du système **POSR** (partially-ordered sequential rules) qui est proposé en 2012 par P. Fournier-Viger, Cheng-Wei Wu, V. S. Tseng, Longbing Cao et Roger Nkambou. [16].

TRuleGrowth est un algorithme permettant de découvrir les règles séquentielles, qui apparaissent dans les bases des séquences. C'est une extension de l'algorithme RuleGrowth, mais il diffère de la contrainte **" Taille de la Fenêtre "** acceptée.

L'entrée de TRuleGrowth est une base des séquences, deux seuils définis par l'utilisateur nommées **Minsup** (une valeur dans [0, 1] représentant un pourcentage) et **Minconf** (une valeur dans [0, 1] représentant un pourcentage) et le paramètre Taille de la Fenêtre (un entier > = 0) , tel que ce dernier représente une contrainte de Taille de la Fenêtre défini par l'utilisateur . Certains utilisateurs intéressés par des contraintes selon les besoins des applications spécifiques, telles que la longueur, l'attribut des éléments, la durée, l'intervalle, etc……

Ainsi L'algorithme TRuleGrowth permet de spécifier des paramètres optionnels:

❖ **" Longueur antécédente minimale "** permet de spécifier le nombre maximum d'éléments pouvant apparaître dans le côté gauche (antécédent) d'une règle.

❖ **" Longueur maximale conséquente "** permet de spécifier le nombre maximum d'éléments qui peuvent apparaître dans le côté droit (conséquent) d'une règle. **[17]**

↳ D'abord dans le cadre du prétraitement des données, il convertit les données de fichier texte en séquences appropriées et stocke tous les occurrences de chaque item pour chaque séquence en utilisant la table de hachage.

Par exemple :

Les occurrences de **a** dans la séquence **seq2 = ({a, d}, {c}, {b}, {a, b, e, f})** sont **1** et **4** (a apparaît dans le premier itemset et le quatrième itemset), et les occurrences de **b** dans **seq2** sont **3** et **4** (**b** apparaît dans le troisième itemset et le quatrième itemset).

↳ Ainsi , pour vérifier si la règle (**i⇒j**) respecte la contrainte de Taille de la Fenêtre, il faut comparer chaque occurrence de **item i** avec chaque occurrence de **item j** pour chaque séquence en utilisant les tables de hachage. Autrement dit :

S'il y a existé une occurrence **x** de **item i** et une occurrence **y** de **item j** tel que **y – x > 0** et **y - x + 1 ≤** Taille de Fenêtre, alors il est conclu que **l'item i** se produit avant **l'item j** dans la séquence et la règle (**i⇒j**) respectent la Taille de la Fenêtre glissante.

Par exemple :

Considérons les **items a** et **b** dans la séquence **seq2 = ({a, d}, {c}, {b}, {a, b, e, f})** et Taille de la Fenêtre **= 3**.

Par comparant les occurrences de **a** et **b**, TRuleGrowth trouve cet **item a** apparaît avant **l'item b**, en respectant la Taille de Fenêtre de glissement car pour l'occurrence **1** de l'**item a** et l'occurrence **3** de **l'item b**, $3 - 1 > 0$ et $3 - 1 + 1 = 3 \leq 3$ (Taille de Fenêtre).

↳ Pour générer des règles séquentielles large et valables à partir de règles plus petites, comme RuleGrowth , l'algorithme TRuleGrowth exécute les deux procédures EXPANDLEFT et EXPANDRIGHT et génèrent toutes les règles séquentielles ayant un support et une confiance respectivement supérieurs à **Minsup** et **Minconf ,** et respectent un nombre donnée d'itemsets consécutifs apparaissent dans la Fenêtre coulissante**. [15]**

3.4.2/ La procédure principal de l'algorithme TRuleGrowth:

ALGORITHME TRULEGROWTH

TRULEGROWTH (SD, Minsup, Minconf, window-size)

1/ Scan the database SD once. For each item **c** found, record the sids of the sequences that contains **c** in a variable sids(**c**).

2/ FOR each pair of items **i , j** such that | sids(i)| / | S| ≥ Minsup. and | sids(j)| / | S| ≥ Minsup {

3/ sids (i⇒j) := Ø , sids(j⇒i) := Ø.

4/ FOR each sid s ∈ (sids(i) ∩ sids(j)) {

5/ IF **i** occurs before **j** in s, sids (i⇒j) := sids(i⇒j) ∪ {s}.

6/ IF **j** occurs before **i** in s, sids (j⇒i) := sids(j⇒i) ∪ {s}.

7/ IF (|sids(i⇒j)| / | S|) ≥ Minsup & size of left rule ≤ maxleft & size of right rule ≤ maxright and rule lie within window-size THEN {

8/ EXPANDLEFT ({i} ⇒{j}, sids(i), sids(i⇒j)).

9/ EXPANDRIGHT ({i}⇒{j}, sids(i), sids(j), sids(i⇒j)).

10/ IF (| sids(i⇒j)| / |sids(i)|) ≥ Minconf THEN OUTPUT

rule {i}⇒{j} with its confidence and support. }

11/ IF (|sids(j⇒i)| / | S|) ≥ Minsup & size of left rule ≤ maxleft & size of right rule ≤ maxright and rule lie within window-size THEN {

12/ EXPANDLEFT({j}⇒{i}, sids(j), sids(j⇒i)).

13/ EXPANDRIGHT({j}⇒{i}, sids(j), sids(i), sids(j⇒i)).

14/ IF (| sids(j⇒i)| / |sids(j)|) ≥ minconf THEN OUTPUT

rule {j}⇒{i} with its confidence and support. }}

Figure 3.5 : La Procédure Principal de l'algorithme TRuleGrowth. **[15]**

3.4.3/ Les étapes de l'algorithme TRuleGrowth:

↳ On peut expliquer les différentes étapes de L'algorithme TRuleGrowth, comme montré ci-dessous :

Etape1: Scannez la base de données de séquences SD une fois, pour chaque item **c** trouvé, notez-les identificateur (sids) des séquences qui contient **c** dans une variable sids(c).

Etape2: Pour chaque paire d'items **i, j** tels que le support de i ≥ support minimal et le support de j ≥ support minimal {

Etape3 : Initialise les identificateurs des séquences sids (i⇒j): = Ø. sids (j⇒i): = Ø.

Etape4 : Pour chaque séquences s qui contient **i** et **j** (sid s ∈ (sids (i) ∩ sids (j))) {

Etape5: SI **i** apparaît avant **j** dans la séquence s, ajouter cette séquence au liste des séquences appartient la règle (i⇒j)

Etape6: Si **j** apparaît avant **i** dans S, ajouter cette séquence au liste des séquences appartient la règle (j⇒i).

Etape7: Vérifier si (le support de règle (i⇒j) ≥ support minimal) et la (taille de antécédent ≥ le max antécédent) et (la taille de conséquence ≥ le max conséquence) et (la règle appartient la taille de fenêtre défini)

Etape8: le Procédure Expansion à gauche de règle

Etape9 : le Procédure Expansion à droite de règle

Etape10:si la confiance de règle (i⇒j) ≥ confiance minimal extraire la règle (i⇒j) avec son support et confiance

Etape11 : Vérifier si (le support de règle (j⇒i) ≥ support minimal) et la (taille de antécédent ≥ le max antécédent) et (la taille de conséquence ≥ le max conséquence) et (La règle (j⇒i) appartient de fenêtre de taille demande (window_size))

Etape12 : le Procédure Expansion à gauche

Etape13 : le Procédure Expansion à droite

Etape14 : si la confiance de règle (j⇒i) ≥ confiance minimal Extrait la règle (j⇒i) avec son support et confiance.

3.4.4/ La procédure Expansion à gauche de l'algorithme TRuleGrowth:

La procédure EXPANDLEFT de TRuleGrowth

EXPANDLEFT (I⇒J, *sids*(I), *sids*(I⇒J))

1/ FOR each sid ∈ sids(I⇒J) { Hash I := Ø. Hash J := Ø.

2/ FOR each itemset X in sequence sid, from the last one to the first one. {

3/ REMOVE all items from Hash I and Hash J seen more

than window_size – 1 itemsets before.

4/ IF |Hash J| was equal to |J| and became smaller after removing items THEN | Hash I|:= Ø.

5/ IF |Hash J| = |J| THEN add each item c ∈ I∩X to Hash I

with the position of X in sequence sid.

6/ IF | Hash J| < |J| THEN add each item d ∈ J∩X to Hash J with the position of X in

sequence sid.

7/ IF | Hash I| = |I| and | Hash J| = |J| THEN add sid to variable sids(I∪{c} ⇒J) for each

item c ∉ I∪J occurring before the first item of J in the window }}

8/ FOR each item c where |sids(I∪{c} ⇒J)| / |S| ≥ Minsup { sids(I∪{c}) := Ø.

9/ FOR each sid ∈sids(I) such that sid ∈sids(c) {

10/ IF c and I occur within a window of size window_size

THEN sids(I∪{c}):= sids(I∪{c}) ∪{sid}. }

11/ EXPANDLEFT (I∪{c} ⇒J, sids(I∪{c}), sids(I∪{c} ⇒J)).

12/ IF |sids(I∪{c} ⇒J)| / | sids(I∪{c})| ≥ Minconf THEN

OUTPUT rule I∪{c} ⇒J. }

Figure 3.6: La Procédure Expansion à gauche de l'algorithme TRuleGrowth. . **[15]**

❖ **Les étapes de procédure Expansion à gauche de l'algorithme TRuleGrowth :**

↳ On peut expliquer les différentes étapes de La Procédure Expansion à gauche de L'algorithme TRuleGrowth, comme montré ci-dessous :

EXPANDLEFT (I⇒J, *sids*(I), *sids*(I⇒J)) :

Etape1 : pour chaque séquence sid où la règle (I⇒J) est présente, initialise le table de hagage Hash I := Ø, Hash J := Ø.

Etape2 : pour chaque itemset X appartient in séquence Sid, du dernier au Premier {

Etape3 : tous les éléments trouvés plus de (window_size – 1) itemsets avant sont supprimés de tables "Hash I" et "Hash J" parce que nous considérons qu'ils tombent en dehors de la fenêtre définie par l'utilisateur .

Etape4 : Si | Hash J | était égal à | J | et est devenu plus petit après avoir enlevé des items Alors | Hash I | : = Ø

Etape5 : Si la taille de la table "Hash J" est égale à la taille de "J" alors
Ajouter chaque élément c (c ∈I∩X dans la table "Hash I". avec la position de X dans la séquence sid.

Etape6 : Si la taille de la table "Hash J" est inférieur à la taille de "J" alors
Ajouter chaque élément d (d ∈ I∩X) dans la table "Hash J" , avec la position de X dans la séquence sid.

Etape7 : Si la taille de "HashI" égale à la taille de "I" et la taille de "HashJ" égal à la taille de "J" Alors : ajoute sid à variable sids (I∪ {c} ⇒J) pour chaque item c (c ∉I∪J) se produisant avant le premier élément de J dans la fenêtre.

Etape8: Pour chaque item c
Vérifier si le support de la règle (I∪{*c*} ⇒J) est le plus grand que le support minimal
Alors Initialise s*ids* (I∪{*c*}) := Ø.

Etape9 : Pour chaque sid ∈sids (I) tel que sid ∈sids (c) {

Etape10 : SI c et I apparaissent dans une fenêtre de taille window_size alors :
Ajouter sid de sids(I∪{c}) }. }

Etape11 : Le Procédure Expansion à gauche

Etape12 : Si la règle I∪ {c} => J a une plus grande confiance que confiance minimal, alors la règle est considérée comme une règle valide et extraire.

3.4.5/ La procédure Expansion à droite de L'algorithme TRuleGrowth:

La* Procédure *EXPANDRIGHT de L'algorithme TRuleGrowth
EXPANDRIGHT (I⇒J, sids (I), sids(I⇒J))
1/ FOR each sid ∈sids(I⇒J) { Hash I := Ø. Hash J := Ø.
2/ FOR each itemset X in sequence sid, from the first one to the last one{
3/ REMOVE all items from Hash I and Hash J seen more
than (window_size – 1) itemsets before.
4 /IF | Hash I | was equal to |I| and became smaller than it
after removing items THEN | Hash J | := Ø.
5 /IF | Hash I | = |I| THEN add each item c ∈ J∩X to Hash J
with the position of X in sequence sid.
6 / IF |Hash I| < |I| THEN add each item d ∈ I∩X to Hash I
with the position of X in sequence sid.
7 / IF |Hash J| = |J| and | Hash I | = |I| THEN add sid to a variable sids(I⇒J∪{c}) for each
item c ∉ I∪J occurring after the last item of I in the window. }}
8 / FOR each item c where |sids(I⇒J∪{c})| / |S| ≥ Minsup {
9 /EXPANDLEFT(I⇒J∪{c}, sids(I), sids(I⇒J∪{c})).
10 / EXPANDRIGHT (I⇒J∪{c}, sids(I), sids(I⇒J∪{c})).
11/ IF |sids(I∪{c} ⇒J)| / |sids(I∪{c})| ≥ Minconf
THEN OUTPUT rule I∪{c} ⇒J. }}

Figure 3.7: La Procédure Expansion à droite de l'algorithme TRuleGrowth. **[15]**

❖ Les étapes de Procédure Expansion à droite de l'algorithme TRuleGrowth:

↳ On peut expliquer les différentes étapes de La Procédure Expansion à droite de L'algorithme TRuleGrowth, comme montré ci-dessous :

EXPANDRIGHT (I⇒J, sids (I), sids(I⇒J)):

Etape1 : pour chaque séquence sid où la règle (I⇒J) est présente, initialise le table de hagage "Hash I" := Ø, "Hash J := Ø.

Etape2 : pour chaque itemset X appartient in séquence sid, du dernier au Premier {

Etape3 : tous les items trouvés plus de (window_size – 1) itemsets avant sont supprimés de tables "Hash I" et "Hash J" parce que nous considérons qu'ils tombent en dehors de la fenêtre définie par l'utilisateur

Etape4 :

Etape5 : Si la taille de la table "Hash I" est égale à la taille de "I" alors Ajouter chaque item c ($c \in J \cap X$) dans la table "Hash J". avec la position de X dans la séquence sid.

Etape6 : Si la taille de la table "Hash I" est inférieur à la taille de "I" alors Ajouter chaque item d ($d \in I \cap X$) dans la table "Hash I", avec la position de X dans la séquence sid.

Etape7 :

Si la taille de "Hash I" égale à la taille de "I" et la taille de "Hash J" égal à la taille de "J" Alors : ajoute sid à variable sids ($I \Rightarrow J \cup \{c\}$) pour chaque item c ($c \notin I \cup J$) se produisant avant le premier élément de I dans la fenêtre.

Etape 8: Pour chaque item c

Vérifier si le support de la règle ($I \Rightarrow J \cup \{c\}$) est le plus grand que le support minimal Alors

Etape9 : La procédure Expansion à gauche

EXPANDLEFT ($I \Rightarrow J \cup \{c\}$, sids(I), sids($I \Rightarrow J \cup \{c\}$)).

Etape10 : La procédure Expansion à droite

EXPANDRIGHT ($I \Rightarrow J \cup \{c\}$, sids(I), sids($I \Rightarrow J \cup \{c\}$)).

Etape11: Si la règle ($I \Rightarrow J \cup \{c\}$)a une plus grande confiance que confiance minimal, alors la règle est considérée comme une règle valide et extraire.

3.4.6/ Exemple d'exécution:

Soit la base des séquences SD qui sont représentés sur le tableau ci-dessous, tel que, nous définissons Minsup = 0.7, Minconf = 0.8 et Taille de Fenêtre = 3,

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

***Tableau 3.2** : Base des séquences SD.* **[17]**

↳ Les étapes de l'exécution TRuleGrowth :

1/ Scannez la base des séquences SD une fois, pour chaque item trouvé, notez-les identificateur (sids) des séquences qui contient cet item, comme le montre la Figure ci-dessous :

Items	Séquences
1	S1, S2, S3, S4
2	S1, S2, S3, S4
3	S1, S2, S3, S4
4	S1, S2, S3
5	S2, S3, S4
6	S1, S3, S4
7	S4

***Tableau 3.3 :** Liste des séquences pour chaque Item*

2/ Sélectionnez tous les items qu'ont un support supérieur ou égale au support minimal (Minsup= 0.7) , qui définit par l'utilisateur.

Item	Support	Item	Support
1	1	4	0.75
2	1	5	0.75
3	1	6	0.75

***Tableau 3 .4**: les valeurs de Support pour chaque Item*

3/ A partir les items trouvées dans la phase précèdent, Découvrez tous les règles de longueur (1-1) :

Liste des règles séquentielles générées de taille (1-1)					
{1} ==> {2}	{2} ==> {1}	{3} ==> {1}	{4} ==> {1}	{5} ==> {1}	{6} ==> {1}
{1} ==> {3}	{2} ==> {3}	{3} ==> {2}	{4} ==> {2}	{5} ==> {2}	{6} ==> {2}
{1} ==> {4}	{2} ==> {4}	{3} ==> {4}	{4} ==> {3}	{5} ==> {3}	{6} ==> {3}
{1} ==> {5}	{2} ==> {5}	{3} ==> {5}	{4} ==> {5}	{5} ==> {4}	{6} ==> {4}
{1} ==> {6}	{2} ==> {6}	{3} ==> {6}	{4} ==> {6}	{5} ==> {6}	{6} ==> {5}

Tableau 3.5*: Liste des règles séquentielles générées de Taille (1-1)*

4/ On vérifier que ces règles produit dans la fenêtre de la taille défini

(Taille de Fenêtre =3) et ses supports supérieurs ou égale à Minsup.

Par exemple :

↳ La règle **{1} ==> {5}** avec un support 0.25 inferieur de **Minsup,** et ne produit pas dans de la fenêtre la taille défini, alors elle rejet.

↳ La règle **{1} ==> {2}** avec un support 100% supérieur de **Minsup** et produit dans la fenêtre de la taille défini, alors elle est valable.

Donc les règles de taille (1-1) obtenues sont : **{1} ==> {2}**

{1} ==> {3}

{4} ==> {3}

5/Puis avec l'utilisation de deux procédures récursives *EXPANDRIGHT* et EXPANDLEFT de règle, il commence la génération des règles large à partir les règles plus petites générées précédemment (règles de longueur 1-1), comme le montre la figure ci-dessous :

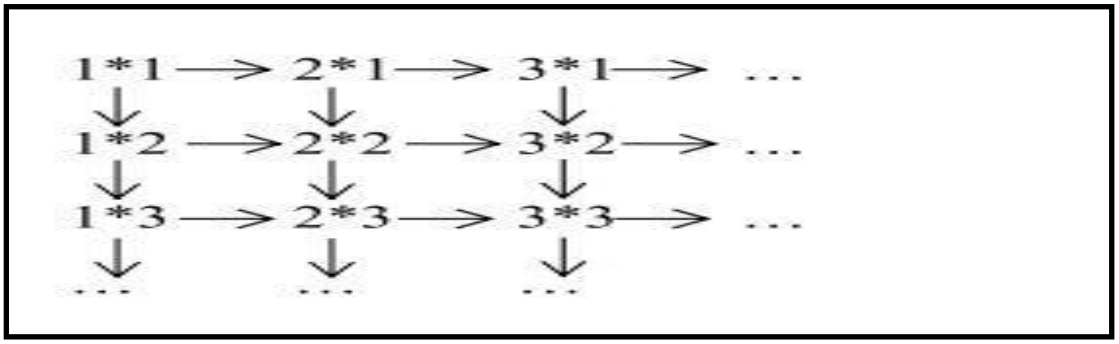

Figure 3.8 : *L'ordre de découverte des règles par les expansions gauche / droite.* **[15]**

Par exemple : On prend la règle : **{1} ==> {3}**

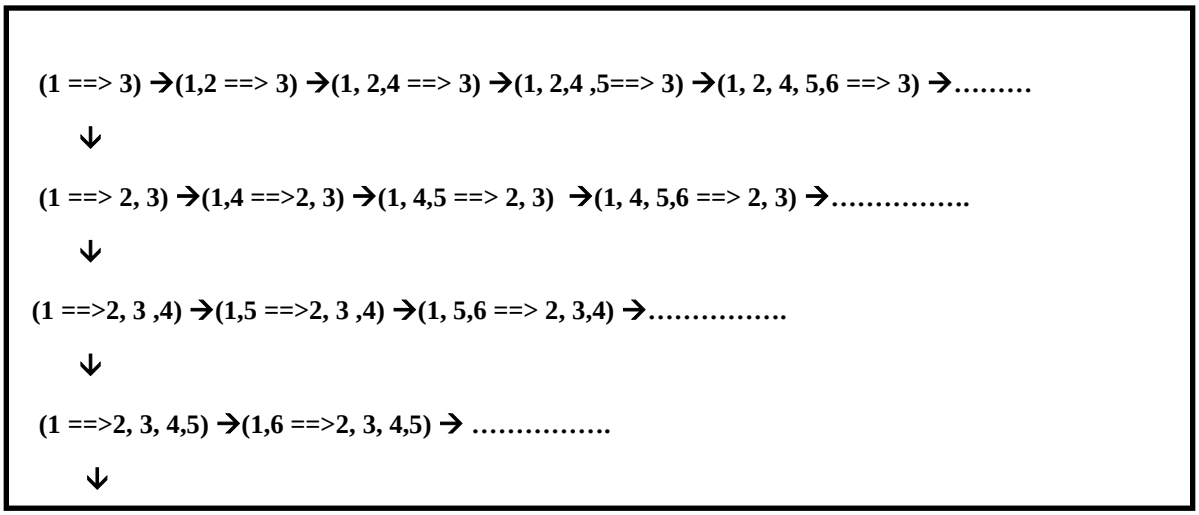

Figure 3.9 : *Liste des règles découvertes par les expansions gauche / droite de la règle :* ***{1} => {3}***

6/ Toutes les règles générées et ont une confiance supérieure ou égale au confiance minimale (**Minconf = 0 .8**) sont sélectionnées en tant que règles séquentielles valides et les règles restantes sont rejetées.

Donc, la listes des règles séquentielles valides résultante est :

Règles	Support	Confiance
{1} ==> {2}	**100 %**	**100 %**
{1} ==> {3}	**100 %**	**100 %**
{4} ==> {3}	**75 %**	**100 %**
{1} ==> {2, 3}	**100 %**	**100 %**

Tableau 3.6: *listes des règles séquentielles valides résultantes*

3.5/ Conclusion :

Nous avons présentée dans ce chapitre, deux algorithmes pour l'extraction des règles séquentielles partiellement ordonnées : RuleGrowth, TRuleGrowth.

Dans le chapitre suivant, on va voir les outils d'implémentation de notre application,

ainsi discussion sur les résultats obtenues après l'exécution de chaque algorithme choisi.

CHAPITER *IV*: ÉTUDE EXPÉRIMENTALE

Chapitre IV*: *Etude Expérimentale	
4.1*/Introduction*	***Page 50***
4.2*/ Conception générale*	***Page 50***
4.2.1*/ Cycle de développement en V*	***Page 50***
4.3*/ Spécification des besoins*	***Page 51***
4.3.1 */ Spécification des besoins fonctionnels*	***Page 52***
4.3.2 */ Spécification des besoins non fonctionnels*	***Page 52***
4.4*/ Implémentation*	***Page 53***
4.4.1*/ Choix du langage de programmation : Java*	***Page 53***
4.4.2*/ Choix de l'architecture de l'application*	***Page 53***
4.4.3*/ Choix de l'outil de développement : IDE ECLIPSE*	***Page 54***
4.5*/ Structure des Données*	***Page 55***
4.5.1*/ Format de Fichier d'Entrée*	***Page 56***
4.5.2*/Caractéristiques de quelques bases des séquences réelles*	***Page 57***
4.6*/ Description de l'application*	***Page 58***
4.7*/ Discussion des Résultats*	***Page 67***
4.7.1*/ Expérience pour évaluer l'influence du Support Minimal*	***Page 67***
4.7.2*/ Expérience pour évaluer l'influence de la Confiance Minimale*	***Page 68***
4.7.3*/ Expérience pour évaluer l'influence de la Taille de Fenêtre*	***Page 69***
4.7.4*/ Expérience pour évaluer l'évolutivité de l'algorithme*	***Page 70***
4.7.5 */Format de Fichier de Sortie*	***Page 71***
4.8*/ Comparison entre RuleGrowth et TRuleGrowth*	***Page 72***
4.9*/ Conclusion*	***Page 73***

4.1/ Introduction:

Une étape essentielle de tout cycle de développement d'un logiciel ou application consiste à effectuer une étude préalable. Le but de cette phase est de comprendre le contexte du système, avec une vision claire des aspects fonctionnels et organisationnels de l'application.

Dans ce chapitre, nous présentons différentes étapes de conception d'application au cours du développement du niveau de détail, afin d'implémentation une application qui nous permet d'étudier les deux algorithmes **: RuleGrowth** et **TRuleGrowth.**

4.2/ Conception Générale :

Après avoir tracé les grandes lignes de phase de spécification de besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception. Cette phase a pour objectif de déduire et approfondit la spécification de l'architecture de système.

4.2.1/ Cycle de développement en V :

Pour la conception, le développement et la réalisation de notre application, nous avons opté l'application du processus de développement en V qui demeure actuellement le cycle de vie le plus connu et certainement le plus convenable aux différents type des projets.

Ce processus nous a accompagné du début de projet jusqu'à l'implémentation. Son principe est qu'avec toute décomposition doit être décrite la recomposition, et que toute description d'un composant doit être accompagnée de test qui permettra de s'assurer qu'il correspond à sa description. Le schéma ci-dessous représente les différentes phases du modèle en V :

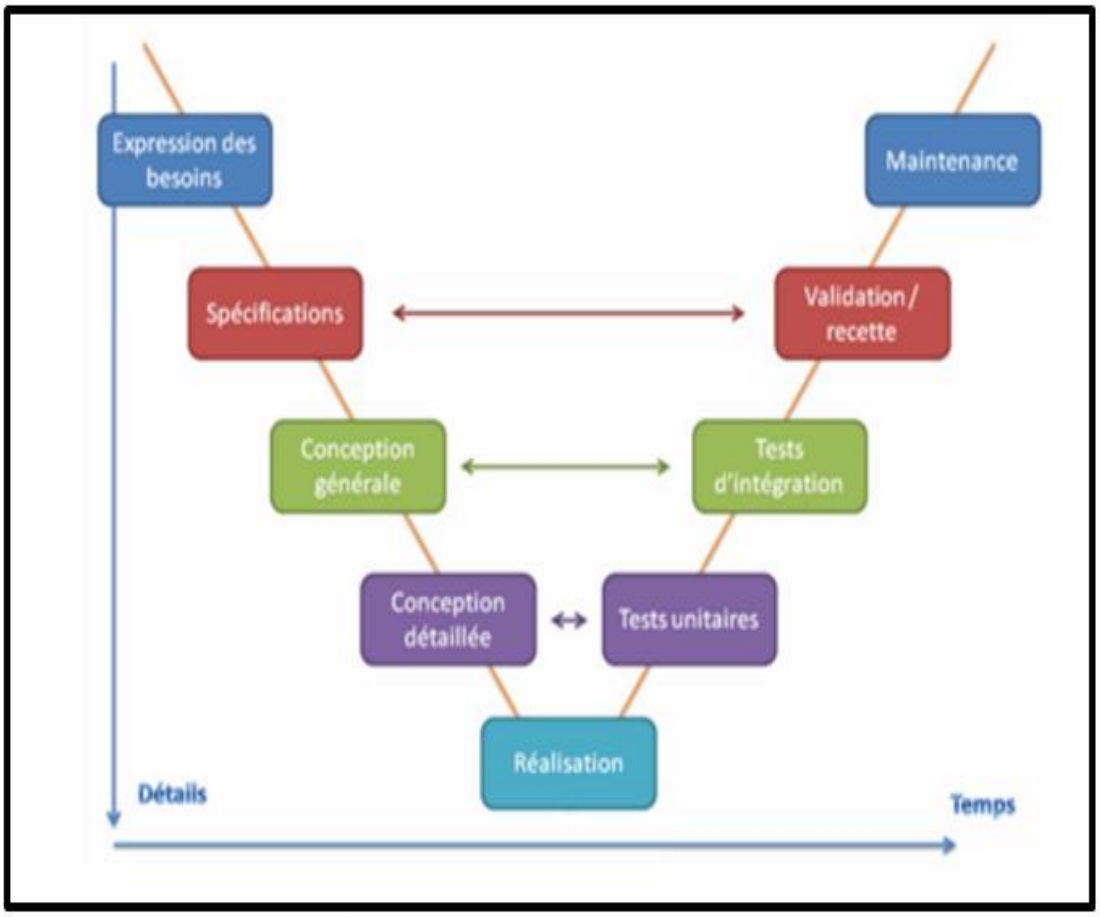

Figure 4.1 : Modèle en V. [13]

4.3/ Spécification des besoins:

La spécification des besoins constitue la phase de départ de toute application à développer dans laquelle nous allons identifier les besoins de notre application. Nous distinguons deux types des besoins, fonctionnels qui présentent les fonctionnalités attendues de notre application et les besoins non fonctionnels pour le développement d'une application satisfaisante ainsi de trouver un accord commun entre les spécialistes et les utilisateurs pour réussir le projet**. [18]**

4.3.1 / Spécification des Besoins Fonctionnels :

Après une étude détaillée de système, cette partie est réservée à la description des exigences fonctionnelles des différents acteurs de l'application.

Dans notre application, les besoins utilisateur :

- Authentification de l'utilisateur
- Création et ajout des données pour la génération des fichiers d'entrée (sous forme txt,...)
- Génération des règles séquentielles en exécutant l'un des algorithmes cité dans le chapitre précédant, par l'utilisation des fichiers des données et d'un ensemble des paramètres spécifiques en entrée pour avoir le résultat en sortie.
- Analyse et comparaison en utilisant quelques outils d'analyse des résultats [18].

4.3.2 / Spécification des Besoins non Fonctionnels :

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement, et ce qui concerne notre application, nous avons dégagé les besoins suivants :

- **La disponibilité**: l'application doit être disponible pour être utilisé par n'importe quel utilisateur.
- **La confidentialité** de l'accès à l'application par des utilisateurs bien déterminés.
- **La fiabilité** : les données fournies par l'application doivent être fiables.
- **La convivialité de l'interface graphique :** l'application doit fournir une interface conviviale et simple pour tout type d'utilisateur car elle présente le premier contact de l'utilisateur avec l'application.
- **Une solution ouverte et évoluée** : l'application peut être améliorée par l'ajout d'autres modules pour garantir la souplesse, l'évolutivité et l'ouverture de la solution.
- La possibilité de retourner à l'interface principale de l'application à partir de n'importe quelle fenêtre de celle-ci. [**18]**

4.4/ Implémentation :

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

4.4.1 Choix du langage de programmation : Java

Java est un langage de programmation orienté objet, développé par **Sun Microsystems** (était un constructeur d'ordinateurs et un éditeur de logiciels américain)et destiné à fonctionner dans une machine virtuelle, il permet de créer des logiciels compatibles avec des nombreux systèmes d'exploitation. Java est non seulement un langage de programmation puissant, mais aussi un environnement de développement qui est continuellement étendu pour fournir des nouvelles caractéristiques et des bibliothèques permettant de gérer de manière élégante des problèmes traditionnellement complexes dans les langages de programmation classiques, tels que le multithreading, les accès aux bases des données, la programmation réseau, l'informatique répartie. En plus java est considéré comme un langage adaptable aux plusieurs domaines puisque une application web implémentée par celle-ci peut avoir des extensions ou des modifications dans le futur.

De plus, Java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé.

La version que nous avons utilisée est **JDK 7** (java développement kit).

4.4.2/ Choix de l'architecture de l'application:

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

Fondamentalement, une architecture 1-tiers conserve tous les éléments d'une application, compris l'interface exécutable et les données en un seul endroit. Les développeurs voient ces types de systèmes comme le plus simple et le plus direct. Certains experts décrivent comme des applications qui pourraient être installées et exécutées sur un seul ordinateur. La nécessité pour les modèles distribués pour les applications Web et des solutions d'hébergement Cloud a créé de nombreuses situations où les architectures à un seul niveau ne sont pas suffisantes. Cela fait à trois niveaux ou de l'architecture multi-niveaux pour devenir plus populaire. Les avantages d'une solution multi-niveaux sont souvent évidents. Ils peuvent fournir une meilleure sécurité, une meilleure performance et plus d'évolutivité. Toutefois, l'appel d'une architecture 1 -tiers peut se rapporter aux coûts qui sont impliqués, où il serait plus logique de garder des applications plus simples contenues dans une plate-forme facile. L'architecture 1-tiers peut être bénéfique lorsque nous traitons avec des données qui se rapportent à un seul utilisateur (ou un petit nombre d'utilisateurs) et nous avons une quantité relativement faible de données. Ils sont peu coûteux à déployer et à maintenir. [**19]**

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

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

4.4.3 / Choix de L'outil de développement : IDE ECLIPSE

Eclipse est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions. La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plugin : toutes les fonctionnalités de cet atelier logiciel sont développées entant que plug-in. [13]

Dans le cadre de notre projet, nous avons utilisé la version Eclipse Java EE IDE Neon.3 Release (4.6.3), avec plusieurs plugins Parmi ces plugins, on cite les suivants :

◈ **Windows Builder :** est un composant de Design qui appartiens à la bibliothèque SWT et Swing, il est très facile de créer des applications Java GUI sans dépenser beaucoup de temps à écrire du code, il permet d'utiliser le concepteur qui possède des outils visuels pour créer des formes simples, des tableaux , le code Java sera généré pour vous, Ajouter facilement des contrôles par glisser-déposer, ajouter des gestionnaires d'événements pour vos commandes, modifier diverses propriétés des contrôles en utilisant un éditeur de propriété. Windows Builder construit un arbre de syntaxe abstraite (AST) pour naviguer dans le code source et aussi afficher et gérer la présentation visuelle**. [13]**

◈ **La Bibliothèque SPMF :**

SPMF (Sequential Pattern Mining Framework) est une bibliothèque de Fouille de données (data mining) open-source a été créé en 2009 , qu'est une bibliothèque multiplateforme implémentée en Java, spécialisé pour la découverte de modèles dans les bases de données de transaction et de séquence telles que les ensembles d'éléments fréquentes, les règles d'association et les règles séquentiels.

Il fournit également des implémentations de plus de 120 algorithmes de data mining ,et une simple interface utilisateur pour un test rapide et une interface de ligne de commande pour une intégration facile avec d'autres systèmes.

Au cours des cinq dernières années, SPMF a été utilisé dans plus de 310 documents de recherche pour résoudre des problèmes appliqués dans un large éventail de domaines d'attribution d'auteur, la prévision de détail, la chimie....

Les implémentations des algorithmes de SPMF sont optimisées périodiquement et couramment utilisé comme points de référence dans les articles de recherche**. [17]**

4.5 / Structure des Données :

Les données utilisés pour exécuter nos algorithmes sont sous format (*.txt)*, ces données sont importés depuis des bases des séquences réelles qui ont été définit par les utilisateurs, ces fichiers contiennent un nombre de séquences, qu'ils contiennent des items numérotés, chaque item correspond à un article bien défini, ces Base des séquences ont des taille différente.

4.5.1/ Format de Fichier d'Entrée :

C'est un fichier texte où chaque ligne représente une séquence d'une base des séquences. Chaque élément d'une séquence est un entier positif et les éléments du même itemset dans une séquence sont séparés par des espaces uniques.

Notez qu'il est supposé que les itemsets d'une même séquence sont triés en fonction d'un ordre total et qu'aucun itemsets ne peut apparaître deux fois dans la même séquence.

La valeur "-1" indique la fin d'un itemset (ensemble d'éléments).

La valeur "-2" indique la fin d'une séquence (elle apparaît à la fin de chaque ligne).

Par exemple, Le fichier d'entrée " **contextPrefixSpan.txt** " contient l'ensemble des séquences comme montre la Figure suivant :

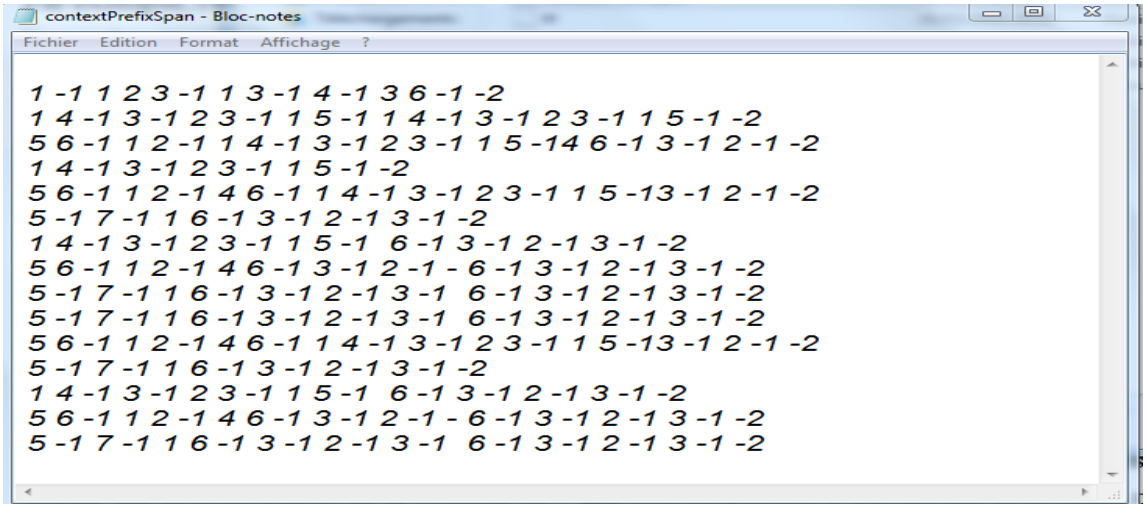

Figure 4. 2 : *Fichier de base des séquences d'entrée.* **[17]**

↳ On remarque que la première ligne représente une séquence où l'itemset {1} est suivi par l'itemset {1, 2, 3}, suivi de l'itemsets {1, 3}, suivi de l'itemset {4}, suivi de l'itemset {3, 6}, tel que chaque deux itemsets séparé par (-1) et en fin de séquence on trouve (-2).

Les lignes suivantes suivent le même format.

↳ Notez qu'il est également possible d'utiliser un fichier texte contenant un texte

(plusieurs phrases). **[17]**

4.5.2 / Caractéristiques de quelque Bases des séquences réelles :

Dans notre travail, nous avons utilisés plusieurs fichiers de données. Ces jeux de données ont été choisis, parce qu'ils sont des ensembles des données réelles ayant des caractéristiques variées, parmi ces fichiers on trouve :

1/Kosarak : Il s'agit d'un très grand ensemble de données contenant 990 000 séquences de données de flux de données provenant d'un portail d'actualités hongrois, cependant cet ensemble de données est très volumineux. Par conséquent et pour rendre l'expérience plus rapide, nous ont utilisé un sous-ensemble de seulement 10 000 séquences tel que chaque séquence a une longueur moyenne de 7,97 items. **[17]**

2/ BMSWebView1 (Gazelle) (KDD CUP 2000) : Cet ensemble de données contient 59 601 séquences de données Click Stream provenant d'un commerce électronique, Il contient 497 éléments distincts. La longueur moyenne des séquences est de 2,42 éléments, dans cet ensemble de données, il y a quelques séquences longues. Par exemple, 318 séquences contiennent plus de 20 éléments. **[17]**

3/BMSWebView2 (Gazelle) (KDD CUP 2000) . Ceci est un deuxième jeu de données utilisé dans le concours KDD-CUP 2000. Il contient 77 512 séquences de données de flux de données. Il contient 3340 objets distincts. La longueur moyenne des séquences est de 4,62 éléments. **[17]**

4/ FIFA : un jeu de données de 20 450 séquences de données de flux sur le site de la FIFA World Cup 98. Il contient 2 990 éléments distincts (pages Web). La longueur moyenne de la séquence est de 34,74 éléments. Ce jeu de données a été créé en traitant une partie des logs web de la coupe du monde 98. **[17]**

5/Signe : un ensemble de données d'énoncé en langue des signes contenant environ 800 séquences. Ce fichier créé par le Centre national pour la langue des signes et les ressources gestuelles à l'Université de Boston, consiste en une collection d'énoncés, où chaque énoncé associe un segment de vidéo à une transcription détaillée. Chaque énoncé contient un certain nombre de champs gestuels et grammaticaux de l'ASL (ex.: Augmentation du sourcil, inclinaison de la tête vers l'avant….), chacun se produisant sur un intervalle de temps. C'est un ensemble de données modérément dense avec des séquences longues. **[17]**

4.6/ Description de l'application :

La conception des interfaces de notre application est une étape très importante puisque toutes les interactions avec l'application passent à travers ces interfaces, nous devons alors guider l'utilisateur avec les messages d'erreur et avec des notifications si besoin.

Dans cette partie, nous allons présenter quelques cas d'utilisations, sous forme d'un guide utilisateur.

Pour accéder à notre application, on doit s'authentifier, comme toute application, la sécurité d'accès est nécessaire. La Figure ci-après donne l'interface d'authentification à travers laquelle l'utilisateur s'identifie. Il saisit son nom d'utilisateur et son mot de passe, si les informations saisis correct, l'application permet à l'utilisateur accède l'interface principale :

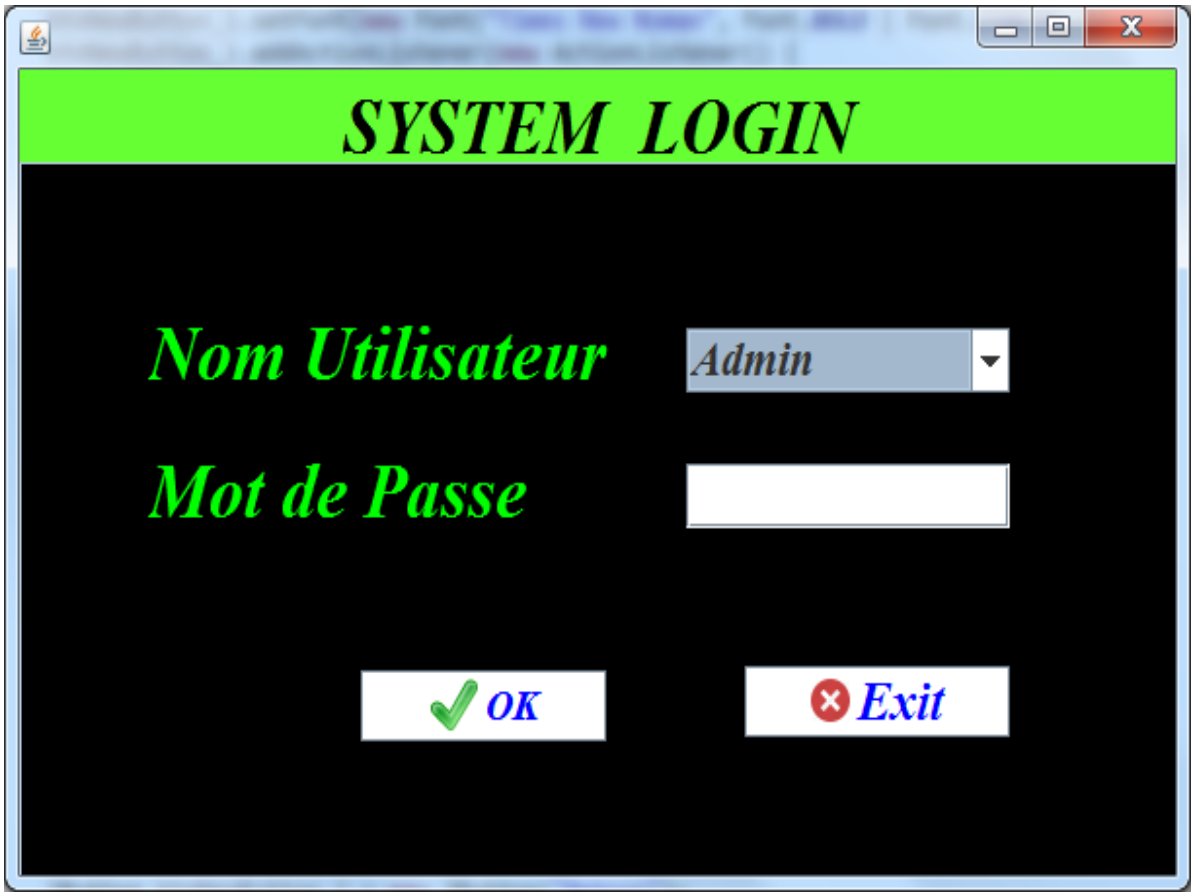

Figure 4.3 : Interface d'authentification

La deuxième interface est l'interface principale, à travers laquelle nous choisissons le type d'étude, où il existe deux types :

- ***Etude Exécutive :*** l'exécution de l'algorithme en tenu compte, comme entrée un seul Fichier.

- ***Etude Comparative :*** l'exécution de l'algorithme en tenu compte, comme entrée plusieurs Fichier pour comparer et analyse les résultats.

Cette interface est représentée ci-après :

Figure 4.4 : Interface Principale

- Dans le cas du choix ***L'Etude Exécutive*** L'interface suivante est l'interface qui regroupe les algorithmes qui doit être exécuté par l'utilisateur en utilisant comme entrée un seul Fichier, en passant plusieurs paramètres, parmi ces paramètres on cite :

- ***Le Fichier d'Entrée :*** c'est un fichier de plusieurs séquences qui contient des items en extension .txt.

- ***Support Minimal :*** c'est le support minimal saisi par l'utilisateur, pour définir le nombre d'occurrence de chaque partie de la règle.

- ***Confiance Minimale :*** c'est la confiance minimale saisi par l'utilisateur pour dire que cette règle est valide.

- ***Taille de fenêtre :*** c'est une valeur saisi par l'utilisateur, pour génère seulement les règles qui respect la contrainte de Taille de la Fenêtre, cette paramètre utilise seulement pour l'algorithme TRuleGrowth.

- ***Max antécédent, Max conséquent :*** sont des valeurs optionnelles saisi par l'utilisateur, pour détermine la taille de antécédent et conséquent pour chaque règle séquentielle généré.

- ***Le Fichier de Sortie :*** Il y a deux types :

1/ Un Fichier texte : qui contient les différents règles générés après l'exécution de l'algorithme approprié, ce dernier contient aussi le support de la règle ainsi la confiance.

2/ Un Liste de Motif : qui contient les différents règles générés après l'exécution de l'algorithme choisi, et aussi le support de la règle ainsi la confiance. Ce dernier contient également des statistiques telles que le nom, la taille de fichier et le nombre des Motifs.

Nous présentons cette interface par cette figure ci-après :

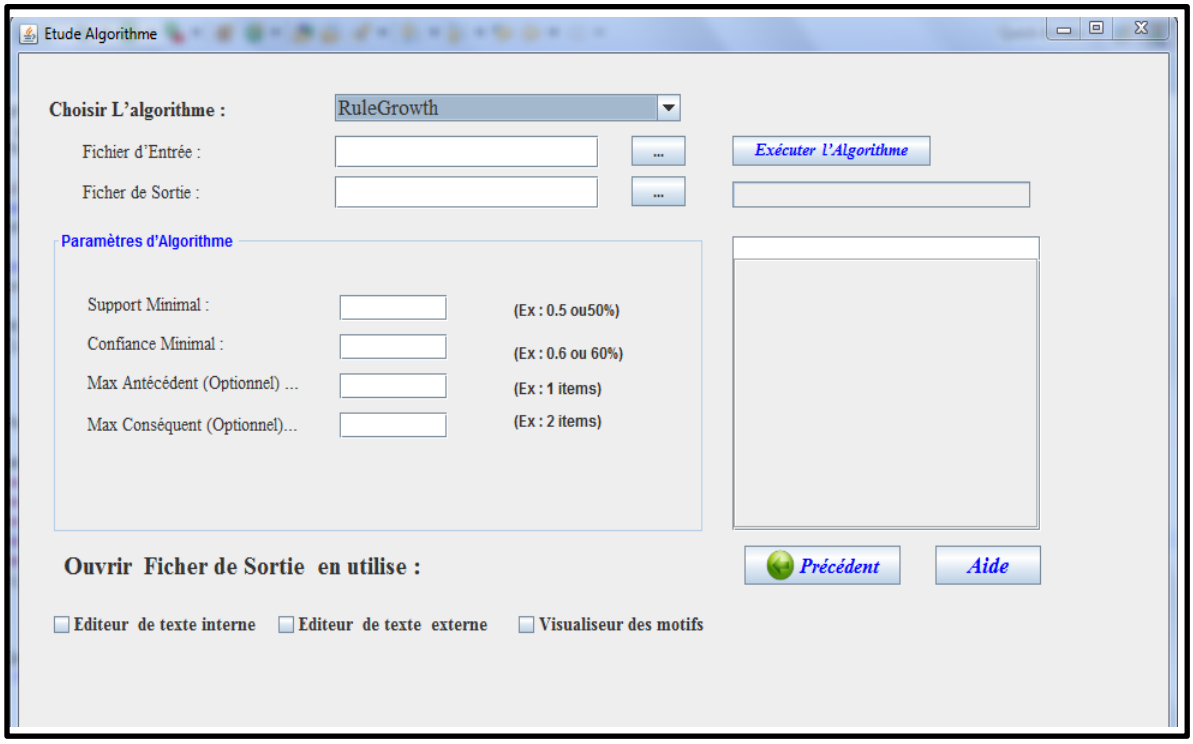

Figure 4.5 : *Interface de L'Etude Exécutive pour la génération des règles séquentielles*

D'après cette interface, on peut générer plusieurs scénarios pour extraire des règles séquentielles, on va citer quelques scénarios :

1/ L'algorithme RuleGrowth :

❖ Nous choisissons l'algorithme **RuleGrowth** de la liste déroulante en haut, puis nous donnons le chemin de Fichiers d'Entrées ainsi le chemin de sauvegarde le Fichier de Sortie, qui seras généré automatiquement lors de l'exécution.

❖ Ensuite, nous introduisons les différents paramètres correspond à l'algorithme : (Support minimal et la Confiance minimale) et on peut ajouter les paramètres optionnelle Max antécédent, Max conséquent.

❖ Maintenant, nous cliquons sur le bouton « **Exécuter l'algorithme »** pour voir les résultats obtenus, comme montré sur ce Figure.

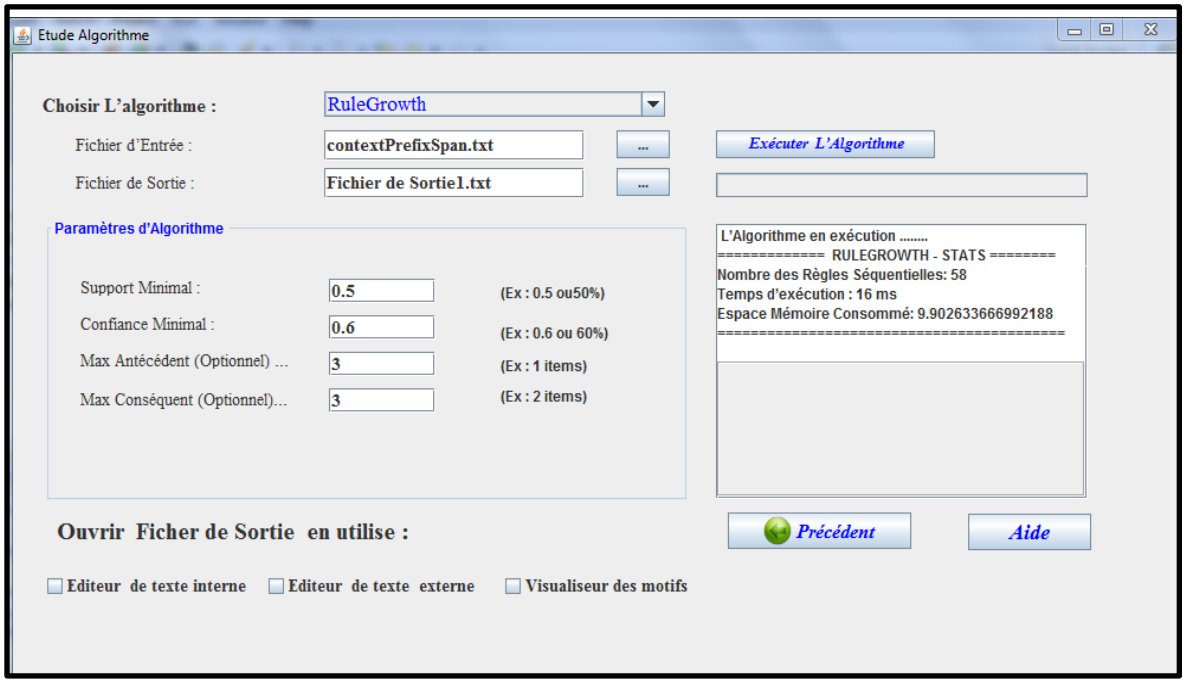

Figure 4.6 : Résultats obtenus après l'exécution de l'algorithme **RuleGrowth**

❖ Ainsi, pour afficher le résultat, choisi le type de Fichier de Sortie qu'est utilisé, par exemple on choisit L'éditeur de texte interne (affichage sur l'application) en cliquant sur le bouton « **Editeur de texte interne ».** Les résultats obtenus sont illustrés sur cette figure.

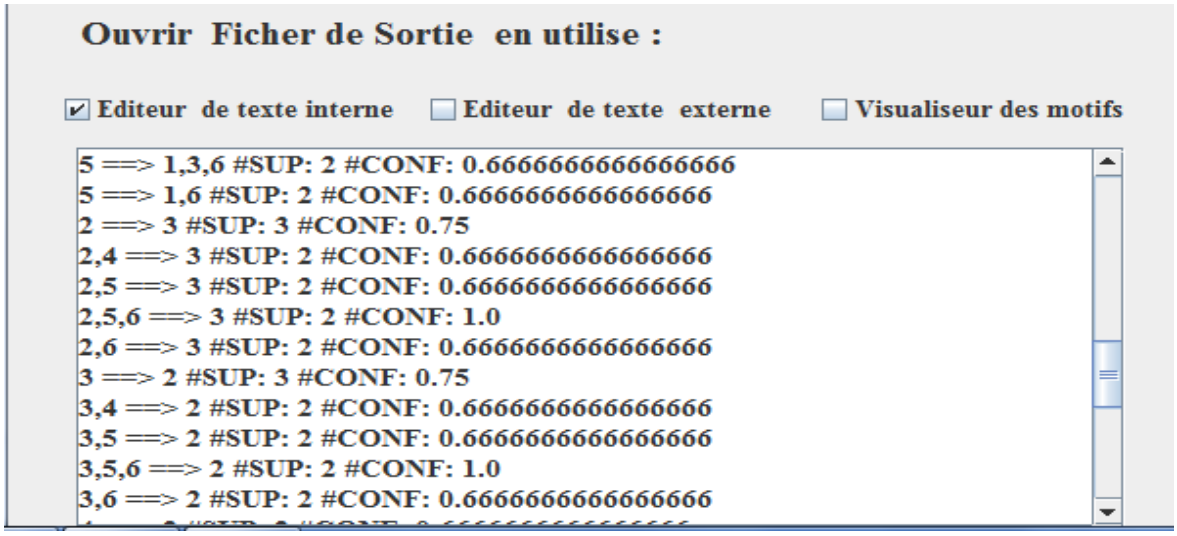

Figure 4.7 : *Fichier de Sortie interne après l'exécution de l'algorithme* ***RuleGrowth***

2/ L'algorithme TRuleGrowth :

❖ Nous choisissons l'algorithme **TRuleGrowth** de la liste déroulante en haut, puis nous donnons le chemin de Fichier d'Entrée, ainsi le chemin de sauvegarde le Fichier de Sortie, qui seras généré automatiquement lors de l'exécution.

❖ Ensuite, nous introduisons les différents paramètres correspond à l'algorithme : (Support minimal, Confiance minimale et la Taille de Fenêtre) et on peut ajouter les paramètres optionnelle Max antécédent, Max conséquent.

❖ Maintenant, nous cliquons sur le bouton « **Exécuter l'algorithme** » pour voir les résultats obtenus, comme montré sur ce Figure :

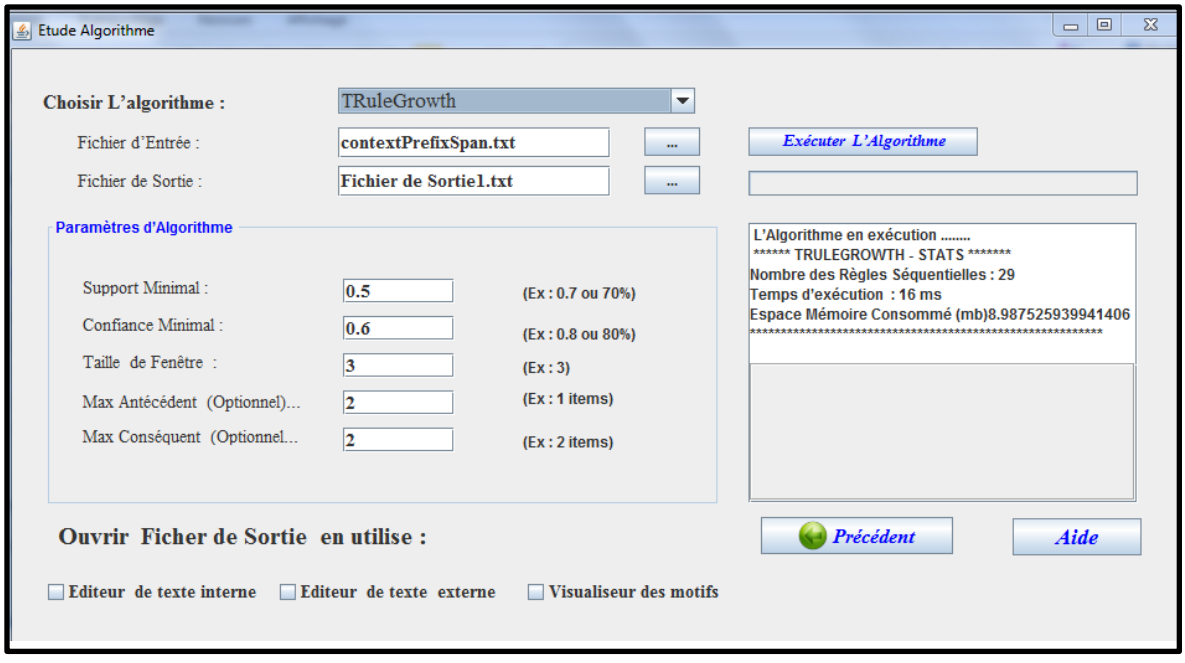

Figure 4.8*: Résultats obtenus après l'exécution de l'algorithme* ***TRuleGrowth***

◈ Notez que cette interface contient deux boutons supplémentaires :

- ***précèdent :*** qui permet de revenir à l'interface principale

- ***Aide :*** qui permet l'accès au site de la bibliothèque SPMF qui nous donne des informations supplémentaires sur les algorithmes.

◈ Pour voir les résultats obtenus, choisi le type de Fichier de Sortie qu'est utilisé :

1/ Choix L'éditeur de texte interne :

En sélectionnant le type **Editeur de texte interne**. Le résultat obtenu est illustrés sur cette Figure :

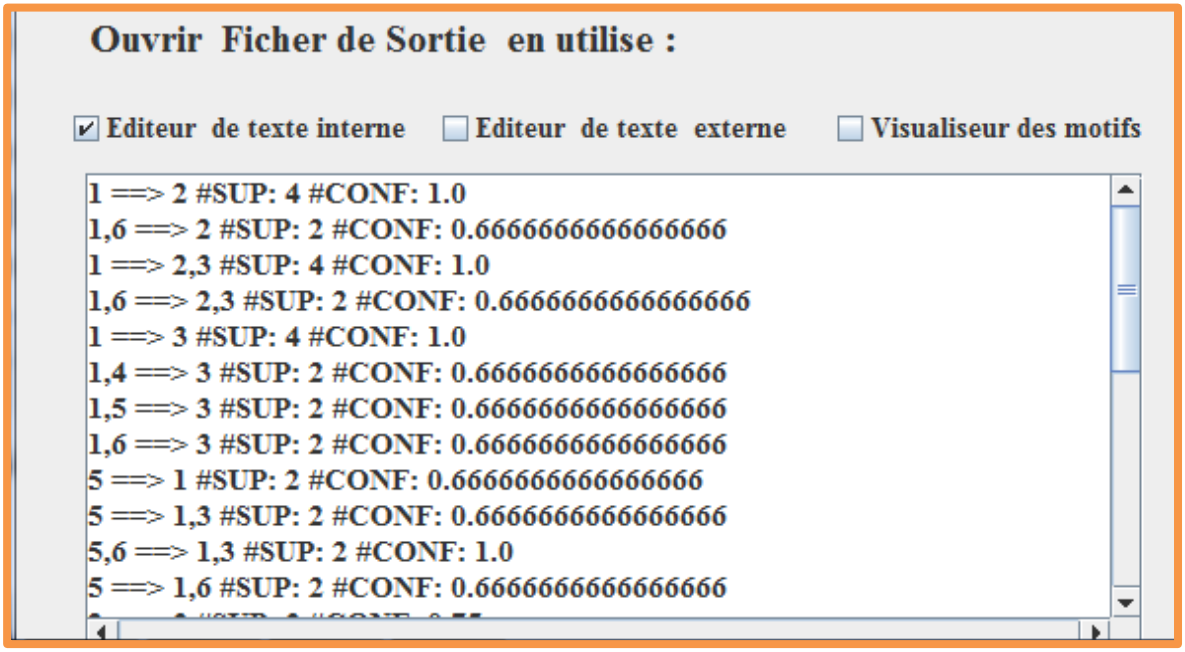

Figure 4.9 : *Fichier de Sortie interne après l'exécution de l'algorithme TRuleGrowth*

2/ Choix L'éditeur de texte externe :

En sélectionnant le type **Editeur de texte externe**. Le résultat obtenu est un fichier texte contient les différentes règles générées après l'exécution et aussi le support et confiance de la règle, comme montre sur cette figure :

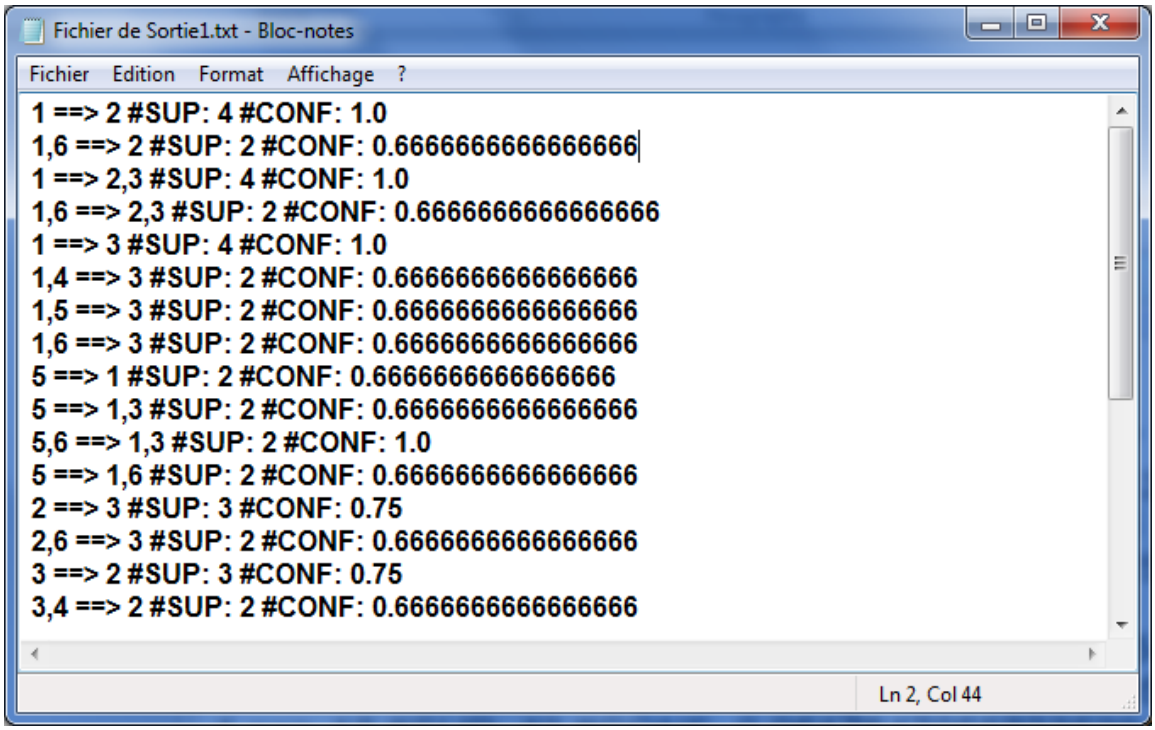

Figure 4.10 : *Fichier de Sortie externe après l'exécution de l'algorithme TRuleGrowth*

3/ Choix de visualiseur des motifs:

En sélectionnant le type visualiseur des motifs. Le résultat obtenu est un fichier de liste Motifs contient les différents règles générés après l'exécution, et aussi le support et la confiance de la règle, en plus des statistiques telles que le nom , la taille de fichier et le nombre des Motifs , comme montre sur cette figure :

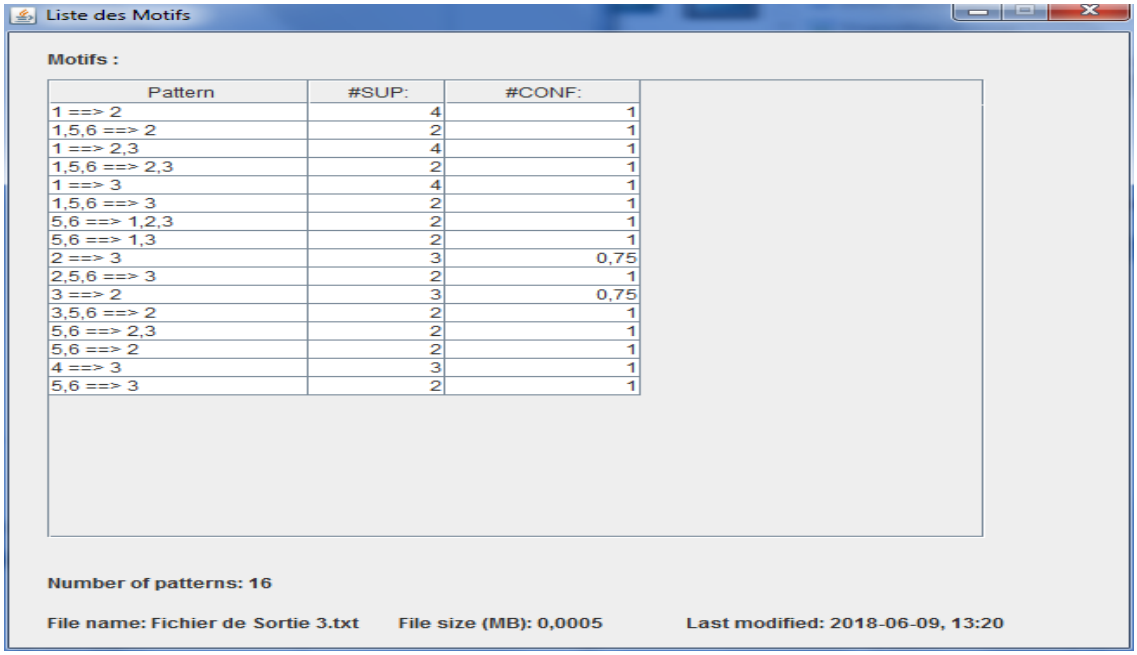

Figure 4.11: *Liste de Motifs affichée après l'exécution de l'algorithme TRuleGrowth*

❖ Par retour à l'interface Principal et sélectionnant le choix de ***L'Etude Comparative*** l'interface suivante est l'interface qui regroupe les algorithmes qui doit être exécuté par l'utilisateur en utilisant comme entrée plusieurs Fichier, pour comparer et analyse les résultats obtenues, Cette interface est représentée ci-après :

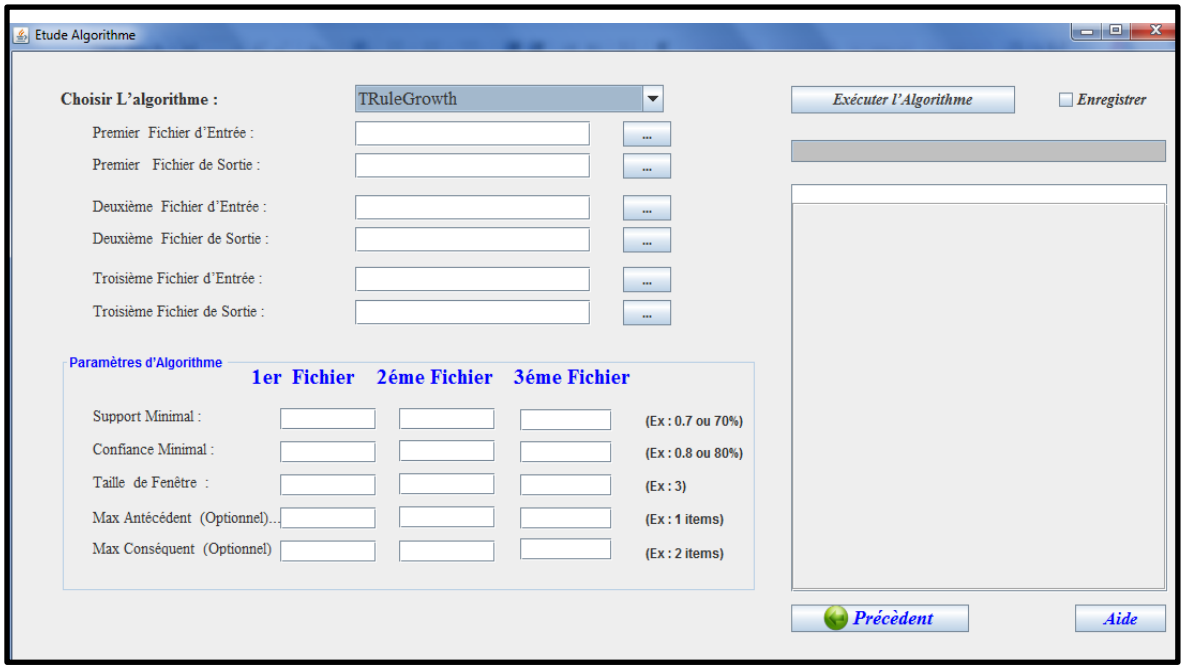

Figure 4.12 : *Interface comparative pour l'analyse des résultats obtenus*

D'après cette interface, on peut exécuter les algorithmes précèdent en utilisant plusieurs Fichier des Entries pour étudier et évaluer la performance de ces algorithmes en comparer et analyser les résultats obtenues.

Par exemple on prendre ***L'Algorithme TRuleGrowth.***

❖ Nous choisissons l'algorithme **TRuleGrowth** de la liste déroulante en haut, puis nous donnons le chemin de trois Fichiers d'Entrées ainsi le chemin de sauvegarde les trois Fichiers de Sorties, qui seront généré automatiquement lors de l'exécution.

❖ Ensuite, nous introduisons les différents paramètres correspond à l'algorithme :

(Support minimal, Confiance minimale et la Taille de Fenêtre) et on peut ajouter les paramètres optionnelle Max antécédent, Max conséquent.

❖ Maintenant, nous cliquons sur le bouton « **Exécuter l'algorithme »** pour voir les résultats obtenus, comme montré sur ce Figure :

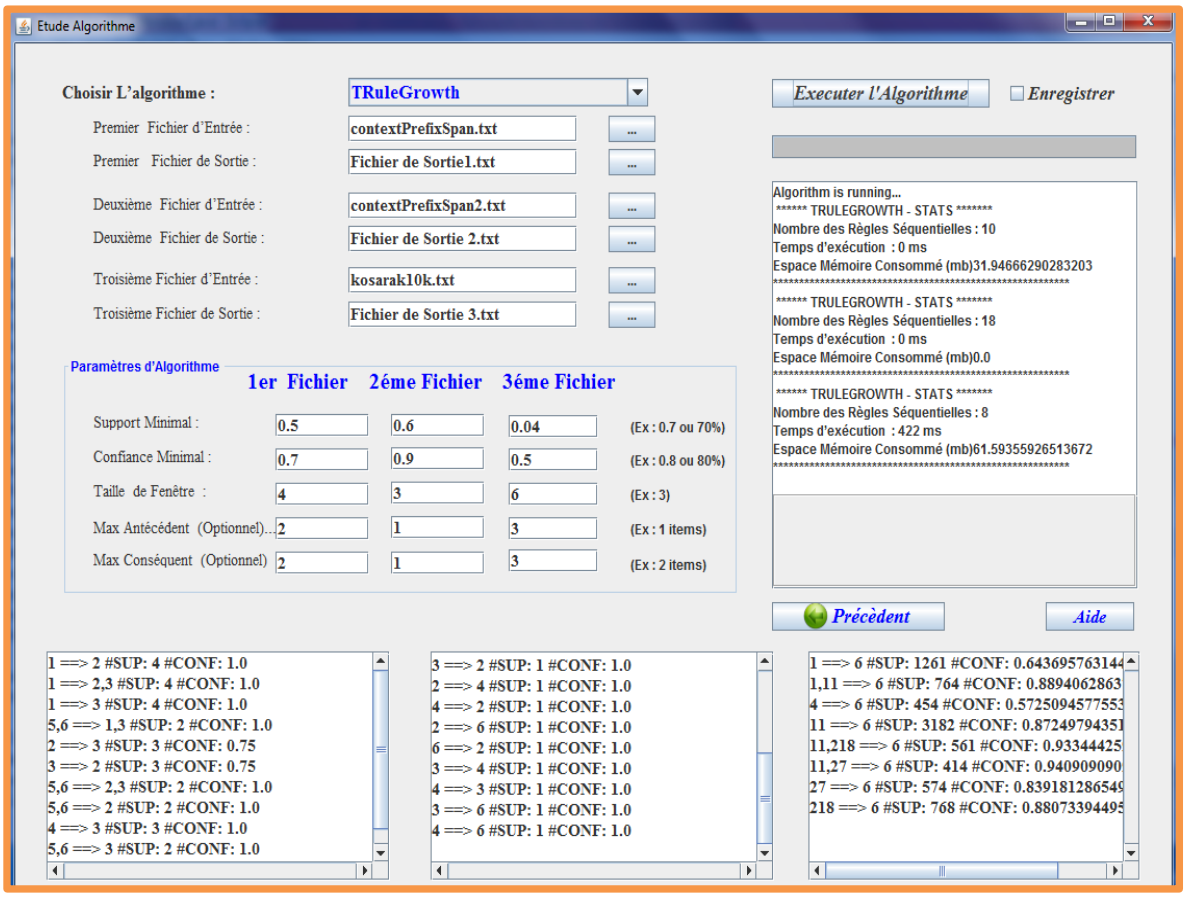

Figure 4.13 : *Interface comparative après l'exécution de l'algorithme TRuleGrowth*

↳ Dans cette interface on affiche les résultats obtenus après l'exécution de l'algorithme **TRuleGrowth** avec plusieurs fichiers des données réelles, tel qu'elle contient :

1/ des statistique sur le Nombre des règles obtenues, Temps d'exécution et L'espace Mémoire consommé.

2/ Liste des règles séquentielles générées avec ses support et confiance pour chaque fichier d'entrée.

4.7 /Discussion des Résultats :

Les résultats obtenus après l'exécution de l'algorithme TRuleGrowth plusieurs fois, sont décrit ci-après, selon les différents paramètres choisis par l'utilisateur, en fonction des besoins de ses applications, Nous détaillerons chaque résultat obtenu comme suit :

4.7.1/ Expérience pour évaluer l'influence de Support Minimal :

La première expérience consiste à exécuter TRuleGrowth, pour des trois ensembles des données avec différents valeurs pour **Minsup** et une valeur fixe pour Minconf et Taille de Fenêtre pour évaluer l'influence de **Minsup** sur la performance relative de l'algorithme.

Temps d'exécution, utilisation maximale de la mémoire et le nombre de règles trouvées sont illustrés sur le tableau suivant :

Algorithme TruleGrowth avec Minconf=0.08, Taille de Fenêtre =3												
Paramètres/ Donnée	**Nombre des règles Obtenues**				**Espace Mémoire consommé en (mb)**				**Temps d'exécutionen (ms)**			
	Minsup = 0 .01	Minsup= 0 .02	Minsup = 0 .04	Minsup = 0 .06	Minsup = 0 .01	Minsup= 0 .02	Minsup = 0 .04	Minsup = 0 .06	Minsup = 0 .01	Minsup = 0 .02	Minsup = 0 .04	Minsup = 0 .06
Kosarak	**49**	**47**	**15**	**06**	**144,999**	**180,310**	**147.120**	**156,096**	**789**	**534**	**388**	**251**
FIFA	**2370**	**1141**	**437**	**196**	**247,485**	**247.464**	**247.280**	**246.366**	**650962**	**358175**	**211867**	**127180**
SIGE	**886**	**398**	**245**	**158**	**73.5775**	**81.9683**	**90.2843**	**51.8569**	**5478**	**2220**	**1261**	**1045**

Tableau 4.1: Résultats obtenus après l'exécution TRuleGrowth avec Minconf =0.08 et Taille de Fenêtre =3

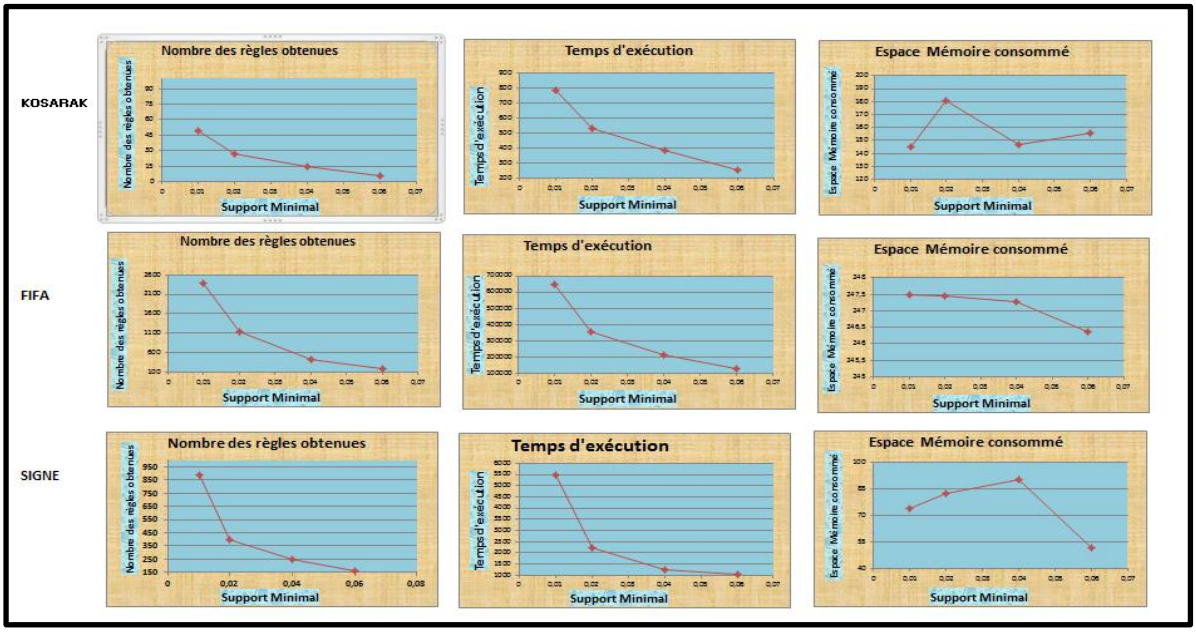

Figure 4.14: Résultats obtenus pour **Minsup =0.01 ,0.02, 0.04, et 0.06** pour chaque fichier de donnée

4.7.2/ Expérience pour évaluer l'influence de Confiance Minimal:

La deuxième expérience consiste à exécuter l'algorithme avec différentes valeurs de **Minconf** sur des trois ensembles des données. Pour cette expérience, fixe les valeurs de Minsup et la taille de fenêtre et donnes des valeurs déférentes pour **Minconf**.

Les résultats sont montrés sur le tableau suivant :

Algorithme TruleGrowth avec Minsup=0.08, Taille de Fenêtre =3												
Paramètres/ Donnée	**Nombre des règles Obtenues**				**Espace Mémoire consommé en (mb)**				**Temps d'exécutionen (ms)**			
	Minconf= 0.1	**Minconf = 0.3**	**Minconf = 0.6**	**Minconf = 0.9**	**Minconf = 0.1**	**Minconf = 0.3**	**Minconf = 0.6**	**Minconf = 0.9**	**Minconf = 0.1**	**Minconf = 0.3**	**Minconf = 0.6**	**Minconf = 0.9**
Kosarak	**68**	**38**	**16**	**2**	**39,700**	**38,790**	**45,316**	**55,798**	**1326**	**1068**	**1129**	**1394**
SIGE	**1190**	**1190**	**103**	**12**	**35,481**	**44,550**	**52,991**	**61,351**	**7941**	**5282**	**7925**	**7737**
BMS2	**35**	**21**	**4**	**0**	**150.748**	**152.499**	**146.869**	**148.474**	**2340**	**1981**	**2512**	**2340**

Tableau 4.2: Résultats obtenus après l'exécution TRuleGrowth avec Minsup =0.08 et Taille de Fenêtre =3

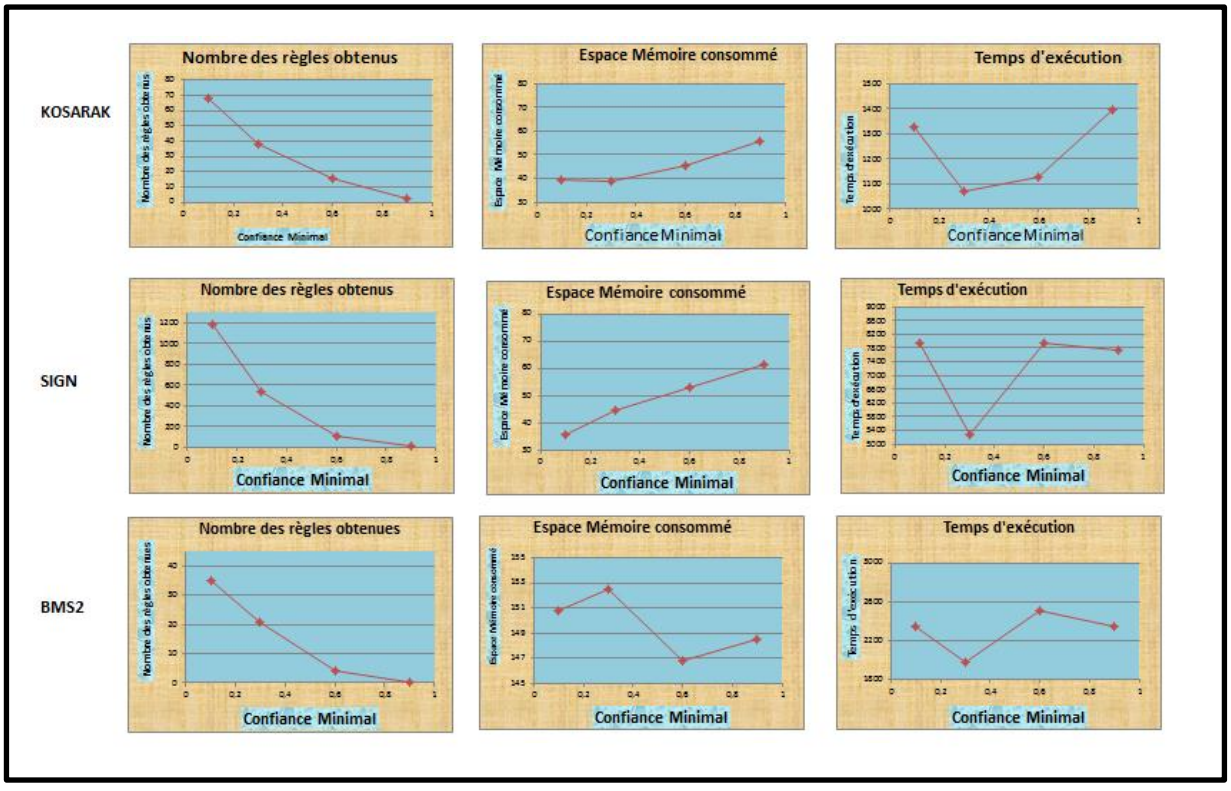

***Figure** 4.15:Résultats obtenus pour **Minconf = 0.1, 0.3, 0.6 et 0.9** pour chaque fichier de donnée*

4.7.3/ Expérience pour évaluer l'influence de la Taille de Fenêtre (window_size) :

La troisième expérience consiste à évaluer l'algorithme avec différentes valeurs de taille de fenêtre (T.F) sur des trois ensembles des données.

Pour cette expérience, fixe les valeurs de Minsup et Minconf et donnes des valeurs différentes pour la taille de fenêtre.

Les résultats sont montrés sur le tableau suivant :

Algorithme TRuleGrowth avec Minsup=0.07 , Minconf =0.08												
Paramètres/ Donnée	**Nombre des règles Obtenues**				**Espace Mémoire consommé en (mb)**				**Temps d'exécutionen (ms)**			
	T.F=2	**T.F=4**	**T.F=6**	**T.F=8**	**T.F=2**	**T.F=4**	**T.F=6**	**T.F=8**	**T.F=2**	**T.F=4**	**T.F=6**	**T.F=8**
BMS2	**21**	**53**	**76**	**99**	**153.952**	**153.93**	**155.035**	**161.969**	**2184**	**2137**	**2356**	**2980**
Kosarak	**5**	**8**	**10**	**10**	**152.669**	**172.848**	**172.063**	**165.207**	**234**	**266**	**281**	**296**
SIGNE	**69**	**241**	**586**	**1214**	**175.73**	**181.921**	**174.445**	**182.559**	**687**	**1170**	**2075**	**3603**

Tableau 4.3: Résultats obtenus après l'exécution de TRuleGrowth avec Minsup= 0.07, Minconf =0.08

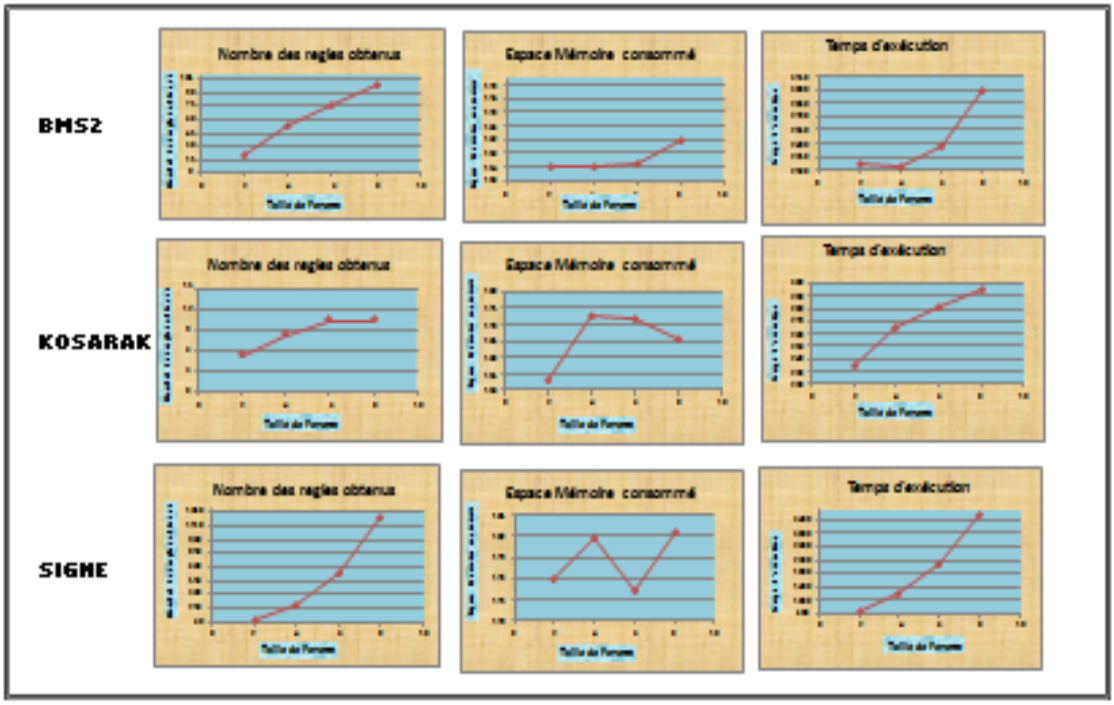

***Figure 4.16**: Résultats obtenus pour Taille de Fenêtre =2, 4, 6 et 8 pour chaque fichier de donnée*

D'après les graphes affiche on peut remarquer :

1/ Comme la valeur de la Taille de la fenêtre augmente, le nombre des règles découvertes et le temps d'exécution augmente également, car le fait que, à très longue période des nombreuses règles peuvent se produire, durer plus longtemps d'exécution, et lorsque la valeur de la Taille de la fenêtre diminue, les règles détectées et le temps d'exécution diminuent également. Donc La relation entre la taille de la fenêtre d'un part, et le nombre des règles détectées et le temps d'exécution d'autre part est proportionnelle

2/ Pour l'espace de mémoire consommée, nous remarquons qu'elle est liée à la taille de la fenêtre mais elle varie aussi en fonction de la nature de l'ensemble de données utilisé

4 .7.4/ Expérience pour évaluer l'évolutivité de l'algorithme :

La quatrième expérience évalue l'évolutivité de TRuleGrowth par rapport au nombre de séquences | S |. Pour cette expérience, l'ensemble de données original de Kosarak a été utilisé car il est un très grand jeu de données contenant 990 000 séquences,

Pour l'expérience, l'algorithme ont été exécutés avec Minsup = 0.003, Minconf = 0.5 et Taille de Fenêtre a été défini à 10, alors que | S | a été varié des plusieurs valeurs.

Les résultats de l'expérience sont montrés sur le tableau suivant :

Algorithme TruleGrowth avec Minsup = 0.003 , Minconf = 0.5 , window_size =10			
Donnée (nombre des séquences)	**Nombre des règles obtenues**	**Espace Mémoire consommé en (mb)**	**Temps d'exécution en (ms)**
Kosarak10 (10000 seq)	**211**	**105,387382**	**8112**
Kosarak25 (25000 seq)	**217**	**157,2616**	**20810**
Kosarak (990 000 seq)	**215**	**226,1873931**	**255308**

Tableau 4.4:Résultats obtenus après l'exécution TRuleGrowth avec diffèrent Taille de base de séquences

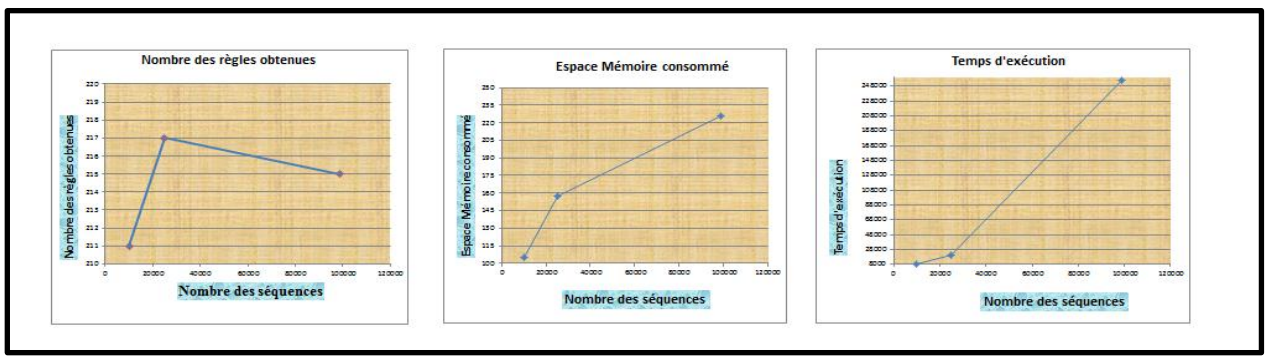

Figure 4.17: Résultats obtenus pour différent Taille de fichier des données

↳ A partir le graphe , on peut observer :

L'espace mémoire consommé et le Temps d'exécution croît linéairement avec la taille de | S |, Mais le nombre des règles séquentielles générées n'augmente pas toujours parce qu'il varie en fonction de la nature et les caractéristiques de l'ensemble des données réelles utilisé.

4.7.5/ Format de Fichier de Sortie :

Après chaque exécution de l'algorithme, des fichiers de sortie seront générées, on va voire un exemple de fichier de sortie ci-après :

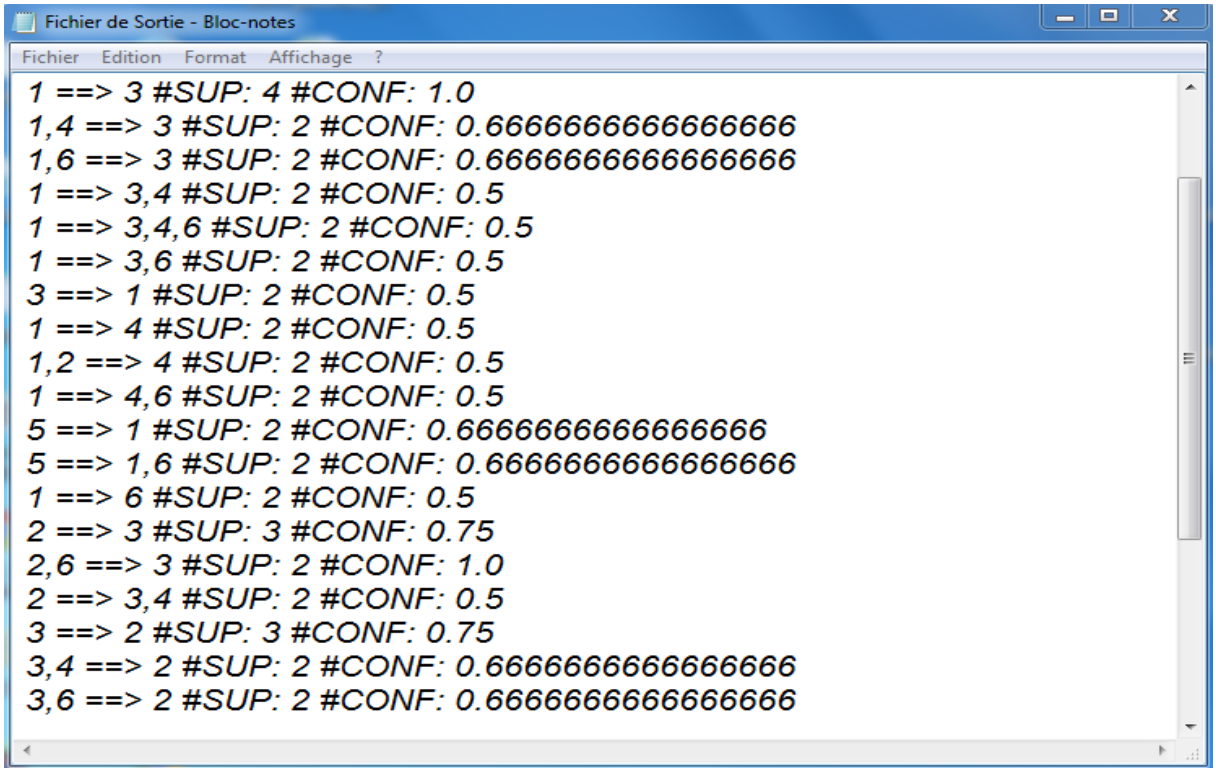

Figure 4.18: Fichier de Sortie

Ce fichier présent les différentes règles générées après l'exécution de l'algorithme approprié, il définit comme un fichier texte. Chaque ligne est une règle séquentielle contient trois parties essentielles :

- ❖ règle séquentielle qui contient une antécédent (partie gauche) et un conséquent (partie droite)
- ❖ Le support de la règle séquentielle
- ❖ La confiance de la règle séquentielle

4.8 /Comparison entre RuleGrowth et TruleGrowth :

La performance des deux algorithmes **RuleGrowth** et **TRuleGrowth** est analysée par effectuer des expériences sur l'ensemble des données réelles de Kosarak .

Des expériences ont été effectuées pour évaluer l'effet de Minsup et Taille de Fenêtre sur le nombre des règles générées, le temps d'exécution et la mémoire consommé, en exécutant RuleGrowth et TRuleGrowth avec différentes valeurs de Minsup et de Taille de Fenêtre.

Les résultats sont montrés sur le tableau suivant :

AlgorithmeTRuleGrowthet RuleGrowth avec Minconf=0.08									
Paramètres/ Algorithmes	**Nombre des règles obtenu**			**Espace Mémoire consommé en (mb)**			**Temps d'exécution en (ms)**		
	Minsup = 0 .02	**Minsup = 0 .04**	**Minsup = 0 .06**	**Minsup = 0 .02**	**Minsup = 0 .04**	**Minsup = 0 .06**	**Minsup = 0 .02**	**Minsup = 0 .04**	**Minsup = 0 .06**
RuleGrowth	**117**	**37**	**16**	**49,051**	**55,802**	**40,139**	**683**	**180**	**145**
TRuleGrowth T.F=5	**32**	**17**	**10**	**62,564**	**82,392**	**89,543**	**789**	**363**	**442**
TRuleGrowth T.F=10	**50**	**22**	**12**	**84,582**	**90,980**	**81,448**	**735**	**541**	**329**
TRuleGrowth T.F=15	**66**	**25**	**15**	**64,282**	**83,685**	**64,561**	**1141**	**458**	**449**

Tableau 4.5: Résultats obtenus après l'exécution RuleGrowth et TruleGrowth

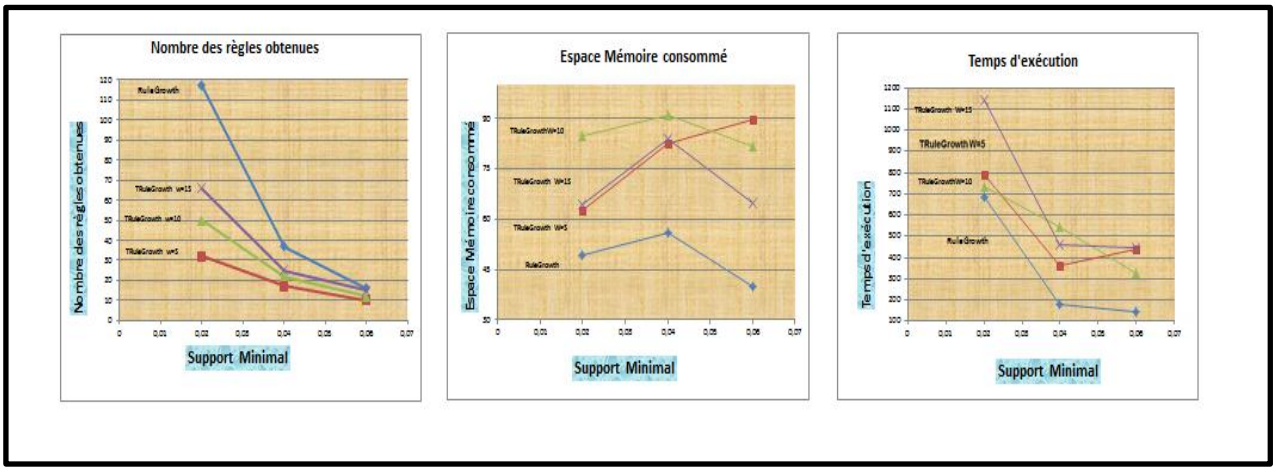

Figure 4.19: Résultats d'exécution obtenus pour les deux algorithmes

D'après ces résultats on peut dire le suivant :

- ❖ Nous avons observé que TRuleGrowth peut être plus lent que RuleGrowth. C'est parce que TRuleGrowtha effectuer des calculs supplémentaires qu'est plus coûteux en termes de temps d'exécution pour vérifier la contrainte de Taille de la Fenêtre.
- ❖ Même si TRuleGrowth prend plus de temps que RuleGrowth pour un grand Taille de la Fenêtre, il génère beaucoup moins des règles, parce que il extraire seulement les règles qu'est respect la contrainte de Taille de la Fenêtre.
- ❖ Pour le consommation de la mémoire, RuleGrowth généralement n'utilise légèrement moins de mémoire que TRuleGrowth , parce que ce dernier garde la trace de toutes les occurrences de chaque item fréquent au lieu de seulement les premières et dernières occurrences .

Globalement, en l'absence de contrainte de taille de fenêtre, RuleGrowth génère plus de nombre des règles séquentielles que l'algorithme TRuleGrowth. Mais ce dernier est fonctionné mieux que l'algorithme RuleGrowth en termes de qualité des règles séquentielles générées

4.9/ Conclusion :

Dans ce chapitre, nous avons vu les différents outils utilisés pour implémenter notre application, ainsi que les interfaces d'application, puis nous avons présenté tous les résultats obtenus après plusieurs d'exécution de l'algorithme étudié.

CONCLUSION GENERALE

CONCLUSION GÉNÉRALE

Au terme de ce projet de fin d'étude, Nous pouvons conclure que la génération des règles séquentielles avec la contrainte de taille de fenêtre est une tâche difficile pour obtenir des résultats raisonnable et facilement exploitable. En effet, les méthodes de génération des règles séquentielles nécessitent une optimisation pour diminuer le temps d'exécution ainsi la consommation de l'espace mémoire.

Plusieurs facteurs influencent les résultats obtenus :

- ❖ Le choix des paramètres utilisés comme le support minimal et la confiance minimale.
- ❖ Le choix de la taille de fenêtre.
- ❖ Les caractéristiques de la base de séquences utilisées en entrée.

Dans notre travail, nous avons étudié les performances de l'algorithme **TRuleGrowth,** qui est un algorithme d'extraction des règles séquentielles partiellement ordonnées, en fonction de plusieurs paramètres, tel que le support minimal, la confiance minimale et la taille de fenêtre.

En plus, nous avons comparé les performances des deux algorithmes **RuleGrowth** et **TRuleGrowth** en fonction de différentes valeurs de taille de fenêtre, pour évaluer les avantages et les inconvénients de l'utilisation de cette contrainte et leur effet sur le nombre des règles générées.

Et comme perspective, nous proposons d'étendre ce travail à d'autres algorithmes de génération des règles séquentielles. Nous proposons aussi d'améliorer notre application par :

- ↳ Améliorer la conception et la structure de stockage pour permettre une sauvegarde et une récupération efficaces des règles séquentielles générées.
- ↳ Réaliser un module d'analyse qui permet de comparer et d'interpréter les résultats obtenus de façon automatique, pour une exploitation simple et efficace par l'utilisateur.
- ↳ Faire des optimisations dans la recherche des règles séquentielles.

Bibliographie

Un mémoire ou une thèse :

[1]: Alice-Maria MARASCU, Titre : Extraction des motifs séquentiels dans les flux de données, thèse de doctorat en Informatique, Université de Nice-Sophia Antipolis Ecole Doctorale STIC (Sciences et Technologies de l'Information et de la Communication) - 2009.

[3] : Hassan Saneifar, Titre : Clustering de motifs séquentiels (Application à la détection d'intrusions), Mémoire de stage de Master 2, Recherche en Informatique Université Montpellier II - (LIRMM) - Juillet 2008 ·

[7]: MAMI Mohammed Nassim, Titre: Extraction des connaissances dans un environnement distribué : synthèse, Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique Option: Système d'Information et de Connaissance (S.I.C), Université Abou Bakr Belkaid – Tlemcen, Faculté des Sciences, Département d'Informatique 2012-2013.

[8] : Mr. MOUNA Azzeddine, Titre : Datamining distribue dans les grilles : approche règles d'association, Mémoire de magister en Informatique, Université des sciences et technologie d'Oran, 2012/2013.

[9] : Mr. ALLIA Mohamed Rachid, Mlle. BOUADI Tassadit , Titre : Fouille de données : Règles séquentielles, Mémoire en Informatique, Université Montpellier II.

[13] : BENYOUNES Sami, Titre : Génération des Règles d'Association, Mémoire de fin d'étude de Master, Filière : Informatique, Spécialité : Technologie de l'Information et de Communication, Université Mohamed Boudiaf, M'sila - 2015/2016.

[18] Aymen HEDIDAR, Titre : CONCEPTION ET REALISATION D'UNE APPLICATION MOBILE M-BANKING, Université virtuelle de Tunis, 2011/2012.

Articles :

[2]: Fayyad, U., Piatetsky- shapiro, G. & Smyth, P. (1996a). From data mining to knowledge discovery in databases. AI Magazine, 17, 3754.

[4]: Friedman, J.H. (1997). Data mining and statistics: What's the connection?

[5]: Cios, K.J., Pedrycz, W. & Swiniarski, R.W. (1998). Data mining methods for knowledge discovery. IEEE Transactions on Neural Networks, 9, 15331534.

[6]: Zaïane,O.R. (1999). Cours , principles of knowledge discovery in databases.111.

[10]: Rakesh Agrawal and Jhon Shafer. Parallel mining of association rules: Design, implementation and experience. Research Report RJ 10004, IBM Almaden Research Center, San Jose, California, February 1996.

[11] : R. Agrawal, T. Imielinski, et A. Swami (1993). Mining Association Rules between sets of Items in Large Databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 207-216, Washington D.C., May 1993.

[12] : Nicolas Pasquier, Extraction de Bases pour les Règles d'Association à partir des Itemsets Fermés Fréquents, Laboratoire d'Informatique (LIMOS) - Université Clermont-Ferrand II.

[14] : Philippe Fournier-Viger , Une introduction à l'exploitation des règles séquentielles, Publié le 18.08.2015 .

[15]: Philippe Fournier-Viger , Cheng-Wei Wu, Vincent S. Tseng, Longbing Cao and Roger Nkambou , Mining Partially-Ordered Sequential Rules Common to Multiple Sequences, Publié le 2015 .

[16]: Mr. Sandipkumar C. Sagare, Prof. Dr. S. K. Shirgave , SPOSR: A System for Mining Partially-Ordered Sequential Rules , Publié le 04/2017 .

Site web:

[17]:http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php, consulté le : 05/04/2018

[19]:https://www.techopedia.com/definition/17374/one-tier-architecture, consulté le : 03/05/2018.

Résumé :

Ce travail s'inscrit dans le domaine de la fouille de données, qui est une étape importante du processus d'Extraction des Connaissances à partir de Données (ECD). Parmi les techniques de la fouille de données figure l'extraction des règles séquentielles. Dans ce contexte, divers algorithmes sont conçus pour la découverte des règles séquentielles communes à plusieurs séquences, mais les règles générées par ces algorithmes sont très spécifiques et beaucoup similaires, autrement dit, ces règles peuvent représenter la même connaissance. Pour résoudre ce problème, une autre approche est explorée, consiste à extraire les Règles Séquentielles Partiellement Ordonnées (POSR). C'est une forme plus générale des règles séquentielles, telles que les éléments de l'antécédent et les éléments du conséquent de chaque règle ne sont pas ordonnées.

Dans ce travail, nous avons étudié les algorithmes RuleGrowth et TRuleGrowth pour la génération des règles séquentielles (POSR), en se basant sur des données réels, afin d'évaluer leurs performances.

Mots clés : *Fouille de données, motifs séquentiels, règles séquentielles, TRuleGrowth.*

Abstract:

This work is part of the field of data mining, which is an important step in the process of Knowledge Extraction from Data (ECD). Among the techniques of data mining is the extraction of sequential rules. In this context, various algorithms are designed for the discovery of sequential rules common to several sequences, but the rules generated by these algorithms are very specific and very similar, in other words, these rules can represent the same knowledge. To solve this problem, another approach is explored, is to extract the Partially Ordered Sequential Rules (POSR). It is a more general form of sequential rules, such that the elements of the antecedent and the consequent elements of each rule are not ordered.

In this work, we studied the RuleGrowth and TRuleGrowth algorithms for the generation of sequential rules (POSR), based on real data, to evaluate their performance.

Keywords: *Data mining, sequential patterns, sequential rules, TRuleGrowth*

ملخص :

يندرج هذا العمل في مجال البحث في البيانات، وهو خطوة هامة في عملية استخراج المعرفة من البيانات. ومن بين تقنيات البحث نجد تقنية استخراج القواعد التسلسلية. في هذا السياق، تم تصميم العديد من الخوارزميات لاكتشاف القواعد التسلسلية المشتركة لعدة تتابعات للبيانات، ولكن القواعد التي تستخرجها هذه الخوارزميات محددة ومتشابهة جدًا، وبعبارة أخرى، يمكن لهذه القواعد أن تمثل نفس المعرفة. لحل هذا الإشكال، يتم استكشاف نهج آخر، هو استخراج القواعد التسلسلية المرتبة جزئيا. وهو شكل أكثر عمومية من القواعد التسلسلية، بحيث لا يتم ترتيب عناصر الشرط والنتيجة لكل قاعدة.

في هذا العمل، قمنا بدراسة الخوارزميتين RuleGrowth و TRuleGrowth لاكتشاف القواعد التسلسلية(POSR) ، استناداً إلى بيانات حقيقية، من أجل تقييم أدائها.

كلمات مفتاحية : البحث في *البيانات، الأنماط التسلسلية، القواعد التسلسلية، خوارزمية " TRuleGrowth "*