République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj
Faculté des Mathématiques et d'Informatique
Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme

Master 2 en informatique

Spécialité : Ingénerie de l'informatique décisionnelle

THEME

Étude comparative entre les algorithmes de haute utilité et les algorithmes de représentation concise.

Présenté par :

- BAKOUR MOULOUD

Soutenu publiquement le :06/22/2023

Devant le jury composé de:

Président : Nouioua Mourad

Examinateur : Saifi abdelhamid

Encadreur : Attia abdelouahab

2022/2023

Dédicace

Je dédie ce travail

À ma mère Belkorchia Merzaka qui n’a jamais eu à exprimer son amour.

À ma grand-mère c’est la personne la plus idéale dans ce monde, que je le dédie.

C’est vrai qu’elle n’est pas avec nous pour récolter le fruit de ses sacrifices, mais elle reste toujours la plus présente.

À mon frère Bakour Bidjad et ma sœur Bakour Noussaiba qu’ils m’ont toujours encouragé.

À toute ma famille Belkorchia A tous les personnes que j’ai autant aimées.

Remerciement

Je remercie avant tout, DIEU le tout puissant qui m'a donnée la santé, la volonté, la patience et l'énergie pour réaliser ce travail.

Au seuil de ce travail je tiens à exprimer mes plus vifs remerciements et ma profonde gratitude à Mr Abdelouahab ATTIA et Mr. Mohamed Bensaadi, pour ses encouragements, ses conseils avisés, orientations et critiques sui m'ont permis de mener à bien cette étude. Et je suis très reconnaissant pour la confiance qu'il nous a donnés

Enfin, très nombreuses de gens qui ont participé à la réalisation de notre travail; mes amis, mes proches, qui m'ont toujours soutenu sans cesse et beaucoup d'autres personnes qu'ils m'excusent de ne plus pouvoir les citer.

Résumé

La fouille d'ensembles d'items à haute utilité (High Utility Itemset Mining - HUIM) est un problème important dans le domaine de la fouille de données, qui vise à découvrir des combinaisons d'articles ayant un impact significatif sur une mesure spécifique, telle que les ventes, les profits ou la satisfaction des clients. Avec l'augmentation du volume de données dans le contexte du Big Data, il est essentiel de développer des algorithmes efficaces pour extraire rapidement ces ensembles d'items à haute utilité.

Dans cette étude comparative, nous avons examiné les performances de les algorithmes de HUIM(FHM)et la représentation concise(MinFHM). Ces algorithmes exploitent des stratégies avancées telles que l'EUCS (Estimated Utility CoOccurrence Structure) et l'utilité list pour améliorer l'efficacité de l'extraction des ensembles d'items à haute utilité.

Nous avons réalisé des expériences sur plusieurs ensembles de données de grande taille pour évaluer les performances de FHM et MinFHM en termes de temps d'exécution, de consommation de mémoire et de nombre d'ensembles d'items à haute utilité extraits. Les résultats ont démontré que ces deux algorithmes étaient capables d'extraire efficacement les ensembles d'items à haute utilité, avec des performances remarquables en termes de rapidité et d'utilisation efficace de la mémoire. MinFHM, en particulier, a permis d'obtenir des résultats plus concis en identifiant les ensembles d'items à haute utilité minimaux.

Cette étude comparative fournit des informations précieuses pour les chercheurs et les praticiens du domaine de la fouille de données, en mettant en évidence les avantages des algorithmes FHM et MinFHM pour l'extraction des ensembles d'items à haute utilité. Ces algorithmes peuvent être utilisés dans des applications de Big Data pour analyser de grands ensembles de données et extraire des connaissances précieuses.

Abstract

High Utility Itemset Mining (HUIM) is an important problem in the field of data mining, aiming to discover combinations of items that have a significant impact on a specific measure, such as sales, profits, or customer satisfaction. With the increasing volume of data in the context of Big Data, it is crucial to develop efficient algorithms for quickly extracting these high utility itemsets.

In this comparative study, we examined the performance of two HUIM algorithms: FHM (Fast High Utility Itemset Mining) and MinFHM (Minimal Fast High Utility Itemset Mining). These algorithms leverage advanced strategies such as Estimated Utility CoOccurrence Structure (EUCS) and utility list to enhance the efficiency of high utility itemset extraction.

We conducted experiments on multiple large-scale datasets to evaluate the performance of FHM and MinFHM in terms of execution time, memory consumption, and the number of extracted high utility itemsets. The results demonstrated that both algorithms were capable of efficiently extracting high utility itemsets, with remarkable performance in terms of speed and efficient memory usage. Particularly, MinFHM yielded more concise results by identifying minimal high utility itemsets.

This comparative study provides valuable insights for researchers and practitioners in the field of data mining, highlighting the advantages of FHM and MinFHM algorithms for high utility itemset extraction. These algorithms can be employed in Big Data applications to analyze large datasets and extract valuable knowledge.

ملخص

يعد تعدين مجموعة الأدوات العالية (HUIM) مشكلة مهمة في مجال التنقيب عن البيانات ، حيث يهدف إلى اكتشاف محاذاة العناصر التي لها تأثير كبير على مقياس معين مثل المبيعات والأرباح ورضا العملاء. مع زيادة حجم البيانات في سياق البيانات الضخمة ، من الضروري تطوير خوارزميات فعالة لاستخراج مجموعات من العناصر ذات الأهمية العالية بسرعة.

في هذه الدراسة المقارنة ، قمنا بالتحقيق في أداء خوارزميتين تعدين لمجموعة عناصر عالية الفائدة(FHM) وتمثيل موجز(MinFHM). تستغل هذه الخوارزميات الاستراتيجيات المتقدمة مثل إستراتيجية عد المرافق الفعالة (EUCS) وقائمة المرافق لتعزيز كفاءة استخراج مجموعات العناصر ذات الأهمية العالية.

أجرينا تجارب على مجموعات بيانات ضخمة متعددة لتقييم أداء FHM و MinFHM من حيث وقت التنفيذ ، واستهلاك الذاكرة ، وعدد مجموعات الكائنات ذات الأهمية العالية المستخرجة. وأظهرت النتائج أن الخوارزميتين قادرتان على استخراج مجموعات العناصر ذات الأهمية العالية بكفاءة وبأداء لافت من حيث السرعة والاستخدام الفعال للذاكرة. على وجه الخصوص ، أظهر MinFHM نتائج أكثر اتساقًا من خلال تحديد مجموعات العناصر ذات الحد الأدنى من الفائدة.

توفر هذه الدراسة المقارنة رؤى قيمة للباحثين والممارسين في مجال التنقيب عن البيانات ، وتسليط الضوء على مزايا خوارزميات FHM و MinFHM في استخراج مجموعات من العناصر ذات الأهمية العالية. يمكن استخدام هذه الخوارزميات في تطبيقات البيانات الضخمة لتحليل مجموعات البيانات الكبيرة واستخراج المعرفة القيمة.

Table des matières

Liste des abréviations

FHM: Faster High-utility Itemset Mining.

HUIM: High-Utility Itemset Mining.

HUI: High-Utility Itemset.

HUI-Miner: High-Utility Itemset- Miner.

CHUI-Miner:Closed High-Utility Itemset- Miner.

GHUI-Miner:Generater High-Utility Itemset- Miner.

IDE: Integrated Development Environment.

JDK : Java Development Kit.

FIM: Frequent Itemset Mining.

EUCS: Estimated Utility CoOccurrence Structure.

TWU: The transaction-weighted utilization

TU: The transaction utility.

Liste des figures

Liste des tableaux

Liste des Algorithmes

Introduction Général

Introduction Générale

L'exploitation des données et la découverte de connaissances à partir de celles-ci sont devenues essentielles dans de nombreux domaines d'activité. La fouille de données, également connue sous le nom de Data Mining, est une discipline qui combine des techniques statistiques, d'apprentissage automatique et de gestion de bases de données pour extraire des informations précieuses à partir de vastes ensembles de données. Elle permet de découvrir des modèles, des tendances et des relations cachées, ouvrant ainsi de nouvelles perspectives pour la prise de décisions éclairées.

Dans ce mémoire, nous nous intéressons plus particulièrement à la fouille d'ensembles d'items à haute utilité (High Utility Itemset Mining - HUIM). L'objectif de cette technique est d'identifier des ensembles d'items qui ont un impact significatif sur une mesure spécifique, telle que les ventes, les profits ou la satisfaction des clients. Avec la croissance exponentielle du volume de données dans le contexte du Big Data, il est devenu primordial de développer des algorithmes efficaces pour extraire rapidement ces ensembles d'items à haute utilité.

Le mémoire est divisé en trois chapitres principaux. Le premier chapitre fournit un aperçu général sur la fouille de données, en expliquant les étapes du processus d'extraction de connaissances à partir de données, les tâches de Data Mining, ainsi que les domaines d'application de cette discipline. Ce chapitre pose les bases nécessaires pour comprendre le contexte dans lequel s'inscrit la fouille d'ensembles d'items à haute utilité.

Le deuxième chapitre se concentre sur le High Utility Itemset Mining (HUIM). Il commence par présenter les concepts fondamentaux de l'extraction des ensembles d'items fréquents, en mettant en évidence les défis spécifiques de cette tâche. Ensuite, il explore les algorithmes utilisés pour extraire les ensembles d'items fréquents, tels que la recherche en largeur et la recherche en profondeur, avec une attention particulière portée à l'algorithme Apriori. Le chapitre se poursuit en introduisant le HUIM, en expliquant les concepts clés tels que l'utilité d'un item, le TWU (Transaction-weighted Utilization) et les principales propriétés de cette technique. Enfin, il présente en détail l'algorithme FHM etMinFHM

Introduction Générale

Le troisième chapitre se concentre sur la réalisation et l'interprétation des résultats obtenus grâce aux algorithmes FHM et MinFHM (Minimal Fast High Utility Itemset Mining). Il commence par présenter les outils de développement utilisés dans le cadre de ce projet, tels que Eclipse, Java, Python, Visual Studio et la bibliothèque SPMF. Ensuite, il décrit l'architecture générale de l'implémentation, en détaillant les différents composants et étapes du processus, du prétraitement des données à la fouille des ensembles d'items à haute utilité, jusqu'à l'interprétation et la comparaison des résultats. Le chapitre aborde également les représentations concises, la visualisation des résultats et l'application de bureau développée.

Enfin, Nous clôturons notre mémoire par une conclusion générale citant les perspectives et les futures améliorations de notre travail.

Chapitre I:

Aperçu Général sur le Data Mining

Chapitre I:Aperçu Général sur le Data Mining

1 Introduction

Les technologies de Data Mining, grâce aux processus d'intelligence artificielle, permettent de traiter de vastes quantités de données afin d'en extraire des informations cruciales (connaissances) nécessaires à une prise de décision efficace. Une connaissance est définie comme un ensemble de relations (règles, phénomènes, exceptions, tendances...) entre les données.

Le Data Mining est apparu dans les années 1990 grâce à l'intersection de facteurs technologiques, économiques et sociopolitiques. Cette émergence a été favorisée par les progrès technologiques, la croissance des données disponibles et les changements dans l'environnement socio-économique. Les volumes massifs de données sont devenus des mines d'informations stratégiques, offrant des avantages considérables aux décideurs et utilisateurs. Le Data Mining permet d'extraire des connaissances cruciales à partir de ces données, facilitant ainsi une prise de décision éclairée et l'optimisation des processus.

Le Data Mining occupe une place centrale dans le processus de découverte de connaissances à partir de données[1] [2] (ECD). Il est considéré comme le moteur de l'ECD, qui vise à extraire des connaissances à partir des données [1]. Le Data Mining utilise des méthodes statistiques, mathématiques et de reconnaissance de formes pour découvrir des règles, des relations, des corrélations et des dépendances au sein d'un grand volume de données. C'est un domaine de recherche essentiel pour extraire des informations utiles à partir de vastes ensembles de données.

Dans ce chapitre, Nous allons donner les concepts fondamentaux du processus d'extractions des connaissances à partir des données (ECD)et fouille de données. Nous débutons en définissant les concepts fondamentaux de la fouille de données, puis nous présentons les tâches et techniques associées à cette discipline.

2 L'extraction des connaissances à partir de donnée

2.1 Définition

« Knowledge Discovery in Databases (KDD) en English » englobe l'ensemble du processus de découverte d'informations ou de connaissances à partir des bases de données. Cela inclut toutes les opérations nécessaires pour extraire de l'information à partir de ces données.[3]

2.2 Etapes du processus ECD

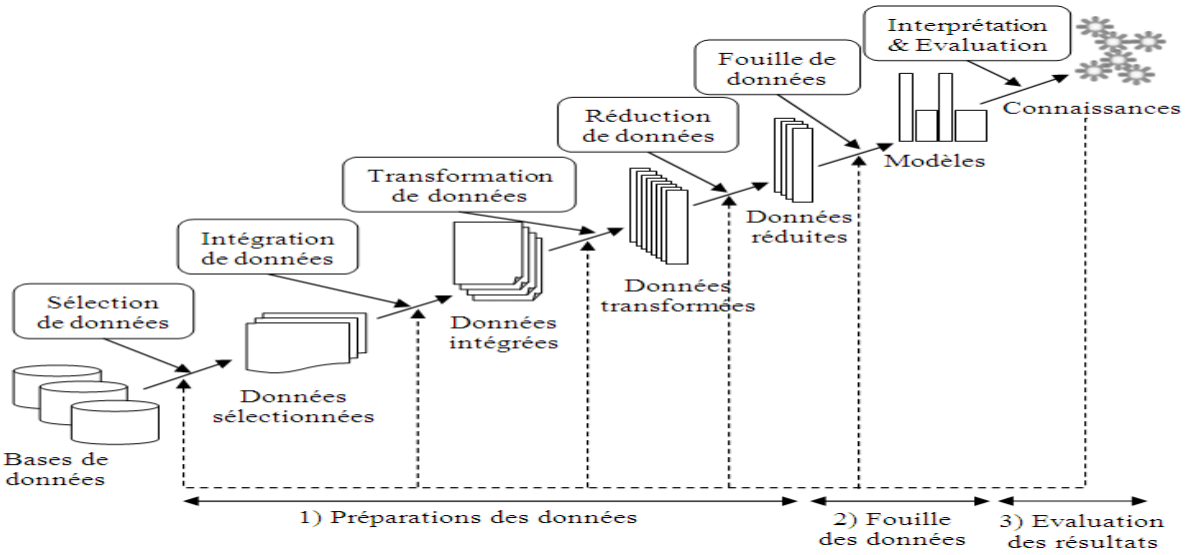


FIGURE 1: ETAPES DU PROCESSUS ECD[4]

L'ECD est un processus interactif et itératif qui vise à extraire des connaissances à partir de données. Il se compose de phases telles que la préparation des données, l'exploration des données et l'analyse des résultats .L'utilisateur peut interagir avec le processus à tout moment, et le processus est ajusté en fonction des résultats obtenus. L'ECD utilise des techniques statistiques, d'apprentissage automatique, de reconnaissance de motifs et de visualisation pour découvrir des informations utiles. (Fayyad et al., 1996)

2.2.1 Collecte et sélection des données

Au cours de cette étape, on procède à la collecte des données qui sont habituellement stockées dans des bases de données ou des entrepôts de données. Les données collectées initialement ne font pas l'objet d'une sélection préalable et nécessitent une étape de tri pour ne conserver que les données pertinentes.

Pour réaliser une sélection optimale des données, il est souvent nécessaire de faire appel à des experts du domaine qui peuvent identifier les facteurs ou variables les plus pertinents pour décrire la problématique. Toutefois, en cas d'indisponibilité d'experts, on peut recourir à des techniques d'analyse (telles que la régression ou les réseaux de neurones ...) pour rechercher les variables les plus déterminantes. Ces techniques permettent d'identifier les variables qui ont le plus d'influence sur l'analyse des données.

2.2.2 Nettoyage de données

Dans cette étape, appelée le nettoyage de données, on s'occupe de vérifier, corriger et filtrer les données collectées afin de garantir leur qualité et leur intégrité. Le nettoyage de données consiste à traiter les erreurs, les valeurs manquantes, les doublons et les valeurs aberrantes qui peuvent affecter la fiabilité des résultats obtenus lors de l'analyse ultérieure. Il est également important de normaliser et de standardiser les données pour assurer leur cohérence et faciliter leur comparaison. Le nettoyage de données vise à obtenir des données propres et fiables pour garantir la validité des résultats lors de l'extraction de connaissances à partir de ces données.

2.2.3 Actions sur les attributs

Des actions peuvent également être entreprises sur les attributs des données dans le cadre du processus de nettoyage et de préparation. Ces actions visent à améliorer la qualité et la pertinence des attributs pour les analyses ultérieures. Parmi les actions courantes sur les attributs, on retrouve :

a) La normalisation

Pour mettre les valeurs des attributs sur une même échelle et éviter les biais liés aux unités de mesure différentes.

b) La discrétisation

Pour regrouper les valeurs continues des attributs en catégories ou intervalles, simplifiant ainsi l'analyse et réduisant la complexité.

c) La réduction de dimension

Pour sélectionner les attributs les plus informatifs et éliminer ceux qui sont redondants ou peu pertinents.

d) La création de nouveaux attributs

En combinant ou dérivant des attributs existants afin d'obtenir de nouvelles informations utiles.

e) La suppression d'attributs

Lorsque certains attributs ne sont pas pertinents pour l'analyse ou se révèlent redondants avec d'autres attributs.

Ces actions permettent d'améliorer la qualité des attributs et de préparer les données de manière plus adaptée pour les étapes suivantes de l'analyse

2.2.4 Construction du modèle (Data Mining)

Dans la phase de fouille de données, des méthodes intelligentes sont utilisées pour extraire des modèles et des motifs à partir des données. Cette étape est au cœur du processus d'extraction de connaissances à partir des données (ECD). Cependant, il est essentiel de souligner que la qualité des modèles extraits et leur coût d'extraction dépendent étroitement des étapes de préparation des données. Une préparation minutieuse des données permet d'obtenir des résultats plus pertinents et fiables lors de la fouille de données.

Cela signifie que pour des informations plus approfondies sur le sujet mentionné, il est recommandé de consulter la ***(Fouille de données (Data Mining)).***

2.2.5 Evaluation et interprétation des connaissances

L'évaluation et l'interprétation des connaissances font partie des étapes finales du processus d'extraction de connaissances à partir des données. Cette étape consiste à évaluer la qualité des modèles ou des motifs extraits, ainsi qu'à interpréter les résultats obtenus.

L'évaluation des connaissances implique généralement l'utilisation de mesures et de techniques pour évaluer la précision, la fiabilité et la pertinence des modèles ou des motifs extraits. Cela permet de déterminer dans quelle mesure les connaissances découvertes sont utiles et fiables pour prendre des décisions.

L'interprétation des connaissances consiste à analyser et à comprendre les modèles ou les motifs extraits dans le contexte du problème ou du domaine d'application. Cela implique souvent de faire des analyses statistiques, de visualiser les résultats et de les interpréter en utilisant des connaissances préexistantes ou des expertises domaines.

Chapitre I:Aperçu Général sur le Data Mining

3 Fouille de données (Data Mining)

Effectivement, les données brutes, même si leur quantité augmente de manière exponentielle, ont peu de valeur intrinsèque. Ce qui est réellement précieux, ce sont les connaissances que nous pouvons en tirer par le biais de la compréhension et de l'analyse de ces données. Cependant, plus le volume de données est important, plus ce processus de compréhension devient complexe.

De nos jours, avec la prolifération des capteurs, les changements dans notre environnement sont enregistrés de manière plus détaillée et à une échelle plus vaste que jamais auparavant. Par conséquent, il est devenu crucial de comprendre ces données pour en tirer des informations significatives.

Comme l'a souligné G. Piatetsky-Shapiro **«** […]as long as the world keeps producing data of all kinds […] at an ever increasing rate, the demand fordata mining will continue to grow**»**[4].La fouille de données est donc devenue une nécessité pour exploiter le plein potentiel des données disponibles et en extraire des connaissances exploitables.

3.1 Définition

Fouille de données ou …

- KDD (Knowledge Discovery in Databases).
- Le Data mining (term English).
- Analyse de données/patterns, business intelligence.
- Extraction automatique de connaissances à partir de données (ECD).

Le Data Mining désigne l'analyse de vastes ensembles de données observationnelles dans le but d'identifier des relations inattendues et de résumer les données de manière novatrice, de manière à ce qu'elles soient compréhensibles et utiles pour le propriétaire des données.[5]

Ils existent d’autres définitions :

- Frawley et Al ont proposé la définition suivante en 1992 : « L'extraction de connaissance à partir des données (ECD) est l'extraction non triviale, à partir des données, d'une information implicite, inconnue auparavant et potentiellement utile. »[6]
- D'après Anand et Buchner le Data Mining offre des algorithmes et des outils pour la découverte de modèles non triviaux, implicites, non connus, potentiellement utiles et compréhensibles à partir d'une grande masse de données.[7]

3.2 Les tâches de Data Mining

Beaucoup de problèmes intellectuels, économiques ou même commerciaux peuvent être exprimés en termes des six tâches suivantes :

Les trois premières tâches sont des exemples du Data Mining supervisé dont le but est d'utiliser les données disponibles pour créer un modèle décrivant une variable particulière prise comme but en termes de ces données. [8]

Le groupement par similitude (ou clustering) et l'analyse des clusters sont des tâches non-supervisées où le but est d'établir un certain rapport entre toutes les variables.

La description appartient à ces deux catégories de tâche, elle est vue comme une tâche supervisée et non-supervisée en même temps.[9]

3.2.1 Classification

Il s'agit de regrouper des éléments dans des catégories prédéfinies en fonction de leurs caractéristiques communes. Par exemple, classer les courriels en spam ou en non-spam.[9]

3.2.2 Estimation

Cela implique d'estimer une valeur numérique en se basant sur des données existantes. Par exemple, estimer le coût d'un projet en fonction de ses spécifications.[10] [9]

3.2.3 Prédiction

Il s'agit de prédire un événement futur ou une valeur en utilisant des modèles statistiques ou d'apprentissage automatique. Par exemple, prédire les ventes d'un produit pour le prochain trimestre.[9]

3.2.4 Groupement par similitude (ou clustering)

Cela consiste à regrouper des éléments similaires sans catégories prédéfinies. Par exemple, regrouper des clients en fonction de leurs habitudes d'achat similaires.[10]

3.2.5 Analyse des clusters

Une fois que les éléments ont été regroupés, cette tâche consiste à analyser les caractéristiques communes des éléments au sein de chaque groupe. Par exemple, analyser les caractéristiques démographiques des différents segments de clients.[10]

3.2.6 Description

Cette tâche consiste à fournir une description ou une interprétation des modèles ou des résultats obtenus. Par exemple, décrire les caractéristiques clés des segments de clients identifiés.[9]

3.2.7 Pattern mining

Le "pattern mining" (ou extraction de motifs) est une tâche spécifique du data mining qui vise à découvrir des motifs fréquents, des règles d'association ou des séquences significatives dans un ensemble de données. Cette tâche est souvent utilisée pour identifier des schémas intéressants ou des tendances cachées qui peuvent être exploités pour prendre des décisions éclairées.

3.3 Domaines d'application du Data Mining

TABLE 1:DOMAINES D'APPLICATION DU DATA MINING

Secteurs d'activité	Exemples d'applications
Bancaire	• Prédire comment les clients réagiront aux variations des taux d'intérêt.
La détection de fraude	• Détection de fraude de cartes de crédits. • Détection de fausses demandes de remboursement médicale.
Assurances	• Formulation des modèles statistiques des risques d'assurance. • Utilisation du modèle de l'exploitation de Poisson / Log-normale
La télécommunication	• Prédiction de modèles d'appels téléphoniques. • Gestion des ressources et de trafic réseau.

4 Conclusion

Le Data Mining consiste à extraire des informations prédictives qui se trouvent dans de vastes bases de données. Cette technologie moderne et puissante permet aux entreprises de se concentrer sur les informations les plus pertinentes stockées dans leurs entrepôts de données. Les outils de Data Mining sont capables de prédire les tendances et les actions futures, ce qui facilite la prise de décisions éclairées. C'est pourquoi le Data Mining est considéré comme une technologie essentielle et précieuse.

Dans ce chapitre introductif, nous avons présenté les concepts clés du Data Mining, qui est une étape essentielle du processus ECD (Extraction des Connaissances à partir des Données). Ce processus a émergé afin d'intégrer et de compléter le Data Mining, permettant ainsi d'automatiser au maximum le traitement des données.

Nous avons exploré les techniques de Data Mining et introduit les principaux algorithmes couramment utilisés pour accomplir certaines des tâches essentielles de la fouille de données, telles que le clustering, la classification et la recherche d'associations.

Dans the prochain chapitre, nous examinerons de manière approfondie les algorithmes utilisés dans le High Utility Itemset Mining (HUIM). Nous étudierons en détail les techniques utilisées pour réduire l'espace de recherche et les différentes méthodes et algorithmes de structuration des données appliqués dans le HUIM. Ces techniques incluent la génération de candidats, la sélection basée sur des seuils d'utilité et l'utilisation de structures de données spécialisées visant à accélérer le processus d'extracti

Chapitre II:
High Utility Itemset Mining HUIM

Chapitre II: High Utility Itemset Mining HUM

1 Introduction

L'exploration de données a révolutionné notre façon d'extraire des connaissances précieuses à partir de vastes ensembles de données. L'une des tâches fondamentales de l'exploration de données est l'extraction des itemsets fréquents, qui consiste à identifier des itemsets qui co-occurrent fréquemment dans une base de données transactionnelle. L'algorithme d'extraction d'itemsets fréquents d'Agrawal, introduit en 1993, a marqué une étape importante dans ce domaine et a été largement utilisé pour découvrir ces itemsets. Cependant, malgré son efficacité, l'extraction d'itemsets fréquents présente des limitations inhérentes qui ont suscité l'exploration d'approches alternatives.[11]

Dans ce chapitre, nous plongerons dans le concept de l'extraction d'itemsets à utilité haut comme une alternative puissante à l'extraction d'itemsets fréquents. Nous commencerons par retracer l'historique de l'algorithme d'extraction d'itemsets fréquents d'Agrawal et ses contributions au domaine. Nous discuterons de son importance dans l'établissement des fondements de l'extraction d'itemsets et son impact sur les recherches ultérieures.[12]

Cependant, à mesure que l'extraction d'itemsets fréquents est devenue plus répandue, les chercheurs ont reconnu ses limitations. Un inconvénient majeur est que l'extraction d'itemsets fréquents ne prend en compte que la fréquence d'occurrence ou le support des itemsets, négligeant l'utilité ou la valeur associée à chaque item. Par exemple, dans un ensemble de données de vente au détail, certains articles peuvent contribuer davantage au chiffre d'affaires ou à la rentabilité globale que d'autres. L'extraction d'itemsets fréquents ne capture pas ces aspects d'utilité, limitant sa capacité à fournir des informations pertinentes pour la prise de décision.[13]

Pour remédier à ces limitations, l'extraction d'itemsets à utilité haut est apparue comme une approche alternative. L'extraction d'itemsets à utilité haut va au-delà de la notion de support et prend en compte l'utilité des items et des itemsets individuels. En incorporant des mesures d'utilité, telles que le profit ou le chiffre d'affaires, l'extraction d'itemsets à utilité haut permet de découvrir des itemsets qui contribuent de manière significative aux objectifs souhaités.

Chapitre II: High Utility Itemset Mining HUM

Tout au long de ce chapitre, nous explorerons en détail le concept de l'extraction d'itemsets à utilité haut. Nous discuterons des limitations de l'extraction d'itemsets fréquents, en mettant l'accent sur l'importance de prendre en compte l'utilité des items.

En comprenant l'extraction d'itemsets à utilité haut, les chercheurs et les praticiens peuvent exploiter sa puissance pour découvrir des modèles significatifs, améliorer les processus de prise de décision et obtenir un avantage concurrentiel dans divers domaines. Ce chapitre vise à fournir une compréhension approfondie de l'extraction d'itemsets à utilité haut, de ses algorithmes, de ses métriques d'évaluation, de son analyse de performance et de ses extensions potentielles.

2 Frequent Itemset Mining(FIM)

L'extraction d'ensembles d'items fréquents (*FIM*) est une tâche qui consiste à extraire tous les ensembles d'items fréquents présents dans un jeu de données, avec une fréquence d'occurrence supérieure à un seuil spécifié. Cette tâche a été proposée au début des années *1990* dans le cadre de l'analyse des paniers d'achat pour découvrir les items qui co-occurrent fréquemment et était initialement appelée "*minage d'ensembles d'items larges*".[11]

2.1 Concepts fondamentaux

Pour mieux comprendre le *FIM*, définissons quelques termes importants :

2.1.1 Item

Un item fait référence à un élément individuel ou attribut dans une base de données transactionnelle. Par exemple, dans une base de données de vente au détail, les items pourraient représenter des produits spécifiques.[14]

2.1.2 Itemset

Un itemset est une collection d'un ou plusieurs items. Il peut s'agir d'un seul item (itemset singleton) ou d'une combinaison de plusieurs items (itemset multi-items). Par exemple, {A}, {B} et {A, B} sont des itemsets.[14]

2.1.3 Frequent itemset

Un Itemset est fréquent si et seulement si son support est supérieur ou égal à un support minimum défini par l'utilisateur.[15]

2.1.4 Mesure de support

La mesure de support quantifie la fréquence ou l'occurrence d'un itemset dans la base de données transactionnelle. Elle est généralement exprimée en pourcentage ou en fraction.[14]

2.1.5 Bases de données transactionnelles

Les bases de données transactionnelles sont constituées d'une collection de transactions, où chaque transaction représente un ensemble d'items. Les transactions peuvent être considérées comme des enregistrements ou des instances dans la base de données.[14]

TABLE 2:BASES DE DONNEES TRANSACTIONNELLES

TID	Transaction
T1	{a, c, d}
T2	{b, c, e}
T3	{a, b, c, e}
T4	{b, e}
T5	{a, b, c, e}

Ces définitions sont essentielles pour comprendre les concepts fondamentaux de l'extraction des ensembles d'items fréquents. Elles fournissent les bases nécessaires pour définir et identifier les itemsets fréquents dans les transactions, en utilisant des mesures de support pour déterminer leur fréquence. En comprenant ces définitions, on peut explorer plus en détail les algorithmes et techniques spécifiques utilisés pour extraire efficacement les ensembles d'items fréquents à partir des bases de données transactionnelles.

2.2 Défis de l'extraction des ensembles d'items fréquents

2.2.1 Espace de recherche

L'extraction d'ensembles d'items fréquents (*FIM*) est un défi consistant à énumérer tous les motifs qui respectent la contrainte de support minimum fixée par l'utilisateur. Cela en fait un problème complexe d'énumération. Il existe toujours une seule réponse correcte à une tâche de *FIM*. Cependant, *FIM* est une problématique difficile. La méthode naïve pour résoudre ce problème serait de considérer tous les ensembles d'items possibles, puis de ne

retenir que ceux qui satisfont la contrainte de support minimum. Cependant, cette approche naïve est inefficace pour la raison suivante. Si une base de données transactionnelle comporte m items distincts, il existe $2^m - 1$ ensembles d'items possibles. Par exemple, avec *1000* items distincts dans une base de données, le nombre d'ensembles d'items possibles serait de $2^{1000} - 1$, ce qui est clairement ingérable avec une approche naïve. Il est important de noter que le problème de *FIM* peut être très complexe même pour une petite base de données. Par exemple, une base de données avec une seule transaction contenant *100* items, et avec un support minimum de *1*, génère un espace de recherche de 2^{100} ensembles d'items. Ainsi, le nombre d'ensembles d'items dans l'espace de recherche est souvent plus important que la taille des données pour le problème de *FIM*. Plusieurs facteurs influencent le nombre d'ensembles d'items dans l'espace de recherche, tels que la similarité entre les transactions de la base de données et la valeur du seuil de support minimum fixé par l'utilisateur.[15]

2.3 La propriété 1(Propriété d'Apriori monotone)

Supposons qu'il existe deux ensembles d'éléments, X et Y.Si $X \subset Y$, alors le support de Y est inférieur ou égal au support de X.[15]

2.4 Algorithmes d'extraction des items fréquents

Afin de découvrir efficacement les ensembles d'items fréquents, il est essentiel de concevoir des algorithmes qui évitent l'exploration exhaustive de l'ensemble de recherche de tous les ensembles d'items possibles et qui traitent chaque ensemble d'items de manière efficace. Plusieurs algorithmes efficaces ont été proposés pour l'extraction des ensembles d'items fréquents, notamment des algorithmes bien connus tels qu'Apriori, FP-Growth, Eclat et LCM. Bien que ces algorithmes partagent la même entrée et la même sortie, leurs différences résident dans les stratégies et les structures de données qu'ils utilisent pour découvrir efficacement les ensembles d'items fréquents[16]. Plus précisément, les algorithmes d'extraction des ensembles d'items fréquents diffèrent dans les aspects suivants

2.4.1 Stratégie de recherche

Ils peuvent utiliser soit une approche de recherche en profondeur (depth-first search), soit une approche de recherche en largeur (Breadth-first search).

2.4.2 Représentation de la base de données

Ils peuvent utiliser soit une représentation horizontale, soit une représentation verticale de la base de données.

2.4.3 Génération des ensembles d'items

Ils varient en termes de la manière dont ils génèrent ou déterminent les prochains ensembles d'items à explorer dans l'espace de recherche.

2.4.4 Évaluation du support

Ils utilisent différentes approches pour calculer le support des ensembles d'items afin de déterminer s'ils respectent la contrainte de support minimum.

TABLE 3:UN RESUME DES ALGORITHMES FIM[15]

Algorithme	Type de recherche	Représentation de la BD
Apriori	Recherche en largeur	Horizontal
F P − Growth	Recherche en profondeur	Horizontal (prefix-tree)
Eclat	Recherche en profondeur	Vertical (TID-lists, diffsets)
LCM	Recherche en profondeur	Horizontal (with transaction merging)

2.5 Recherche en largeur (Breadth-first search)

Dans un algorithme de recherche en largeur (également appelé algorithme de niveau), tel que l'algorithme Apriori, on suppose qu'il y a m items dans une base de données. Cet algorithme explore l'espace de recherche des ensembles d'items en commençant par les 1-ensembles d'items, puis les 2-ensembles, les 3-ensembles, et ainsi de suite jusqu'aux m-ensembles. Par exemple, la (*Figure 2*)représente l'espace de recherche de tous les ensembles d'items possibles pour l'exemple donné. Dans cette figure, l'espace de recherche est représenté sous la forme d'un diagramme de Hasse.

Un algorithme de recherche en largeur va d'abord considérer les 1-ensembles d'items {a}, {b}, {c}, {d} et {e}. Ensuite, il générera les 2-ensembles d'items tels que {a, b}, {a, c}, {a, d}, puis les 3-ensembles d'items, et ainsi de suite, jusqu'à ce qu'il génère l'ensemble {a, b, c, d, e} qui contient tous les items de la base de données.

2.6 Recherche en profondeur (depth-first search)

Les algorithmes de recherche en profondeur tels que FPGrowth, Eclat et LCM commencent par chaque ensemble d'items de taille 1, puis tentent de manière récursive d'ajouter des items à l'ensemble d'items actuel pour générer des ensembles d'items plus grands. Par exemple, dans l'exemple en cours, un algorithme de recherche en profondeur typique explorerait les ensembles d'items dans l'ordre suivant : {a}, {a, b}, {a, b, c}, {a, b, c, d}, {a, b, c, d, e}, {a, b, c, e}, {a, b, d}, {a, b, d, e}, {a, b, e}, {a, c}, {a, c, d}, {a, c, d, e}, {a, c, e}, {a, d}, {a, d, e}, {a, e}, {b}, {b, c}, {b, c, d}, {b, c, d, e}, {b, c, e}, {b, d}, {b, d, e}, {b, e}, {c}, {c, d}, {c, d, e}, {c, e}, {d}, {d, e}, {e}.

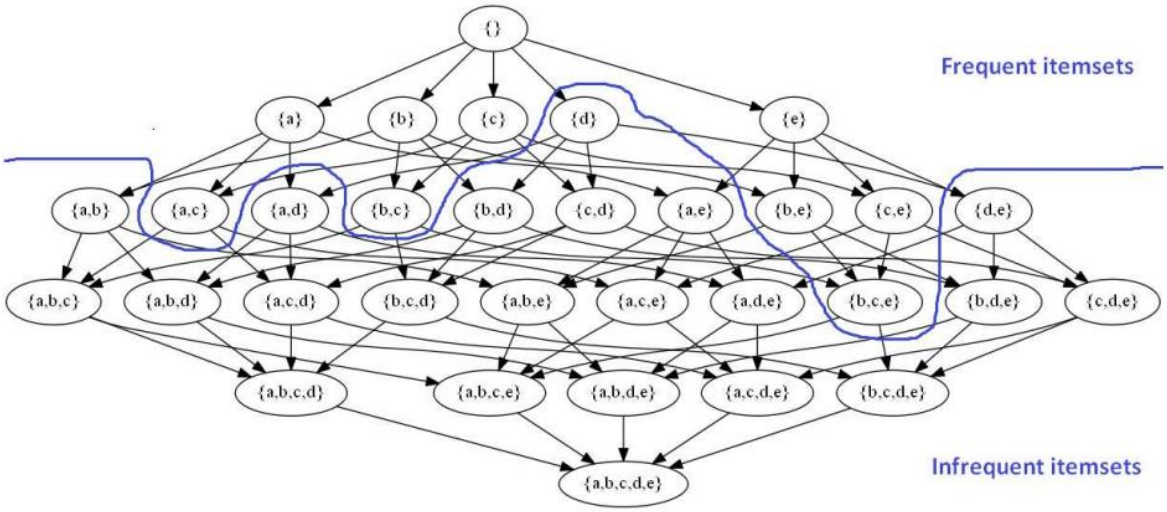


FIGURE 2:L'ESPACE DE RECHERCHE POUR I = {A, B, C, D, E}[15]

2.7 Aperçu de l'algorithme Apriori

L'algorithme Apriori est l'un des algorithmes les plus populaires pour l'extraction des ensembles d'items fréquents[17]. Il utilise des techniques de génération de candidats et d'élagage pour découvrir efficacement les ensembles d'items fréquents dans une base de données transactionnelle(*Error! Reference source not found.*). Voici un aperçu de la procédure de l'algorithme Apriori(*Error! Reference source not found.*):

Algorithm 1: The Apriori Algorithm

Input : D: *a horizontal transaction database,* ***minsup:*** *a user-specified threshold*

Output: *the set of frequent itemsets*

1 *Scan the database to calculate the support of all items in* ***I****;*

2 ***F1*** *=* ***{i|i ∈ I ∧ sup({i}) ≥ minsup}*** *;* *// F1 : frequent 1-itemsets*

3 ***k*** *= 2;*

4 ***while Fk 6 = ∅*** *do*

5 ***Ck*** *= CandidateGeneration (****Fk−1****) ;* *// Ck : candidate k-itemsets*

6 *Remove each candidate* ***X ∈ Ck*** *that contains a* ***(k − 1)-itemset*** *that is not in* ***Fk−1****;*

7 *Scan the database to calculate the support of each candidate* ***X ∈ Ck****;*

8 ***Fk*** *=* ***{X|X ∈ Ck ∧ sup(X) ≥ minsup}*** *;* *// Fk : frequent k-itemsets*

9 ***k = k + 1****;*

10 ***end***

11 ***return*** $\bigcup_{k=1\ldots k} F_K$*;*

ALGORITHM 1:THE APRIORI ALGORITHME

L'algorithme Apriori comprend plusieurs étapes. Tout d'abord, il analyse la base de données de transactions pour calculer le support de chaque item individuel et identifie les items fréquents de *taille 1*. Ensuite, il génère de manière itérative les items de candidats en *joignant* les items fréquents de *taille (k-1)* à partir des itérations précédentes. Cependant, avant de calculer le support de ces candidats, l'algorithme utilise *la propriété d'Apriori* pour éliminer les candidats non fréquents. Cela signifie que si un sous-ensemble d'un candidat n'est pas fréquent, alors ce candidat lui-même ne peut pas être fréquent. Cela réduit l'espace de recherche en éliminant les candidats inutiles. Ensuite, l'algorithme calcule le support de chaque candidat restant en analysant à nouveau la base de données de transactions. Les candidats avec un support supérieur ou égal au seuil minimum sont considérés comme des itemsets fréquents. L'algorithme continue ce processus itératif jusqu'à ce qu'il n'y ait plus itemsets fréquents trouvés. Finalement, il retourne l'union de tous les itemsets fréquents découverts.

2.8 Limitations de Fréquent Itemset Mining

En effet, l'extraction de motifs fréquents est utilisée dans de nombreux domaines, notamment l'analyse des achats, l'étude du comportement des clients, la bioinformatique, l'extraction de données sur le web, et bien d'autres. Elle offre la possibilité de découvrir des motifs pertinents et utiles dans de vastes ensembles de données. Néanmoins, il est essentiel de prendre en considération certaines limitations significatives[16] [18]

2.8.1 Motifs fréquents non pertinents

Une des limitations de l'extraction de motifs fréquents est que bon nombre des motifs découverts peuvent ne pas être intéressants ou ne pas fournir d'informations précieuses. La définition de "fréquent" repose uniquement sur la fréquence d'occurrence dans l'ensemble de données, sans tenir compte de la signification ou de la pertinence des motifs. Par conséquent, des ***motifs triviaux***, prévisibles ou non informatifs peuvent également être identifiés comme fréquents. Par exemple, dans un ensemble de données transactionnelles, des articles courants tels que ***le pain*** ou ***le lait*** peuvent être fréquents, mais leur co-occurrence peut ne révéler aucune association significative. Cette limitation met en évidence la nécessité d'incorporer des mesures d'intérêt ou d'utilité pour filtrer les motifs non intéressants et se concentrer sur ceux qui fournissent réellement des informations précieuses.[18]

2.8.2 Représentation binaire des items

Une autre limitation de l'extraction de motifs fréquents est l'hypothèse d'une représentation binaire des items, où chaque item est soit ***présent (1)*** soit ***absent (0)*** dans une transaction. Cette approche binaire simplifie le processus d'extraction en réduisant la complexité des combinaisons d'ensembles d'items. Cependant, elle ne capture pas les aspects quantitatifs ou les variations des occurrences d'items. Dans de nombreux scénarios du monde réel, la quantité ou la fréquence des items peut contenir des informations importantes. En ne considérant que la présence ou l'absence binaire, des informations précieuses liées aux quantités d'items ou à l'intensité des occurrences sont perdues.[18]

2.8.3 Égalité d'importance des items

Une troisième limitation est l'hypothèse selon laquelle tous les items ont une ***importance*** ou un poids égal. Le processus d'extraction considère tous les items comme également significatifs en termes de leur contribution aux motifs fréquents. Cependant, dans de

nombreuses applications, certains items peuvent avoir une plus grande pertinence ou une plus grande signification. Ignorer ***l'importance*** variable des items peut conduire à la découverte de motifs qui peuvent ne pas être vraiment précieux ou exploitables. L'incorporation de poids ou de ***mesures d'importance*** pour les items peut fournir une compréhension plus nuancée des relations et des associations entre les items dans l'ensemble de données.[18]

Afin de remédier à ces limitations et améliorer l'efficacité et la pertinence de l'extraction de motifs fréquents dans divers domaines, il est nécessaire de développer des techniques et des algorithmes avancés qui dépassent les approches basiques d'extraction de motifs fréquents. Des chercheurs ont proposé différentes méthodes pour relever ces défis, telles que l'incorporation de mesures d'intérêt, l'utilisation d'ensembles d'items pondérés et la prise en compte de l'importance des items. Ces approches visent à filtrer les motifs non pertinents, à tenir compte de la quantité ou de la fréquence des items, ainsi qu'à prendre en considération leur importance relative, afin d'obtenir des résultats plus pertinents et significatifs lors de l'extraction de motifs fréquents. ***Dans la section suivante***, nous aborderons en détail des solutions spécifiques à ces limitations, notamment l'extraction des ensembles d'items à utilité élevée (***High Utility Itemset Mining HUIM***).

3 High Utility Itemset Mining HUIM

L'extraction des ensembles ***d'items à utilité haut*** est une extension de l'extraction des ensembles d'items fréquents qui vise à *surmonter la limitation de considérer uniquement la fréquence des items*. Cette approche permet de découvrir des motifs qui ont *une importance haut* pour les utilisateurs, en prenant en compte des critères tels que *le profit*, *l'utilité* ou *d'autres mesures spécifiques*.

L'objectif de l'extraction des ensembles *d'items à utilité haut* est de trouver des ensembles d'items qui maximisent *une fonction d'utilité* prédéfinie. Cette *fonction d'utilité* peut être définie en fonction de différents facteurs, tels que *le coût*, *le profit*, *la satisfaction* du client, *le temps passé*, ou d'autres considérations spécifiques au domaine d'application.

La découverte des ensembles ***d'items à utilité haut*** est particulièrement pertinente dans des domaines tels que *le marketing*, où l'objectif est d'identifier les ensembles d'items qui peuvent générer ***un profit haut***, ou dans la personnalisation de services, où l'on cherche à recommander des ensembles d'items qui répondent aux besoins spécifiques des utilisateurs.

Cette approche nécessite une modélisation appropriée des données, où chaque item est associé à *une mesure d'utilité* ou de *profit*. Les algorithmes d'extraction des ensembles d'items à utilité haut exploitent ensuite ces informations pour identifier les ensembles d'items les plus pertinents et *les plus rentables*.

L'extraction des ensembles d'items à utilité haut a suscité un intérêt croissant dans la recherche en fouille de données, et de nombreux algorithmes efficaces ont été développés pour résoudre ce problème. Ces algorithmes exploitent des techniques telles que *la génération de candidats, la prune basée sur des seuils d'utilité* et *des structures de données spécialisées* pour accélérer le processus d'extraction.

Cette approche offre de nouvelles perspectives pour découvrir des motifs ***intéressants*** et ***importants*** dans les données, allant au-delà de la simple fréquence des items. Elle permet aux utilisateurs de découvrir des ensembles d'items qui ont un impact significatif sur leurs objectifs et de prendre des décisions plus *éclairées* et *plus rentables*.

Dans la suite de ce chapitre, nous explorerons en détail les différents aspects de l'extraction des ensembles d'items à utilité haut, y compris les différentes techniques et algorithmes utilisés, telles que la génération de candidats, la prune basée sur des seuils d'utilité et des structures de données spécialisées pour accélérer le processus d'extraction.

3.1 Concepts fondamentaux

3.1.1 Base de données quantitative des transactions

Une base de données de transactions quantitatives D est définie par : $D = \{T_0, T_1, ..., T_n\}$, où chaque transaction Tq est un Itemset ($Tq \subseteq I$) avec un identifiant unique q (TID). Chaque item $i \in I$ a une ***utilité externe*** p(i)(***Table 5***) et une ***utilité interne*** q(i, Tc)(***Table 4***) dans la transaction Tc.[18]

TABLE 4:BASE DE DONNEES QUANTITATIVE DES TRANSACTIONS

TID	Transaction
T1	(a, 1), (b, 5), (c, 1), (d, 3), (e, 1)
T2	(b, 4), (c, 3), (d, 3), (e, 1)
T3	(a, 1), (c, 1), (d, 1)
T4	(a, 2), (c, 6), (e, 2)
T5	(b, 2), (c, 2), (e, 1)

TABLE 5:VALEURS D'UTILITE EXTERNE

Item	utilité externe
a	5
b	2
c	1
d	2
e	3

3.1.2 Utilité d'un item

L'utilité d'un item i dans une transaction Tr est notée U(i, Tr) et définie comme étant égale à P(i, D) × Q(i, Tr).[19]

Exemple : L'utilité de l'item a dans T1 est $U(a, T_1)=5$

a) Utilité d'un itemset dans une transaction

L'utilité d'un itemset X dans une transaction Tr est définie comme suit :

$U(X, Tr) = \sum_{I \in X} U(I, Tr)$, Où U(I, Tr) représente l'utilité de l'item I dans la transaction Tr.[19]

Exemple : L'utilité de l'itemset {a,b} a dans T_1 est $U(\{a,b\}, T_1) =15$.

b) Utilité d'un itemset dans une base de données

L'utilité d'un itemset X dans D est définie comme suit :

$U(X, D) = \sum_{T \in D} U(X, T)$,où U(X, T) représente l'utilité de itemset X dans la transaction T.[19]

Exemple : L'utilité de l'itemset {a,c} a dans une base de données est $u(\{a, c\}) = u(a)+u(c) = u(a, T_0)+ u(a, T_2) + u(a, T_3) + u(c, T_0) + u(c, T_2) + u(c, T_3) = 5 + 5 + 10 + 1 + 1 + 6 = 28$.

3.1.3 Itemset à utilité haut

Un itemset X est considéré comme un itemset à utilité haut (high-utility itemset) si son utilité u(X) est supérieure ou égale à un seuil d'utilité minimum spécifié par l'utilisateur (minutil). Sinon, X est considéré comme un itemset à faible utilité (low-utility itemset).

Exemple : Minutil=25.

- U({a, c}) = 28 {a, c} est un itemset à utilité haut .
- U({a, b}) = 15 {a, b} est un itemset à faible utilité.

3.1.4 L'utilité de la transaction

L'utilité de transaction (TU) d'une transaction Tc est la somme de l'utilité des articles de Tc dans Tc, c'est-à-dire TU(Tc) = $\sum_{x \in Tc} u(x, Tc)$.[12]

Exemple : L'utilité de transaction(T_3): TU(T_3) = U(a, T_3)+ U(c, T_3)+ U(d, T_3)= 5+1+2=8.

3.1.5 Transaction-weighted utilization (TWU)

Le TWU (transaction-weighted utilization) d'un itemset X est défini comme la somme de l'utilité des transactions contenant X, c'est-à-dire TWU(X)=$\sum_{Tc \in g(X)} TU(Tc)$.[20]

Exemple : TWU({c, d}) = TU(T_0) + TU(T_1) + TU(T_2) = 25 + 20 + 8 = 53.

3.2 Principales propriétés

3.2.1 Propriété 2 (La mesure d'utilité n'est ni monotone ni anti-monotone)

Soient deux ensembles itemset X et Y tels que X ⊂ Y. La relation entre les utilités de X et Y peut être u(X) < u(Y), u(X) > u(Y) ou u(X) = u(Y).[18]

Exemple : u(a)=20 U({a, c}) = 28 et U({a, c, e})=31 .

3.2.2 Propriété 3 (Le TWU est une borne supérieure monotone sur la mesure d'utilité)

Soit un itemset X. Le TWU de X est supérieur ou égal à son utilité (TWU(X) ≥ u(X)). De plus, le TWU de X est supérieur ou égal à l'utilité de ses sur-ensembles (TWU(X) ≥ u(Y) pour tout Y ⊃ X).[18]

Exemple : u(a)=20 et TWU(a)=55.

3.2.3 Propriété 4 (Élagage de l'espace de recherche en utilisant le TWU)

Pour tous ensemble itemset X, si TWU(X) < minutil, alors X est un ensemble d'articles à faible utilité, ainsi que tous ses sur-ensembles. Cela découle directement de ***la Propriété 3***.[18]

3.3 Algorithmes itemset à haute utilité

Un aperçu de certains algorithmes populaires d'extraction d'ensembles d'items à utilité haut. Le(*Table 6*) présente une comparaison de leurs caractéristiques en termes de type de recherche (recherche en largeur ou en profondeur), nombre de phases (une ou deux), représentation de la base de données (horizontale ou verticale) et algorithme d'extraction d'ensembles fréquents le plus similaire.

TABLE 6:ALGORITHMES POPULAIRES D'EXTRACTION D'ENSEMBLES D'ITEMS A UTILITE ELEVEE

Algorithme	Type de recherche	Nb de phases	Représentation DB	Extensions
Two-Phase	breadth-first	Deux	Horizontale	Apriori
IHUP	depth-first	Deux	Horizontal (arbre de préfixes)	FP-Growth
UP-Growth	depth-first	Deux	Horizontal (arbre de préfixes)	FP-Growth
HUI-Miner	depth-first	Une	Vertical (listes d' utilité)	Eclat
FHM	depth-first	Une	Vertical (listes d' utilité)	Eclat
EFIM	depth-first	Une	Horizontal (avec fusionnement)	LCM

4 L'algorithme FHM

L'algorithme FHM a été développé par Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng et Jerry Chun-Wei Lin. L'algorithme a été introduit pour la première fois dans un article de recherche intitulé "***FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning***" publié en 2014. Les auteurs ont proposé cet algorithme comme une méthode plus rapide et plus efficace pour extraire des ensembles d'articles à

haute utilité à partir d'une base de données transactionnelle. Depuis lors, l'algorithme a été amélioré et étendu dans des articles de recherche ultérieurs, tels que FHM+ et P-FHM+.[20]

L'objectif principal de l'algorithme FHM(*Algorithm 2*) est de réduire le nombre d'opérations de jointure (HUI-Miner) nécessaires lors de l'extraction des ensembles d'éléments fréquents avec une utilité élevée. Cela est réalisé grâce à une nouvelle technique appelée *Estimated Utility Co-Occurrence Structure (EUCS)Error! Reference source not found.,* qui permet d'éliminer certaines parties de l'espace de recherche.

4.1 Concepts fondamentaux

4.1.1 Total order

Soit > un ordre total sur les items I, et X un itemset . Une extension de X, notée Xy, est un itemset obtenu en ajoutant un item y à X, tel que y > i ,$\forall i \in$X.[22]

4.1.2 Utilité restante

Soit $\succ$ un ordre total sur les items de l'ensemble I, et X un itemset. L'utilité restante de X dans une transaction Tc est définie comme re(X, Tc) =$\sum_{i \in Tc \wedge i > x \forall x \in X} U(i, Tc)$.

Dans cette définition, re(X, Tc) représente l'utilité restante de l'itemset X dans la transaction Tc. Elle est calculée en faisant la somme des valeurs d'utilité des items i dans Tc qui viennent après tous les items de X selon l'ordre total $\succ$.[23]

4.1.3 La liste d'utilité

La liste d'utilité d'un itemset X dans une base de données D est composée de plusieurs tuples. Chaque tuple (c, iutil, rutil) correspond à une transaction Tc contenant X. Les éléments iutil et rutil d'un tuple représentent respectivement l'utilité de X dans Tc (u(X, Tc)) et l'utilité restante de X dans Tc (re(X, Tc)(*Error! Reference source not found.*)[21]

4.1.4 The Estimated Utility Co-occurrence Structure (EUCS)

EUCS de l'utilité d'une base de données D est désignée par EUCSD et définie comme $EUCS_D$ ={ (a; b; TW U(ab)) ∈ $I^* \times I^* \times R^+$ },est l'ensemble de tous les éléments ayant une TWU (utilité pondérée totale) supérieure ou égale à une utilité minimale dans D (***Table 7***).[23]

TABLE 7:THE ESTIMATED UTILITY CO-OCCURRENCE STRUCTURE (EUCS)

Item	d	b	a	e
b	45			
a	33	25		
e	45	54	47	
c	53	54	55	76

4.2 Principales propriétés

4.2.1 Propriété 5(Somme des valeurs iutil)

Soit X un itemset .X est un itemset à haute utilité si $\sum_{ul \in UL(X)} ul.iutil \geq minutil$. Sinon, X est considéré comme un itemset à faible utilité.[23]

4.2.2 Propriété 6(Somme des valeurs iutil et rutil)

Considérons un itemset X. Si la somme combinée des valeurs iutil et rutil dans la liste d'utilité de X est inférieure au seuil d'utilité minimum, alors toutes les extensions de X sont considérées comme des itemsets à faible utilité.[23]

4.3 Algorithme

Algorithm 2: The FHM algorithm

Input : D: *a transaction database,* ***minutil:*** *a user-specified threshold*

Output : *the set of high-utility itemsets*

1 *Scan* ***D*** *to calculate the* ***TWU*** *of single items;*

2 $I^* \leftarrow$ *each item* ***i*** *such that* ***TWU(i)*** $\geq$ ***minutil****;*

3 *Let* $>$ *be the total order of* ***TWU*** *ascending values on* I^**;*

4 *Scan* ***D*** *to build the* ***utility-list*** *of each item* $i \in I^*$ *and build the* ***EUCS****;*

5 ***FHMSearch*** *(*$\emptyset$*,*I^**,* ***minutil****,* ***EUCS****);*

ALGORITHM 2:FHM ALGORITHM

L'algorithme "*FHM*" reçoit en **entrée** *D*: une base de données transactionnelle ***(****Error! Reference source not found.****)***, *minutil*: un seuil spécifié par l'utilisateur *minutil =25*.**Sortie** : l'ensemble des ensembles d'items à haute utilité.

4.3.1 Premier balayage (scan) de la base de données

l'algorithme calcule le Total Weighted Utility (*TWU*)*Error! Reference source not found.* de chaque item en utilisant le Transaction Utility de chaque transaction.Les items qui satisfont à la condition TWU supérieure ou égale à *minutil* sont réservés dans une variable appelée ***I**** avec un ordonnancement ascendant basé sur les valeurs de TWU.

TABLE 8:TOTAL WEIGHTED UTILITY (TWU)

Item	**d**	**B**	**a**	**e**	**C**
TWU	53	54	55	76	84

TABLE 9:TRANSACTION UTILITY

TID	**T1**	**T2**	**T3**	**T4**	**T5**
TU	25	20	8	22	9

4.3.2 Deuxième balayage (scan) de la base de données

Créer une nouvelle base de données***(Table 10)***contenant des ***transactions révisées(Table 10)***. Ces transactions révisées ne contiennent que des items appartenant à l'ensemble ***I*[d, b, a, e, c].*** Cette nouvelle base de données est utilisée pour construire les ***listes d'utilité(Table 13)*** et ***EUCS(Table 7).***

TABLE 10:BASE DE DONNEES QUANTITATIVE DES TRANSACTIONS SELON L'ORDRE TOTAL D≻B≻A≻E≻C

TID	**Transaction**
T1	(d, 3),(b, 5),(a, 1),(e, 1),(c, 1)
T2	(d, 3),(b, 4),(e, 1),(c, 3)
T3	(d, 1),(a, 1),(c, 1)
T4	(a, 2),(e, 2),(c, 6)
T5	(b, 2),(e, 1),(c, 2)

TABLE 11:LA LISTE D'UTILITE DE D

tid	iutil	rutil
T1	6	19
T2	6	14
T3	2	6

TABLE 12:LA LISTE D'UTILITE DE B

tid	iutil	rutil
T1	10	9
T2	8	6
T5	4	5

L'algorithme appelle la procédure de recherche récursive "Search" (*Veuillez consulter la section suivante pour obtenir plus de* ***The Search procedure***).

4.4 The Search procedure

Algorithm 3: The Search procedure

Input : P: *an itemset,* ***ExtensionsOfP:*** *a set of extensions of P, the* ***minutil*** *threshold,*

Output: *the set of high-utility itemsets*

```
1   Foreach itemset P x ∈ ExtensionsOfP do
2       If SUM(P x.utilitylist.iutils) ≥ minutil then
3           Output P x;
4       end
5       If SUM(P x.utilitylist.iutils)+SUM(P x.utilitylist.rutils) ≥ minutil then
6           ExtensionsOfPx ← ∅;
7           Foreach itemset P y ∈ ExtensionsOfP such that y ≻ x do
8               If ∃(x, y, c) ∈ EUCS such that c ≥ minutil then
9                   P xy ← P x ∪ P y ;
10                  P xy.utilitylist ← Construct (P, P x, P y);
11                  ExtensionsOfPx ← ExtensionsOfPx ∪ P xy ;
12              end
13          end
```

```
14          |     |  Search (P x, ExtensionsOfPx, minutil);
15          |   end
16        end
```

ALGORITHM 3:THE SEARCH PROCEDURE

L'algorithme "**Search**" est une procédure utilisée dans le cadre de l'extraction d'itemsets à haute utilité. Il prend en entrée un itemset ***P***, un ensemble d'extensions de P appelé ***ExtensionsOfP***, un seuil d'utilité minimum (***Minutil***) et une structure de données appelée ***EUCS***. La procédure commence en itérant sur chaque ***itemset P_x*** dans l'ensemble ***ExtensionsOfP***. Pour chaque ***P_x,*** deux conditions sont vérifiées :

Si la somme des valeurs d'utilité dans ***P_x.utilitylist.iutils*** est supérieure ou égale au seuil ***Minutil***, alors ***l'itemset P_x*** est considéré comme ayant une haute utilité et est renvoyé en sortie.

Si la somme des valeurs d'utilité dans ***P_x.utilitylist.iutils*** et ***P_x.utilitylist.rutils*** est supérieure ou égale au seuil ***Minutil***, cela indique que ***P_x*** et ses extensions doivent être explorés plus en détail. Dans ce cas, un nouvel ensemble vide appelé ***ExtensionsOfPx*** est initialisé.

Ensuite, l'algorithme effectue une autre itération sur chaque ***itemset P_y*** dans l'ensemble ***ExtensionsOfP***, en respectant un ordre total où ***Y*** est supérieur à ***X***. Pour chaque ***P_y***, il vérifie si un ***triplet (x, y, c)*** existe dans la structure ***EUCS***, où c est supérieur ou égal à ***Minutil.*** Si cette condition est satisfaite, cela signifie que ***P_x*** et ***P_y*** peuvent être fusionnés pour former un nouvel ***itemset P_xy***.

L'algorithme crée alors le nouvel ***itemset P_xy*** en fusionnant ***P_x*** et ***P_y***, et utilise la (***The Construct procedure***)pour calculer la liste d'utilité pour ***P_xy(Table 13)***, en utilisant les ***itemsets P***, ***P_x*** et ***P_y*** comme paramètres. Le nouvel ***itemset P_xy*** est ensuite ajouté à l'ensemble ***ExtensionsOfPx.***

Après avoir terminé la boucle, l'algorithme appelle récursivement la procédure de recherche (***Search***) en utilisant ***P_x*** comme le nouvel ***itemset P***, ***ExtensionsOfPx*** comme le nouvel ensemble ***ExtensionsOfP***, et ***Minutil***. Cela permet d'explorer davantage les extensions de ***P_x*** pour trouver d'autres itemsets à haute utilité.

Le processus se répète jusqu'à ce que toutes les combinaisons possibles d'itemsets et de leurs extensions aient été explorées. L'algorithme utilise une approche de recherche en profondeur pour explorer les extensions de chaque itemset avant de passer au suivant. Les conditions d'utilité sont vérifiées pour déterminer si un itemset ou son extension doivent être considérés comme ayant une haute utilité.

4.5 The Construct procedure

Algorithm 4: The Construct procedure

Input : P: *an itemset,* ***P x****: the extension of P with an item x,* ***P y:*** *the extension of P*

Output*: the utility-list of P xy*

```
1   UtilityListOfP xy ← ∅ ;
2   Foreach tuple ex ∈ P x.utilitylist do
3       If ∃ey ∈ P y.utilitylist and ex.tid = exy.tid then
4           If P.utilitylist ≠ ∅ then
5               Search element e ∈ P.utilitylist such that e.tid = ex.tid ;
6               exy ← (ex.tid, ex.iutil + ey.iutil − e.iutil, ey.rutil);
7           end
8           else
9               exy ← (ex.tid, ex.iutil + ey.iutil, ey.rutil);
10          end
11          UtilityListOfP xy ← U tilityListOfP xy ∪{exy};
12      end
13  end
14  return UtilityListPxy;
```

ALGORITHM 4:THE CONSTRUCT PROCEDURE

L'algorithme "*Construct*" reçoit en **entrée** un itemset P ***(initialement vide ∅)***, une extension de P avec un item x ***(P_x initialement d)***, et une extension de P avec un item y ***(P_y initialement b)***. Son objectif est de construire la liste d'utilité de l'itemset fusionné P_xy.

L'algorithme commence par initialiser une liste vide appelée "***UtilityListOfP_xy***" pour stocker la liste d'utilité de ***P_xy (ligne 1)***. Ensuite, pour chaque tuple ***ex ({tid_d, iutil_d, rutil_d})*** dans la liste d'utilité de P_x.utilitylist ***(*****Error! Reference source not found.*****)***, il vérifie s'il existe un tuple ***ey ({tid_b, iutil_b, rutil_b})*** dans la liste d'utilité de ***P_y***.utilitylist

(*Error! Reference source not found.*) tel que ***ex.tid*** (transaction ID) soit égal à ey.tid ***(ligne 3)***. Cela signifie qu'il y a des transactions correspondantes dans les deux extensions.

Si une correspondance est trouvée ***(ligne 3)***, il vérifie si la liste d'utilité de ***P*** est non vide ***(ligne 4)***. Dans ce cas, il recherche l'élément e dans la liste d'utilité de ***P*** tel que ***e.tid*** soit égal à ***ex.tid (ligne 5)***. Cela suppose que la liste d'utilité de ***P*** contient des informations sur les transactions correspondantes. Ensuite, l'algorithme calcule un nouveau tuple ***exy (ePxy)*** en combinant les valeurs d'utilité des tuples ***ex(ed)*** et ***ey(eb),*** tout en soustrayant la valeur d'utilité de l'élément ***e (ligne 6).*** Cela permet de prendre en compte la contribution de ***P*** à l'itemset fusionné. Le tuple ***ePxy ({tid_b, iutil_d + iutil_b - iutil_P, rutil_b})*** est calculé.

Si la liste d'utilité de ***P*** est vide ***(ligne 8)***, l'algorithme calcule simplement un nouveau tuple ***exy(e_db)*** en additionnant les valeurs d'utilité de ***ex*** et ***ey (ligne 9)***. Dans ce cas, il n'y a pas de contribution de ***P*** à prendre en compte. Le tuple ***exy ({tid_b, iutil_d + iutil_b, rutil_b})*** est calculé exemple: ***({T1,16,9}).***

Enfin, le tuple ***exy*** est ajouté à la liste d'utilité de ***P_xy (ligne 11).*** Après avoir traité tous les tuples ex de la liste d'utilité de ***P_x***, l'algorithme retourne la liste d'utilité de ***P_xy (ligne 14)*****Error! Reference source not found.**.

L'algorithme "Construct" construit la liste d'utilité de l'itemset fusionné ***P_xy*** en combinant les informations d'utilité des extensions ***P_x*** et ***P_y***, tout en tenant compte de la contribution de ***P*** si des transactions correspondantes sont présentes dans la liste d'utilité de ***P***.

TABLE 13:LA LISTE D'UTILITE DE ({D,B})

tid	iutil	rutil
T1	16	9
T2	14	6

Chapitre II: High Utility Itemset Mining HUM

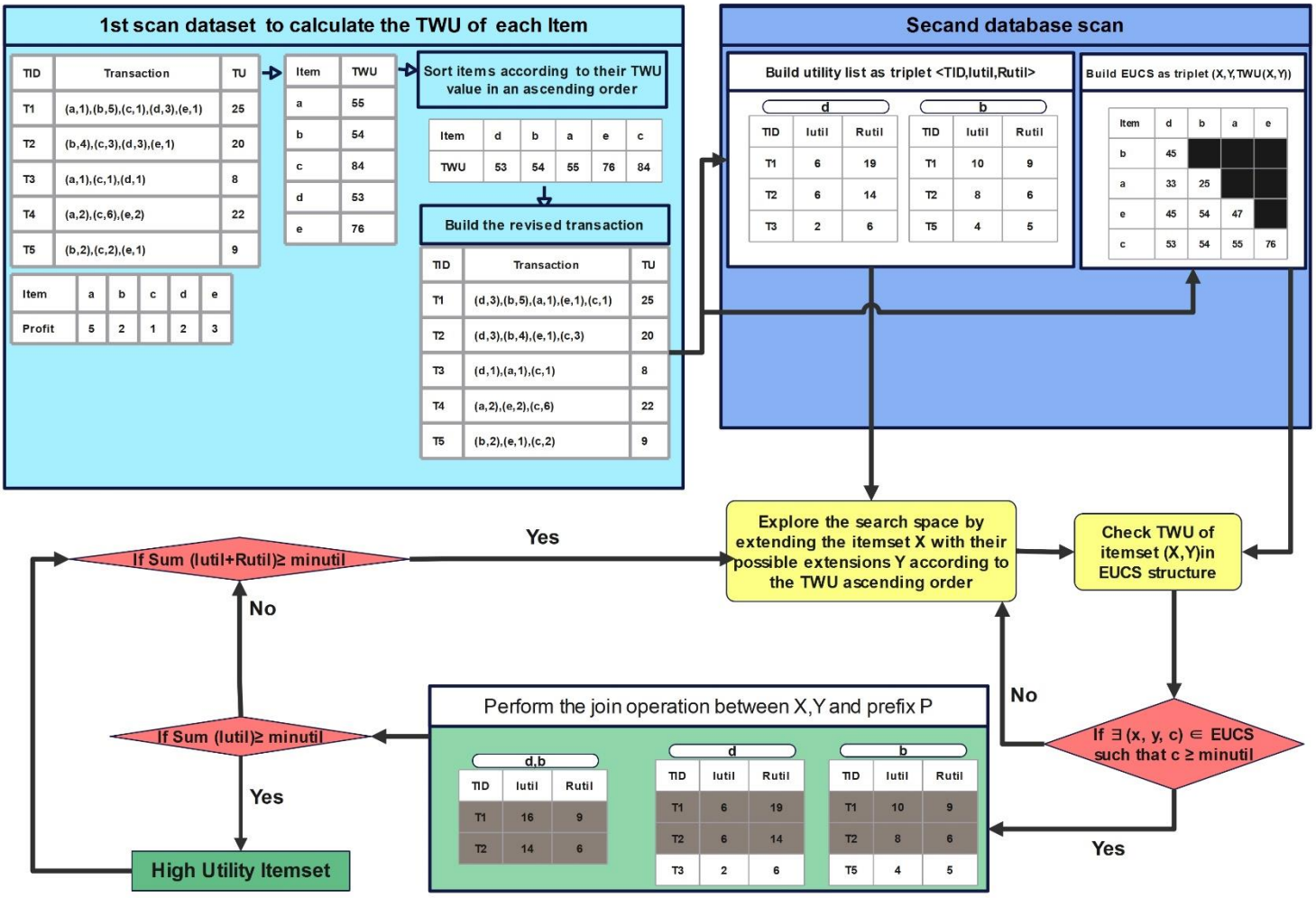


FIGURE 3:FHM ALGORITHM

4.6 Les inconvenient

L'extraction des itemsets à haute utilité est une technique de fouille de données qui vise à identifier des itemsets générant un profit élevé ou une utilité élevée. Elle est utile pour découvrir des itemsets rentables dans différents domaines. Cependant, cette technique peut générer un ***grand nombre d'itemsets***, dont certains peuvent être ***redondants*** et fournir des ***informations similaires***. Pour remédier à cela, les chercheurs ont développé des méthodes permettant de découvrir des ***représentations concises des itemsets à haute utilité (HUI)***. Ces techniques aident à ***réduire la complexité***, à ***améliorer l'efficacité*** et à conserver les itemsets les ***plus précieux*** et ***non redondants*** pour une analyse et une prise de décision ultérieures.

5 Représentations concises

Dans cette section, nous explorerons l'idée de ***la représentation concise*** qui permet de ***compresser*** et de ***réduire*** la taille des ensembles d'items à utilité élevée tout en préservant leur utilité et leur signification. La représentation concise offre plusieurs avantages, notamment la réduction de ***la redondance***, l'amélioration de ***l'efficacité*** des algorithmes d'extraction, et une interprétation plus claire des motifs découverts.

5.1 Concepts fondamentaux

5.1.1 Closed HUI (CHUI)

Un HUI X est considéré comme fermé s'il n'existe aucun autre HUI Y tel que X soit un sous-ensemble de Y ($X \subset Y$) et sup(X) = sup(Y). En d'autres termes, il ne devrait pas exister de HUI Y plus grand avec la même valeur d'utilité que X.[24]

5.1.2 Maximal HUI (MaxHUI)

Un HUI X est considéré comme un HUI maximal s'il n'existe aucun autre HUI Y tel que X soit un sous-ensemble propre de Y ($X \subset Y$). En d'autres termes, il ne devrait pas exister de HUI Y plus grand qui contient X.[25]

5.1.3 Générateur d'ensembles d'items à utilité élevée (GHUI)

Pour être considéré comme un générateur d'ensembles d'items à utilité élevée (GHUI), un ensemble d'items X doit satisfaire à deux conditions. Premièrement, il ne doit y avoir aucun sous-ensemble propre Y de X où la valeur d'utilité de X (sup(X)) est égale à la valeur d'utilité de Y (sup(Y)). Deuxièmement, il doit exister un ensemble d'items Z qui contient X en tant que sous-ensemble ($X \subseteq Z$) et dont la valeur d'utilité (u(Z)) est supérieure ou égale à un seuil d'utilité minimale spécifié (minutil).[26]

5.1.4 Minimal HUIs (MinHUI)

Un ensemble d'items X est considéré comme un ensemble d'items à utilité élevée minimal (MinHUI) si et seulement si $u(X) \geq$ minutil et qu'il n'existe pas d'ensemble d'items $Y \subset X$ tel que $u(Y) \geq$ minutil.[27].Cette représentation proposée est l'opposée des ensembles d'items à utilité élevée maximaux (MaxHUI), c'est-à-dire qu'elle se compose des ensembles d'items ***les plus petits*** qui génèrent un bénéfice élevé ***plutôt*** que des plus grands.

5.2 Principales propriétés

5.2.1 Propriété 7 (Influence de minutil sur le nombre de MinHUIs)

Si minutil est réduit, le nombre de MinHUIs peut augmenter, diminuer ou rester le même. De plus, si minutil = 1, l'ensemble des MinHUIs est égal à I.[27]

5.2.2 Propriété 8 (propriété d'élagage MinHUIs)

Si un ensemble d'items X est un MinHUI, alors les sur-ensembles de X ne sont pas des MinHUIs.[27]

5.3 Algorithmes

Différents algorithmes ont été développés pour efficacement découvrir les formes compactes des ensembles d'items à utilité élevée mentionnés précédemment. Le***(Table 14)***offre un aperçu de ces algorithmes ainsi que de leurs caractéristiques. Dans de nombreux cas, extraire les formes compactes peut être bien plus rapide que découvrir l'ensemble des ensembles d'items à utilité élevée, car cela permet de trouver moins d'ensembles d'items.

TABLE 14:ALGORITHMES POUR EXTRAIRE DES REPRESENTATIONS CONCISES D'ITEMSETS A HAUTE UTILITE

Algorithme	Patterns	Nb de phases	Représentation DB	Extensions
MinFHM	MinHUIs	Une	Vertical (utility-lists)	FHM
GHUI-Miner	GHUIs	Une	Vertical (utility-lists)	FHM
CHUD	CHUIs	Deux	Vertical (utility-lists)	DCI_Closed
CHUI-Miner	CHUIs	Une	Vertical (utility-lists)	DCI_Closed
CLS-Miner	CHUIs	Une	Vertical (utility-lists)	FHM
EFIM-Closed	CHUIs	Une	Horizontal (with merging)	EFIM
GUIDE	MHUIs	Une	Stream	UPGrowth
CHUI-Mine	MHUIs	Une	Vertical (utility-lists)	HUI-Miner

5.4 L'algorithme MinFHM

Algorithm 5: The MinFHM algorithm

Input : D: *a transaction database,* ***minutil****: a user-specified threshold*

Output: *the set of high-utility itemsets*

1 *Scan* ***D*** *to calculate the* ***TWU*** *of single items;*

2 ***MinHUI*** ← *each item* ***i*** *such that* ***utility(i) ≥ minutil*** *;*

3 ***I**** ← *each item* ***i*** *such that* ***TWU(i) ≥ minutil*** *and* ***utility(i) < minutil****;*

4 *Let > be the total order of* ***TWU*** *ascending values on* ***I*****;**

5 *Scan* ***D*** *to build the* ***utility-list*** *of each item* ***i ∈ I**** *and build the* ***EUCS****;*

6 ***Output*** *each item* ***i ∈ I**** *such that* ***SUM({i}.utilitylist.iutils) ≥ minutil****;*

7 ***Search (∅,I*, minutil, EUCS);***

8 ***end***

ALGORITHM 5: THE MINFHM ALGORITHM

L'algorithme " MinFHM " reçoit en **entrée** D: une base de données transactionnelle ***(***Error! Reference source not found.***)***, minutil: un seuil spécifié par l'utilisateur minutil =25. **Sortie** : l'ensemble des ensembles d'items à haute utilité.

5.4.1 Premier balayage (scan) de la base de données

L'algorithme calcule le Total Weighted Utility (***TWU***) de chaque item en utilisant le ***Transaction Utility***(*Error! Reference source not found.*) de chaque item et le Transaction Utility de chaque transaction. Les items dont l'utilité est supérieure ou égale au seuil ***Minutil*** sont ajoutés à l'ensemble ***MinHUI***. Les items qui satisfont à la condition ***TWU*** supérieure ou égale à ***Minutil*** et dont l'utilité est inférieure à ***Minutil*** sont ajoutés à l'ensemble ***I**** avec un ordonnancement ascendant basé sur les valeurs de ***TWU***.

5.4.2 Deuxième balayage (scan) de la base de données

Dans la deuxième partie du balayage de la base de données, l'algorithme crée une nouvelle base de données contenant uniquement les ***transactions révisées***(***Table 10***) qui ne contiennent que des items de l'ensemble ***I*.** Cette nouvelle base de données est utilisée pour

construire les ***listes d'utilité(Table 13)*** et la structure de données ***EUCS(Table 7)***.Enfin, l'algorithme appelle ***Algorithm 6: The Search procedure*** récursive "***Search***", qui explore les extensions des itemsets pour trouver les ensembles d'items à haute utilité.

5.5 The Search procedure

Algorithm 6: The Search procedure

input : P: *an itemset,* ***ExtensionsOfP:*** *a set of extensions of* ***P:*** *minutil: a user-*

output: *the set of high-utility itemsets*

```
1   Foreach itemset P x ∈ ExtensionsOfP do
2       If SUM(P x.utilitylist.iutils)+SUM(P x.utilitylist.rutils) ≥ minutil then
3           ExtensionsOfPx ← ∅;
4           Foreach itemset P y ∈ ExtensionsOfP such that y > x do
5               If ∃(x, y, c) ∈ EUCS such that c ≥ minutil then
6                   P xy ← P x ∪ P y;
7                   P xy.utilitylist ← Construct (P, P x, P y);
8                   ExtensionsOfPx ← ExtensionsOfPx ∪ {P xy};
9                   If SUM(P xy.utilitylist.iutils) ≥ minutil then
10                      If ∃(E) ∈ MinHUI such that E ⊂ Pxy then
11                          Remove Pxy from MinHUI          //Eliminate supersetes
12                      end
13                      If ∃(S) ∈ MinHUI such that Pxy ⊂ S then
14                          Remove S from MinHUI            // remove supersetes
15                      end
16                      Output P x;
17                      MinHUI ← MinHUI ∪ {P x};
18                  end
19              end
20          end
21          Search (P x, ExtensionsOfPx, minutil);
22      end
23  end
```

ALGORITHM 6: THE SEARCH PROCEDURE

L'algorithme reçoit en entrée un ensemble d'itemsets P vide, un ensemble ***d'extensions de P*** trié selon un ordre total, un seuil ***Minutil*** et une structure ***EUCS***. Il utilise une stratégie de recherche en profondeur d'abord pour explorer les itemsets.

Pour chaque ***itemset P_x*** dans ***ExtensionsOfP***, l'algorithme vérifie si la somme des valeurs d'utilité dans ***P_x.utilitylist.iutils*** et ***P_x.utilitylist.rutils*** est supérieure ou égale au seuil ***Minutil***. Si oui, il explore les extensions de ***P_x***.

L'algorithme fusionne ***P_x*** avec chaque itemset ***P_y*** dans ***ExtensionsOfP*** qui satisfait une condition dans la structure ***EUCS***. Il calcule la liste d'utilité pour chaque nouvel itemset ***P_xy*** créé.

Pour chaque ***P_xy***, l'algorithme vérifie si sa somme de valeurs d'utilité dans ***P_xy.utilitylist.iutils*** est supérieure ou égale au ***Minutil***. S'il est suffisant, il vérifie si ***P_xy*** est un sur-ensemble ou un sous-ensemble d'un itemset dans l'ensemble ***MinHUI***. En fonction de ces vérifications, des actions sont effectuées pour éliminer les itemsets redondants.

Les itemsets ***P_x*** qui passent les conditions d'utilité et de non-redondance sont considérés comme ayant une haute utilité et sont renvoyés en sortie.

Après avoir terminé la boucle, l'algorithme appelle récursivement la procédure de recherche (***Search***) en utilisant le nouvel ensemble ***P_x***, les nouvelles extensions de ***P (ExtensionsOfPx)*** et le seuil ***Minutil***. Cela permet d'explorer davantage les extensions pour trouver d'autres itemsets à haute utilité.

Chapitre II: High Utility Itemset Mining HUM

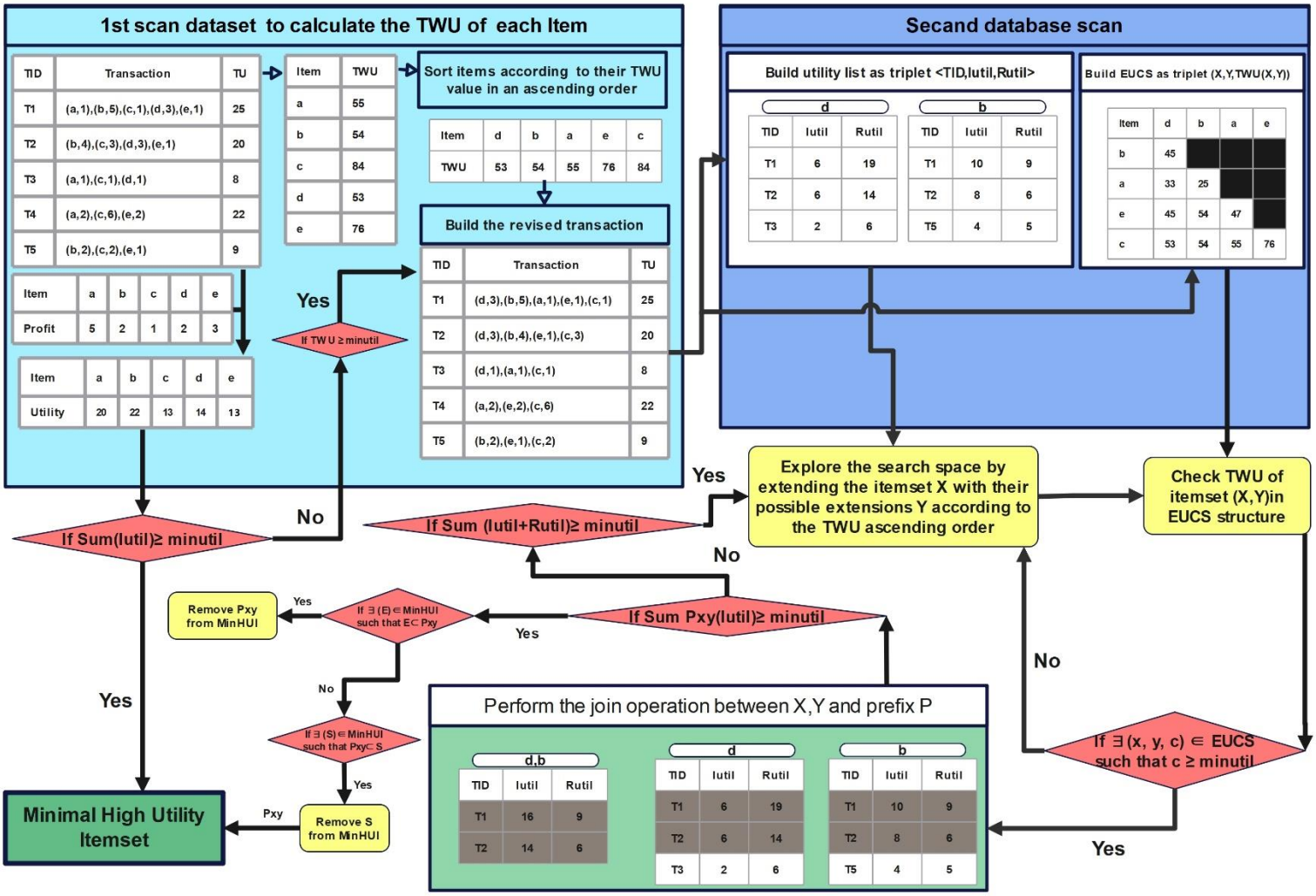


FIGURE 4:MINFHM

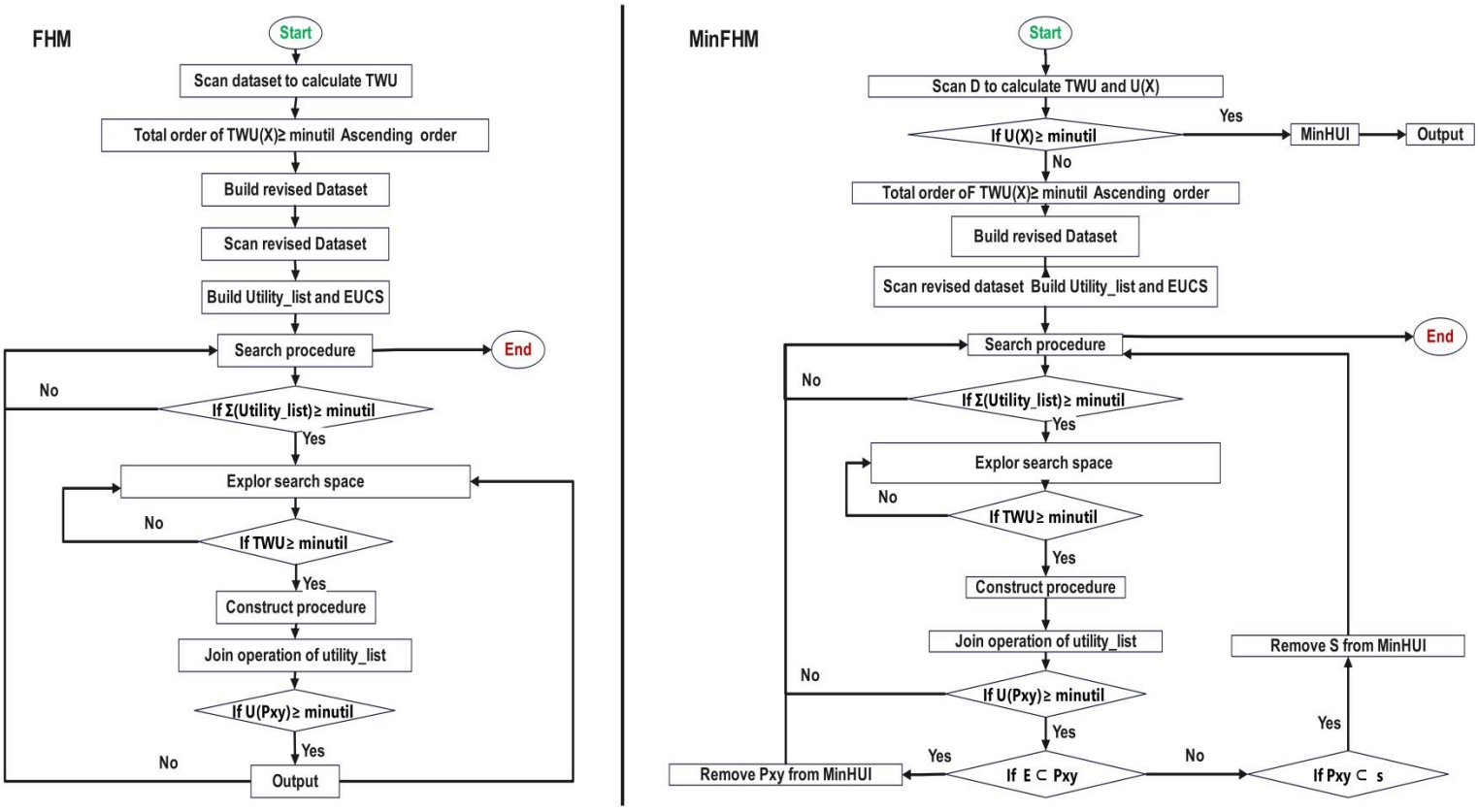


FIGURE 5: FHM ET MINFHM

6 Conclusion

En conclusion, l'extraction des itemset à haute utilité (HUIM) est un domaine de recherche important dans la fouille de données. Les algorithmes HUIM visent à découvrir des motifs ayant une utilité élevée ou une importance pour l'utilisateur, en tenant compte de facteurs tels que le profit, le temps ou d'autres mesures d'utilité. Ces algorithmes abordent les limites de l'extraction des itemset fréquents en se concentrant sur les items qui offrent des avantages significatifs plutôt que sur la fréquence seule.

Dans ce chapitre, nous avons exploré en détail l'implémentation des algorithmes de fouille d'ensembles d'items à haute utilité (HUIM) en utilisant l'EUCS et la liste d'utilité. Nous avons présenté l'algorithme FHM et son extension, MinFHM, en mettant l'accent sur les stratégies utilisées pour optimiser les performances en termes de temps d'exécution et d'utilisation de la mémoire.

Grâce à notre implémentation, nous avons pu constater que l'utilisation de l'EUCS permet de réduire efficacement l'espace de recherche et d'accélérer le processus d'extraction des ensembles d'items à haute utilité. En utilisant des informations préalablement calculées, l'EUCS permet de minimiser les calculs redondants et de gagner du temps lors de l'évaluation de l'utilité des ensembles d'items.

De plus, l'introduction de la liste d'utilité comme une structure de données clé s'est avérée essentielle. Cette structure permet de stocker les informations d'utilité pour chaque item individuel, ce qui facilite les opérations de recherche et de mise à jour lors de l'extraction des ensembles d'items à haute utilité. Cela contribue à améliorer les performances globales de l'algorithme.

Chapitre III:
Réalisation & Interprétation

Chapitre III: Réalisation & Interprétation

1 Introduction

Dans ce chapitre, nous nous concentrons sur les outils de développement utilisés dans notre projet. Nous décrivons les technologies, les langages de programmation et les bibliothèques utilisés, tels que Eclipse, Java, Python, Visual Studio et la bibliothèque SPMF, pour l'extraction des ensembles d'items fréquents. Nous expliquons comment nous avons configuré ces outils et intégré les algorithmes de fouille d'ensembles d'items à haute utilité dans notre environnement de développement.

Pour commencer, Eclipse est un environnement de développement intégré (IDE) couramment utilisé. Nous avons choisi Eclipse comme notre principal IDE pour ce projet en raison de ses fonctionnalités pratiques telles que l'autocomplétion, le débogage et la gestion de projet.

En ce qui concerne les langages de programmation, nous avons utilisé Java et Python. Java est un langage populaire et polyvalent qui offre simplicité, performance et une large adoption dans le domaine de l'informatique. Nous avons implémenté nos algorithmes de fouille d'ensembles d'items à haute utilité en Java. Quant à Python, il est connu pour sa facilité d'utilisation et sa flexibilité, ce qui en fait un choix judicieux pour certaines tâches spécifiques dans notre projet.

Enfin, nous avons intégré la bibliothèque SPMF (Squential Pattern Mining Framework) dans notre environnement de développement. SPMF est une bibliothèque open-source en Java spécialement conçue pour la fouille de données séquentielles, y compris l'extraction d'ensembles d'items fréquents.

Pour compléter ce chapitre, nous fournissons une description détaillée de l'architecture générale de notre implémentation. Nous expliquons la structure du code, les différentes classes et modules utilisés pour mettre en œuvre les algorithmes de fouille d'ensembles d'items à haute utilité. Nous abordons également les considérations de performance, telles que l'utilisation efficace de la mémoire et l'optimisation du temps d'exécution, qui ont été prises en compte lors de l'implémentation des algorithmes.

Chapitre III: Réalisation & Interprétation

2 Présentation des outils de développement

2.1 Eclipse

Eclipse est un environnement de développement intégré (IDE) open-source utilisé principalement pour le développement en Java, bien qu'il supporte plusieurs langages de programmation grâce à des plugins. Il a été initialement développé par IBM puis adopté par la fondation Eclipse, une communauté de développeurs et d'organisations qui collaborent pour améliorer et maintenir la plateforme Eclipse.[28]

2.2 Java

Java est à la fois un langage de programmation et une plate-forme informatique développés par Sun Microsystems en 1995. Il s'agit d'une technologie fondamentale qui permet l'exécution de programmes modernes et performants, utilisée notamment dans la création d'utilitaires, de jeux et d'applications professionnelles. Java est largement adopté, avec une présence sur plus de 850 millions d'ordinateurs de bureau et plus d'un milliard de périphériques dans le monde, y compris des appareils mobiles et des systèmes de diffusion télévisuelle.[29]

2.3 Python (langage)

Est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.[30]

Le langage Python est placé sous une licence libre proche de la licence BSD5 et fonctionne sur la plupart des plateformes informatiques, des smartphones aux ordinateurs centraux6, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et peut aussi être traduit en Java ou .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.[30]

2.4 Visual Studio

Visual Studio est un environnement de développement intégré (IDE) populaire créé par Microsoft. Il prend en charge plusieurs langages de programmation et offre des outils pour l'édition de code, le débogage et la gestion de projets. Avec une interface conviviale et de nombreuses options de personnalisation, il améliore la productivité des développeurs. Visual Studio s'intègre aux systèmes de contrôle de version tels que Git et propose des fonctionnalités de collaboration. Il est largement utilisé pour créer des applications logicielles sur différentes plateformes.[31]

2.5 Une bibliothèque de donnée extra source (SPMF)

SPMF est une bibliothèque open-source écrite en Java pour l'exploration de données dans le domaine de l'extraction de modèles miniers. Elle est distribuée sous licence GPL. Cette bibliothèque offre l'implémentation de 210 algorithmes de traitement des données, et le code source de chaque algorithme peut être facilement intégré dans d'autres logiciels Java. SPMF peut être utilisé de manière autonome grâce à son interface utilisateur simple ou en ligne de commande. De plus, le code source de chaque algorithme peut être facilement intégré dans d'autres logiciels Java. Des wrappers non officiels sont également disponibles pour d'autres langages tels que Python, R et Weka. SPMF se distingue par sa rapidité et sa légèreté, sans dépendance à d'autres bibliothèques. La version actuelle, v2.47, a été publiée le 28 mai 2021.[32]

3 Conception et Implémentation

3.1 L‘architecture générale

La conception globale d'un système ou d'un projet, appelée architecture générale, définit sa structure en incluant les composants, leurs interactions et les principes de conception qui guident leur mise en œuvre. Pour comparer les algorithmes FHM et MinFHM dans le domaine de l'exploration de données à haute utilité, voici une proposition d'architecture :

1. Effectuer une sauvegarde de la base de données d'origine.
2. Réaliser l'étape de prétraitement des données, suivie de l'étape de fouille de données en utilisant les algorithmes FHM et MinFHM.

3. Interpréter et comparer les résultats obtenus à partir des deux algorithmes pour évaluer leur performance et leur précision dans la découverte des données à haute utilité.

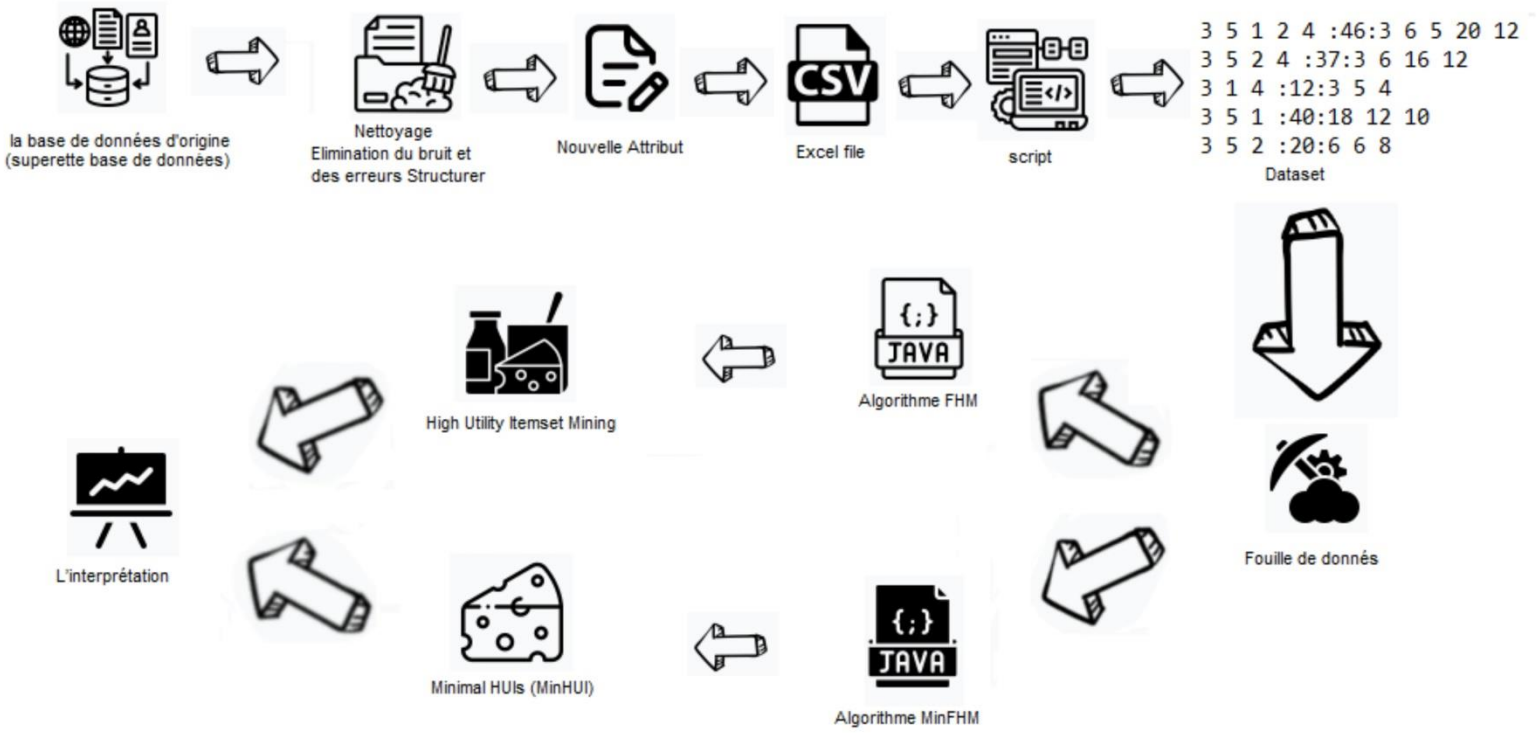


FIGURE 6:L'ARCHITECTURE GENERALE

3.2 Description des composants de l'architecture

3.2.1 La base de données d'origine 1 (cosmétique base de données)

La base de données principale, connue sous le nom de "cosmétique base de données", est le centre névralgique du système qui stocke toutes les informations liées au cosmétique. Elle enregistre les détails des produits, des clients, des ventes et d'autres informations clés. La base de données est organisée et structurée de manière à faciliter les opérations telles que l'ajout, la modification, la recherche et la suppression d'informations. Elle est régulièrement

mise à jour avec de nouvelles données provenant des activités quotidiennes de l'industrie cosmétique. Ces données jouent un rôle vital dans le suivi des stocks, l'analyse des ventes, la gestion des préférences des clients et l'aide à la prise de décisions commerciales éclairées. Nous avons effectué une sauvegarde de cette base de données(*Figure 7*) afin d'étudier le mouvement des produits (les ventes) et d'analyser les résultats afin de déterminer la relation entre les produits.

	A	B	C	D	E	F	G	H	I
1	IDDetailBon	IDBon	IDProduit	Designation	Qte	Prix_Achat	PrixV_Detail	profile	
2	1573	18	4407	KASSA HAODA NOIRE	1	100	150	50	
3	4548	40	548	SAVON DOVE	1	155	170	15	
4	4541	40	1377	BAD SENSODYNE DOUCEUR	1	360	400	40	
5	4564	40	1483	CRE EMILY	1	160	200	40	
6	4544	40	1663	PORTE SAVON FLOWER	1	43	70	27	
7	4543	40	2621	GLD JOHNSONS BEBE 200ML	1	350	400	50	
8	4547	40	2790	COTON TIEGE FAMILY	4	73	90	68	
9	4539	40	3122	DNT COLGATE MAX WHITE ONE INTENSE	1	255	350	95	

FIGURE 7: DATASETS ORIGINAL (COSMETIQUE)

TABLE 15:STATISTIQUES DE LA BASE DE DONNEE 1(COSMETIQUE BASE DE DONNEES)

Le nombre des items	7723
Le nombre des transactions	170374
La taille de la base de donnée	5,95 Mo
Total utility	45229212
Maximum Id of item	16295
Average length of transaction	3.86
Maximum length of transaction	151

3.2.2 L'étape de prétraitement des données

a) Sélection

Dans un premier temps, nous avons choisi de sélectionner uniquement les données essentielles pour notre analyse, à savoir l'identifiant du bon de vente (IDBon), l'identifiant du produit (IDProduit), la quantité(Qte) et le profil(*Figure 8*). Ces données représentent les bons de vente, où chaque bon est identifié de manière unique par l'IDBon et contient des produits.

Chapitre III: Réalisation & Interprétation

Pour optimiser l'espace mémoire, nous avons utilisé les IDProduit plutôt que les noms de produits. Chaque produit est associé à une quantité et un profit, tandis que le reste des données a été ignoré dans cette sélection initiale.

1	IDBon	IDProduit	Qte	profile
2	18	4407	1	50
3	40	548	1	15
4	40	1377	1	40
5	40	1483	1	40

FIGURE 8: SELECTION DES DONNEE

Après la sélection des données, nous avons entrepris une étape de nettoyage des données sélectionnées. Pendant cette étape, nous avons traité les erreurs de saisie, les données erronées et les données manquantes. Pour faciliter ce processus, nous avons développé un script en langage Python(*Figure 10*). Ce script prend en entrée(*Error! Reference source not found.*) un fichier Excel brut contenant les données et génère en sortie un fichier Excel nettoyé(*Figure 11*), où les données sont corrigées et filtrées pour assurer leur qualité.

121	59	7623	1	30
122	60	2444	1	48
123	60	11870	1	-10
334	117	2748	1	8
335	118	2019	2	26
336	119	506	-1	-50
721	210	12616	1	20
722	211	409	1	25
723	211	2335	2	12,5
724	212	11384	2	-50

FIGURE 9:ENTREE (LES DONNEE NEGATIVE)

Chapitre III: Réalisation & Interprétation

```python
# 1-cleaning.py > ...
import os
import pandas as pd

# Specify the path to the Excel file
file_path = r'C:\Users\user\Desktop\script BAKOUR MOULOUD\2---Datasets sélection.xlsx'

# Read the Excel file
df = pd.read_excel(file_path)

# Iterate over the first 4 columns
for column in df.columns[:4]:
    # Check if the column data type is not string
    if df[column].dtype != 'object':
        # Convert the non-string column to string
        df[column] = df[column].astype(str)

    # Replace "-" with an empty string
    df[column] = df[column].str.replace("-", "")

# Get the directory path of the original file
directory = os.path.dirname(file_path)

# Construct the path for the output file in the same directory
output_path = os.path.join(directory, '3---Dataset nettoyage.xlsx')

# Save the modified DataFrame to the output file
df.to_excel(output_path, index=False)

```

FIGURE 10: SCRIPT PYTHON (NETTOYAGE)

120	59	2021	1	28.0		
121	59	7623	1	30.0		
122	60	2444	1	48.0		
123	60	11870	1	10.0		
334	117	2748	1	8.0		
335	118	2019	2	26.0		
336	119	506	1	50.0		
721	210	12616	1	20.0		
722	211	409	1	25.0		
723	211	2335	2	12.5		
724	212	11384	2	50.0		

FIGURE 11: SORTIE UN FICHIER EXCEL NETTOYE

b) Regroupement les ID Produit

Après avoir nettoyé les données, nous avons obtenu un fichier Excel contenant 4 colonnes. Cependant, les bons de vente dans ce fichier étaient détaillés, ce qui signifiait qu'il y avait plusieurs lignes pour un même bon de vente (voir *Figure 12*). Pour résoudre ce problème, nous avons développé un script en langage Python (voir *Figure 13*). Ce script prend en entrée un fichier Excel contenant des bons de vente détaillés et regroupe les données de chaque bon de vente en une seule ligne en utilisant l'identifiant unique (IDBon). En sortie, le script génère un nouveau fichier Excel(voir *Figure 14*) avec 3 colonnes : ITEMS

(contenant tous les IDProduit de ce bon de vente), TU (Transaction Utility, qui est la somme des profits de tous les IDProduit) et UTILITY (les profits de chaque IDProduit séparés par un espace vide). Ce format regroupé facilite l'analyse ultérieure des données consolidées.

	IDBon	IDProdui	Qte	profile
2	18	4407	1	50.0
3	40	548	1	15.0
4	40	1377	1	40.0
5	40	1483	1	40.0
6	40	1663	1	27.0
7	40	2621	1	50.0
8	40	2790	4	68.0
9	40	3122	1	95.0
10	40	3240	4	300.0
11	40	4951	1	30.0

FIGURE 12: FICHIER EXCEL PLUSIEURS LIGNES POUR UN MEME BON DE VENTE

```python
import pandas as pd

# Read the input Excel file
input_file = r'C:\Users\user\Desktop\script BAKOUR MOULOUD\3---Dataset nettoyage.xlsx'
df = pd.read_excel(input_file)

# Initialize dictionary to accumulate sum and profit values
grouped_data = {}

# Process each row and accumulate sum and profit values
for _, row in df.iterrows():
    IDBon = row['IDBon']
    IDProduit = row['IDProduit']
    Qte = row['Qte']
    profile = row['profile']

    if IDBon not in grouped_data:
        grouped_data[IDBon] = {'IDProduit': '', 'Sum': 0, 'Profit': []}

    if grouped_data[IDBon]['IDProduit']:
        grouped_data[IDBon]['IDProduit'] += ' '
    grouped_data[IDBon]['IDProduit'] += str(int(float(IDProduit)))
    grouped_data[IDBon]['Sum'] += profile
    grouped_data[IDBon]['Profit'].append(profile)

# Convert dictionary to DataFrame
grouped_df = pd.DataFrame.from_dict(grouped_data, orient='index')

# Add column for profit of each item
grouped_df['Profit of Each Item'] = grouped_df['Profit'].apply(
    lambda x: ' '.join(str(i) for i in x))

# Save the output to a new Excel file with only three columns
output_file = r'C:\Users\user\Desktop\script BAKOUR MOULOUD\4---Dataset SPMF Format excel.xlsx'
grouped_df.to_excel(output_file, index_label='IDBon', columns=[
                    'IDProduit', 'Sum', 'Profit of Each Item'], header=['ITEMS', 'TU', 'UTILITY'])

```

FIGURE 13:SCRIPT PYTHON (REGROUPE LES ID PRODUIT)

	ITEMS	TU	UTILITY
2	4407	50	50.0
3	548 1377 1483 1663 2621 2790 3122 3240 4951 5854 7396 7732 88	4511	15.0 40.0 40.0 27.0 50.0 68.0 95.0 300.0 30.0 168.0 9.0 72.0 65.0 600.0 160.0 520.0 720.0 150.0 90.0 200.0 140.0 50.0 20.0 380.0 160.0 60.0 50.0 2
4	61 109 2957 12384	425	70.0 55.0 200.0 100.0
5	6149 7623 12650	145	50.0 30.0 65.0
6	44 1552 8531 10612 12616 12616	202	60.0 12.0 25.0 45.0 40.0 20.0
7	4 78 475 5033 10105 10418 11668	500	60.0 55.0 80.0 70.0 55.0 60.0 120.0
8	10967	1060	1060.0

FIGURE 14: LE FICHIER EXCEL RESULTANT REGROUPE LES ID DES PRODUITS

c) Format SPMF

Après avoir regroupé les ID Produit, nous avons développé un script en langage Python(voir *Figure 15*)pour convertir le fichier Excel(voir *Figure 14*) obtenu en un format SPMF. Ce script prend en entrée le fichier Excel obtenu et produit en sortie un format

SPMF(voir *Figure 16*) où chaque ligne est divisée en trois parties distinctes. La première partie contient les ID Produit séparés par ":". La deuxième partie contient l'utilité de la transaction, également séparée par ":". Enfin, la troisième partie contient le profit associé à chaque ID Produit.

```python
import pandas as pd

# Read the input Excel file
input_file = r'C:\Users\user\Desktop\script BAKOUR MOULOUD\4---Dataset SPMF Format excel.xlsx'
df = pd.read_excel(input_file)

# Remove the first column
df = df.iloc[:, 1:]

# Convert DataFrame to text format with ":" separator
text_data = df.astype(str).apply(':'.join, axis=1)

# Save the text data to a text file
output_file = r'C:\Users\user\Desktop\script BAKOUR MOULOUD\Dataset SPMF Format text type Float.txt'
text_data.to_csv(output_file, index=False, header=False)

print("Conversion completed successfully!")
```

FIGURE 15: SCRIPT FORMAT SPMF

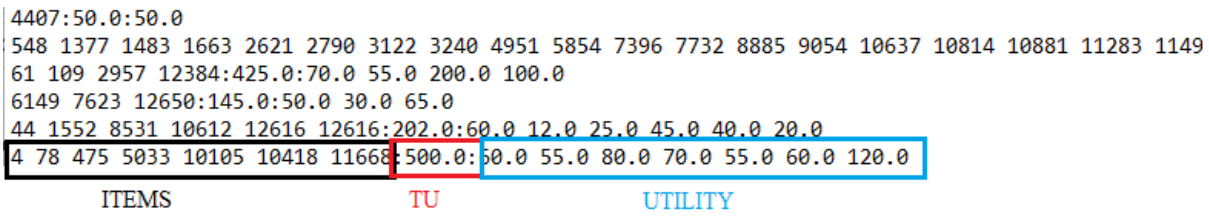


FIGURE 16: LE FORMAT SPMF (AVEC DES VALEURS DE TYPE FLOTTANT)

Nous avons développé un script en langage Python(voir *Figure 17*) pour convertir le format SPMF (avec des valeurs de type flottant)(voir *Figure 16*) en format SPMF (avec des valeurs de type entier)(voir *Figure 18*). Cela est nécessaire car certains algorithmes de HUIM (High Utility Itemset Mining) n'acceptent que des valeurs entières.

```python
# Replace with the actual input file path
input_file_path = r"C:\Users\user\Desktop\script BAKOUR MOULOUD\Dataset SPMF Format text type Float.txt"
# Replace with the desired output file path
output_file_path = r"C:\Users\user\Desktop\script BAKOUR MOULOUD\Dataset SPMF Format text type Integer.txt"

# Read the input file
with open(input_file_path, "r") as input_file:
    lines = input_file.readlines()

# Process each line
transactions = []  # List to store the processed transactions
for line in lines:
    # Split the line using ":" as the separator
    transaction_parts = line.strip().split(":")

    # Convert and keep the individual float values in split_0 separated by spaces
    split_0_values = transaction_parts[0].strip().split()
    split_0 = " ".join(value for value in split_0_values)

    # Convert and keep the individual float values in split_1 separated by spaces
    split_1_values = transaction_parts[1].strip().split()
    split_1 = " ".join(str(int(float(value))) for value in split_1_values)

    # Convert and keep the individual float values in split_2 separated by spaces
    split_2_values = transaction_parts[2].strip().split()
    split_2 = " ".join(str(int(float(value))) for value in split_2_values)

    # Store the processed transaction
    transaction = (split_0, split_1, split_2)
    transactions.append(transaction)

# Write the processed transactions to the output file
with open(output_file_path, "w") as output_file:
    for transaction in transactions:
        output_file.write(
            f"{transaction[0]}:{transaction[1]}:{transaction[2]}\n")
```

FIGURE 17: SCRIPT CONCERT LE FORMAT SPMF (AVEC DES VALEURS DE TYPE FLOTTANT)) EN FORMAT SPMF (AVEC DES VALEURS DE TYPE ENTIER)

```
4407:50:50
548 1377 1483 1663 2621 2790 3122 3240 4951 5854 7396 7732 8885 9054 10637 10814 10881 11283
61 109 2957 12384:425:70 55 200 100
6149 7623 12650:145:50 30 65
44 1552 8531 10612 12616 12616:202:60 12 25 45 40 20
4 78 475 5033 10105 10418 11668:500:60 55 80 70 55 60 120
```

FIGURE 18:LE FORMAT SPMF (AVEC DES VALEURS DE TYPE ENTIER)

d) La deuxième base de donnée utilisée(Chicago_Crimes_2001_to_2017_utility)

La base de données a été convertie à partir de la base de données "Crimes à Chicago" par *Zhongjie Zhang*. Le jeu de données enregistre les crimes qui se sont produits à Chicago de 2001 à 2017.

Chaque transaction correspond à un <mois, zone>. Une transaction décrit les crimes qui se sont produits dans une zone spécifique au cours d'un mois donné. Les éléments représentent les types de crimes, tandis que l'utilité d'un élément dans une transaction indique le nombre d'occurrences de ce crime. Nous avons téléchargé cette base de données à partir du lien suivant https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php#c1

Ou https://www.kaggle.com/datasets/currie32/crimes-in-chicago

TABLE 16:STATISTIQUES DE LA BASE DE DONNEE 1(CHICAGO_CRIMES_2001_TO_2017_UTILITY)

Le nombre des items	30
Le nombre des transactions	367103
La taille de la base de donnée	4,00 Mo
Total utility	1032793
Maximum Id of item	30
Average length of transaction	1.9
Maximum length of transaction	15

3.2.3 L'étape de fouille de données

a) Nous avons téléchargé la bibliothèque SPMF à partir du lien suivant : https://www.philippe-fournier-viger.com/spmf/index.php?link=download.php

FIGURE 19:LA BIBLIOTHEQUE SPMF

b) Nous avons téléchargé Eclipse IDE à partir du lien suivant : https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-developers

Chapitre III: Réalisation & Interprétation

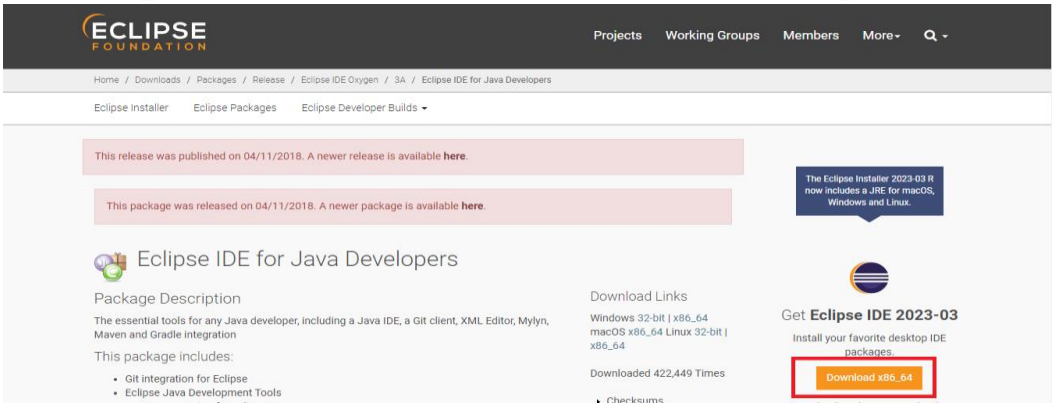


FIGURE 20:ECLIPSE IDE

c) Nous avons créé un nouveau projet Java et ajouté la bibliothèque SPMF au répertoire source (src) de ce projet.

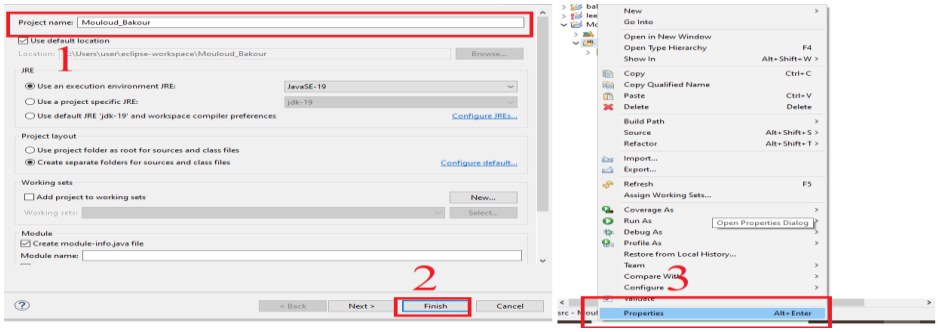


FIGURE 21: CREE UN NOUVEAU PROJET JAVA

```
INPUT FILE: /C:/Users/user/eclipse-workspace/bakour_mouloud/bin/ca/pfv/spmf/test/contextPasquier99.txt
PARAMETERS: [##, 0.6]

TIME (S)
minsup   300000 250000  200000  150000  120000  90000   80000   65000   60000   55000   50000   45000   40000   35000   20000
FHM      1,61    1,31    1,39    1,4     1,46    1,67    1,64    1,82    1,83    1,86    2       2,06    2,16    3,49    2,27

MEMORY (MB)
minsup   300000 250000  200000  150000  120000  90000   80000   65000   60000   55000   50000   45000   40000   35000   20000
FHM      54,81   66,83   80,82   81,12   114,73  32,79   46,43   59,46   65,36   75,9    95,91   58,94   64,5    201,78  201,78

OUTPUT_SIZE (LINES)
minsup   300000 250000  200000  150000  120000  90000   80000   65000   60000   55000   50000   45000   40000   35000   20000
FHM      5       7       10      15      20      40      56      91      108     120     144     166     204     303     1791
```

FIGURE 22:FIGURE 25: EXEMPLE DE RESULTAT DE TEST

Chapitre III: Réalisation & Interprétation

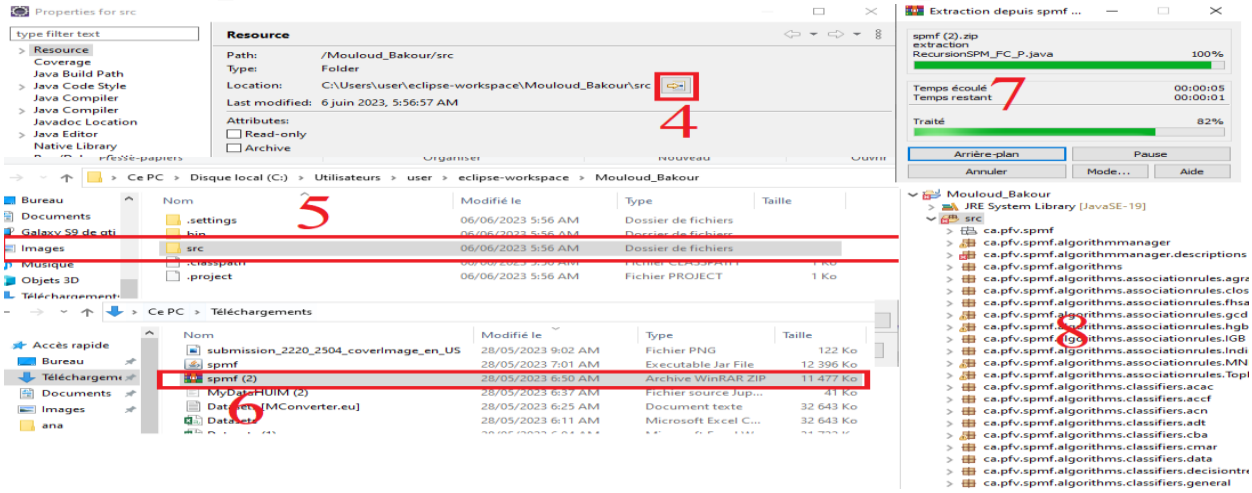


FIGURE 23:AJOUTE LA BIBLIOTHEQUE SPMF

Remarque : *Pour les étapes 6 et 7 vous devez copier et extraire les fichiers de l'archive WinRAR dans le répertoire du même projet*.

d) Nous avons fait des tests

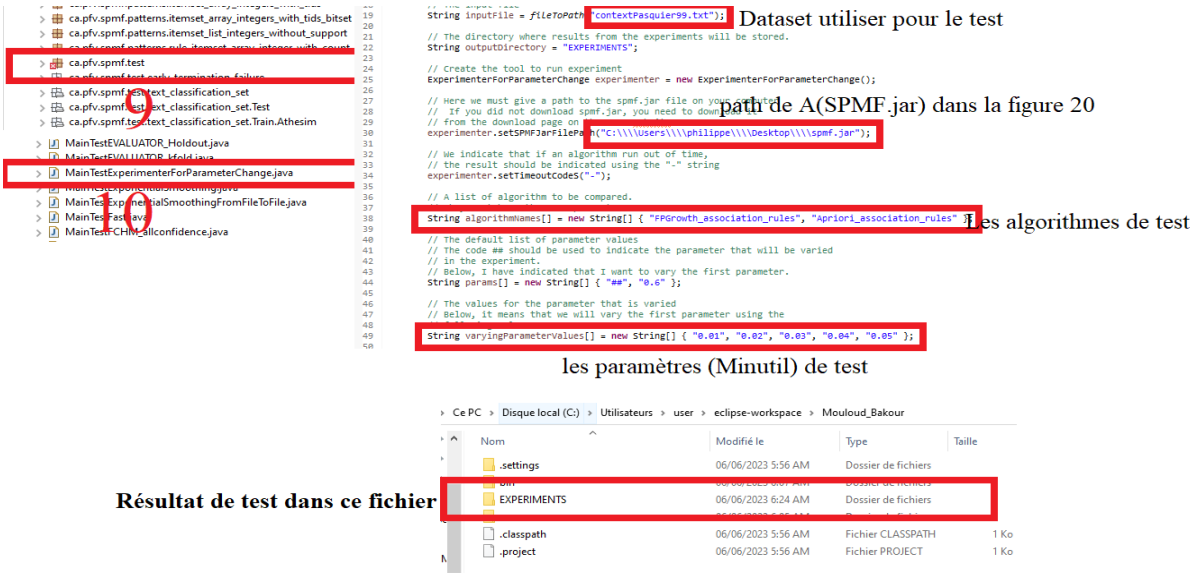


FIGURE 24:EXPERIMENTERFORPARAMETERCHANGE

3.2.4 Interprétation et comparaison les résultats

Nous avons réalisé des expériences pour évaluer les performances de l'algorithme proposé. Les expériences ont été effectuées sur un ordinateur équipé d'un processeur *Core i5 de 6e génération 64 bits*, fonctionnant sous *Windows 10* et disposant de *8 Go de RAM libre*.

Chapitre III: Réalisation & Interprétation

Nous avons comparé les performances de *FHM* avec *HUI-Miner*, *UPGrowth*,en HUIM,CHUI-Miner, GHUI-Miner et MinFHM en représentations concises.

TABLE 17:RESULTATS DE L'EXECUTION DES ALGORITHMES FHM, HUI-MINER,UPGROWTH,MINFHM,CHUI-MINER ET GHUI-MINER SUR UN DATASET COSMETIQUES

algorithme		**Minutil**							
		35000	**45000**	**55000**	**80000**	**90000**	**120000**	**200000**	**300000**
FHM	**temps d'exécution (S)**	3,99	2,28	2,09	1,73	1,82	1,56	1,41	1,38
	Mémoire (MB)	213,21	107,5	88	47,64	76,38	114,24	80,9	55,25
	nombre de HUIM	303	166	120	56	40	20	10	5
HUI-Miner	**temps d'exécution (S)**	14,36	7,92	6,85	3,95	3,47	2,21	1,2	1,17
	mémoire	352,97	63,28	31,18	86	75,46	61,6	38,54	28,85
	nombre de HUIM	303	166	120	56	40	20	10	5
UP Growth	**temps d'exécution (S)**	39,32	6,68	5,6	3,53	3,15	2,68	1,99	1,75
	mémoire	143,61	143,61	37,83	201	173,37	66,23	134,75	71,57
	nombre de HUIM	-	271	180	74	51	31	16	10
Min	**temps d'exécution**	1,55	1,62	1,6	1,58	1,63	1,5	1,4	1,38

FHM	**Mémoire**	52,5	61,85	54,42	76,7	74,5	41,81	89,11	87,38
	nombre de HUIM	223	154	113	54	38	18	9	5
CHUI-Miner	**temps d'exécution**	2,79	2,58	2,49	1,87	1,75	1,57	1,35	1,31
	Mémoire (MB)	99,5	105	157,99	92,5	74,46	45,22	71,5	46,6
	nombre de HUIM	260	166	120	56	40	20	10	5
GHUI-Miner	**temps d'exécution**	6,91	3,02	2,9	2,53	2,33	2,08	1,85	1,93
	Mémoire (MB)	382,52	192,04	94,89	122,63	106,82	81,99	47,87	63,12
	nombre de HUIM	32167	166	120	56	40	20	10	5

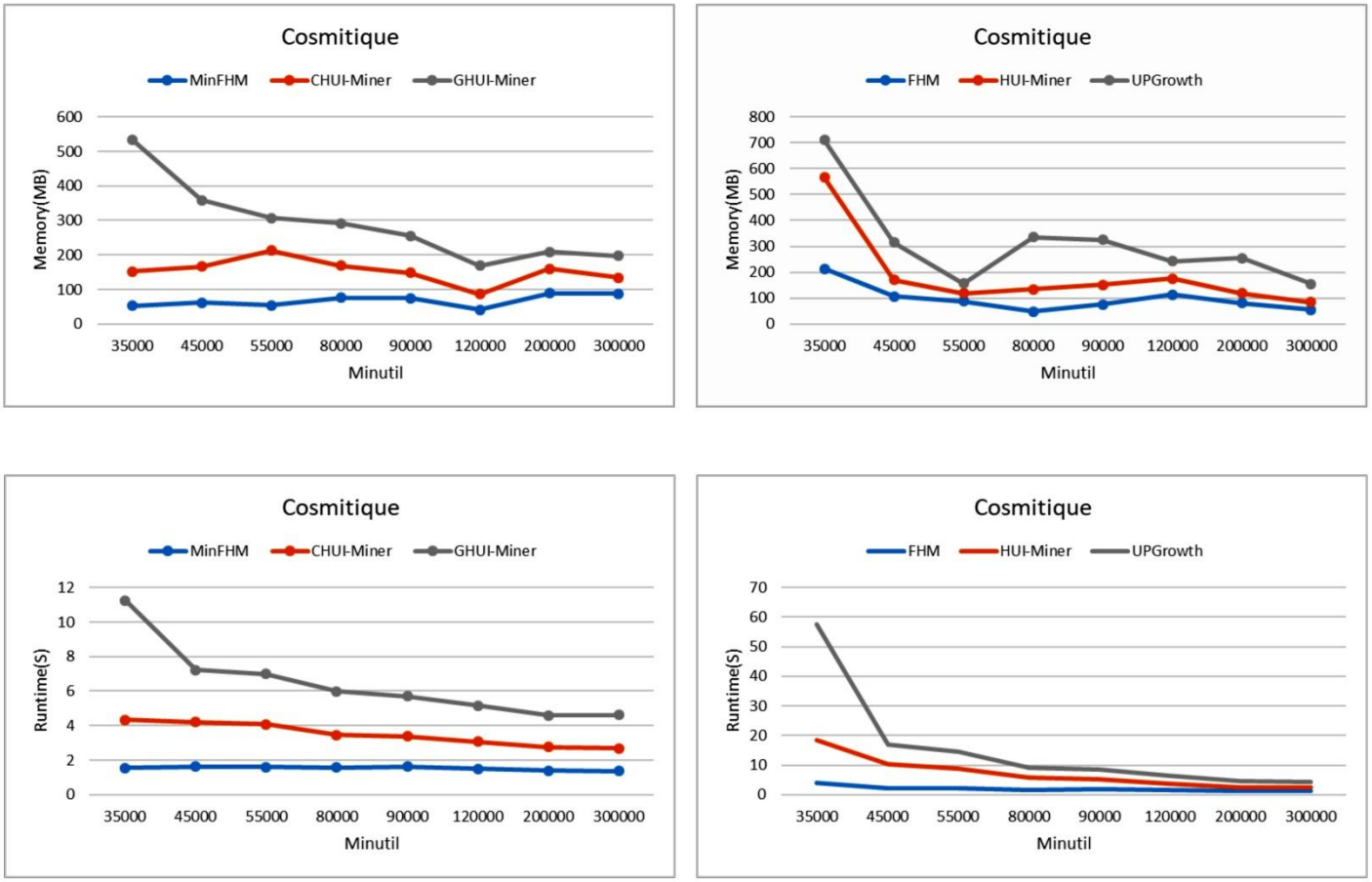


FIGURE 25:RESULTATS DE L'EXECUTION DES ALGORITHMES FHM, HUI-MINER,UPGROWTH,MINFHM,CHUI-MINER ET GHUI-MINER SUR UN DATASET COSMETIQUES

TABLE 18:RESULTATS DE L'EXECUTION DES ALGORITHMES FHM, HUI-MINER,UPGROWTH,MINFHM,CHUI-MINER ET GHUI-MINER SUR UN DATASET CHICAGO_CRIMES_2001_TO_2017_UTILITY

algorithme		Minutil							
		1	10	100	1000	10000	50000	100000	200000
FHM	**temps d'exécution (S)**	2,12	2,27	1,88	1,81	1,6	1,42	1,34	1,18
	Mémoire (MB)	76,25	89,12	62,49	43,06	94,43	75,4	59,06	30,57
	nombre de HUIM	131989	121104	50432	5297	247	23	6	1

HUI-Miner	**temps d'exécution (S)**	2,02	2,12	1,81	1,77	1,6	1,45	1,52	1,28
	mémoire	74,46	72,49	24,43	108,98	79,26	59,98	65,55	34,03
	nombre de HUIM	131989	121104	50432	5297	247	23	6	1
UP Growth	**temps d'exécution (S)**	36,97	34,53	20,1	7,47	2,83	2,06	1,73	1,7
	mémoire	98,96	98,03	24,43	85,23	64,73	61,52	61,48	52,53
	nombre de HUIM	131989	121104	50432	5297	247	23	6	1
Min FHM	**temps d'exécution (S)**	0,99	1,06	1,14	1,24	1,23	1,2	1,3	1,38
	Mémoire (MB)	12,73	22,43	19,95	21,53	24,76	16,85	54	65,29
	nombre de HUIM	30	29	25	22	12	8	4	1
CHUI-Miner	**temps d'exécution (S)**	2,44	2,48	2,3	2,12	1,82	1,57	1,68	1,36
	Mémoire (MB)	24,15	54,76	48,15	109,48	88,84	55,1	49,68	19,72
	nombre de HUIM	20626	20383	15295	4265	247	23	6	1
GHUI-	**temps d'exécution**	5,94	5,78	5,43	4,03	2,75	2,29	2,03	1,86

Miner	Mémoire (MB)	119,3	390,62	355,83	187,04	170,04	199,51	145,09	65,93
	nombre de HUIM	25238	24876	18114	5358	274	23	6	1

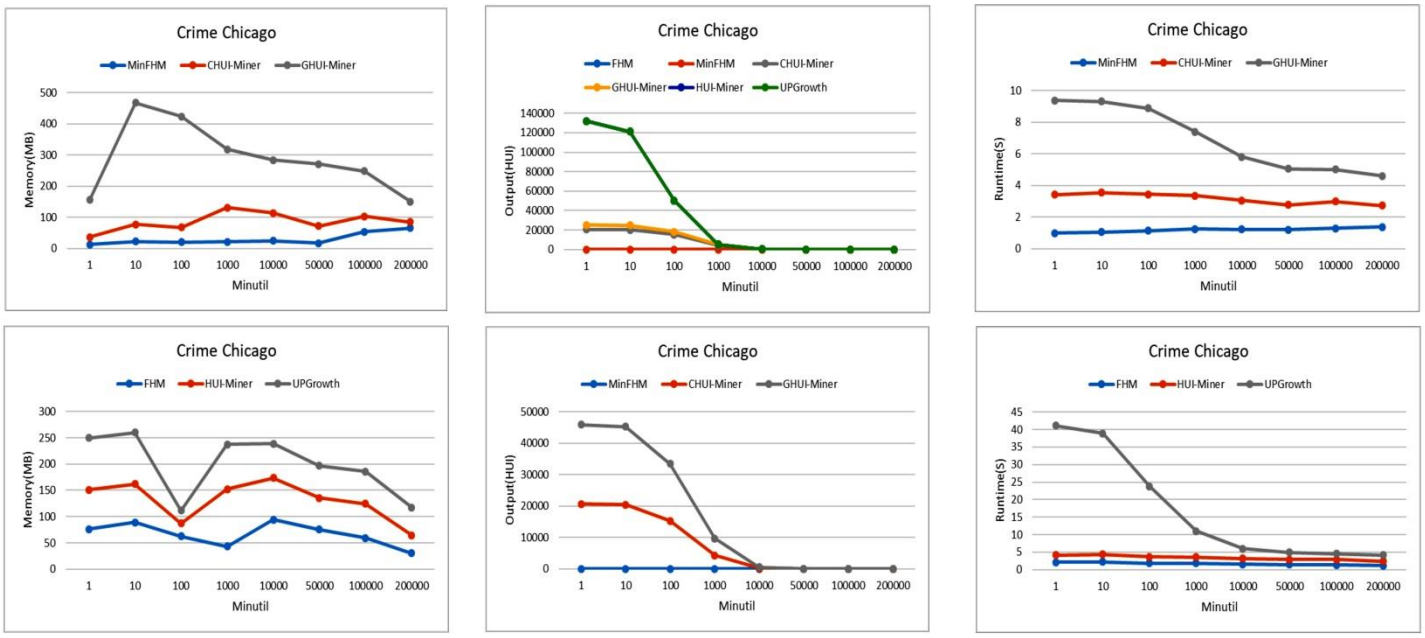


FIGURE 26:RESULTATS DE L'EXECUTION DES ALGORITHMES FHM, HUI-MINER,UPGROWTH,MINFHM,CHUI-MINER ET GHUI-MINER SUR UN DATASET CHICAGO_CRIMES_2001_TO_2017_UTILITY

3.2.5 High Utility Itemset Mining

Nous avons d'abord exécuté les algorithmes FHM, HUI-Miner et UPGrowth sur chaque ensemble de données, en réduisant progressivement le seuil minutil jusqu'à ce que les algorithmes prennent trop de temps pour s'exécuter, dépassent la mémoire disponible ou qu'un gagnant clair soit observé. Pour chaque ensemble de données, nous avons enregistré le temps d'exécution(S), la MÉMOIRE (MB) et la TAILLE DE SORTIE (nombre d'ensembles d'éléments à haute utilité).

La comparaison des temps d'exécution est présentée dans (***Figure 25,Figure 26").*** Sur chaque ensemble de données, FHM est le premier algorithme, suivi de HUI-Miner, puis de UPGrowth, qui prend beaucoup de temps.

La comparaison de la mémoire utilisée est présentée dans (***Figure 26,Figure 25"***). Sur tous les ensembles de données, FHM utilise moins d'espace que HUI-Miner, tandis que UPGrowth utilise une quantité considérable de mémoire.

3.2.6 Représentations concises

Dans un premier temps, nous avons exécuté les algorithmes MinFHM, CHUI-Miner et GHUI-Miner sur chaque ensemble de données. Nous avons progressivement réduit le seuil minutil jusqu'à ce que les algorithmes prennent trop de temps pour s'exécuter, dépassent la mémoire disponible ou qu'un gagnant clair soit observé. Pour chaque ensemble de données, nous avons enregistré les temps d'exécution (S), la mémoire utilisée (MB) et la taille de la sortie (nombre d'ensembles d'éléments à haute utilité).

La comparaison des temps d'exécution est présentée dans (***Figure 25,Figure 26"***). Sur l'ensemble de données des cosmétiques,MinFHM est le premier algorithme, suivi de CHUI-Miner, puis de GHUI-Miner. Sur l'ensemble de données des crimes de Chicago de 2001 à 2017 (***Figure 26,Figure 25)***MinFHM est également le premier algorithme, suivi de CHUI-Miner, puis de GHUI-Miner, qui prend beaucoup de temps.

La comparaison de la mémoire utilisée est présentée dans (***Figure 25,Figure 26***). Sur tous les ensembles de données, MinFHM utilise moins d'espace que CHUI-Miner, tandis que GHUI-Miner utilise une quantité considérable de mémoire.

La comparaison de la taille de la sortie (nombre d'ensembles d'éléments à haute utilité) est présentée dans (***Figure 25,Figure 26***). Dans la (***Figure 25***), MinFHM retourne moins d'ensembles d'éléments à haute utilité que CHUI-Miner, puis GHUI-Miner en retourne un grand nombre.

Dans la (***Figure 26***), les algorithmes de représentation concise retournent moins d'ensembles d'éléments à haute utilité que les algorithmes de recherche d'ensembles d'éléments à haute utilité.

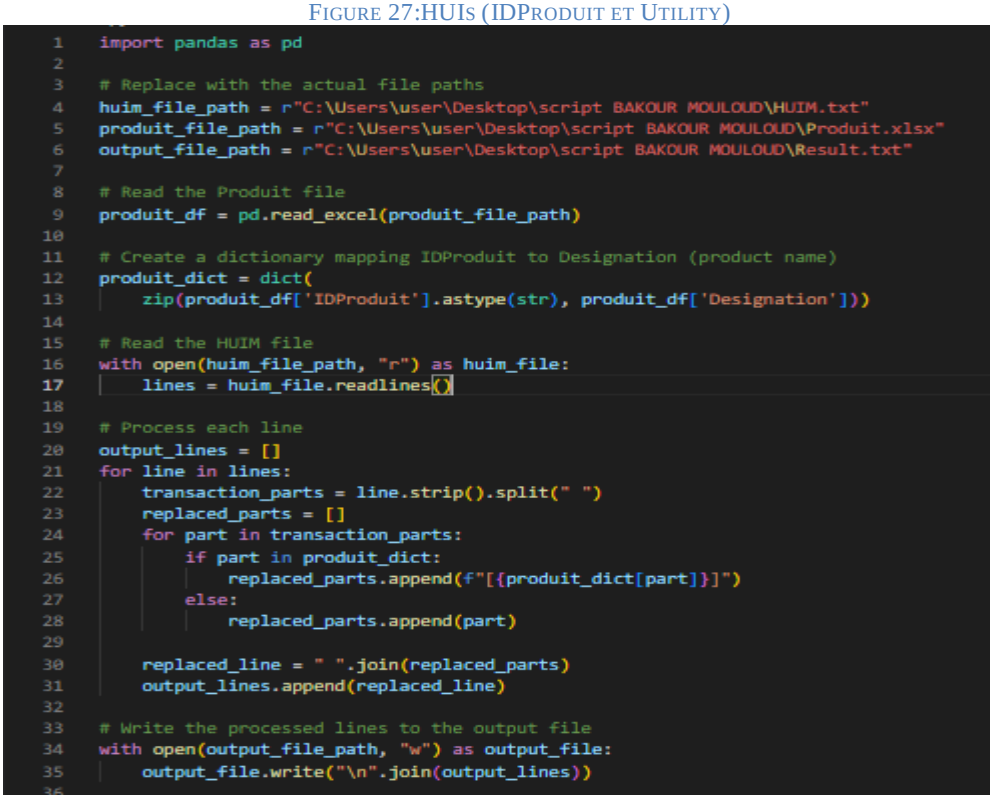
3.2.7 Visualisation les résultats

Nous avons développé un script Python qui prend en entrée un fichier texte contenant les HUI (IDProduit) avec leurs utility, ainsi qu'un autre fichier Excel comportant deux colonnes (la première colonne contenant les IDProduit et la deuxième colonne contenant les

désignations des produits). Le script renvoie en sortie un fichier texte contenant les HUI (noms des produits) avec leurs utility.

```
11407 #UTIL: 86080
8755 #UTIL: 76000
629 #UTIL: 76000
8998 #UTIL: 164000
7449 #UTIL: 101700
6340 #UTIL: 118080
9733 #UTIL: 94800
3424 #UTIL: 106750
12052 #UTIL: 66360
```

FIGURE 27:HUIS (IDPRODUIT ET UTILITY)



```python
import pandas as pd

# Replace with the actual file paths
huim_file_path = r"C:\Users\user\Desktop\script BAKOUR MOULOUD\HUIM.txt"
produit_file_path = r"C:\Users\user\Desktop\script BAKOUR MOULOUD\Produit.xlsx"
output_file_path = r"C:\Users\user\Desktop\script BAKOUR MOULOUD\Result.txt"

# Read the Produit file
produit_df = pd.read_excel(produit_file_path)

# Create a dictionary mapping IDProduit to Designation (product name)
produit_dict = dict(
    zip(produit_df['IDProduit'].astype(str), produit_df['Designation']))

# Read the HUIM file
with open(huim_file_path, "r") as huim_file:
    lines = huim_file.readlines()

# Process each line
output_lines = []
for line in lines:
    transaction_parts = line.strip().split(" ")
    replaced_parts = []
    for part in transaction_parts:
        if part in produit_dict:
            replaced_parts.append(f"[{produit_dict[part]}]")
        else:
            replaced_parts.append(part)

    replaced_line = " ".join(replaced_parts)
    output_lines.append(replaced_line)

# Write the processed lines to the output file
with open(output_file_path, "w") as output_file:
    output_file.write("\n".join(output_lines))
```

FIGURE 28SCRIPT PYTHON CONVERTIR LES RESULTAT

```
[FDT INFALLIBLE LOREAL 1ER] #UTIL: 66420
[LANTIE NATURAL COLORS] #UTIL: 68200
[SH LOREAL EXPERTISE 250ML] #UTIL: 65330
[COF DUBAI ] #UTIL: 83160
[BLONDORE LOREAL INFINIE PLATINE 500G] #UTIL: 76050
[DOSE KIRATINE 65ML] #UTIL: 86080
[APAREIL CIRE PRO WAX11] #UTIL: 76000
[BLOND DOR CHIC SILVER 400G] #UTIL: 76000
[REMINGTON 5890] #UTIL: 164000
[STICKE FOR EVER] #UTIL: 101700
[FDT LOREAL INFALLIBLE] #UTIL: 118080
```

FIGURE 29:LES RESULTAT(DESIGNATION ET UTILITY)

3.2.8 Application de bureau

Nous avons développé une application de bureau pour regrouper tous les scripts précédents.

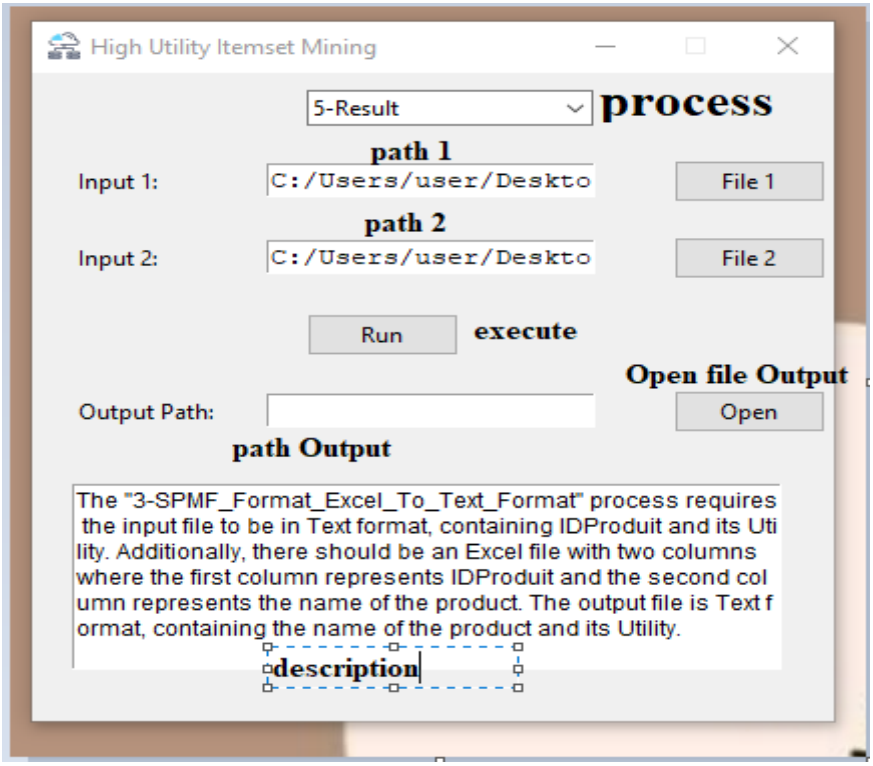


FIGURE 30:APPLICATION DE BUREAU

```
C: > Users > user > Desktop > desktop > app.py > handle_combobox_selection
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
from tkinter.filedialog import askopenfilenames
import pandas as pd
import os
import threading
import tkinter.messagebox as messagebox
from PIL import Image, ImageTk


def Integer():
    def cleaning_thread():
        elementToAnimate = ttk.Label(frame, text="Processing...")
        elementToAnimate.grid(row=0, column=0)
        # Replace with the actual input file path
        input_file_path = input_text1.get("1.0", tk.END).strip()
        # Replace with the desired output file path
        directory = os.path.dirname(input_file_path)
        output_path = os.path.join(
            directory, '5---Dataset SPMF Format text type Integer.txt')

        # Read the input file
        with open(input_file_path, "r") as input_file:
            lines = input_file.readlines()

```

FIGURE 31:CODE SOURCE DE APPLICATION DE BUREAU

4 Conclusion

En conclusion de ce chapitre, nous avons examiné les outils de développement utilisés dans notre projet, tels que Eclipse, Java, Python, Visual Studio et la bibliothèque SPMF. Nous avons décrit comment ces outils ont été configurés et intégrés dans notre environnement de développement pour mettre en œuvre les algorithmes de fouille d'ensembles d'items à haute utilité.

Nous avons constaté que l'utilisation d'Eclipse comme IDE principal nous a permis de bénéficier de fonctionnalités avancées telles que l'autocomplétion, le débogage et la gestion de projet, facilitant ainsi le développement de notre application.

Java et Python se sont révélés être des langages de programmation adaptés à notre projet. Java nous a offert la simplicité et la performance nécessaires pour implémenter efficacement les algorithmes de fouille d'ensembles d'items à haute utilité, tandis que Python nous a apporté sa facilité d'utilisation et sa flexibilité dans certaines parties spécifiques de notre projet.

L'intégration de la bibliothèque SPMF a été cruciale pour l'extraction des ensembles d'items fréquents. Cette bibliothèque open-source en Java a fourni des fonctionnalités spécialisées dans la fouille de données séquentielles, ce qui a facilité notre travail dans ce domaine.

Enfin, nous avons fourni une description détaillée de l'architecture générale de notre implémentation, en expliquant la structure du code, les différentes classes et modules utilisés pour mettre en œuvre les algorithmes de fouille d'ensembles d'items à haute utilité. Nous avons également souligné l'importance des considérations de performance, telles que l'utilisation efficace de la mémoire et l'optimisation du temps d'exécution, qui ont été prises en compte tout au long de l'implémentation des algorithmes.

Ce chapitre nous a permis de mieux comprendre les outils et les technologies utilisés dans notre projet, ainsi que l'architecture générale de notre implémentation. Ces informations sont essentielles pour assurer le bon déroulement du développement et l'efficacité des algorithmes de fouille d'ensembles d'items à haute utilité.L'implémentation pratique de FHM et MinFHM a été réalisée en utilisant des outils tels que Eclipse, Java, Python et Visual Studio, ainsi que la bibliothèque SPMF. Ces outils ont été essentiels pour la mise en œuvre efficace des

algorithmes et pour les tests sur des ensembles de données réels. Les résultats obtenus ont démontré l'efficacité des algorithmes dans l'extraction précise des ensembles d'items à haute utilité.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons exploré en détail le domaine de la fouille de données, en mettant particulièrement l'accent sur la fouille d'ensembles d'items à haute utilité (HUIM). Nous avons étudié les techniques d'extraction de connaissances utilisées dans ce domaine et analysé les algorithmes clés pour extraire des ensembles d'items à haute utilité.

Tout d'abord, nous avons compris l'importance croissante de la fouille de données dans notre monde moderne, où la quantité de données générées chaque jour est immense. La fouille de données nous offre une opportunité précieuse de découvrir des connaissances cachées, des modèles et des tendances à partir de ces données massives.

Ensuite, nous avons examiné les différentes tâches et applications de la fouille de données, en comprenant les étapes du processus de fouille de données, de la préparation des données à l'interprétation des résultats. Nous avons également exploré l'évolution de la fouille de données dans le contexte des systèmes de bases de données, en comprenant comment elle s'est développée pour répondre aux défis croissants liés à l'analyse des données.

Dans le chapitre central de ce mémoire, nous avons sommes concentrés sur la fouille des ensembles d'items à haute utilité. Nous avons étudié les algorithmes d'extraction des ensembles d'items fréquents, tels que la recherche en largeur et la recherche en profondeur, ainsi que l'algorithme Apriori. Nous avons souligné les limitations de la fouille des ensembles d'items fréquents et avons identifié la nécessité d'extraire des ensembles d'items ayant un impact significatif sur une mesure spécifique.

Pour répondre à cette nécessité, nous avons présenté les algorithmes FHM (Fast High Utility Itemset Mining) et MinFHM (Minimal Fast High Utility Itemset Mining) pour l'extraction des ensembles d'items à haute utilité. Nous avons décrit en détail le fonctionnement de ces algorithmes, en mettant en évidence leurs concepts fondamentaux et leurs principales propriétés. Nous avons également clarifié que l'EUCS (Efficient Utility Counting Strategy) et l'utilité list sont des structures de données utilisées pour réduire l'espace de recherche et accélérer le calcul de l'utilité des itemsets.

La fouille de données, notamment la fouille d'ensembles d'items à haute utilité, offre des opportunités passionnantes pour découvrir des connaissances précieuses à partir de données massives. Les techniques et les algorithmes étudiés dans ce mémoire constituent une base

solide pour l'extraction de connaissances dans divers domaines tels que le commerce, la santé et la finance. Ces approches permettent d'identifier des ensembles d'items ayant un impact significatif sur des mesures spécifiques, facilitant ainsi la compréhension des modèles et des tendances cachés dans les données.

Perspectives

Dans nos perspectives, notre objectif principal est d'améliorer les performances en termes de temps d'exécution, d'utilisation de la mémoire et du nombre d'ensembles d'items à haute utilité (HUI) retournés.

En ce qui concerne le temps d'exécution, nous visons à développer des algorithmes plus efficaces et optimisés pour extraire les ensembles d'items à haute utilité. Cela peut impliquer l'utilisation de structures de données avancées ou de techniques de recherche permettant de réduire le temps nécessaire pour explorer l'espace de recherche et identifier rapidement les HUI pertinents.

Nous avons également observé que les algorithmes de recherche d'items fréquents se concentrent uniquement sur le nombre d'occurrences des items, indépendamment de leur utilité, tandis que les algorithmes de recherche d'ensembles d'items à haute utilité se concentrent uniquement sur l'utilité des items. Cependant, dans les deux cas (FIM et HUIM), le temps n'est pas considéré comme important. L'objectif principal est de déterminer si un item est fréquent ou s'il a une haute utilité, sans tenir compte du temps nécessaire pour réaliser cette évaluation.

Dans nos futures recherches, nous souhaitons introduire le temps en tant que facteur clé, en combinaison avec l'utilité, dans les méthodes de recherche d'ensembles d'items à haute utilité. Cela nous permettra de prendre en compte à la fois l'utilité des items et le temps nécessaire pour les extraire, ce qui conduira à des résultats plus pertinents et plus efficaces. Ainsi, nous serons en mesure d'identifier les HUI qui sont à la fois significatifs en termes d'utilité élevée et extraits dans une petite période de temps.

Bibliographiques

[1] S. Bandyopadhyay and U. Maulik, “Knowledge Discovery and Data Mining,” in *Advanced Methods for Knowledge Discovery from Complex Data*, Springer-Verlag, 2006, pp. 3–42. doi: 10.1007/1-84628-284-5_1.

[2] J. Han, “Data Mining: Concepts and Techniques Second Edition.” [Online]. Available: www.mkp.com

[3] J. Han, “Data Mining: Concepts and Techniques Second Edition.” [Online]. Available: www.mkp.com

[4] G. Piatetsky-Shapiro, “Data mining and knowledge discovery 1996 to 2005: Overcoming the hype and moving from ‘university’ to ‘business’ and ‘analytics,’” *Data Min Knowl Discov*, vol. 15, no. 1, pp. 99–105, Aug. 2007, doi: 10.1007/s10618-006-0058-2.

[5] D. J. (David J.) Hand, Heikki. Mannila, and Padhraic. Smyth, *Principles of data mining*. MIT Press, 2001.

[6] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, “Knowledge Discovery in Databases: An Overview,” 1992.

[7] B. Agard, C. Morency, M. Trépanier, and G. Madituc, “MINING PUBLIC TRANSPORT USER BEHAVIOUR FROM SMART CARD DATA,” 2006.

[8] M. J. A. Berry, Gordon. Linoff, and M. J. A. Berry, *Mastering data mining : the art and science of customer relationship management*. Wiley Computer Pub, 2000.

[9] M. J. Berry, G. S. Linoff, and F. Marketing, “Customer Relationship Management Second Edition Data Mining Techniques.”

[10] D. T. Larose and C. D. Larose, “DISCOVERING KNOWLEDGE IN DATA An Introduction to Data Mining Second Edition Wiley Series on Methods and Applications in Data Mining.”

[11] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules.”

[12] P. Fournier-Viger, J. Chun-Wei Lin, T. Truong Chi, and R. Nkambou, “A Survey of High Utility Itemset Mining.”

[13] P. Fournier, V. Jerry, C.-W. Lin, R. Nkambou, B. Vo, and V. S. Tseng Editors, “Studies in Big Data 51 High-Utility Pattern Mining Theory, Algorithms and Applications.” [Online]. Available: http://www.springer.com/series/11970

[14] J. M. Luna, P. Fournier-Viger, and S. Ventura, “Frequent Itemset Mining: a 25 Years Review.”

[15] P. Fournier-Viger *et al.*, “A Survey of Itemset Mining.”

[16] P. Fournier-Viger *et al.*, “A Survey of Itemset Mining.”

[17] R. Agrawal and R. S&ant, “Fast Algorithms for Mining Association Rules.”

[18] P. Fournier-Viger, J. Chun-Wei Lin, T. Truong Chi, and R. Nkambou, “A Survey of High Utility Itemset Mining.”

[19] P. Pardeshi1 and U. Patil2, “Mining Top-k High Utility Itemset using Efficient Algorithms,” *International Journal of Science and Research*, doi: 10.21275/ART20194775.

[20] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, “FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning.”

[21] V. S. Tseng, C. W. Wu, P. Fournier-Viger, and P. S. Yu, “Efficient Algorithms for Mining Top-K High Utility Itemsets,” in *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society, Jan. 2016, pp. 54–67. doi: 10.1109/TKDE.2015.2458860.

[22] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, “FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning.”

[23] Q. H. Duong, B. Liao, P. Fournier-Viger, and T. L. Dam, “An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies,” *Knowl Based Syst*, vol. 104, pp. 106–122, Jul. 2016, doi: 10.1016/j.knosys.2016.04.016.

[24] V. S. Tseng, C. W. Wu, P. Fournier-Viger, and P. S. Yu, “Efficient algorithms for mining the concise and lossless representation of high utility itemsets,” *IEEE Trans Knowl Data Eng*, vol. 27, no. 3, pp. 726–739, Mar. 2015, doi: 10.1109/TKDE.2014.2345377.

[25] B. E. Shie, P. S. Yu, and V. S. Tseng, “Efficient algorithms for mining maximal high utility itemsets from data streams with different models,” *Expert Syst Appl*, vol. 39, no. 17, pp. 12947–12960, Dec. 2012, doi: 10.1016/J.ESWA.2012.05.035.

[26] P. Fournier-Viger, C. W. Wu, and V. S. Tseng, “Novel concise representations of high utility itemsets using generator patterns,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8933, pp. 30–43, 2014, doi: 10.1007/978-3-319-14717-8_3.

[27] P. Fournier-Viger, J. Chun-Wei Lin, C.-W. Wu, V. S. Tseng, and U. Faghihi, “Mining Minimal High-Utility Itemsets.”

[28] “Eclipse (projet) — Wikipédia.” https://fr.wikipedia.org/wiki/Eclipse_(projet) (accessed May 29, 2023).

[29] “Qu’est-ce que Java et pourquoi en ai-je besoin ?” https://www.java.com/fr/download/help/whatis_java.html (accessed May 29, 2023).

[30] “Python (langage) — Wikipédia.” https://fr.wikipedia.org/wiki/Python_(langage) (accessed May 30, 2023).

[31] “Visual Studio - Wikipedia.” https://en.wikipedia.org/wiki/Visual_Studio (accessed May 30, 2023).

[32] “SPMF: A Java Open-Source Data Mining Library.” https://www.philippe-fournier-viger.com/spmf/index.php?link=algorithms.php (accessed May 29, 2023).