UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
MOHAMED AMIR KOALAL

VERS UNE APPROCHE DE RÉDUCTION DU NOMBRE DE RÈGLES D'ASSOCIATION

MAI 2022

Résumé

Les règles d'association est une méthode transparente qui permet de comprendre les raisons qui justifient les résultats obtenus. Toutefois, la méthode des règles d'association souffre d'une limitation majeure. Le nombre de règles obtenues peut parfois être très important. Selon plusieurs travaux, les règles ayant des scores inférieurs à des seuils préalablement établis sont supprimées. Ce principe a longtemps été utilisé pour ne pas avoir d'explosions combinatoires. Néanmoins, il a souvent impliqué la perte de règles d'association importantes, qui en raison du contexte de leur extraction ne pouvaient être récurrentes et donc ne pouvaient avoir des scores supérieurs à ceux établis par les utilisateurs. D'autre part, il est évident que les seuils sont généralement établis d'une façon quasi arbitraire.

Nous proposons dans notre mémoire une approche permettant de réduire le nombre de règles d'association en utilisant une méthode largement utilisée dans la simplification des expressions de la logique booléenne, à savoir la méthode de Quine-McClusckey. Notre méthodologie se distingue par deux « phases ». Dans la première phase, le nombre de règles d'association initiales est réduit en appliquant la méthode de Quine-McCluskey. Dans la deuxième phase, nous vérifions si notre approche engendre une perte des informations importantes.

Les expérimentations que nous avons menées ont montré que notre approche réduit significativement le nombre de règles, tout en gardant les informations importantes.

Abstract

Association rules is a transparent method that allows us to understand the reasons behind the results obtained. However, the method of association rules suffers from a major limitation. The number of rules obtained can sometimes be very large. According to several works, rules with scores below previously established thresholds are deleted. This principle has been used for a long time to avoid combinatorial explosions. Nevertheless, it has often implied the loss of important association rules, which due to the context of their extraction could not be recurrent and therefore could not have scores higher than those established by the users. On the other hand, it is obvious that thresholds are usually established in a quasi-arbitrary way.

In our dissertation, we propose an approach to reduce the number of association rules by using a method widely used in the simplification of Boolean logic expressions, namely the Quine-McClusckey method. Our methodology is characterized by two "phases". In the first phase, the number of initial association rules is reduced by applying the Quine-McCluskey method. In the second phase, we measure the rate of loss of important information.

The experiments we conducted showed that our approach significantly reduces the number of rules, while keeping the important information.

Remerciement

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon parcours et qui m'ont aidée lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps remercier, mon directeur de recherche Ismail Biskri, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion. Je serais toujours honoré de travailler avec lui.

Je tiens aussi à remercier, mes professeurs évaluateurs Nadia Ghazzali et Boucif Amar Bensaber qui ont accepté d'évaluer mon travail.

Je voudrais exprimer ma reconnaissance envers les amis et collègues de laboratoire qui m'ont apporté leur soutien moral et intellectuel tout au long de mes études.

Je remercie également toutes les personnes, qui de prêt ou de loin, m'ont soutenu et encouragé pour réussir mes études.

Enfin, je tiens à montrer ma gratitude envers mes parents et mes proches pour leurs soutiens, encouragements et leur aide si précieuse durant mes études.

Table des matières

Liste des Figures

Liste des Tableaux

Liste des algorithmes

Chapitre I: Introduction

Chapitre I

Introduction

Au cours des dernières années, le forage de données a permis de réaliser des prouesses extraordinaires. Ce dernier est très utilisé dans la prise des décisions intelligentes sur les marchés ; faire des prédictions ; classifier des objets ; analyser les comportements ; détecter les intrusions, etc.

Le forage de données est une branche de l'intelligence artificielle dont le concept est de plus en plus populaire dans de nombreux domaines. Fondamentalement, là où il y a des données, le forage de données est applicable. En fait, dans la plupart des transactions que nous faisons, des données sont générées et téléchargées dans des banques de données, sous plusieurs formes telles que les images ; les textes ; les sons, etc. Les organisations stockent, traitent et analysent les données de plus en plus, et cette tendance n'a jamais arrêté de s'accentuer.

Plusieurs méthodes ont été introduites pour réaliser les différentes tâches du forage de données. Parmi celles-ci, on trouve une méthode largement utilisée, à savoir les règles d'association. Ces dernières présentent l'information sous une forme relationnelle, entre des événements antécédents menant à des conséquents. Leur efficacité et leur facilité d'application rendent leurs utilisations très vastes. Elles sont notamment populaires dans les prédictions des achats aussi appelées le panier de la ménagère [2], en traitement de textes [4], etc.

Les règles d'association permettent l'extraction des connaissances cachées, à partir d'un grand volume de données. Toutefois, l'inconvénient des règles d'association est leur grand nombre après leur extraction [4]. Cette problématique a fait l'objet de plusieurs recherches visant à réduire le nombre de règles d'association. Parmi les approches proposées, on trouve l'utilisation de métriques visant à limiter leur nombre. Bien que le nombre de règles puisse être réduit par ces métriques, ces approches peuvent engendrer la perte de certaines règles d'association importantes.

Notre objectif de recherche dans ce travail est de mettre en place une approche permettant de réduire le nombre de règles sans perdre les informations importantes. Nous avons utilisé une méthode de minimisation de fonctions booléennes, à savoir la méthode de Quine-McCluskey. Cette dernière peut être appliquée à tout type de données et ne nécessite que des données binarisées. De plus, aucun paramètre ou métrique n'est requis de l'utilisateur.

Les différentes parties de notre implémentation permettent d'appliquer la méthode de Quine-McClusckey aux règles d'association en transformant les règles sous la forme booléenne. Ils permettent aussi de revenir depuis la forme booléenne vers la forme d'une règle d'association pour faciliter la lisibilité des règles à l'utilisateur.

Dans notre approche, nous avons utilisé la classification de textes arabes, de programmes exécutables et enfin de fleurs Iris, pour vérifier si notre approche engendre une perte d'informations importantes après la réduction du nombre de règles d'association. Notre choix s'est posé sur les systèmes de classification du fait que ces derniers sont très sensibles aux changements dans leurs bases de connaissances. En effet, si des informations importantes ont été perdues après la réduction, la précision de la classification sera négativement impactée. Nous étudions la perte des informations importantes comme suit :

1 : dans le cas où le système de classification conserve sa performance après la réduction (pas de baisse significative du taux de précision de la classification). Alors, les informations importantes ont été conservées.

2 : dans l'éventualité où le système de classification perd significativement sa performance après la réduction (baisse notable du taux de précision de la classification). Alors, les informations importantes ont été perdues.

Le présent mémoire est constitué de sept chapitres. Dans le **deuxième chapitre**, nous présenterons les règles d'association et les principales mesures utilisées dans leur extraction, comme : le support, la confiance et le lift. Puis nous parlerons par la suite de quelques algorithmes d'extraction des règles d'association et nous donnerons les détails des principales étapes de ces derniers.

Le **troisième chapitre** présente l'état de l'art, dans lequel nous présenterons les détails de quelques travaux dans la littérature, qui ont proposé des approches qui traitent des règles d'association.

Le **quatrième chapitre** présentera la méthode de Quine-McCluskey. Nous avons opté pour cette méthode de réduction de fonction booléenne dans le but de réduire le nombre de règles d'association et la sauvegarde des informations importantes. Nous commençons par l'origine de la méthode de Quine-McCluskey puis nous présenterons son principe de base. Par la suite, nous montrons, exemple à l'appui, les étapes de fonctionnement de la méthode.

Le **cinquième chapitre** présente notre méthodologie pour la réduction du nombre de règles d'association. Nous présenterons d'abord, notre architecture globale du système. Nous détaillerons la phase de réduction et mettrons en lumière notre approche qui permet d'appliquer la méthode de Quine-McCluskey sur les règles d'association. Nous détaillerons ensuite, la phase de vérification de la perte d'information. Enfin, nous présenterons l'implémentation de notre système.

Dans le **sixième chapitre**, nous verrons les résultats des expérimentations obtenus. Dans un premier temps, nous présentons les banques de données utilisées. Ensuite, nous étudions les résultats de la méthode proposée, sur le taux de réduction et la perte d'informations importantes.

Chapitre 2 : Les règles d’association

Chapitre 2
Les règles d'association

2.1. Introduction

Les règles d'association sont une représentation de l'information, dont l'idée principale est d'établir une relation entre deux ou plusieurs données[1]. Ces dernières sont extraites et validées par des mesures permettant de les filtrer, telles que le support, la confiance et le lift [2], etc.

2.2. Définition

Les règles d'association : Une règle d'association est une information généralement présentée sous la forme suivante : Antécédent → conséquent.

L'antécédent est également appelé le corps de la règle et le conséquent la tête de la règle[3]. La partie de l'antécédent représente les événements déclencheurs menant à un résultat appelé conséquent. Une fois que les règles d'association sont générées, elles peuvent avoir :
n (antécédent)→m (conséquent), où : n et m, sont supérieurs ou égaux à 1 [4].

Dans ce qui suit, nous allons détailler les notions utilisées dans l'extraction des règles d'association telles que: Transaction , Ensemble d'items(item-set), Support, Confiance, Lift [5].

2.2.1 Transaction et ensemble d'items

TABLEAU 2. 1: EXEMPLE DE TRANSACTIONS

TID	Transactions
1	Pression, Combustible, Chaleur, Feu, Vent
2	Combustible, Oxygène, Feu
3	Chaleur, Combustible, Vent, Feu
4	Chaleur, Combustible, Oxygène, Feu
5	Chaleur, Combustible, Oxygène, Vent, Feu
6	Combustible, Oxygène, Vent

TID : identifiant de transaction

Selon le contexte, une transaction peut être : une image, un son, un texte, etc. Une transaction est un ensemble fini d'items. Soit T = {t1, t 2 ... t $_i$} un ensemble de i transactions distinctes, une transaction T_i est un ensemble d'items qui occurrent ensemble. Dans le tableau 2.1 ci-dessus,

chaque ligne représente une transaction. Chaque élément dans la ligne représente un item. Comme exemple : pour la transaction 2, « Combustible » est un item. Donc, l'ensemble : {Combustible, Oxygène, Feu}, est un ensemble d'items(item-set) qui forme la transaction 2, et {Combustible, Oxygène, Vent} forme la transaction 6.

2.2.2 Ensembles d'items fréquents et support

Un ensemble d'items (item-set) regroupe des items (généralement deux au minimum) d'une même transaction, Exemple : {Chaleur, Feu} issu de la transaction 4 est un item-set. Un item-set est fréquent, si son support est supérieur ou égal à un seuil appelé minsup. Ce seuil représente la valeur minimale du support que doit avoir un item ou un item-set pour être considéré comme fréquent.

Le support d'un item-set AB est égal au nombre de fois que cet ensemble apparait dans la banque de données [5].

$$\sup(AB) = \frac{nombre\ d'occurence(AB)}{Nombre\ total\ de\ transactions}$$

Exemple:

Le support de l'item-set : {Chaleur, Combustible, Vent, Feu} est égal à:

$$\sup(\text{Chaleur, Combustible, Vent, Feu}) = \frac{nombre\ d'occurence(\text{Chaleur, Combustible, Vent, Feu})}{Nombre\ total\ de\ transactions}$$

$depuis\ le\ tableau\ 2.1, nous\ avons$: (Chaleur, Combustible, Vent, Feu), occurrent trois fois ensemble donc :

$nombre\ d'occurence(\text{Chaleur, Combustible, Vent, Feu}) = 3$

Nombre total de transactions = 6.

$donc$: $\sup(\text{Chaleur, Combustible, Vent, Feu}) = 3/6$.

2.2.3 support, confiance et Lift d'une règle d'association

Afin d'accepter une règle d'association A → B, des métriques importantes pour mesurer sa force sont utilisées: le support, la confiance et le lift [5].

2.2.3.1 le support d'une règle d'association

Le support d'une règle d'association A→B, est égal au nombre de fois que A et B apparaissent ensemble dans la banque de données.

$$\sup(A \rightarrow B) = \frac{nombre\ d'occurence(A \cup B)}{Nombre\ total\ de\ transactions}$$

Exemple:

Le support de la règle: $\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}$, est égal à:

$$\sup(\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}) = \frac{nombre\ d'occurence(\ (\text{Chaleur, Combustible, Vent}) \cup \text{Feu})}{Nombre\ total\ de\ transactions}$$

$depuis\ le\ tableau$ 2.1, $le\ nombre\ d'occurence\ de$ $(\text{Chaleur, Combustible, Vent}) \cup \text{Feu}$ $est\ trois\ fois$

$Nombre\ total\ de\ transactions$ = 6.

$donc$: $\sup(\text{ Chaleur, Combustible, Vent} \rightarrow Feu) = \frac{3}{6} = 0.5$.

3.2.3.2 la confiance d'une règle d'association

La confiance est une indication de la fréquence à laquelle la règle est vraie. Elle est égale au support de la règle divisé par le support de son antécédent [5].

$$\text{Conf}(A \rightarrow B) = sup(A \cup B)/\sup(A)$$

Exemple:

La confiance de la règle $\{\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}\}$ est égale à:

$$\text{Conf}(\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu})$$
$$= sup(\ (\text{Chaleur, Combustible, Vent}) \cup \text{Feu}\ )/\sup(\text{Chaleur, Combustible, Vent})$$

$nous\ avons\ sup((\text{Chaleur, Combustible, Vent}) \cup \text{Feu}) = \frac{3}{6} = 0.5$

$nous\ avons\ sup(\text{Chaleur, Combustible, Vent}) = \frac{3}{6} = 0.5$

$donc\ \text{Conf}(\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}) = \frac{0.5}{0.5} = 1.$

2.2.3.3 le Lift d'une règle d'association

Le lift est égal à la confiance de la règle divisée par le support de son conséquent [5].

$$\text{Lift}(A \rightarrow B) = Conf(A \rightarrow B)/\text{sup}(B)$$

Exemple:

Le lift de la règle $\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}$ est égal à:

$$\text{Lift}\ (\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}) = Conf(\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu})/\text{sup}(Feu)$$

$nous\ avons\ Conf(\text{Chaleur, Combustible, Vent} \rightarrow \text{Feu}) = 1$

$nous\ avons\ sup(Feu) = \frac{5}{6} = 0.83$

$donc\ \text{lift}\ (\text{Oxygène, Combustible, Chaleur} \rightarrow \text{Feu}) = \frac{1}{0.83} = 1.20.$

2.3 Extraction des règles d'association

L'extraction des règles d'association nécessite un ensemble d'items et de transactions. Plusieurs algorithmes d'extraction des règles d'association existent comme : Apriori et FP-growgth [6]. L'extraction des règles d'association est généralement réalisée à l'aide de l'algorithme Apriori [7]. Ces algorithmes passent par deux étapes à savoir, la Recherche des ensembles d'items fréquents puis la production des règles d'association. Les règles ayant un support ≥ minsup, une confiance ≥ minconf et un Lift ≥ minlift sont retenus.

Définition : minsup, minconf, minlift sont des seuils définis par l'utilisateur. Ils représentent les valeurs minimales que doit avoir une règle, un item ou un item-set pour être accepté.

2.3.1 L'algorithme Apriori

En 1994 Agrawal et Srikant proposent l'algorithme Apriori. Ce dernier est la base de tous les algorithmes de recherche des règles d'association [7]. Cet algorithme se déroule en plusieurs étapes. En premier lieu, un minsup, un minconf et un minlift sont déterminés initialement par l'utilisateur. En deuxième lieu, l'algorithme recherche les items fréquents ayant un support supérieur ou égal au minsup. Ensuite, les item-sets fréquents sont recherchés et filtrés en comparant leurs supports avec le minsup. Enfin, la génération des règles d'association est réalisée en faisant plusieurs combinaisons entre les items présents dans chaque item-set fréquent. Une règle est acceptée seulement si son support, sa confiance et son lift sont supérieurs ou égaux aux seuils établis par l'utilisateur.

2.3.1.1 Recherche des items fréquents

ALGORITHME 2. 1: RECHERCHE DES ITEMS FRÉQUENTS

Algorithme: recherche des items fréquents

```
Pour chaque item I dans chaque transaction
    Si support(I) non calculé
        Si Support(I)>=minsup
            Retenir I comme item_fréquent
```

L'algorithme apriori commence par trouver les items fréquents dans la banque de données. Cette tâche est réalisée en parcourant chaque item présent dans une transaction, et en calculant son support (si le support de l'item n'a pas déjà été calculé). Si le support de l'item est supérieur ou égal au seuil (déterminé par l'utilisateur), l'item en question est retenu comme fréquent.

En reprenant le tableau 2.1, le support de tous les items est calculé. Exemple : si le minsup donné par l'utilisateur est égal à 0.5, seuls les items dont le support dépasse où est égal au seuil de support seront acceptés et utilisés pour la suite de l'algorithme. Le tableau 2.3 ci-dessous présente les items fréquents.

TABLEAU 2. 2: SUPPORT DES ITEMS

	Items	Support
1	Oxygène	0.6
2	Feu	0.8
3	Pression	0.16
4	Combustible	1
5	Chaleur	0.6
6	Vent	0.6

TABLEAU 2. 3: ITEMS FRÉQUENTS

	Items	Support
1	Oxygène	0.6
2	Feu	0.8
3	Combustible	1
4	Chaleur	0.6
5	Vent	0.6

2.3.1.2 RECHERCHE DES ITEMS-SETS FRÉQUENTS

ALGORITHME 2. 2: RECHERCHE DES ITEMS-SETS FRÉQUENTS

Algorithme: recherche des items-sets fréquents

```
Pour chaque item I dans chaque transaction
      Si Support (I) < minsup
            Supprimer (I)

Pour chaque transaction Tr
      SE = extraire_des_sous_ensemble (Tr)
      Pour chaque sous-ensemble S dans SE
            Si Support (S)>=minsup et Support (S) non
      calculé
                  Retenir S comme item-set fréquent
```

La prochaine étape de l'algorithme est de trouver les items-sets fréquents. L'algorithme commence par supprimer tous les items non fréquents depuis les transactions. Ensuite, l'algorithme génère des sous-ensembles d'un minimum de deux éléments depuis chaque transaction. Exemple : pour une transaction T= {Vent, Feu, Chaleur}, les sous-ensembles suivants sont générés :

{Vent, Feu}, {Vent, Chaleur}, {Feu, Chaleur} et {Vent, Feu, Chaleur}. Chaque sous-ensemble est appelé item-set. Enfin, les supports de tous les items-sets sont calculés. Exemple : si le minsup donné par l'utilisateur est égal à 0.5, seuls les items-sets, dont les supports, dépassant ou égalent le seuil sont acceptés. Ces derniers seront utilisés pour la suite de l'algorithme. La figure 2.1 ci-dessous présente les items-sets fréquents du tableau 2.1.

FIGURE 2. 1: LISTE DES ITEMS-SETS FRÉQUENTS

1	Oxygene /\ Combustible	66.7
2	Oxygene /\ Feu /\ Combustible	50.0
3	Oxygene /\ Feu	50.0
4	Vent /\ Combustible	66.7
5	Vent /\ Chaleur /\ Combustible /\ Feu	50.0
6	Vent /\ Chaleur /\ Combustible	50.0
7	Vent /\ Chaleur /\ Feu	50.0
8	Vent /\ Chaleur	50.0
9	Vent /\ Feu /\ Combustible	50.0
10	Vent /\ Feu	50.0
11	Chaleur /\ Combustible /\ Feu	66.7
12	Chaleur /\ Combustible	66.7
13	Chaleur /\ Feu	66.7
14	Feu /\ Combustible	83.3

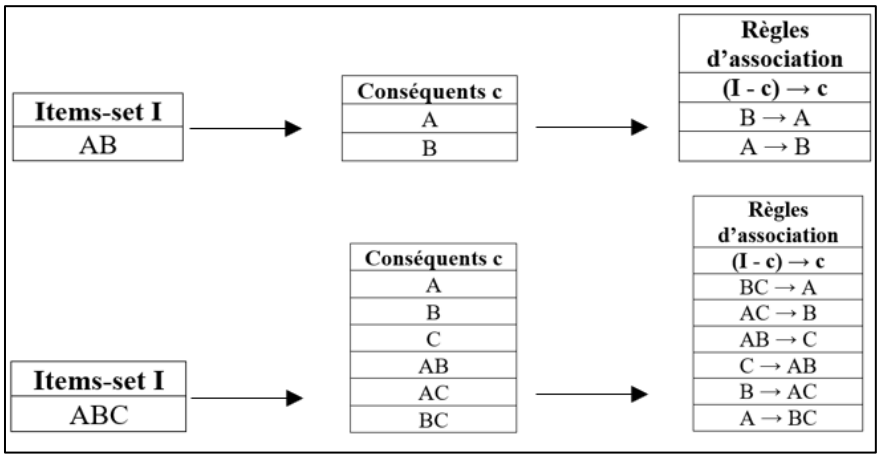
2.3.1.3 GÉNÉRATION DES RÈGLES D'ASSOCIATION

Les règles d'association sont générées en utilisant les items-sets fréquents. Rappelons que les règles d'association sont sous la forme n(antécédent)→m(conséquent), où n et m >=1.

Pour générer une règle d'association, l'algorithme permute les items présents dans un item-set **I** entre antécédents et conséquents. Pour ce faire, il commence par générer toutes les combinaisons possibles depuis **I** et les considère comme des conséquents. Chaque combinaison contient **n-1** items au maximum et un seul au minimum. **n** est le nombre d'items présents dans l'item-set **I**.

Exemple : pour l'item-set « ABC », **n** est égal à 3, alors, un conséquent peut contenir deux items au maximum et un seul au minimum. Les conséquents qui seront générés depuis l'item-set « ABC » sont les suivants : A, B, C, AB, AC et BC. Ensuite, l'algorithme génère les règles d'association avec la méthode suivante : $\mathbf{(I - c_i) \rightarrow c_i}$, où « $\mathbf{c_i}$ » est un $i_{ème}$ conséquent. La figure 2.2 ci-dessous montre un exemple d'application.

FIGURE 2. 2: GÉNÉRATION DES RÈGLES D'ASSOCIATION



Exemple :

Pour l'item-set « ABC » et le conséquent « AB » nous aurons $\mathbf{(I - c_i) \rightarrow c_i = (ABC - AB) \rightarrow AB = C \rightarrow AB}$

ALGORITHME 3. 3: GÉNÉRATION DES RÈGLES D'ASSOCIATION

```
Algorithme: génération des règles d'association

Pour chaque item-set_fréquent I Faire

      L = ensemble_C(I) //donne une liste qui contient tous les
conséquents possibles depuis l'item-set I.

      Pour chaque conséquent c dans L Faire

            Regle_association = (I - c) → c

            Si Support (Regle_association)>=minsup et
            Conf(Regle_association)>=minconf et

            Lift (Regle_association) )>=minlift Alors

                  Retenir (Regle_association)
```

Une fois les règles d'association en main, les supports des règles sont calculés comme premier filtre, suivi de la confiance et enfin du Lift. Seules les règles avec des scores supérieurs ou égaux aux seuils (minsup, minconf, minlift) seront retenues.

En appliquant l'algorithme apriori avec un minsup 0.5, minconf 0.6, et le minlift 0.9 sur le tableau 2.1, nous aurons les règles d'association présentées dans le Tableau 2.4 ci-dessous :

TABLEAU 2. 4:RÈGLES D'ASSOCIATION GÉNÉRÉES

N	Antécédent	Conséquent	Lift	Support (%)	Confiance (%)
1	Chaleur	Feu-Vent	1.5	50	75
2	Feu-Vent	Chaleur	1.5	50	100
3	Chaleur	Combustible-Feu	1.2	66.667	100
4	Feu	Combustible-Chaleur	1.2	66.667	80
5	Feu	Vent-Chaleur	1.2	50	60
6	Combustible-Feu	Chaleur	1.2	66.667	80
7	Combustible-Chaleur	Feu	1.2	66.667	100
8	Chaleur	Feu	1.2	66.667	100
9	Feu	Chaleur	1.2	66.667	80
10	Vent-Chaleur	Feu	1.2	50	100
11	Chaleur	Combustible-Vent	1.125	50	75
12	Feu-Chaleur	Vent	1.125	50	75
13	Vent	Chaleur	1.125	50	75
14	Chaleur	Vent	1.125	50	75
15	Vent	Feu-Chaleur	1.125	50	75
16	Combustible-Vent	Chaleur	1.125	50	75
17	Vent	Combustible-Chaleur	1.125	50	75
18	Combustible-Chaleur	Vent	1.125	50	75
19	Combustible	Chaleur	1	66.667	66.667
20	Chaleur	Combustible	1	66.667	100
21	Combustible	Oxygène	1	66.667	66.667
22	Oxygène	Combustible	1	66.667	100
23	Combustible	Feu	1	83.333	83.333
24	Feu	Combustible	1	83.333	100
25	Combustible	Vent	1	66.667	66.667
26	Vent	Combustible	1	66.667	100
27	Combustible	Feu-Chaleur	1	66.667	66.667
28	Feu-Oxygène	Combustible	1	50	100
29	Feu-Chaleur	Combustible	1	66.667	100
30	Feu-Vent	Combustible	1	50	100
31	Combustible	Feu-Vent	1	50	50
32	Combustible	Feu-Oxygène	1	50	50
33	Vent-Chaleur	Combustible	1	50	100
34	Combustible	Vent-Chaleur	1	50	50
35	Vent	Feu	0.9	50	75
36	Combustible-Vent	Feu	0.9	50	75
37	Oxygène	Feu	0.9	50	75
38	Vent	Combustible-Feu	0.9	50	75
39	Combustible-Oxygène	Feu	0.9	50	75
40	Oxygène	Combustible-Feu	0.9	50	75

2.3.2. Algorithme FP-Growth.

L'algorithme FP-Growth [8] est un algorithme d'extraction des règles d'association utilisant les motifs fréquents. L'algorithme compresse la banque de données en un arbre de motifs fréquents (FP-tree). Pour générer cet arbre, l'algorithme scanne la banque de données une première fois pour obtenir le nombre d'occurrences de chaque item. Il rejette les items qui n'apparaissent pas fréquemment, puis ordonne les items retenus par leur nombre d'occurrences dans la banque de données, dans une liste L. Exemple : depuis le tableau 2.1, si le minsup est 3, L = {Combustible (6), Feu (5), Chaleur (4), Oxygène (4), Vent (4)}. L'algorithme crée une première racine « null » et y attache l'item le mieux classé dans L, dans notre exemple : « Combustible ». Ensuite, l'algorithme scanne encore la banque de données et range les items de chaque transaction par leur classement dans L, exemple : la transaction 2 : {Combustible, Oxygène, Feu}, devient {Combustible, Feu, Oxygène}. Puis, commence à élargir l'arbre en liant les items de chaque transaction sous forme de nœuds. Exemple : pour la transaction {Combustible, Feu, Oxygène}, l'algorithme lie Feu à Combustible et Oxygène à Feu. Si un nœud est un préfixe pour d'autres transactions, son compte « w » est augmenté de 1.

En général, l'algorithme considère l'augmentation d'une branche pour une transaction et lorsque chaque nœud suit un préfixe commun, son compte augmente de 1, puis l'algorithme crée un nœud pour l'élément qui suit le préfixe et le relie [6].

ALGORITHME 2. 4: FP-GROWTH

```
R=null
Pour chaque item I dans transaction 1
      R = R ← I_w=1
Pour chaque transaction T de 2..a..n
      Si item T_nI_1 dans R et longueur (null ← R_I1) = 1
            R_i = nœud T_nI_1 dans R
            Pour chaque item I dans transaction T_n
                  N=R_i
                  Si T_nI_i ≠ N
                        N = N ← I_i,w=1
                  Sinon
                        R_i = T_nI_i,w=w+1
                  Attacher N a R_i
      Sinon
            N = null ← I_1
            Pour chaque item I dans transaction T_n
                  N = N ← I_w=1
```

R: arbre FP-Tree, Ri: Nœud i dans l'arbre FP-tree, W: Compteur d'itérations, T_nI_i: Item i dans la transaction n.

Pour le tableau 2.1, l'algorithme construit l'arbre en parcourant les transactions. Pour chaque transaction, l'algorithme prend en considération que les items fréquents. Exemple d'exécution :

T1 : combustible, feu, Chaleur, vent

R=null

R= null ←$combustible_{w1}$

R = null ←$combustible_{w1}$←Feu_{w1}

R=null ←$combustible_{w1}$←Feu_{w1}←$Chaleur_{w1}$←$Vent_{w1}$

T2 : combustible, feu, oxygène

R=null ←$combustible_{w1}$←Feu_{w1}←$Chaleur_{w1}$←$Vent_{w1}$

R_1= $combustible_{w=2}$

N=R_1

R_2=$feu_{w=2}$

N=R_2

N=feu_{w2}←$oxygene_{w1}$

R_2 «-- feu_{w2}←$oxygene_{w1}$

T3 : Combustible, Feu, Chaleur, Vent

R=null←combustible$_{w2}$←(feu$_{w2}$←oxygene$_{w1}$) ←Chaleur$_{w1}$←Vent$_{w1}$

R_1 = combustible$_{w2}$

N= R_1

R_1= combustible$_{w=3}$

N=R_1

R_2=feu$_{w3}$

N=R_2

R_3=Chaleur$_{w=2}$

N=R_3

R_4=vent$_{w=2}$

T4 : Combustible, Feu, Chaleur, Oxygène

R=null←combustible$_{w3}$←(feu$_{w3}$←oxygene$_{w1}$) ←Chaleur$_{w2}$←Vent$_{w2}$

R_1= combustible$_{w=4}$

N=R_1

R_2=feu$_{w=4}$

N=R_2

R_3=Chaleur$_{w=3}$

N=R_3

N= Chaleur$_{w3}$←oxygene$_{w1}$

R_3 «-- Chaleur$_{w3}$←oxygene$_{w1}$

R_4 = vent$_{w2}$

T5 : Combustible, feu, Chaleur, oxygène, vent

R=null←combustible$_{w4}$←(feu$_{w4}$←oxygene$_{w1}$) ←(Chaleur$_{w3}$←oxygene$_{w1}$) ←Vent$_{w2}$

R_1= combustible$_{w=5}$

N=R_1

R_2=feu$_{w=5}$

N=R_2

R_3=Chaleur$_{w4}$

N= Chaleur$_{w4}$←oxygene$_{w1}$

N= Chaleur$_{w4}$←oxygene$_{w2}$←vent$_{w1}$

R_3 «-- Chaleur$_{w4}$←oxygene$_{w2}$←vent$_{w1}$

R_4=vent$_{w2}$

T6 combustible, oxygène, vent

R=null←combustible$_{w5}$←(feu$_{w5}$←oxygene$_{w1}$) ←(Chaleur$_{w4}$←oxygene$_{w2}$←Vent$_{w1}$) ←Vent$_{w2}$

R_1 = combustible$_{w5}$

N= R_1

R_1= combustible$_{w=6}$

N=R_1

N= combustible$_{w=5}$←oxygene$_{w1}$

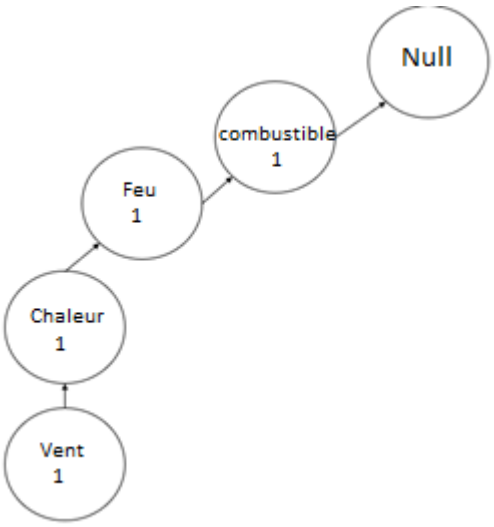
N= combustible$_{w=5}$←oxygene$_{w1}$←vent$_{w1}$

R_1 «-- combustible$_{w=5}$←oxygene$_{w1}$←vent$_{w1}$

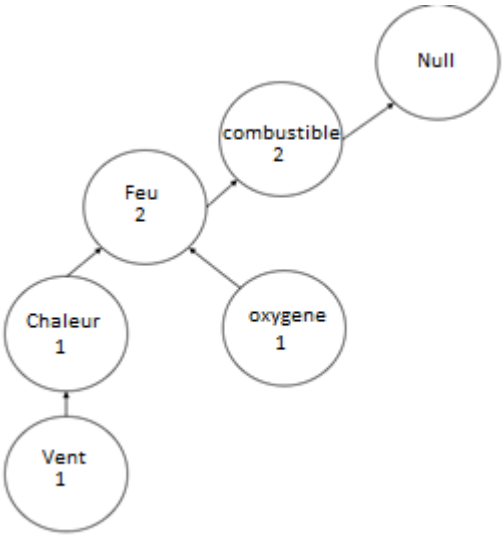
R=null←(combustible$_{w=6}$←oxygene$_{w1}$←vent$_{w1}$)← (feu$_{w5}$←oxygene$_{w1}$) ←(Chaleur$_{w4}$←oxygene$_{w2}$←Vent$_{w1}$) ←Vent$_{w2}$

On peut voir ci-dessous un exemple de déroulement sous forme de graphe comme suit :

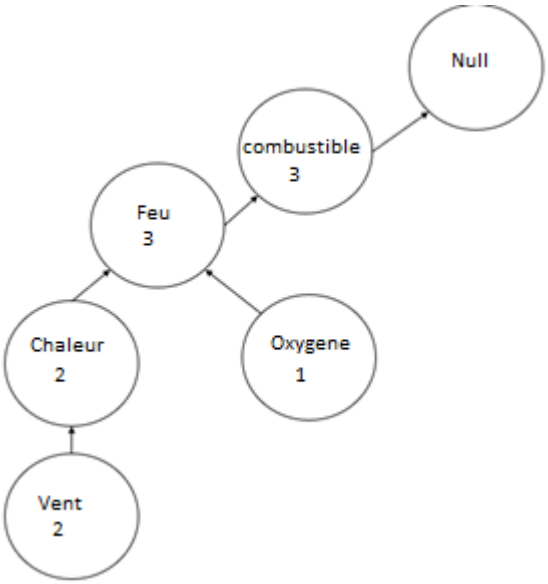
Combustible, Feu, Chaleur, Vent



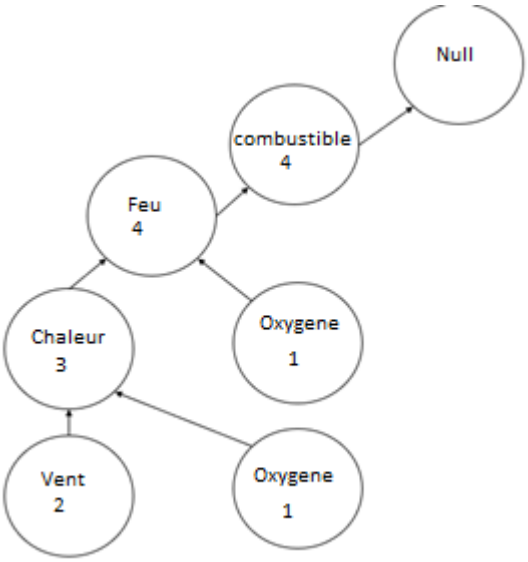
Combustible, Feu, Oxygène



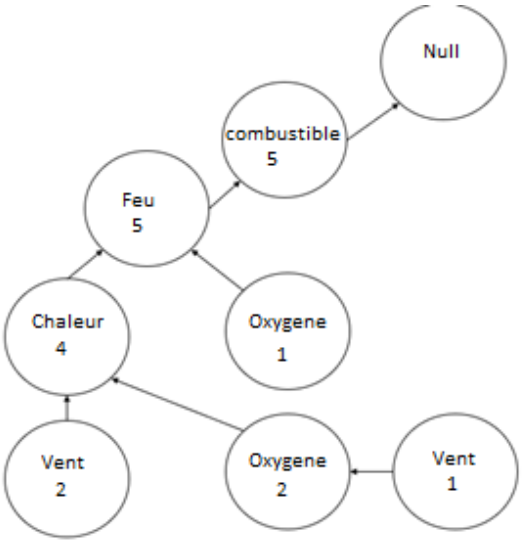
Combustible, Feu, Chaleur, Vent



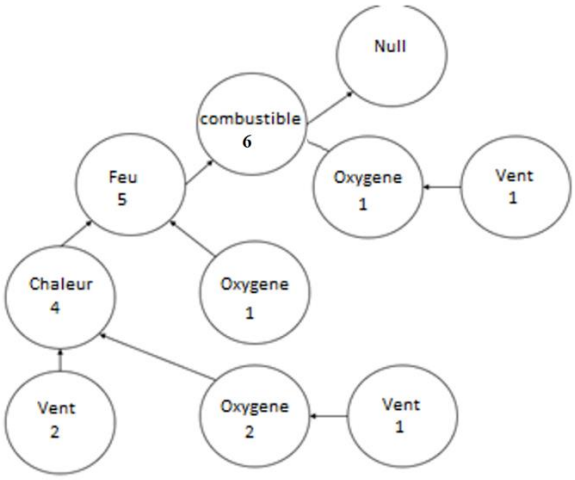
Combustible, Feu, Chaleur, Oxygène



Combustible, Feu, Chaleur, Oxygène, Vent



Combustible, Oxygène, Vent



L'algorithme génère ensuite les items-sets fréquents en parcourant l'arbre (FP-tree) utilisant la méthode du Depth first (profondeur d'abord) [9]. Exemple : en commençant par Vent, l'algorithme compte deux fois (Vent Chaleur), puisque le compte de Vent est égal à 2 avec une liaison directe avec Chaleur : (Vent (2) → Chaleur (4)), et 1 en liaison indirecte (Vent (1) → Oxygène (2) → Chaleur (4)), donc Vent Chaleur apparaisse ensemble 3 fois et seront fréquents si le minsup=3. Le total des item-sets fréquents sera comme suit :

[Vent Chaleur Feu, Vent Chaleur Feu Combustible, Vent Chaleur Combustible, Vent Feu Chaleur, Vent Chaleur, Vent Feu, Vent, Combustible, Oxygène Feu, Oxygène Feu Combustible, Oxygène Combustible, Chaleur Feu, Chaleur Feu Combustible, Chaleur Combustible, Feu Combustible].

À partir de ces items set fréquents, les règles d'association seront générées, suivant le même principe de permutation entre antécédents et conséquents comme l'algorithme Apriori.

2.4. Conclusion

Dans ce chapitre, nous avons pu mettre en lumière les règles d'association et les différentes mesures nécessaires à leur extraction. Nous avons vu deux algorithmes populaires d'extraction des règles d'association à savoir Apriori et FP-Growgth, que nous avons illustré par moyen d'exemples. Nous avons pu constater qu'à partir d'une petite banque de données nous pouvons extraire un grand nombre de règles d'association. Une base plus conséquente pourrait générer des milliers de règles.

Dans le chapitre suivant, nous présentons des travaux trouvés en littérature sur la réduction du nombre de règles d'association et l'état de la situation.

Chapitre 3: réduction du nombre de règles d'association : état de la situation

Chapitre 3:

Réduction du nombre de règles d'association : état de la situation

3.1 Introduction

Au cours de ces dernières années, plusieurs méthodes ont été proposées pour réduire le nombre de règles d'association. Dans la littérature, on trouve plusieurs travaux présentant différentes approches. Dans ce qui suit, nous allons présenter quelques travaux qui ont utilisé les règles d'association sous différents aspects.

3.2 Littérature

3.2.1 Reduction du nombre de règles d'association

Ashrafi et al [10] présentent plusieurs méthodes d'élimination des règles d'association redondantes. Ces dernières sont des règles qui portent des significations similaires par le fait de partager soit les mêmes conséquents ou les mêmes antécédents. Les méthodes proposées tiennent compte de la confiance et du support de chaque règle avant d'écarter toute règle considérée redondante. Les méthodes vérifient le sous-ensemble de chaque antécédent et de conséquents pour identifier les règles redondantes. En outre, elles ne forcent jamais à abandonner une règle de support ou de confiance élevée. Les méthodes présentées sont les suivantes : 1-« la méthode de dominance totale », cette dernière recherche une règle propre, et ses règles de sous-conséquents, ou sous-antécédents correspondants. Si les règles de sous-conséquents ou sous-antécédents ont un support et une confiance plus élevés que la règle propre, la règle propre est donc considérée comme redondante.

Exemple : si, on considère l'ensemble de règles d'association : R = [A→BC, A→B, A→C]. A→BC est une règle propre et A→B, A→C, sont des règles de sous-conséquences. Pour R*= [WX→Y, W→Y, W→Y], WX→Y est une règle propre et W→Y, X→Y, sont des règles de sous-antécédents. La méthode jugera les règles propres comme redondantes, si leurs supports et leurs confiances sont inférieurs aux supports et confiances des règles de sous-antécédents ou sous-conséquents.

2-La deuxième méthode est « la méthode de dominance partielle » : une règle propre est éliminée, si plusieurs règles (pas toutes) de sous-conséquents ou sous-antécédents dominent la règle propre en termes de confiance.

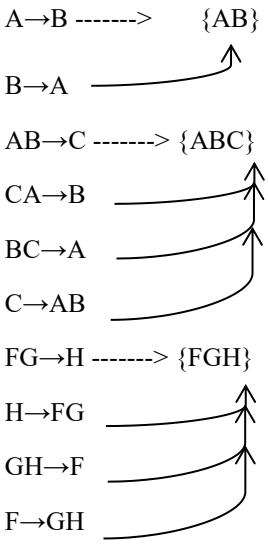
Finalement, 3-« La méthode de la dominance d'indifférence » : élimine les sous conséquents ou sous antécédents, dont les supports et les confiances sont inférieurs à ceux des règles propres.

Les méthodes proposées par les auteurs sont intéressantes, car elles exploitent les règles redondantes pour réduire le nombre de règles d'association. Toutefois, l'utilisation de métriques telles que le support et la confiance peut être arbitraire et subjective d'un chercheur à un autre. De plus, aucun test permettant de savoir si les approches présentées conservent les informations importantes n'a été réalisé.

Jeypal et Al [11] ont proposé, eux aussi, une approche pour éliminer les règles d'association redondantes. Dans cet article, les auteurs définissent les règles redondantes comme suit : Les règles qui se forment en combinant seulement les items entre antécédents et les conséquents sont des règles redondantes, exemple : A→B et B→A ou AC→B et BC→A, selon les auteurs, ces règles sont redondantes, car elles sont formées simplement en permutant les items entre antécédent et conséquent.

Leur approche consiste à former des nœuds en rassemblant l'ensemble des éléments d'une règle (antécédents et conséquents). Premièrement, une règle forme un nœud initial, puis une nouvelle règle forme un nouveau nœud, si cette dernière pointe vers le nœud initial, alors la nouvelle règle est éliminée de l'ensemble des règles.

Exemple : Pour l'ensemble des règles d'association R = [A→B, AB→C, BC→A, FG→H, CA→B, H→FG, GH→F, F→GH, C→AB, B→A]. L'algorithme prend la première règle qu'il trouve : A→B, puis un premier nœud est créé avec les items de cette dernière. Ensuite, il cherche les règles qui ont les mêmes items, et les pointent vers la première règle. Ce processus s'arrête quand il n'est plus possible de former de nouveaux nœuds :



L'algorithme donne comme résultat la réduction suivante : [AB→C, FG→H, A→B]

L'utilisation d'une représentation graphique peut être très bénéfique pour étudier les règles d'association. De plus, exploiter la redondance des règles est une très bonne stratégie pour réduire le nombre de règles. Toutefois, les auteurs se fient aux métriques de support et de confiance pour valider les résultats obtenus, nous jugeons cette approche de validation peu convaincante, car l'utilisation de métriques est là encore, arbitraire d'une personne a une autre.

Dans [12] Vera et Al, présentent une nouvelle définition de la redondance des règles d'association. Une méthode de post-traitement est développée pour éliminer la redondance en calculant la Clôture des items pour déterminer si une règle est redondante ou pas. Les règles passent par un processus où les axiomes d'Armstrong sont appliqués pour en éliminer encore plus. La Clôture d'un item E, est annotée E^+, elle représente l'ensemble des items qui peuvent être déduits en partant de E. exemple : pour l'ensemble R= [A→B, B→C, D→B], la Clôture de « A » = A^+= ABC puisque « A » conduit à « B » et « B » conduit à « C ».

Le processus proposé par les auteurs se déroule en travaillant avec deux ensembles F et G où : $F = Sc \cup R_i$; $G= (F - R_i)$ (E_i→ conséquent) ou ($F - R_i$) (antécédent →E_i). « Sc » est l'ensemble des règles connues non redondantes, R_i la règle étudiée et E_i un item dans l'antécédent de la règle R_i. Si la Clôture d'un item est la même dans F et G, la règle est considérée comme redondante.

Exemple : Pour l'ensemble des règles d'association non redondantes Sc= [A→B, CD→K], et une règle à étudier : ED→AB, nous avons : F= Sc ∪ R_1= [A→B, CD→K, ED→AB]. La méthode commence par l'item « E » et calcule sa Clôture dans F, puis dans G.

Notons que G= (F – R_1) ∪ (E → AB) = [A→B, CD→K, E→AB]. La Clôture de « E » dans F est égale a : E^+_F = E, et dans G : E^+_G = EAB. Puisque $E^+_F \neq E^+_G$, l'item « E » n'est pas redondant. Nous continuions avec l'item « D », nous avons : G= (F – R_1) ∪ (D → AB) = [A→B, CD→K, D→AB].

D^+_F = D, et D^+_G = DAB, puisque $D^+_F \neq D^+_G$, l'item « D », n'est pas redondant. Maintenant avec la partie du conséquent nous commençons par « A ». Nous avons : A^+_F = EDAB.

G= (F – R_1) ∪ (ED → A) = [A→B, CD→K, ED→A]. Alors, A^+_G = EDAB. Puisque A^+_F = A^+_G, la règle ED→AB est redondante.

À la fin du processus les axiomes d'Armstrong sont appliqués pour réduire encore plus le nombre des règles d'association.

L'approche proposée semble intéressante, mais les auteurs ont utilisé les axiomes d'Armstrong, notamment la transitivité, ce qui pose un problème majeur : la transitivité n'est pas acceptée avec les règles d'association. D'autre part, les auteurs utilisent là encore, les mesures de support et de confiance pour valider les résultats, ce qui est arbitraire.

Sarma et Al [13] ont introduit une nouvelle mesure, appelée Inter Item-set distance ou Spread. Cette mesure est basée sur les approches de l'algorithme apriori en vue de réduire le nombre de règles d'association. Une analyse de l'algorithme est effectuée et les résultats sont présentés et comparés aux résultats de l'algorithme apriori conventionnel. La notion du spread peut être interprétée comme l'évaluation de la distribution des item-sets dans la banque de données en comptant le nombre d'absences d'un item-set depuis sa première occurrence jusqu'à sa dernière. Cette mesure donne une indication de la rareté des item-sets. Pour bien interpréter cette notion, il est utile de prendre en exemple le support. En effet, si le support compte le nombre d'occurrences, le spread compte le nombre d'absences. Plus la valeur du spread est grande plus un item-set est rare. Le Spread est égal à la somme des écarts(absence) divisée par le support -1.

Exemple:

Dans ce qui suit, Spread (AB) = 2 / 4. Puisqu'AB apparait dans les transactions 1,3,4,5,7, l'écart sera donc le nombre d'absences entre la transaction 1 et 7. l'item-set AB est absent dans les transactions 2 et 6, alors l'écart est égal à 2. Le support d'AB est égal à 5.

TID	Items
1	A B D E
2	B C E
3	A B C D E
4	A B E
5	A B C E
6	C D
7	A B
8	C D
9	B C D
10	A D

Item-set	support	spread
AB	5 (50%)	2/4 (=0.5)
AD	3 (30%)	6/2 (=3.0)
AE	4 (40%)	1/3 (=0.33)
BC	4 (40%)	4/3 (=1.33)
BD	3 (30%)	6/2 (=3.0)
BE	5 (50%)	0/4 (=0.0)
CD	4 (40%)	3/3 (=1.0)
CE	3 (30%)	1/2 (=0.5)
ABE	4 (40%)	1/3 (=0.33)

Item-set Spread	Règles d'association
AB (2/4 =0.5) BE (0/4 =0.0)	A → B (5/6 = 80.33%), B → A (5/7 = 70.14%) B → E (5/7 = 70.14%), E → B (5/5 = 100%)
AE (1/3=0.33) BC(4/3 =1.33) CD (3/3=1.0) ABE(1/3=0.33)	A → E (4/6 = 66.67%), E → A (4/5 = 80.0%) C → B (4/6 = 66.67%) C → D (4/6 = 66.67%), D → C (4/6 = 66.67%) AB → E (4/5 = 80.0%), E → AB (4/5 = 80.0%) AE → B (4/4 = 100%), BE → A (4/5 =80%) A → BE (4/6 = 66.67%)
CE (1/2=0.5)	E → C (3/5 = 60%)

L'approche proposée tente de mettre en place une nouvelle mesure pour filtrer les règles d'association. L'approche met en valeur la rareté des items dans une banque de données. La mesure du Spread peut très bien filtrer les règles en plus des mesures conventionnelles comme le support, mais reste, elle aussi, arbitraire.

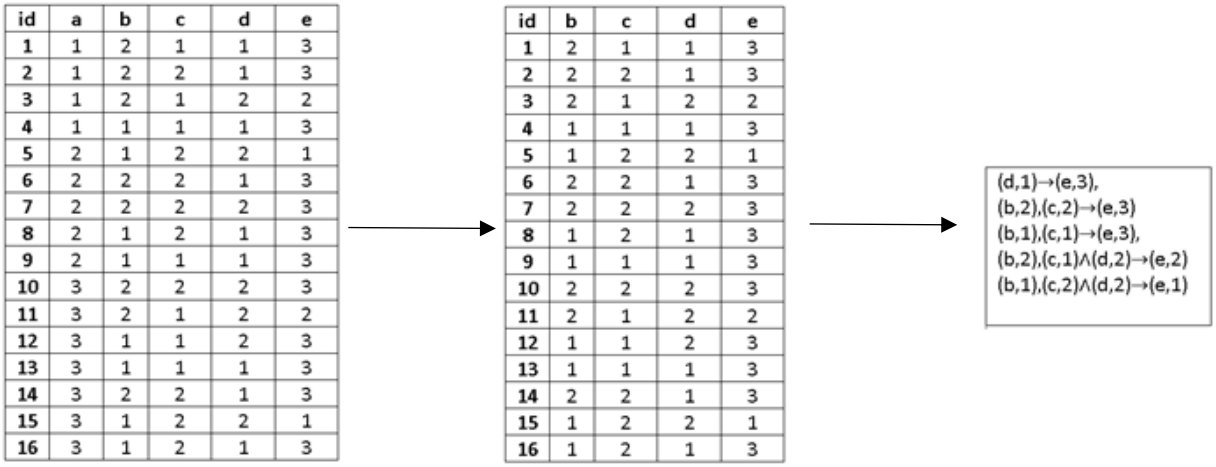
Chen et Qixin [14] ont utilisé une approche mathématique pour la réduction des règles d'association, en calculant la dimension fractale (FDR). Cette dernière est employée comme seuil permettant de filtrer les item-sets. Le processus se déroule en sélectionnant un item-set, et en calculant sa dimension fractale D. Si D se situe dans un petit seuil, les auteurs considèrent cet item-set non important, et ne contribue presque pas aux caractéristiques globales de l'ensemble des données.

Les auteurs ont présenté des résultats de réduction du nombre de règles d'association, mais aucun test permettant de vérifier si la réduction a mené à une perte des informations importantes n'a été réalisé. De plus, les auteurs ne donnent aucun exemple d'application.

Ma et Yu.liang [15] proposent un nouvel algorithme permettant d'extraire un nombre réduit de règles d'association, en combinant l'extraction des règles d'association avec la théorie des roughs sets (RST). Le processus commence par la réduction d'attributs en supprimant les attributs non-noyaux. Un attribut non-noyaux est un attribut pouvant être supprimé de la table sans créer de l'ambiguïté (deux lignes ou plus avec les mêmes antécédents, mais avec des décisions différentes). Puis, génère les règles d'association en utilisant les attributs noyaux. Pour qu'une règle soit retenue, elle doit satisfaire deux critères. Le premier est le score de support qui doit être supérieur ou égal au minsup. Le deuxième est une comparaison entre le support de l'antécédent et le support de la règle, si le support de l'antécédent est inférieur au support de la règle, la règle est supprimée.

Exemple:

Admettons une table décisionnelle, où a, b, c, d sont des antécédents et « e » est la décision (Conséquent). Nous commençons par supprimer les attributs non-noyaux, dans notre exemple seul l'attribut « a » est non-noyau, le reste des attributs est utilisé pour générer des règles d'association un peu comme nous l'avons vu avec Apriori, par exemple, depuis la première ligne, nous pouvons extraire les règles : $b_2 \rightarrow e_3$, $c_1 \rightarrow e_3$, $d_1 \rightarrow e_3$, $b_2c_1 \rightarrow e_3$, $b_2d_1 \rightarrow e_3$, $c_1d_1 \rightarrow e_3$, $b_2c_1d_1 \rightarrow e_3$. Une règle d'association est retenue, si le support des antécédents est égal au support de la règle. Dans notre exemple, le support de $(d_1) = 9$ et le support de la règle $(d_1) \rightarrow (e_3) = 9$. Donc, cette règle est retenue.



Des résultats de réduction des règles d'association ont été obtenus par cette approche. Cette dernière se base sur la définition : le support d'un antécédent ne peut être inférieur au support de la règle. Les règles sont encore une fois filtrées arbitrairement.

Xiangjun Dong, Feng Hao, Long Zhao, Tiantian Xu [16] présentent une nouvelle méthode pour exploiter les règles d'association dites positives et négatives PNAR. Si une règle de la forme A→B, est une règle d'association positive, les règles des trois autres formes A→¬B, ¬A→B, ¬A→¬B, sont des règles d'association négatives (NAR), où la négation exprime l'absence d'un item. Les auteurs cherchent à éliminer le maximum de redondance dans ces règles d'association.

La méthode proposée utilise le raisonnement logique pour élaguer les PNAR redondantes. En outre, la méthode combine le coefficient de corrélation et les confiances minimales pour garantir que les PNAR extraites sont fortement corrélées et que leur nombre peut être limité de manière flexible. Le support d'une règle sup (A → ¬B) = sup(A) − sup (A → B), où ¬B exprime l'absence de B donnant ainsi une indication de la fréquence où A ne conduit pas à B. plus d'exemples de mesures sont disponibles dans le papier des auteurs.

Explorer l'absence des items peut rapporter beaucoup d'informations. De plus exprimer l'absence par la négation est une représentation qui ouvre la porte à plusieurs utilisations. Nous nous inspirons de cette représentation pour présenter nos règles sous la forme booléenne. Les résultats expérimentaux montrent que la méthode permet d'éliminer les PNAR que les auteurs ont

jugées redondantes. Toutefois, la méthode se base sur les mesures de supports et de confiances, ce qui peut être quasi-arbitraire.

Bastide et Al [17] utilisent la Clôture pour réduire le nombre de règles d'association. L'approche utilise des ensembles générateurs. Ces derniers sont des items-sets fréquents contenants au minimum un item. La Clôture d'un générateur est calculée pour obtenir un nouvel ensemble. À partir de ce dernier, et du générateur, une règle est générée en mettant le générateur comme antécédent et le reste de l'ensemble comme conséquent. À la fin, les règles dont les supports et confiances sont inférieurs aux minsup, minconf sont supprimées.

Exemple : pour un ensemble de transactions : T= [ACD, BCE, ABCE, BE, ABCE, BCE], avec un minsup égal à 3. L'item « A » a un support égal à 3 donc il est fréquent, nous l'utiliserons comme générateur. Parmi les Clôtures de « A » nous avons A+ = AC, cette dernière a le plus grand support par rapport aux autres Clôtures de « A », alors elle sera utilisée pour générer des règles. La suite consiste à supprimer tout ce qui a un score de support inférieur au minsup. À la fin, les règles sont générées.

ACD
BCE
ABCE
BE
ABCE
BCE

Générateur	Closure
A	AC
B	BE
C	C
D	ACD
E	BE
AC	AC
BC	BCE
BE	BE
CE	BCE

Générateur	Closure
A	AC
B	BE
C	C
E	BE
AC	AC
BC	BCE
BE	BE
CE	BCE

Générateur	Closure	Règle d'association
A	AC	A→C
B	BE	B→E
C	C	
E	BE	E→B
AC	AC	
BC	BCE	BC→E
BE	BE	
CE	BCE	CE→B

La méthode proposée est intéressante et a pu produire un nombre réduit de règles d'association par rapport à d'autres approches d'extraction présentées par les auteurs. Toutefois, les auteurs utilisent les valeurs de support et de confiance ce qui est encore une fois subjectif. De plus, l'approche n'a pas été validée en termes de perte des informations importantes.

Al-zawaidah et Al [18] présentent une nouvelle approche d'exploration des règles d'association dans le but de les filtrer. L'approche proposée est dérivée de l'approche Apriori avec des caractéristiques ajoutées pour améliorer les performances de l'exploration de données. L'algorithme utilise la mesure de Levrage pour filtrer les règles trouvées.

$$Levrage(a \rightarrow b) = (\sup(a\,b)) - (\sup(a)\,.\,\sup(b)).$$

Cette dernière, mesure la différence entre ce qui serait attendu si l'antécédent et le conséquent apparaissait ensemble et ce qui serait s'ils étaient statistiquement dépendants.

Des expériences approfondies ont été réalisées et comparées aux performances d'algorithmes existants trouvés dans la littérature. Les résultats expérimentaux ont montré que l'approche peut extraire des règles d'association avec un nombre réduit. Toutefois, les auteurs ont utilisé des mesures comme le support et la confiance comme paramètre et la nouvelle mesure est elle-même un nouveau seuil tout comme le support ou la confiance. De plus, les auteurs n'ont pas réalisé de tests permettant de tester la pertinence des règles obtenues.

Han et Al [19] ont étendu le champ d'études de l'extraction de règles d'association d'un seul niveau à des règles de multiples niveaux. Une règle d'association de multiples niveaux est une règle plus précise(spécifique) qu'une règle d'association régulière.

Exemple : une règle « lait → pain » exprime que si un client achète du lait, il achèterait probablement du pain, tandis qu'une règle de multiples niveaux «lait de 2% → pain blanc» précisera qu'un client qui achète du lait de 2% achèterait probablement du pain blanc.

Nous avons donc dans le niveau 1 : « lait → pain » qui est plus général et dans le niveau 2 : « lait de 2% → pain blanc » qui est plus précis. Le premier niveau est une racine pour le deuxième.

Les auteurs tentent aussi de mieux réduire le nombre de règles en éliminant les règles redondantes, en comparant la confiance d'une règle de multiples niveaux avec celle de sa racine, si la confiance tombe dans un intervalle déterminé par l'utilisateur, alors la règle n'est pas redondante.

L'étude des performances menée par les auteurs montre que différents algorithmes peuvent avoir de bons résultats pour différentes distributions de données en termes de temps d'extraction des règles d'association. Le nombre de règles a été réduit par les scores de support et de confiance. Mais, cela reste bien sûr arbitraire.

Abdullah et Al [20] proposent une nouvelle mesure appelée support relatif critique (CRS) pour réduire le nombre de règles d'association. La valeur CRS est comprise entre 0 et 1, et est déterminée, en multipliant la valeur la plus élevée entre le support de l'antécédent divisé par le conséquent ou dans l'autre sens, et leur coefficient de similarité de Jaccard. L'utilisateur doit arbitrairement fixer un seuil de CRS pour supprimer les règles. Malheureusement, les auteurs ne présentent aucun test permettant de vérifier si la réduction engendre une perte des informations importantes.

$$CRS(X \to Y) = \max\left(\frac{sup(X)}{sup(Y)}, \frac{sup(Y)}{sup(X)}\right) * \left(\frac{\sup(X \to Y)}{\sup(X) + Sup(Y) - \sup(X \to Y)}\right)$$

Exemple:

Avec les règles suivantes B→A, D→AB, avec les supports sup(B)=0.5; sup(A)=1; sup(D)=0.66; sup (AB)=0.5; sup(B→A) =0.5; sup(D→AB) =0.33 :

$$CRS(B \to A) = \max\left(\frac{sup(B)}{sup(A)}, \frac{sup(A)}{sup(B)}\right) * \left(\frac{\sup(B \to A)}{\sup(B) + Sup(A) - \sup(B \to A)}\right) = 1.$$

$$CRS(D \to AB) = \max\left(\frac{sup(D)}{sup(AB)}, \frac{sup(AB)}{sup(D)}\right) * \left(\frac{\sup(D \to AB)}{\sup(D) + Sup(AB) - \sup(D \to AB)}\right) = 0.39.$$

3.2.2 D'autres travaux sur les règles d'association

Dans cette section, nous présentons d'autres travaux réalisés sur les règles d'association. Nous nous abstenons de détailler ces derniers, car, ils n'apportent pas d'informations connexes avec notre travail. Nous invitons le lecteur à examiner ces derniers pour plus de détails.

Ghareb et Al [21] proposent une méthode de catégorisation hybride de textes arabes utilisant les règles d'association. Les auteurs utilisent deux classifieur : le classifieur associative (AC), et le classifieur bayésien (CB). Pour limiter le nombre de règles d'association, les valeurs de score de support et confiance sont utilisées. Dans la première étape, les règles d'association de classification sont découvertes et filtrées avec les mesures de support et de confiance. La méthode a été évaluée sur trois ensembles de données de textes arabes avec plusieurs catégories. Les résultats expérimentaux ont montré que la méthode hybride (AC/CB) est plus efficace que le classifieur associative AC et le classifieur bayésien CB quand ils sont utilisés seuls. Nous notons là encore que le nombre de règles d'association est limité par des seuils, et que le taux de classification change d'un seuil à un autre.

Sharma et Al [22] ont téléchargé 14 696 rapports de bogues de trois produits : Seamonkey, Firefox et Bugzilla du projet open source Mozilla. Ils ont attribué une valeur numérique de 1 à 7 à l'attribut de gravité et de 1 à 5 à l'attribut de priorité. Ils ont extrait les règles en utilisant l'algorithme Apiori avec les outils ARMADA (Association Rule Miner And Deduction Analysis) dans MATLAB. Le nombre minimal de supports est de 3 et la confiance minimale de 10% pour extraire les règles d'association. Les auteurs tentent de limiter le nombre de règles en manipulant les mesures de support et de confiance et valident les résultats en se basant sur des mesures de support et la confiance.

Alwedyan et Al [23] étudient l'un des algorithmes de classification associative (CA) bien connus, à savoir MCAR, la méthode bayésienne Naïve (NB) et l'algorithme de la machine à vecteur de support (SVM) sur différents ensembles de données arabes. Les valeurs de support et confiance et lift sont utilisées pour limiter le nombre des règles d'association. Les auteurs ont démontré avec plusieurs résultats expérimentaux sur différents ensembles de données de catégorisation de textes arabes que l'algorithme MCAR surpasse les algorithmes NB et SVM pour toutes les mesures. Ce qui démontre l'efficacité des règles d'association dans la classification.

Alhalees et Ala M [24] proposent aussi une approche visant à exploiter les règles d'association pour la classification des textes arabe. Leur méthode commence comme suit : d'abord les données sont prétraitées à l'aide de techniques de traitement du langage naturel telles que la tokenisation et l'abréviation. Ensuite, la méthode de l'entropie maximale est utilisée pour classer les documents arabes. L'expérimentation de l'approche utilise des données réelles, puis est comparée avec d'autres systèmes existants. Toutefois, plusieurs valeurs de support et confiance sont utilisées pour limiter le nombre des règles d'association faisant ainsi changer le taux de précision selon les seuils donnés.

Haralambous et Al [25] ont aussi étudié différentes techniques de classification de textes arabes en variant le nombre de règles d'association en fonction du support et la confiance. Les auteurs ont préparé deux versions de corpus : l'une avec des mots racinés et l'autre avec des mots déracinés. Les phrases sont considérées comme des transactions sélectionnées de deux manières : par la technique Tf-idf et par des techniques de grammaire de dépendance. Enfin, deux types de classifieurs sont appliqués : les règles d'association de classification et SVM. Les résultats ont montré que les règles de classification sont meilleures que les SVM pour les transactions de petite taille et deviennent encore meilleurs lorsque la taille des transactions devient plus grande. Toutefois, nous notons que le nombre de règles retenues a aussi un impact sur la classification, ce qui prouve que l'utilisation de seuils arbitraire peut faire perdre des informations importantes.

Bogorny et Al [26] présentent un framework pour l'extraction de règles d'association à partir de banques de données géographiques en utilisant les connaissances de base à la fois dans le prétraitement des données et la génération d'ensembles fréquents. Les expériences ont montré qu'indépendamment du nombre d'éléments, une dépendance suffit à élaguer de nombreuses règles. La principale contribution de l'approche concerne l'utilisateur de l'exploration de données, qui analysera des règles visuellement.

3.2 Conclusion

Toutes les recherches présentées utilisent les règles d'association pour classifier ou tentent de réduire le nombre de règles avec différentes approches en utilisant des seuils. Malheureusement, nous n'avons trouvé aucun travail étudiant l'information que la réduction peut faire perdre aux banques de données. De plus, l'utilisation de seuils pour réduire les règles est arbitraire d'une recherche à l'autre. Ainsi, il est essentiel de présenter une recherche qui assure la réduction des règles d'association sans utiliser de seuils et d'étudier la perte d'informations pour valider l'approche.

Chapitre 4 : Quine McCluskey

Chapitre 4

Quine McCluskey

4.1. Introduction

La méthode de Quine-McCluskey est largement utilisée pour la minimisation des fonctions booléennes. Elle permet de trouver de manière déterministe la forme minimale des fonctions. Cette méthode est facile à mettre en œuvre et à utiliser. Elle est également connue sous le nom de méthode de tabulation, car la méthode passe par des étapes déterministes pour minimiser des fonctions booléennes en sélectionnant des éléments essentiels appelés impliquants premiers, à l'aide d'un tableau [27]. Dans ce chapitre, nous verrons l'origine de Quine-McCluskey, son principe, puis enfin, des exemples d'application de la méthode.

4.2. Origine et principe Quine-McClusckey

4.2.1 Origine de Quine-McClusckey

Quine-McCluskey est basée sur le travail de John Stuart Mill [28], lorsqu'il a présenté ses méthodes des canons inductifs en 1843 [29], comme une tentative d'isoler une cause à partir d'une séquence complexe d'événements. Plus précisément, elle est basée sur la méthode de l'accord de Mill, où deux ou plusieurs cas d'un événement (effets) sont comparés pour voir ce qu'ils ont en commun. Ce point commun est identifié comme la cause [30]**.**

La Méthode a été développée par Willard V. Quine, puis étendue par Edward J. McCluskey. Son déroulement part d'une combinaison unique de présence/absence dans la table de vérité. La méthode minimise les paires de combinaisons différant par un seul bit. Et donc, ne tolère qu'une seule différence entre deux situations [31].

4.2.2 Principe de base

Par exemple, si on considère cette fonction booléenne : S = BCDE + ABCDE. Dans les deux termes, les éléments B, C, D, E sont présents. Seul l'élément « A » à deux états différents, étant absent dans le premier terme et présent dans le second. En utilisant les canons inductifs de Mill (1843) (méthode de l'accord), on peut déduire que « A » n'a pas d'effet direct sur le résultat,

car le point persistant dans les deux situations est BCDE. On tire la même conclusion en utilisant la sémantique logique en algèbre de Boole :

Il est évident que BCDE est un facteur commun entre les deux cas, donc l'expression : BCDE + ABCDE peut être réécrite comme BCDE (ꟾA + A). Puisque l'union de l'ensemble « ꟾA » et de l'ensemble « A » est logiquement égale à 1, le résultat de cette expression est « BCDE ». La même conclusion peut être obtenue avec Quine-McCluskey, seulement Quine-McCluskey notera la différence avec un « - » donnant :

-BCDE, où le « - » est utilisé pour indiquer l'élimination de l'élément A, qui étant variable, n'a pas d'influence sur le résultat et il est ignoré. On peut ainsi conclure que la méthode de Quine-McCluskey trouve quel est le minimum de conditions (ou combinaison de conditions) nécessaires et/ou suffisantes pour déclencher le résultat [28].

Les annotations suivantes sont utilisées dans la méthode de Quine-McCluskey :

1- « 1 » représente le vrai.
2- « 0 » représente le faux ou la négation.
3- « - » Représente que la variable est ignorée, car elle n'a pas d'effet direct sur le résultat.

4.3. Fonctionnement de la méthode de Quine-McCluskey

La méthode de Quine-McCluskey travaille avec des termes de la logique booléenne, chaque variable d'un terme apparaît exactement une fois sous forme vrai ou faux (1 ou 0). Le but de la méthode est de trouver les impliquants premiers. Ces derniers sont des éléments qui ne peuvent pas être combinés avec d'autres éléments pour éliminer une variable, exemple : pour l'expression E=WY + WX + WXZ. WY est un impliquant premier, car il ne peut être combiné avec aucun autre terme de l'expression E.

Enfin, la méthode cherche les impliquants premiers essentiels. Ces derniers sont des impliquants premiers qui couvrent à eux seuls une sortie de la fonction, nous allons détailler la notion d'impliquant premier essentiel plus loin. Nous pouvons exprimer l'ensemble des étapes de la méthode de Quine-McClusckey comme suit :

Admettons la fonction suivante : S= ¬W¬XY¬Z + W¬X¬Y¬Z + ¬WXY¬Z + W¬X¬YZ + W¬XY¬Z + W¬XYZ + WXY¬Z + WXYZ, où (¬W¬XY¬Z, W¬X¬Y¬Z, ¬WXY¬Z, W¬X¬YZ, W¬XY¬Z, W¬XYZ, WXY¬Z, WXYZ) sont des termes.

Étape 1 – la première étape de la méthode de Quine-McCluskey est la binarisation des termes et la formation de groupes. La transition vers la forme binaire permet de :

- Préserver les positions des variables.
- Faciliter la comparaison des termes.
- Accélérer la minimisation.

Exemple d'application:

TABLEAU 4. 1: TERMES DE LA FONCTION S EN BINAIRE

ID	Terme	Forme binaire des termes			
		W	X	Y	Z
1	¬W¬XY¬Z	0	0	1	0
2	W¬X¬Y¬Z	1	0	0	0
3	¬WXY¬Z	0	1	1	0
4	W¬X¬YZ	1	0	0	1
5	W¬XY¬Z	1	0	1	0
6	W¬XYZ	1	0	1	1
7	WXY¬Z	1	1	1	0
8	WXYZ	1	1	1	1

Pour utiliser la méthode de Quine-McCluskey, l'utilisateur doit reconnaitre les termes qu'il faut comparer. Par exemple le terme 1 (¬W¬XY¬Z) contient beaucoup moins de 1 (bit 1) comparé au terme 6 (W¬XYZ), ça ne sert à rien de les comparer, car il n'y a aucune chance qu'ils puissent être combinés. De même pour les termes 1 et 2 qui ont le même nombre de 1s. Pour faire gagner du temps à l'utilisateur, les termes qui ont le même nombre de bits 1 sont groupés ensemble. Les groupes sont dans un ordre croissant en fonction du nombre de 1s dans les termes, ce qui permet une comparaison plus rapide et plus facile.

TABLEAU 4. 2: RANGEMENT DES TERMES SELON LE NOMBRE DE 1

GROUPE	TID	Terme	Forme binaire des termes			
			W	X	Y	Z
1	1	⅂W⅂XY⅂Z	0	0	1	0
	2	W⅂X⅂Y⅂Z	1	0	0	0
2	3	⅂WXY⅂Z	0	1	1	0
	4	W⅂X⅂YZ	1	0	0	1
	5	W⅂XY⅂Z	1	0	1	0
3	6	W⅂XYZ	1	0	1	1
	7	WXY⅂Z	1	1	1	0
4	8	WXYZ	1	1	1	1

TID: Identifiant de terme

Ainsi, l'utilisateur peut comparer seulement les termes qui ont le plus de chance d'être combinés.

Étape 2 – chaque terme présent dans un groupe i est comparé avec les termes d'un groupe i+1. S'il y a une différence d'un seul bit entre deux termes, alors cette différence est marquée avec un « - » dans la position du bit différent. Les autres bits restent tels quels. Les nouveaux termes issus de la comparaison entre le groupe i et i+1 seront placés dans un nouveau groupe. Exemple : dans le tableau 4.2, les combinaisons entre le groupe 1 et 2 donneront naissance au groupe 5, et les combinaisons entre le groupe 2 et 3 donneront naissance au groupe 6, etc.

Exemple d'application :

De manière itérative, tous les termes sont comparés. Nous donnons à chaque combinaison un identifiant (TID). Ce dernier, indique qu'un nouveau terme t_i a été formé en combinant les termes t_n et t_m, exemple : dans le tableau 4.3, le terme ⅂WYZ (TID : 1,3) a été formé en combinant les termes 1 et 3, soit respectivement ⅂W⅂XY⅂Z et ⅂WXY⅂Z. À la fin des comparaisons, les termes qui n'ont pas été combinés sont considérés comme des impliquants premiers. Toute redondance est omise.

En comparant les termes ⅂W⅂XY⅂Z (0010) et ⅂WXY⅂Z (0110), ces derniers ont une seule différence (le bit dans la position 2). Cette différence est marquée par « - », donnant ainsi :

⅂W-Y⅂Z (0-10). Peut-être réécrit : ⅂WY⅂Z (0-10). Les combinaisons entre les groupes 1 et 2 seront placées dans un nouveau groupe 5.

TABLEAU 4. 3: MINIMISATION DE LA FONCTION S

GROUPE	TID	Terme	W	X	Y	Z	Retenu
5	1,3	⅂WY⅂Z	0	-	1	0	
	1,5	⅂XY⅂Z	-	0	1	0	
	2,4	W⅂X⅂Y	1	0	0	-	
	2,5	W⅂X⅂Z	1	0	-	0	
6	3,7	XY⅂Z	-	1	1	0	
	4,6	W⅂XZ	1	0	-	1	
	5,6	W⅂XY	1	0	1	-	
	5,7	WY⅂Z	1	-	1	0	
7	6,8	WYZ	1	-	1	1	
	7,8	WXY	1	1	1	-	

TID : identifiant des termes.

Étape 3 - l'étape 2 est répétée, avec les termes nouvellement formés jusqu'à ce que tous les impliquants premiers soient trouvés. Tout terme (initial ou issu d'une minimisation) qui n'a pas pu être combiné est considéré comme impliquant premier.

Exemple d'application:

TABLEAU 4. 4: MINIMISATION DE LA FONCTION S, PHASE 2

GROUPE	TID	Terme	W	X	Y	Z	Retenu
8	1,3,5,7	Y⅂Z	-	-	1	0	
	1,5,3,7	Y⅂Z	-	-	1	0	
	2,4,5,6	W⅂X	1	0	-	-	
	2,5,4,6	W⅂X	1	0	-	-	
9	5,6,7,8	WY	1	-	1	-	
	5,7,6,8	WY	1	-	1	-	

TID : identifiant des termes. En rouge : termes redondants.

Les termes des groupes 8 et 9 ne peuvent pas être combinés, ils sont donc des impliquants premiers. En enlevant la redondance, nous obtenons le tableau 4.5.

TABLEAU 4. 5: IMPLIQUANTS PREMIERS

TID	Terme	W	X	Y	Z	Retenu
1,3,5,7	Y⎤Z	-	-	1	0	#
2,4,5,6	W⎤X	1	0	-	-	#
5,6,7,8	WY	1	-	1	-	#

TID : identifiant des termes.

Étape 4 - La table des impliquants premiers est formulée. Elle est constituée d'un ensemble de lignes et de colonnes. Les impliquants premiers peuvent être placés par ligne et les termes (ou identifiants des termes TID) peuvent être placés par colonne. Un « 1 » est placé dans les cellules qui correspondent aux termes qui sont couverts par chacun des impliquants premiers. Exemple : Y⎤Z, a été formé par la minimisation avec les termes 1,3,5,7. Donc, un 1 sera mis dans les colonnes correspondantes.

TABLEAU 4. 5: TABLEAU DES IMPLIQUANTS PREMIERS

Impliquants premiers	TID							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Y⎤Z	1		1		1		1	
W⎤X		1		1	1	1		
WY					1	1	1	1

Étape 5 - Les impliquants premiers essentiels sont trouvés en observant chaque colonne. Si un terme n'est couvert que par un seul impliquant premier, alors cet impliquant premier est un impliquant premier essentiel. Ces derniers feront partie de la fonction booléenne simplifiée.

Exemple d'application :

L'impliquant premier essentiel doit couvrir à lui seul une colonne du tableau. Chaque impliquant est représenté par les TID (les termes impliqués dans la minimisation, lors de leur création).

On remarque que Y⎤Z, couvre à lui seul la colonne (TID) 1. Cela veut à dire que : Y⎤Z, est un impliquant premier essentiel. Les 1s dans la ligne d'Y⎤Z, et dans les colonnes couvertes, sont supprimés.

TABLEAU 4. 6: RECHERCHE DES IMPLIQUANTS PREMIERS ESSENTIELS, ITÉRATION 1

Impliquants premiers	TID							
	1	2	3	4	5	6	7	8
Y⊻Z	1		1		1		1	
W⊻X		1		1	1	1		
WY					1	1	1	1

Donc nous avons: S = Y⊻Z.

Étape 6 - Le tableau des impliquants premiers est parcouru, en supprimant les 1s dans la ligne de chacun des impliquants premiers essentiels et les colonnes correspondant aux termes qui sont couverts par ce dernier. L'étape 5 est répétée pour le tableau des impliquants premiers réduits. La méthode s'arrête lorsque tous les « 1 » dans le tableau des impliquants premiers sont supprimés.

TABLEAU 4. 7: RECHERCHE DES IMPLIQUANTS PREMIERS ESSENTIELS, ITÉRATION 2

Impliquants premiers	TID							
	1	2	3	4	5	6	7	8
W⊻X		1		1		1		
WY						1	1	1

W⊻X, couvre à lui seul la colonne (TID) 2 et 4. Donc W⊻X est un impliquant premier essentiel. Les 1 dans la ligne de W⊻X et dans les colonnes couvertes sont supprimés.

Donc: S = Y⊻Z + W⊻X.

TABLEAU 4. 8: RECHERCHE DES IMPLIQUANTS PREMIERS ESSENTIELS, ITÉRATION 3

Impliquants premiers	TID							
	1	2	3	4	5	6	7	8
WY							1	1

Il ne reste que WY, alors il est aussi un impliquant premier essentiel, car c'est le dernier élément restant. Ces 1 sont supprimés.

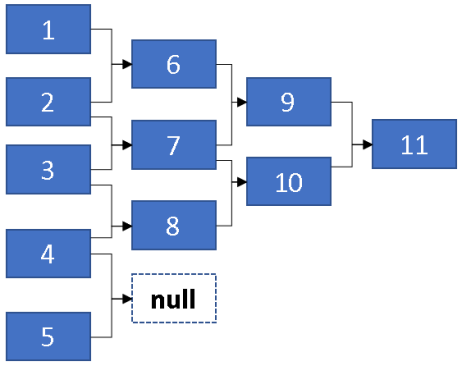
L'algorithme s'arrête puisqu'il y a plus de 1s dans le tableau.

Finalement : S’ = Y1Z + W1X + WY.

4.4 Algorithme de Quine-McCluskey

L’algorithme de Quine-McCluskey fonctionne en combinant un groupe i avec un groupe i+1. Un groupe contient les éléments ayant le même nombre de bits 1. Le déroulement de la combinaison est comme suit : si on reprend les termes du tableau 4.3, G_1 contiendra les termes ayant le nombre de bits 1 égale à un, G_2 contiendra les termes ayant le nombre de bits 1 égale à 2, G_3 contiendra les termes ayant le nombre de bits 1 égale à 3, et finalement G4 aura les termes ayant le nombre de bits 1 égale à 4. Un nouveau groupe sera créé à condition qu’au moins un terme d’un groupe G_i ait été combiné avec un terme d’un groupe G_{i+1}, la figure 4.1 montre un exemple de déroulement des combinaisons et formation de groupes.

FIGURE 4. 1 REPRÉSENTATION DES COMBINAISONS DE GROUPES



4.4.1 Algorithme de Quine-McClusckey

PSEUDO ALGORITHME 4. 1: RANGEMENT DES TERMES SELON LE NOMBRE DE BITS 1

```
NM: nombre maximum de bits 1
T: ensemble des termes binarisés
NB: Nombre maximum de 1
Pour i de 0 à NM:
        Pour chaque terme t dans T:
                N=Compter le nombre de bits 1 dans t
                        Si N==i:
                                Créer groupe(i)
                                Pour chaque terme t2 dans T:
                                        N2=Compter le nombre de bit 1 dans t2
                                        Si N2==i:
```

```
                                                    Ajouter la ligne t2 au groupe(i)
```

L'algorithme 4.1 permet de ranger les termes selon le nombre de bits 1. Il parcourt les termes binarisés et compte le nombre de 1s. Les termes qui ont le même nombre de bit 1 sont ajoutés au même groupe G_i. De cette façon les termes de chaque Groupe G_i peuvent être comparés avec les termes d'un groupe G_{i+1} pour réaliser les minimisations, si possible.

PSEUDO ALGORITHME 4. 2: COMPARAISON DES TERMES

```
GR: liste des Groupes
NG: liste des nouveaux Groupes
OG: liste des anciens Groupes
LC: liste des Combinaisons
TGR:  taille de GR
TNG:  taille de NG
Pour chaque groupe Gi dans OG
    Créer_groupe=0
    NG = [] de taille 0
    Pour chaque terme t dans Gi
        Pour chaque terme t2 dans Gi+1
        Combinaison=Comparer t avec t2
        Si Combinaison ! = null:
            Ajouter à LC (Combinaison)
            Créer_groupe=1
        Si Créer_groupe==1
            Créer groupe (GTGR+1)
            Ajouter GTGR+1 à NG
            Pour chaque combinaison C dans LC
                 Ajouter combinaison C au groupe (GTGR+1)
        GR = GR + NG
        OG = NG
    Si TNG == 0
        Arrêter la comparaison
```

Pour comparer les termes, comme exprimé avec l'algorithme 4.2, chaque terme d'un groupe G_i se voit alors comparé à tous les termes de G_{i+1}. Si deux termes n'ont qu'une seule différence entre eux, un nouveau groupe G_{TGR+1} est créé et la combinaison des deux termes est ajoutée à ce groupe. L'algorithme arrête les comparaisons une fois qu'il n'y a plus de nouveaux groupes.

PSEUDO ALGORITHME 4. 3: RASSEMBLEMENT DES IMPLIQUANTS PREMIERS

```
GR: liste Des Groupes
TC: liste des Termes combinés
LIP: liste des impliquants premiers
Pour chaque terme t dans GR:
    Si TC ne contient pas t:
      Ajouter t à LIP
```

Cette partie de code parcourt tous les groupes générés et cherche tous les impliquants premiers. Un terme est un impliquant premier, s'il n'a pas été combiné avec aucun autre terme.

4.4.1.3 Table des impliquants premiers

PSEUDO ALGORITHME 4. 4: RECHERCHE DES IMPLIQUANTS PREMIERS ESSENTIELS

```
Grille: Tableau des impliquants premiers
LIPE: liste des impliquants premiers essentiels
Pour chacun des impliquants premiers IP dans Grille
  Pour chaque TID dans IP
    Si compter_Les_Un_de_TID == 1
      Ajouter IP a LIPE
      Supprimer les «1 » qui ont le même TID dans IP
      Supprimer les «1 » de IP
```

Les impliquants premiers sont rangés dans un tableau. L'algorithme parcourt le tableau, et pour chaque impliquant premier, il compte le nombre de 1s dans la colonne correspondante à son TID. Si l'impliquant premier couvre à lui seul un TID (colonne), il est retenu comme impliquant premier essentiel. Les 1s sont supprimés dans les colonnes correspondantes ainsi que ceux dans la ligne de chacun des impliquants premiers essentiels. Le tableau 4.11 ci-dessous montre un exemple de déroulement. Les impliquants « a » et « d » sont essentiels et sont retenus.

TABLEAU 4. 9: TABLE DES IMPLIQUANTS PREMIERS

Impliquant premier	TID				
	1	**2**	**3**	**4**	**5**
a	1		1		
b		1	1		1
c		1	1		1
d				1	1

PSEUDO ALGORITHME 4. 5: RECHERCHE DES IMPLIQUANTS PREMIERS RESTANTS

```
Grille: Tableau des impliquants premiers
LIP: liste des impliquants premiers
Pour chacun des impliquants premiers IP dans Grille
  MAXLINE=0
  Pour chaque TID dans IP
    MAX=Compter les «1 » dans IP + compter les «1 » qui ont
    le même TID qu'IP
    Si MAX >= MAXLINE
      MAXLINE=MAX
      Supprimer les «1 » qui ont le même TID dans IP
      Supprimer les «1 » d'IP
      Ajouter IP à LIP
```

Les impliquants premiers restants sont recherchés. L'algorithme parcourt la table, et pour chacun des impliquants restants, compte le nombre de 1s de la même façon que les impliquant premiers essentiels. Mais, il retient l'impliquant qui élimine le plus de « 1 ». En reprenant le Tableau 4.11 ci-dessus, les impliquants b et c éliminent tous les deux le même nombre de « 1 », l'algorithme prendra l'un d'entre eux.

TABLEAU 4. 10: TABLE DES IMPLIQUANTS PREMIERS : ÉLIMINATION DES 1S DES IMPLIQUANTS PREMIERS RESTANTS

Impliquant premier	TID				
	1	2	3	4	5
a					
b		1			
c		1			
d					

4.6. Conclusion

Dans ce chapitre, nous avons présenté la méthode de Quine-McCluskey. Nous avons parlé de son origine et de son principe visant à trouver le nombre minimal d'événements nécessaires pour déclencher un résultat, non pas en trouvant une intersection entre les événements, mais en éliminant les événements variables et en gardant seulement les événements persistants. Nous avons déroulé un exemple de minimisation en utilisant la méthode, où nous avons vu que la méthode peut minimiser une fonction booléenne sans aucun paramètre à donner par l'utilisateur.

Dans le chapitre suivant intitulé: Méthodologie, nous verrons notre méthodologie en ce qui concerne la réduction du nombre de règles d'association.

Chapitre 5: Méthodologie

Chapitre 5
Méthodologie :

5.1 Introduction

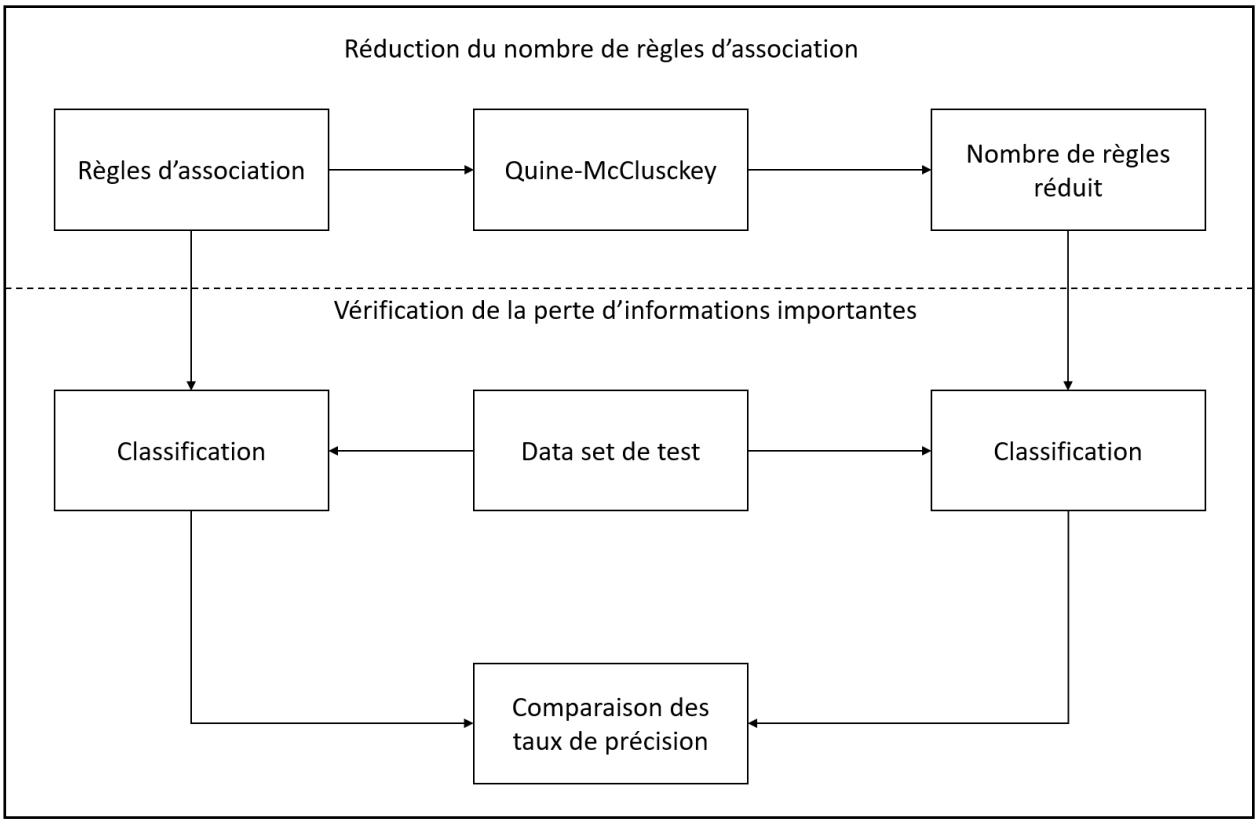
Dans ce chapitre, tout d'abord nous verrons l'architecture globale de notre système. Nous détaillerons chaque étape tout en montrant l'objectif de celle-ci.

5.2. Architecture de notre système :

La figure 5.1 résume l'architecture générale de notre projet. Notre système s'exécute dans les étapes suivantes :

1. Le système a comme entrée un ensemble « E » de règles d'association.
2. Le nombre de règles d'association est réduit en appliquant Quine-McCluskey, pour obtenir un nouvel ensemble de règles « E' ».
3. Le système utilise l'ensemble « E » comme base de connaissances pour classifier un data-set de test T, et sauvegarde les résultats.
4. Le système utilise ensuite l'ensemble « E' » comme base de connaissances pour classifier le même data-set de test T, et sauvegarde les nouveaux résultats.
5. Les deux résultats sont comparés, pour déduire si une perte des informations importantes a été engendrée.

FIGURE 5. 1 ARCHITECTURE GLOBALE DU SYSTEME



Procédure Programme_Principal (x, x', D)

Variable

t, t' : taux de précision

x : règles d'association initiale

x': règles d'association après la réduction

D : data-set de test

Début

s = S(x)

s' = QUINE-MCCLUSKEY(s)

t = classifier(D, s)

t' = classifier(D, s')

Comparer t avec t'

Fin

5.3 Phase de réduction du nombre de règles d'association :

Avant d'utiliser une méthode de minimisation de fonction booléenne telle que la méthode de Quine-McCluskey sur les règles d'association, nous avons besoin de mettre en œuvre une approche permettant de présenter les règles sous une forme booléenne.

Puisque la méthode de Quine-McCluskey minimise une fonction S contenant des termes booléens, en utilisant les annotations suivantes : 1 représente le vrai ; le 0 la négation ; le « - » qu'une variable est ignorée.

Alors, nous proposons l'approche suivante pour appliquer la méthode de Quine-McCluskey sur les règles d'association :

1- Pour écrire les termes de la fonction S nous nous inspirons des règles d'association négatives [16], où la négation d'un item représente son absence. Exemple : a¬b→c, est égal à : a→c, puisque ¬b représente l'absence de b.

Donc, les termes de la fonction S peuvent être soit des antécédents pour un $i_{ème}$ conséquent, soit des conséquents pour un $i_{ème}$ antécédent. Exemple : les antécédents dans les règles ab → c, a → c peuvent être réécrits respectivement : ab, a¬b.

Pour les règles a → b, a → c, leurs conséquents peuvent être réécrit respectivement: b¬c, ¬bc.

2- Les règles ayant les mêmes conséquents sont minimisées ensemble et les règles ayant les mêmes antécédents sont minimisées ensemble. Nous proposons alors deux types de minimisations : par les mêmes antécédents et par les mêmes conséquents.

« La minimisation par les mêmes conséquents consiste à minimiser les règles ayant le même conséquent ».

« La minimisation par les mêmes antécédents consiste à minimiser les règles ayant le même antécédent ».

À ce stade seulement, la minimisation des règles d'association est effectuée « si possible ». Lors de la minimisation, nous utilisons les mêmes annotations de Quine-McCluskey, mais avec les significations suivantes :

1- « 1 » représente la présence d'un item.
2- « 0 » représente l'absence d'un item.
3- « - » représente que l'item est ignoré, car il n'a pas d'effet direct sur la règle d'association.

Exemple :

**Cas de minimisation possible

[abd → c
  ab¬d→ c]

Résultat : ab→ c

Puisque les deux règles ont le même conséquent, nous appliquons la minimisation par mêmes conséquents. La méthode donne ab→ c, parce qu'« ab » (antécédent) est persistant entre les deux règles et une seule différence existe entre elles. Ce résultat peut s'expliquer par le fait qu'on peut déduire des deux règles qu'« ab » donne à lui seul « c » ; « abd » donne « c » ; mais jamais « d » donnant « c ». Donc « d » est ignoré, car il n'a pas d'influence sur le résultat et la méthode déduit qu'« ab » est suffisant pour avoir « c ».

**Cas de minimisation impossible :

Exemple 1:

[abc → d
  a¬b¬c→ d]

Résultat : aucune minimisation possible.

ou

Exemple 2 :

[abc¬e→d
  a¬b¬ce→d]

Résultat : aucune minimisation possible.

Avoir un point commun ne suffit pas. La méthode ne donne pas de résultat, même si « a » est persistant entre les règles. En effet, Quine-McCluskey ne cherche pas à trouver une intersection, donc aucun résultat n'est obtenu.

En règle générale, une seule différence est tolérée pour effectuer la minimisation. Ceci fait de Quine-McCluskey une méthode capable de réduire efficacement les banques de données.

5.4.5 Exemple d'application :

Admettons l'ensemble « E » des règles d'association suivantes :

Règles d'association
ab→m
ab→mn
ab→e
b→e
df→e
f→e

Dans l'ensemble « E » nous pouvons sortir deux sous-ensembles. Le premier contenant les règles ayant le même conséquent « e ». Et le deuxième contenant les règles ayant le même antécédent « ab ».

5.4.5.1 la réduction par mêmes conséquents :

ab→e
b→e
df→e
f→e

1. Notre algorithme commence par former la fonction booléen S comme suit : D'abord, il rassemble les items qui forment les antécédents.

E_a=a,b,d,f ← ab → e, b → e, df → e, f → e

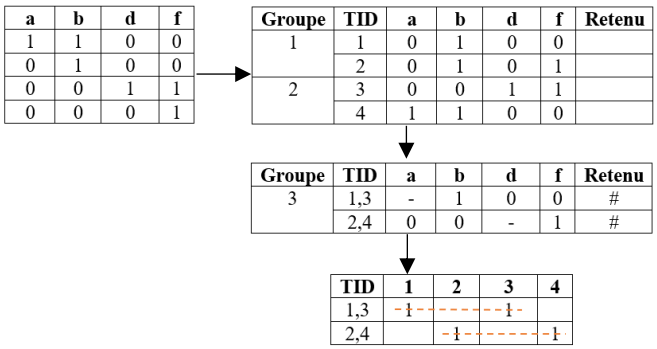
E_a: est la liste des items qui forment les antécédents.

Maintenant chaque antécédent d'une règle est comparé avec E_a. Si un item est absent, son absence est représentée par sa négation.

ab¬d¬f →e
¬ab¬d¬f →e
¬a¬bdf →e
¬a¬b¬df →e

Finalement, la fonction S va recevoir les antécédents sous leur forme booléenne.
S= ab$\bar{d}\bar{f}$ + $\bar{a}\bar{b}$d$\bar{f}$ + $\bar{a}\bar{b}$df + $\bar{a}$b$\bar{d}$f.

2. Maintenant la fonction S est minimisée en appliquant la méthode de Quine-McCluskey.



a	b	d	f
1	1	0	0
0	1	0	0
0	0	1	1
0	0	0	1

Groupe	TID	a	b	d	f	Retenu
1	1	0	1	0	0	
	2	0	1	0	1	
2	3	0	0	1	1	
	4	1	1	0	0	

Groupe	TID	a	b	d	f	Retenu
3	1,3	-	1	0	0	#
	2,4	0	0	-	1	#

TID	1	2	3	4
1,3	1		1	
2,4		1		1

Le résultat est : S'= b$\bar{d}\bar{f}$ + $\bar{a}\bar{b}$f.

Donc, les nouvelles règles sont :

Règles résultantes
b$\bar{d}\bar{f}$ →e
$\bar{a}\bar{b}$f→e

Nous pouvons les exprimer sans la négation :

Règles résultantes
b →e
f→e

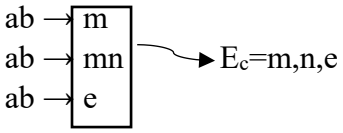
5.4.5.2 la réduction par mêmes antécédents :

Depuis l'ensemble « E », nous avons les règles qui partagent le même antécédent « ab »:

ab→m
ab→mn
ab→e

Réduction par mêmes antécédents, donc :

1. L'algorithme commence par former la fonction booléen S comme suit :
   D'abord, il commence par rassembler les items qui forment les antécédents.



   E_c: est la liste des items qui forment les conséquents.

   Maintenant chaque antécédent d'une règle est comparé avec E_c. Si un item est absent, son absence est représentée par sa négation.

   ab →m˥n˥e
   ab →mn˥e
   ab →˥m˥ne

   Finalement, la fonction S va recevoir les conséquents sous leurs formes booléennes.
   S= m˥n˥e + mn˥e + ˥m˥ne.

2. Maintenant la fonction S est minimisée en appliquant Quine-McCluskey.

Le résultat est : S'= m˥e + ˥m˥ne.

Donc, les nouvelles règles résultantes sont :

Règles résultantes
ab → m˥e
ab → ˥m˥ne

Nous pouvons les réécrire sans la négation :

Règles résultantes
ab → m
ab → e

5.3 Phase de vérification de la perte des informations importantes :

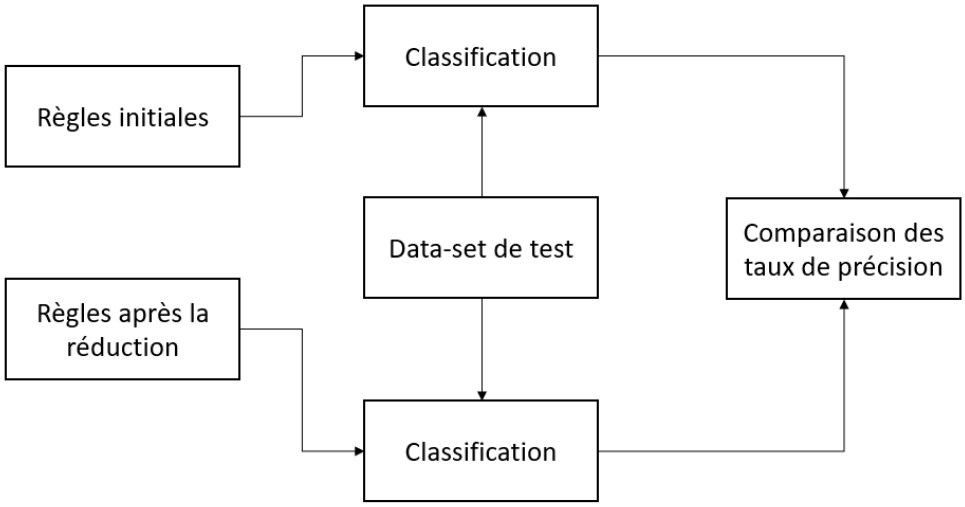
Il est clair qu'une réduction sur un ensemble de données comme les règles d'association ne peut être sans conséquence. En effet, l'enjeu dans une réduction du nombre de règles d'association est la perte d'informations importantes. Le but de cette étape est de vérifier si une telle perte a été engendrée.

Puisque nos données sont des règles d'association, nous avons : « E' », un ensemble de règles d'association obtenu en réduisant « E » (un ensemble de règles initial). En utilisant « E » et « E' » dans une même tache F commune et en comparant les résultats, nous pouvons savoir si la réduction par notre approche a engendré une perte des informations importantes.

Nous avons choisi la classification comme tache commune, car cette approche est un très bon moyen permettant de savoir si notre méthode engendre une perte des informations importantes, en comparant les performances des classifications entre les deux états du système. Le premier, est l'état avant la réduction et le deuxième, après la réduction. En effet, dans l'éventualité où les performances des classifications (taux de précision) baissent, ceci est une indication incontestable que des informations importantes ont été perdues.

Comme le montre la figure 5.2 ci-dessous, nous comparons les taux de précision des classifications avant la réduction avec ceux obtenus après la réduction. Notre système commence par l'utilisation des règles d'association initiales comme base de connaissances, puis classifie plusieurs data-set de tests et sauvegarde les taux de précision. Ensuite, il utilise les règles obtenues après la réduction comme base de connaissances et classifie les mêmes data-sets de tests. Enfin, les nouveaux taux de précision sont comparés aux premiers.

FIGURE 5. 2 LA VERIFICATION DE LA PERTE DES INFORMATIONS



5.4. Implémentation :

5.4.1 Langage choisi pour l'implémentation

Nous avons implémenté notre système en utilisant le langage Python 3. Nous l'avons utilisé pour mieux manipuler des structures de données telles que les matrices. Notre système est portatif et ne dépend pas d'un système d'exploitation spécifique.

5.4.2 Liste des scripts Python

-Exctract_By_Consequents.py : Ce script réalise l'extraction des règles ayant les mêmes conséquents. Il commence par rassembler les conséquents qui se répètent au moins deux fois. Puis, transforme leurs antécédents en termes. La fonction de binarisation a comme premier paramètre X=c, représente le type d'extraction ; le deuxième paramètre représente les termes ; et le troisième Y représente le conséquent des règles.

Ensemble de règles → Extract_by_Conséquents → X : termes : Y → Binarisation → X : S_{bin} : Y

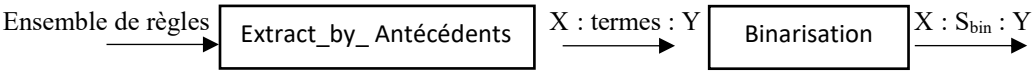
Exemple :

[a→b, ab→b, e→d, f→d, n→m] → Extract_by_Conséquents → [c: a⅂b + ab : b], [c: e⅂f +⅂ef : d]

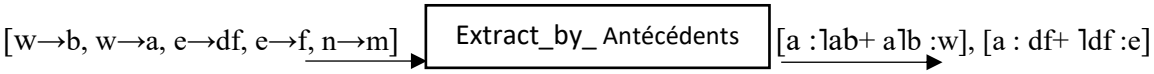
[c : a⅂b + ab : b], [c: e⅂f + ⅂ef : d] → Binariser → [c : 10 + 11 : b], [c : 10 + 01 : d]

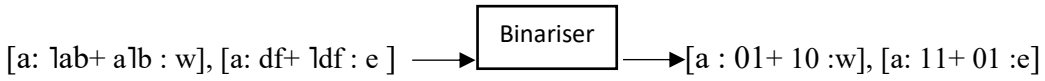
```
Fonction Extract_by_Conséquents (x : liste) : liste
Variable
      x : règles d'association
      S : fonction booléenne
      Ea : liste des items formant les antécédents
      Ec : liste des items formant les conséquents
Début
      Si réduction par même conséquents alors
            Ea = liste_Antécedents(x)
            Pour chaque règle R dans x Faire
                  Y=Conséquents(R)
                  Terme = ""
                  Pour chaque item I dans Ea Faire
                        Si I présent dans antécédents(R) alors
                              Terme = Terme + I
                        Sinon
                              Terme = Terme + ¬I
            S = S + terme
            Résultat = Résultat + [c,S,Y]
Retourner Résultat
Fin
```

-Exctract_By_Antécédents.py : Ce script réalise l'extraction des règles ayant les mêmes antécédents. Le paramètre X=a : représente l'extraction par les mêmes antécédents et le paramètre Y : l'antécédent des règles.



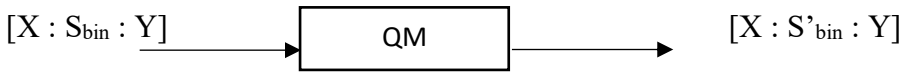
Exemple :



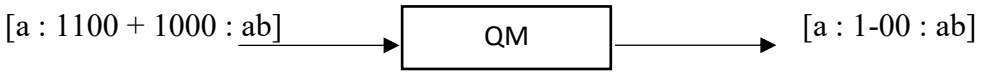



```
Fonction Extract_by_Antécédents (x : liste) : liste
Variable
      x : règles d'association
      S : fonction booléenne
      Ea : liste des items formants les antécédents
      Ec : liste des items formant les conséquents
Début
      Si réduction par même antécédent alors
            Ec = liste_Conséquents(x)
            Pour chaque règle R dans x Faire
                  Y=Antécédent(R)
                  Terme = ""
                  Pour chaque item I dans Ec Faire
                        Si I présent dans conséquents(R) alors
                              Terme = Terme + I
                        Sinon
                              Terme = Terme + ¬I
            S = S + terme
            Résultat = Resultat + [a,S,Y]
Retourner Résultat
Fin
```

-Quine-McCluskey.py : Ce script représente la méthode de Quine-McCluckey. Ce dernier reçoit comme premier paramètre un ensemble de termes sous forme binaire. Le deuxième paramètre sert à conserver le conséquent ou l'antécédent.

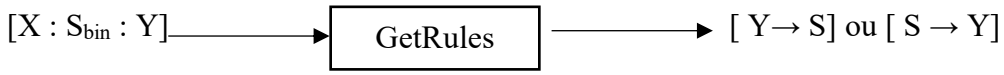


Exemple :




```
Fonction QUINE-MCCLUSKEY (X : caractère, Sbin : liste, Y : chaine de
caractère) : liste
Début
      Si X= 'c'
            S'bin=QUINE-MCCLUSKEY(Sbin)
            Retourner (c, S'bin,Y)
      Si X= 'a'
            S'bin=QUINE-MCCLUSKEY(Sbin)
            Retourner (a, S'bin,Y)
Fin
```

-GetRules.py : Permet de ramener les règles de la forme binaire à la forme d'une règle régulière. Le paramètre X est égal à : « a » représentant que Y est antécédent ou « c » que Y est conséquent.



Exemple :

```
Fonction GetRules (X: caractère, S_bin: liste, Y: chaine de caractère) : liste
Variable :
      Antécédent : Antécédent d'une règle d'association
      Conséquent : Conséquent d'une règle d'association
Début
      Si X= 'c'
            Pour chaque élément e dans S'_bin :
                  Si e == 0 :
                        Antécédent = Antécédent + le
                  Si e == 1 :
                        Antécédent = Antécédent + e
            Retourner [Antécédent → Y]
      Si X= 'a'
            Pour chaque élément e dans S'_bin :
                  Si e == 0 :
                        Conséquent = conséquent + le
                  Si e == 1 :
                        Conséquent = conséquent + e
            Retourner [Y → conséquent]
Fin
```

5.4.3 Suppression de la redondance des règles d'association après la réduction

Après la réduction du nombre de règles, il peut arriver que la base de connaissances soit submergée de règles redondantes. Par exemple : {ab →F, a → F} donnerait a- →F, qui se traduit par a→ F, et {ac → F, a→ F} donnerait a-→ F se traduisant par a→ F, on a donc une redondance. Dès lors, toute redondance doit être éliminée. Nous ne gardons qu'une seule règle représentative : a→ F.

5.5 Conclusion

Dans ce chapitre nous avons vu notre méthodologie permettant d'appliquer la méthode de Quine-McClusckey aux règles d'association. Dans le chapitre suivant, nous allons voir les différentes expérimentations que nous avons menés.

Chapitre 6 : Expérimentations et résultats

Chapitre 6

Expérimentations et résultats

6.1. Introduction

Dans ce chapitre, nous allons discuter des résultats que nous avons obtenus et de notre approche de validation. Nous avons utilisé quatre banques de données constituées de règles d'association, et de règles de classification. Les règles de classification sont un type de règles d'association dont le conséquent est une classe. Nous avons utilisé ce type de règles d'association, car notre approche de validation utilise la classification où les règles de classification y sont très utilisées en littérature.

La première et la deuxième banque de données sont constituées respectivement de règles de classification et des règles d'association régulières traitant la langue arabe. La troisième banque de données est constituée de règles de classification dans le but de classifier les programmes entre programmes malveillants et bénins. La dernière base contient des règles d'association dans le but de classifier des fleurs Iris.

Définition :

Règle d'association régulière : comme nous l'avons vu dans le deuxième chapitre de notre mémoire, une règle d'association régulière est une règle de la forme :

n(antécédents) → m(conséquents) ou n et m sont >= 1.

Exemple : pain, confiture → lait.

Règle d'association de classe : ces règles sont de la même forme que les règles régulières, a l'exception qu'elles ne peuvent avoir qu'un seul item dans la partie du conséquent qui représente une classe. Leur forme est la suivante : n(antécédents) → Classe, ou n est >=1.

Exemple : Maladie, Traitement → Médecine.

6.2. Expérimentations réalisées :

Toutes les expérimentations ont un seul but, à savoir la réduction du nombre de règles d'association et la vérification de la perte d'informations. Nous avons utilisé adéquatement nos méthodes de réduction : par même conséquents (RMC) ; par mêmes antécédents (RMA). Dans la première et troisième banque de données, nous avons utilisé la RMC, car les règles ont un seul item dans le conséquent qui représente une classe. Pour la deuxième et quatrième banque de données nous avons utilisé la RMC, RMA, et la combinaison des deux. Le but est de multiplier les contextes pour mieux étudier notre approche.

6.3. Banques de données utilisées.

Pour réaliser nos expérimentations, nous avons utilisé les banques de données suivantes :

6.3.1 SANAD: Single-Label Arabic News Articles Dataset for Automatic Text Categorization.

Cette banque de données contient des documents textuels en langue arabe, avec plus de 190 mille articles de journaux. Les articles de cette banque de données sont issus des journaux arabophones: Akhbarouna, Arabiya et Khaleej. Ces articles parlent de plusieurs sujets : Culture, Finance, Médicale, Politiques, Religion, Sports et technologies. Cette banque de données a été utilisée dans 29 articles traitants la langue arabe. Nous avons utilisé cinq classes depuis cette dernière pour extraire nos règles d'association à savoir : Culture, Finance, Médicale, Religion, Sports, chaque classe contient 100 documents constituant la base d'apprentissage et 30 documents de chaque classe pour la base de test. Pour extraire les règles d'association nous avons utilisé des méthodes qui ont montrées leur efficacité dans l'extraction des règles d'association depuis la langue arabe tel que : le stemming, la méthode TF-IDF, la binarisation et enfin, l'extraction des règles d'association avec l'algorithme Apriori. Nous avons extrait deux types de règles d'association à savoir : les règles d'association régulières et les règles d'association de classe.

Lien : https://data.mendeley.com/datasets/57zpx667y9/2

6.3.2 Programmes malveillants/bénins.

Cette banque de données contient des programmes exécutables bénins et malveillants que nous avons utilisés dans un autre travail. Ces programmes sont généralement difficiles à obtenir et une demande d'autorisation est requise pour éviter toutes utilisations non académiques. Plusieurs plateformes proposent ce service telles que virusign.com et malshare.com. Nous avons pu réunir 390 programmes malveillants et 214 programmes bénins.

Lien : https://doi.org/10.1177%2F1550147719889907

6.3.3 Iris.

Cette petite banque de données contient des caractéristiques de fleurs d'Iris, avec 151 échantillons. Cette banque de données a été utilisée dans 100 articles.

Lien : http://archive.ics.uci.edu/ml/datasets/Iris

6.4. Qualité des règles d'association extraites.

# banque de données	Type de règle	Nombre de règles	minsup	minconf	minlift
1	**RC**	467	55%	80%	1
2	**AR**	1430	55%	80%	1
3	**RC**	1413	60%	70%	1
4	**AR**	270	50%	75%	1

AR : règles d'association régulières; RC : règles d'association de classe

6.5. Validation de notre approche :

Pour valider notre approche, nous avons pris en considération les critères suivants :

6.5.1 la réduction du nombre de règles d'association.

Pour répondre à ce critère, nous devons rependre à la question suivante : peut-on réduire le nombre de règles d'association en utilisant notre approche ? Pour répondre à cette question, nous mesurons le taux de réduction du nombre de règles d'association.

$$Taux\ de\ reduction = 1 - \frac{nombre\ de\ regles\ apres\ la\ reduction}{nombre\ de\ regles\ avant\ la\ reduction}$$

6.5.2 la perte des informations importantes.

Pour satisfaire ce critère, nous devons vérifier si notre approche engendre une perte des informations importantes. En théorie, quand des informations importantes sont perdues, les performances d'un système qui a besoin de ces dernières dans des tâches telles que : la prédiction, la classification, etc., vont baisser. En pratique, nous pourrons déduire si notre approche engendre une perte des informations importantes en étudiant les taux de précision des classifications. En effet, si les taux de précision baissent significativement après la réduction, ceci est un signe incontestable que des informations importantes ont été perdues. Dans le cas contraire, les informations importantes ont été conservées.

6.6. Résultats obtenus :

TABLEAU 6. 1: LES RÉSULTATS OBTENUS

Sujet	# banque de données	Type de règle	Type de réduction	Taux de précision avant la réduction	Taux de précision après la réduction	Nombre de règles avant la réduction	Nombre de règles après la réduction
Langue arabe	**1**	**RC**	RMC1	87.33%	87.33%	467	171
	2	**AR**	RMC2	88.66%	88.66%	1430	437
			RMA2		88.66%		525
			RMC2+RMA2		92%		225
Programmes malveillants/bénins	**3**	**RC**	RMC3	82.73%	85.10%	1413	141
Iris	**4**	**AR**	RMC4	100%	100%	270	119
			RMA4		100%		119
			RMC4+RMA4		100%		37
			Moyenne	**92%**	**92.71%**		

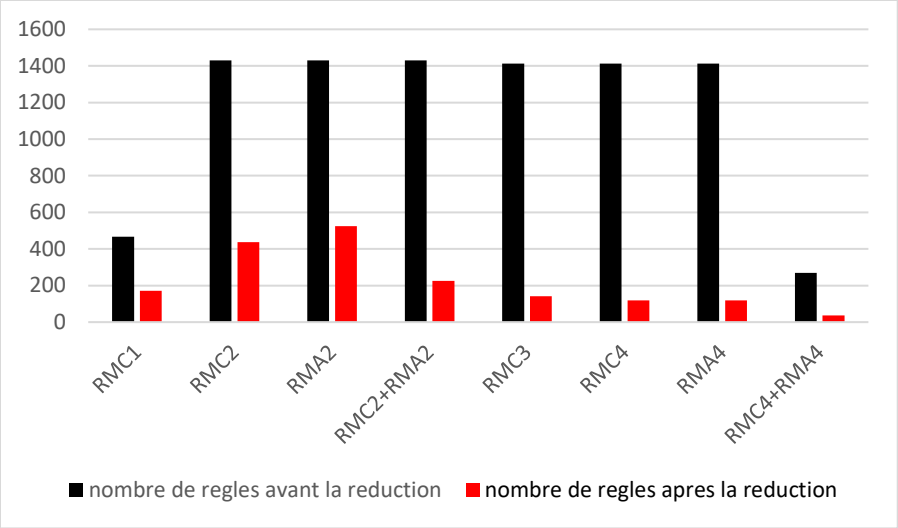
AR : règles d'association régulières; RC : règles d'association de classe

6.6.1. Étude de la réduction du nombre de règles d'association:

Comme le montre le tableau 6.1 ci-dessus, pour la première banque de données nous avons pu réduire le nombre de règles significativement à 171 au lieu de 467(taux de réduction de 63.3%). Pour la deuxième banque de données, le nombre de règles a été réduit utilisant la RMC de 1430 à 437(taux de réduction de 69.4%) règles ; de 1430 à 525(taux de réduction de 63.2%) avec la RMA; et de 1430 à 225(taux de réduction de 84.2%) en combinant les deux approches de réduction. Pour la troisième base, nous avons pu réduire le nombre de règles de 1413 à 141(taux de réduction de 90%) avec la RMC. Pour la dernière base, nous avons obtenu une réduction de 270 à 119(taux de réduction de 60%) règles avec la RMC, et avec la RMA de 270 à 119 règles. Finalement, pour la combinaison de deux, le nombre de règles a été réduit à 37(taux de réduction de 86.2%) règles.

Les résultats obtenus montrent que notre approche peu en effet réduire significativement le nombre de règles d'association sans fixer aucun seuil.

FIGURE 6. 1 COMPARAISON ENTRE LE NOMBRE DE RÈGLES AVANT ET APRÈS LA RÉDUCTION



6.6.2 Étude de la perte des informations importantes :

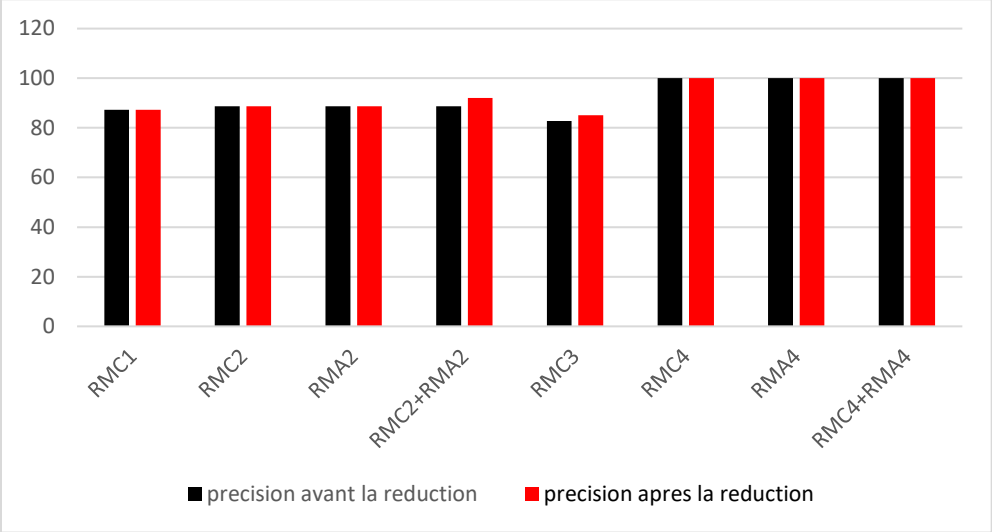
Nous étudions les taux de précision de la classification pour avoir une indication fiable et concrète permettant de vérifier si notre approche a engendré une perte d'informations importantes. Nous avons donné aux banques de données initiales les mêmes taches que celles obtenues après la réduction. Le taux de précision de la classification est calculé comme suit [32]:

$$Taux\ de\ precision = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

VP (Vrai Positif) : est le nombre d'objets correctement classifiés. FP (Faux Positif) : est le nombre d'objets incorrectement classifiés.

Comme le montre le tableau 6.1 ci-dessus, le taux de précision de la classification est resté le même pour la première base à savoir 87.33%. Pour la deuxième banque de données, le taux de précision de la classification avec la RMC et RMA a est resté le même (88.66%), mais il a été amélioré avec la combinaison des deux approches de réduction (de 88.66% à 92%). Pour la troisième banque de données, le taux de précision de la classification a été amélioré de 82.73% à 85.10%. Et enfin, nous pouvons voir une que les taux de précision de la classification pour la quatrième banque de données est resté le même.

FIGURE 6. 2 TAUX DE PRÉCISION



Les taux de précision de la classification sont restés très proches, ce qui indique que notre approche a pu réduire le nombre de règles d'association et garder le minimum d'informations importantes et suffisantes pour avoir des résultats satisfaisants. Nous pouvons aussi sortir quelques améliorations pour la classification, mais rappelons d'abord que notre but principal n'était pas d'améliorer la classification, mais d'utiliser la classification comme un test nous permettant de savoir si notre approche engendre une perte des informations importantes. Nous pouvons noter quelques bénéfices, comme les bases de connaissances qui ont été allégées en termes de taille(mémoire) et le temps moyen de la classification qui a été améliorée (de 1.4 s avant la réduction et 0.3 s après la réduction). D'autres améliorations peuvent apparaitre, mais, cela nécessite bien sûr, un autre travail complet visant cet objectif pour validation.

6.3.2.1 Visualisation de la perte des informations :

Dans ce qui suit, nous étudions les règles perdues ou conservées par leur importance. Dans le cadre de la classification, plus une règle est précise plus elle est importante pour la classification. La précision d'une règle d'association est calculée avec l'équation suivante [33]:

$$Precision\ (a \rightarrow b) = \frac{\mathrm{NT} \in \mathrm{C_i}}{\mathrm{NTT} \in \mathrm{C_i} + \ \Sigma_{\mathrm{n}}^{\mathrm{m}}((\mathrm{NT} \notin \mathrm{C_i})\ _n)}$$

Où $\mathrm{NT} \in \mathrm{C_i}$ est le nombre de transactions de classe $\mathrm{C_i}$ où la règle $a \rightarrow b$ est vraie (équivalent au nombre d'occurrence de l'antécédent avec le conséquent dans un ensemble de transactions).

$\mathrm{NTT} \in \mathrm{C_i}$ est le nombre total de transactions de classe $\mathrm{C_i}$.

$\mathrm{NT} \notin \mathrm{C_i}$ est le nombre de transactions de classe autre que $\mathrm{C_i}$ où la règle $a \rightarrow b$ est vraie. m est le nombre des autres classes autre que $\mathrm{C_i}$.

Cette équation nous donne l'importance d'une règle d'association dans le cadre de la classification. En effet, plus une règle a une précision élevée plus elle a un impact important sur la classification. La formule de la précision exprime que plus une règle représente une seule classe avec un score élevé, plus elle est précise. Exemple : admettons 3 classes de documents contenant chacune 100 transactions. Si une règle $a \rightarrow b$ issue de la classe C_1 est vraie dans 80 transactions de classe C_1, mais dans aucune des autres classes, la précision de $a \rightarrow b$ est égale à 0.8.

$$Precision\ (a \rightarrow b) = \frac{80}{100 + (0 + 0)}$$

Maintenant, admettons une règle $a \rightarrow c$ qui est vraie dans 80 transactions de classe C_1, dans 30 transactions de classe C_2 et finalement dans 10 transactions de classe C_3. La précision de $a \rightarrow c$ est égale à 0.57.

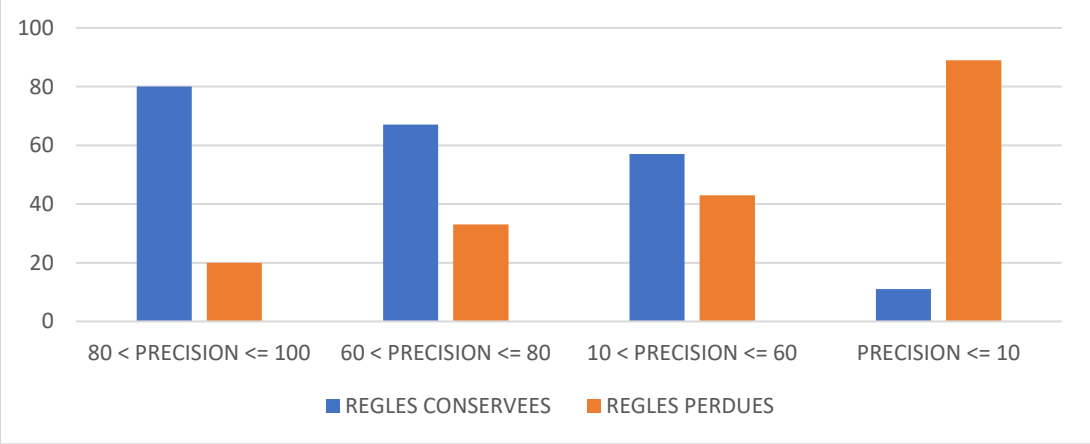
$$Precision\ (a \rightarrow c) = \frac{80}{100 + (30 + 10)}$$

Donc on peut dire que même si les deux règles soient vraies dans le même nombre de transactions de classe C_1, la règle $a \rightarrow b$ est plus précise (importante) que $a \rightarrow c$, puisque la deuxième règle représente plusieurs autres classes alors que la première règle représente une seule avec un score élevé. Ceci montre à quel point le fait de prédire plus d'une classe est pénalisant.

Dans la figure 6.3 ci-dessous, les règles de la banque de données 1 sont ordonnées par leurs précisions. Nous calculons les taux des règles manquantes (perdues) et conservées après la réduction. Le taux de perte est égal à : $Tp = \frac{\text{Np}}{\text{Np+Nc}}$. Et le taux de conservation : $Tc = \frac{\text{Nc}}{\text{Np+Nc}}$. Où N_c est le nombre de règles conservées et N_p le nombre de règles perdues.

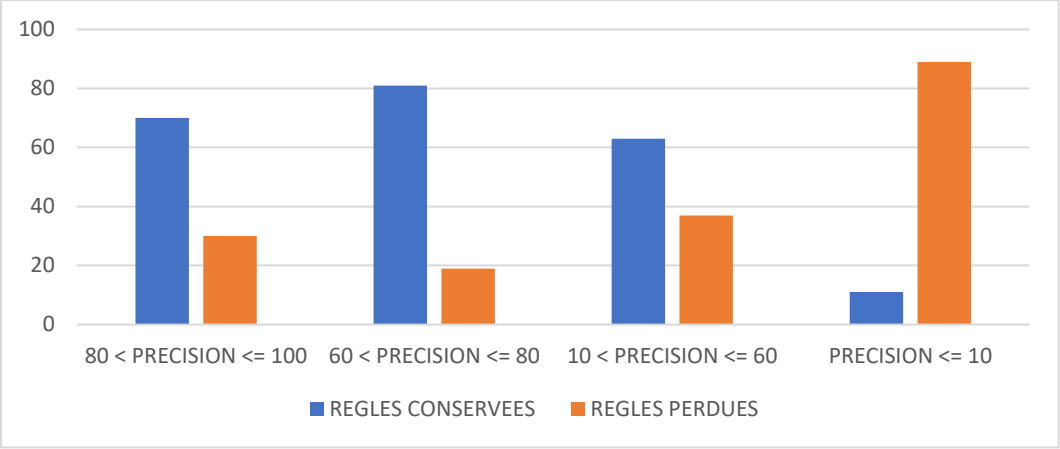
Nous pouvons voir que le taux de conservation des règles dont la précision se situe entre 80% et 100% sont largement supérieur à celui des règles perdues. Pour les règles dont la précision se situe entre 60% et 80%, le taux de conservation est supérieur au taux de perte. Pour les règles ayant une précision entre 10% et 60%, le taux de conservation est légèrement supérieur au taux de perte, et enfin, pour les règles dont la précision est inférieure ou égale à 10% le taux de perte est largement supérieur au taux de conservation.

FIGURE 6. 3: DATA-SET 1 : ÉTUDE DE LA PERTE D'INFORMATION PAR LA RMC



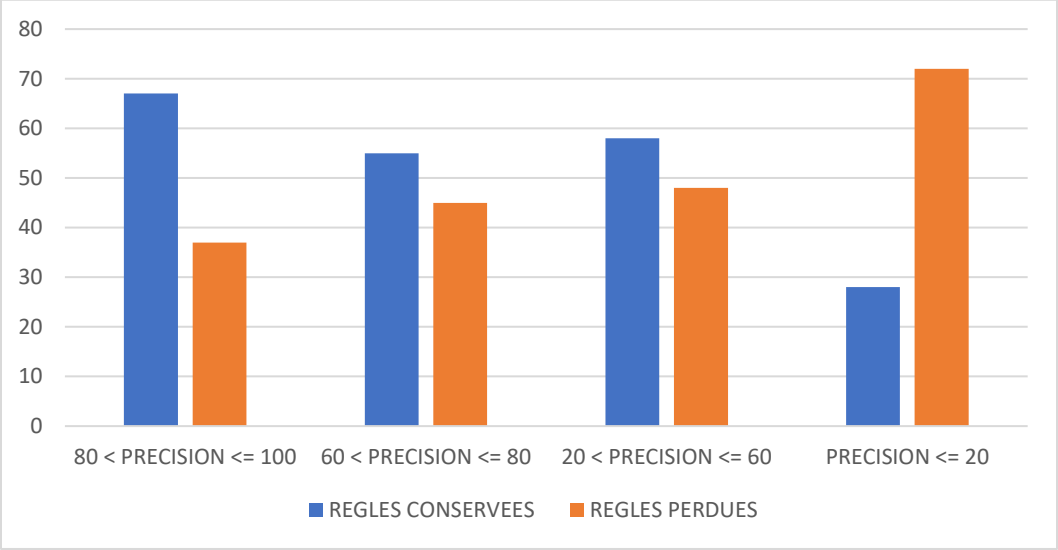
Pour la banque de données 2, en utilisant la RMC nous pouvons voir dans la figure 6.4 ci-dessous que le taux de conservation des règles dont la précision se situe entre 80% et 100%, 60% et 80%, 10% et 60% le taux de conservation est supérieur au taux de perte. Et enfin, pour les règles dont la précision est inférieure ou égale à 10% le taux de perte est largement supérieur au taux de conservation.

FIGURE 6. 4: DATA-SET 2 : ÉTUDE DE LA PERTE D'INFORMATION POUR RÈGLES D'ASSOCIATION PAR LA RMC



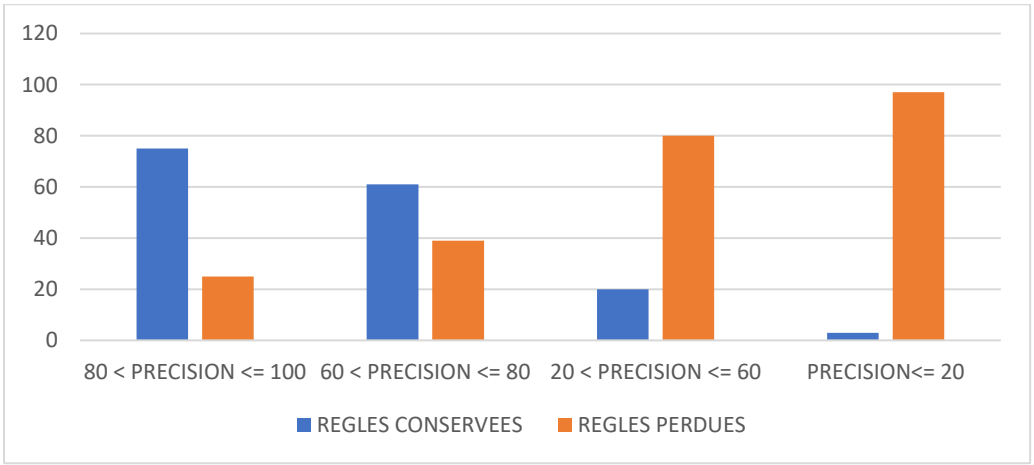
Voyons maintenant quelles sont les règles perdues lors de la réduction pour la deuxième banque de données avec la RMA. Nous pouvons voir que le taux de conservation des règles dont la précision supérieure à 80% le taux de conservation est largement supérieur au taux de perte. Pour les règles dont la précision est entre 20% et 80% le taux de conservation est légèrement supérieur au taux de perte. Pour les règles dont la précision est inférieure ou égale à 20% le taux de perte est largement supérieur au taux de conservation.

FIGURE 6. 5: DATA-SET 2 : ÉTUDE DE LA PERTE D'INFORMATION PAR LA RMA



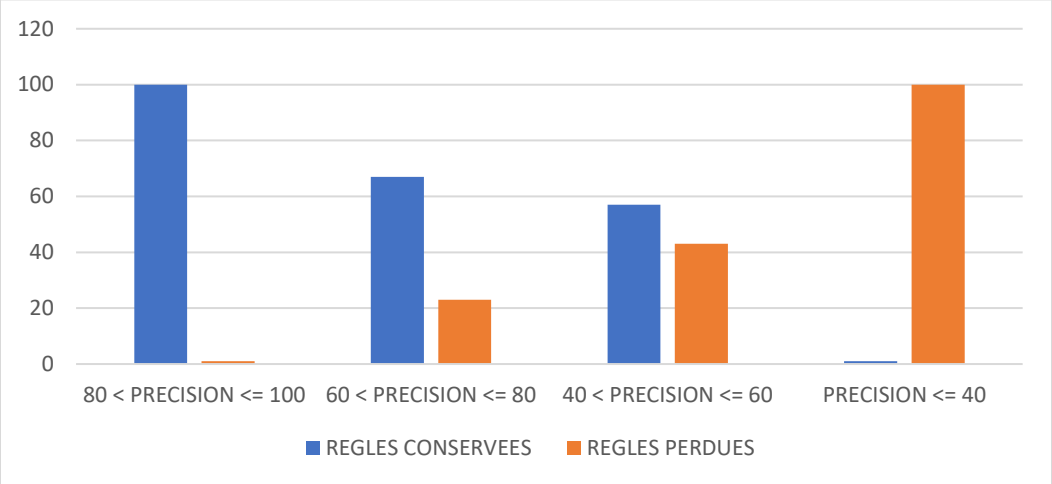
En combinant RMA et RMC nous obtenons les mêmes constats. Le taux de conservation des règles dont la précision est supérieure à 60% est supérieur au taux de perte. Pour les règles dont la précision est inférieure ou égale à 60% le taux de perte est supérieur au taux de conservation.

FIGURE 6. 6: DATA-SET 2 : ÉTUDE DE LA PERTE D'INFORMATIONS AVEC LA RMC + RMA



Pour la troisième banque de données, nous pouvons voir que le taux de conservation des règles dont la précision se situe entre 80% et 100% ont été toutes conservées. Pour les règles dont la précision se situe entre 60% et 80%, le taux de conservation est largement supérieur au taux de perte. Pour les règles ayant une précision entre 40% et 60%, le taux de conservation est légèrement supérieur au taux de perte, et enfin, pour les règles dont la précision est inférieure ou égale à 40% toutes les règles ont été perdues.

FIGURE 6. 7: DATA-SET 3 : ÉTUDE DE LA PERTE D'INFORMATIONS AVEC LA RMC.



6.7. Discussion

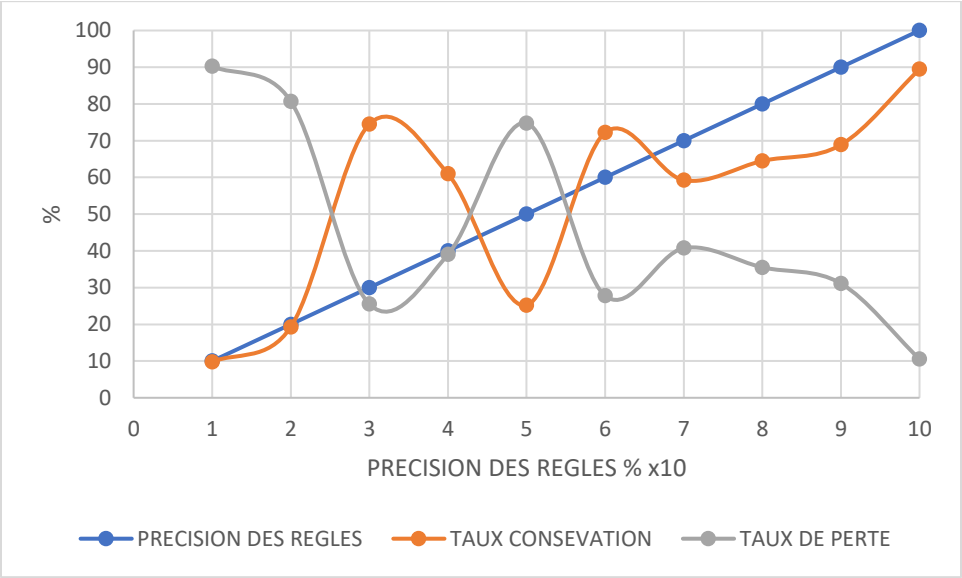
En appliquant notre approche de réduction du nombre de règles d'association et étudier les résultats obtenus nous avons remarqué les points suivants :

***La réduction du nombre de règles :** le nombre de règles d'associations est réduit ce qui allège les data-sets en termes de mémoire. Ainsi, nous avons pu satisfaire notre premier critère, qui est la possibilité de réduction du nombre de règles.

***La conservation des informations importantes :** Pour ce critère, nous avons constaté que les taux de précision de la classification sont restés proches et améliorés dans certains cas. Ceci montre que les règles qui ont été majoritairement conservées sont des règles importantes en termes d'informations. Nous avons aussi pu vérifier cela, en étudiant le taux des règles perdues et conservées par leurs scores de précision. Cette étude a montré que notre approche a tendance à conserver les règles ayant une précision élevée, autrement dit, les règles importantes. Nous pouvons aussi étudier les précisions des règles, les taux de pertes et de conservations de tous les data-sets. Nous avons divisé nos règles par intervalle de 10% de précision.

Dans la figure 6.8 ci-dessous et depuis les figures 6.3, 6.4, 6.5, 6.6 et 6.7, nous pouvons voir une tendance : plus le taux de précision des règles augmente, plus le taux de perte diminue, le taux de conservation, compte à lui, augmente et inversement:

FIGURE 6.8: ANALYSE DES TAUX DE PERTE, CONSERVATION ET LA PRÉCISION DES RÈGLES.



6.8. Conclusion

Pour toutes les expérimentations réalisées, en ce qui concerne la réduction, nous avons pu réduire le nombre de règles significativement. Pour la perte des règles importantes, nous avons pu déterminer que notre approche conserve les informations importantes en comparant les taux de classification d'avant et après les réductions, qui sont restés très proches. En étudiant les règles perdues et conservées nous avons pu constater que la méthode a tendance de conserver les règles à scores élevés en majorité, avec quelques pertes de règles de scores élevés et la conservation de certaines règles de scores faibles aussi. Ceci démontre qu'avoir un score élevé ne signifie pas pour autant qu'une règle est plus importante qu'une autre en termes d'informations. Nous pouvons donc conclure que notre approche a pu alléger considérablement les data-sets et conserver les informations importantes, sans fixer aucun seuil. Et que notre méthode ne juge aucune règle par son score, la privilégiant a d'autres.

Chapitre 7 : Conclusion.

Chapitre 7
Conclusion.

Les règles d'association ont largement démontré leurs intérêts d'application dans différents domaines tels que la classification de documents textuels ou la prédiction de comportements, etc. En effet, leur forme d'antécédents conduisant à des conséquents est facile à comprendre par l'homme et à traiter par les machines. Plusieurs recherches ont introduit des mécanismes et des notions permettant de tirer le maximum d'informations en utilisant les règles d'association.

Dans ce travail, nous avons présenté les mesures généralement utilisées pour filtrer les règles d'association comme le support, la confiance, et le lift, ainsi que deux algorithmes d'extraction des règles d'association à savoir, l'algorithme Apriori et FP-Growgth. À partir d'exemples d'application, nous avons vu que depuis un petit nombre de transactions, plusieurs règles pouvaient être générées. Cette problématique a fait l'œuvre de plusieurs recherches, visant à réduire le nombre de règles d'association.

Dans ce mémoire, nous avons traité cette problématique, tout en cherchant à conserver les informations importantes. Nous avons utilisé une méthode de minimisation des fonctions booléenne, à savoir la méthode de Quine-McClusckey.

Pour utiliser la méthode de Quine-McClusckey, il nous a fallu déterminer de nouvelles annotations, permettant de garder la lisibilité des règles d'association ainsi que permettre la minimisation des règles. Si la méthode de Quine-McCluskey a comme entrée une fonction booléenne, donc nos règles devaient subir une transformation vers cette forme. Nous nous sommes inspirés des règles d'association négatives, où la négation « ¬ » d'une variable signifie son absence. De ce fait, il nous restait plus qu'à trouver comment mettre en binaire les règles d'association. Nous avons annoté l'absence par 0 et la présence par 1. Pour la minimisation nous avons considéré le tiret « - » pour dire que l'item est ignoré, car il n'a pas d'effet direct sur le résultat.

Nous avons proposé deux approches de réduction, à savoir la réduction par mêmes antécédents et par mêmes conséquents.

En introduisant ce mémoire, nous avons précisé que l'utilisation de seuils tels que le minsup, minconf, minlift ou autre, pouvait supprimer des règles d'association qui bien que leurs

scores soient faibles, elles détiennent des informations importantes. Dont la nécessiter de trouver une approche qui ne dépend d'aucun seuil. Nous avons émis deux hypothèses : la première est que la réduction du nombre de règles d'association par notre méthode serait bénéfique en termes d'allègement des banques de données et la conservation des informations importantes. La deuxième : est que notre méthode pourrait faire perdre des informations importantes. Nous avons donc cherché à étudier notre méthode dans un environnement très sensible aux changements pouvant être apportés sur sa base de connaissances à savoir les systèmes de classification.

Ce contexte rend les règles intéressantes, car ces dernières sont porteuses d'informations permettant une classification correcte. Cette spécificité était très importante pour notre travail, puisque la réduction de règles porteuse d'informations utilisées dans la classification est critique. En effet, une réduction peut faire perdre des informations importantes conduisant ainsi à une baisse considérable de la précision de la classification. C'est pourquoi nous avons veillé à étudier notre approche selon le taux de réduction des règles et la précision de la classification après la réduction.

Les expérimentations ont montré que la méthode proposée a pu réduire le nombre de règles d'association de manière considérable. L'étude de la perte d'information nous a permis de constater que les taux de précision sont restés très proches.

Nous avons aussi étudié les taux des règles perdues et conservées après la réduction. Nous avons pu déduire que la méthode de Quine-McCluskey a tendance de conserver les informations à précisions élevés, mais de garder des informations de faibles scores aussi, ce qui indique qu'avoir un score élevé ne donne aucun privilège aux règles.

La méthode de Quine-McClusckey s'est montrée très efficace, et absolument simple à mettre en œuvre. Sans lui fixer aucun paramètre, cette dernière a pu réduire le nombre de règles et conserver les règles importantes. Son fonctionnement dans un environnement binaire la rend applicable à tout type de donnée. L'utilisateur n'a besoin que de trouver comment mettre les informations sous la forme booléenne et laisser la méthode faire la minimisation.

Comme perspective, nous proposons d'étendre notre recherche sur la classification des programmes malveillants et de déterminer leurs familles (Virus, botnet, etc.). En effet, dans ce mémoire, nous avons classifié les programmes seulement par leur nature (Programme malveillant ou bénin). Nous avons pu voir dans un précédent travail que les appels d'API pouvaient aider à

déterminer si un programme est malveillant ou non. Ces programmes sont exécutés dans des environnements confinés, afin de prévenir tout effet pouvant nuire à notre machine. Nous tenterons une nouvelle approche visant à extraire les différents enchainements de tâches effectuées par les programmes sous forme de règles d'association. Ces enchainements seront très importants, car ils pourront déterminer la nature d'un programme, mais aussi sa famille. Puisque nous utiliserons les règles d'association, nous nous attendons à avoir un nombre considérable de règles. Nous croyons que l'utilisation de la méthode de Quine-McClusckey pourrait là encore résoudre le problème du nombre de règles et conservera les informations importantes, améliorant ainsi la performance du système.

Bibliographie:

[1]. Bokhabrine, Ayoub, Ismaïl Biskri, and Nadia Ghazzali. 2020. "Textual Clustering: Towards a More Efficient Descriptors of Texts." In *International Conference on Computational Collective Intelligence*, 801-10. Springer.

[2]. Turčínek, Pavel, and Jana Turčínková. 2015. 'Exploring consumer behavior: Use of association rules', *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 63: 1031-42.

[3]. Zimmermann, Albrecht, and Luc De Raedt. 2004. "Corclass: Correlated association rule mining for classification." In *International Conference on Discovery Science*, 60-72. Springer.

[4]. Zaki, Mohammed J. 2000. "Generating non-redundant association rules." In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 34-43.

[5]. Lin, Wen-Yang, Ming-Cheng Tseng, and Ja-Hwung Su. 2002. "A confidence-lift support specification for interesting associations mining." In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 148-58. Springer.

[6]. Zeng, Yi, Shiqun Yin, Jiangyue Liu, and Miao Zhang. 2015. 'Research of improved FP-Growth algorithm in association rules mining', *Scientific Programming*, 2015.

[7]. Agrawal, Rakesh, and Ramakrishnan Srikant. 1994. "Fast algorithms for mining association rules." In *Proc. 20th int. conf. very large data bases, VLDB*, 487-99. Citeseer.

[8]. Han, Jiawei, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques* (Elsevier).

[9]. Kosters, Walter A, Wim Pijls, and Viara Popova. 2003. "Complexity analysis of depth first and fp-growth implementations of apriori." In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, 284-92. Springer.

[10]. Ashrafi, Mafruz Zaman, David Taniar, and Kate Smith. 2007. 'Redundant association rules reduction techniques', *International Journal of Business Intelligence and Data Mining*, 2: 29-63.

[11]. Jeyapal, Akilandeswari, and Jothi Ganesan. 2016. 'Elimination of Redundant Association Rules—An Efficient Linear Approach.' in, *Computational Intelligence, Cyber Security and Computational Models* (Springer).

[12]. Vera, Julio César Díaz, Guillermo Manuel Negrín Ortiz, Carlos Molina, and María Amparo Vila. 2020. 'Knowledge redundancy approach to reduce size in association rules', *Informatica*, 44.

[13]. Sarma, Pankaj Kumar Deva, and Anjana Kakati Mahanta. 2012. 'Reduction of number of association rules with inter itemset distance in transaction databases', *International Journal of Database Management Systems*, 4: 61.

[14]. Chen, Qixin. 2012. 'Application on mining association rules with attributes reduction based on fractal dimension.' in, *Information Engineering and Applications* (Springer).

[15]. Ma, Yu-liang, and Wen-jun Yan. 2006. 'Value reduction algorithm in rough sets based on association rules support', *Journal of Zhejiang University-Science A*, 7: 219-22.

[16]. Dong, Xiangjun, Feng Hao, Long Zhao, and Tiantian Xu. 2020. 'An efficient method for pruning redundant negative and positive association rules', *Neurocomputing*, 393: 245-58.

[17]. Bastide, Yves, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. 2000. "Mining minimal non-redundant association rules using frequent closed itemsets." In *International Conference on Computational Logic*, 972-86. Springer.

[18]. AL-Zawaidah, Farah Hanna, Yosef Hasan Jbara, and AL Marwan. 2011. 'An improved algorithm for mining association rules in large databases', *World of Computer science and information technology journal*, 1: 311-16.

[19]. Han, Jiawei, and Yongjian Fu. 1999. 'Mining multiple-level association rules in large databases', *IEEE Transactions on knowledge and data engineering*, 11: 798-805.

[20]. Abdullah, Zailani, Tutut Herawan, Noraziah Ahmad, and Mustafa Mat Deris. 2011. 'Mining significant association rules from educational data using critical relative support approach', *Procedia-Social and Behavioral Sciences*, 28: 97-101.
[21]. GHAREB, ABDULLAH S, ABDUL RAZAK HAMDAN, AZURALIZA ABU BAKAR, and MOHD RIDZWAN YAAKUB. 2015. 'HYBRID STATISTICAL RULE-BASED CLASSIFIER FOR ARABIC TEXT MINING', *Journal of Theoretical & Applied Information Technology*, 71.
[22]. Sharma, Meera, Abhishek Tandon, Madhu Kumari, and VB Singh. 2017. 'Reduction of redundant rules in association rule mining-based bug assignment', *International Journal of Reliability, Quality and Safety Engineering*, 24: 1740005.
[23]. Alwedyan, Jaber, Wa'el Musa Hadi, Ma'an Salam, and Hussein Y Mansour. 2011. "Categorize arabic data sets using multi-class classification based on association rule approach." In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, 1-8.
[24]. El-Halees, Alaa M. 2015. 'Arabic text classification using maximum entropy', *IUG Journal of Natural Studies*, 15.
[25]. Haralambous, Yannis, Yassir Elidrissi, and Philippe Lenca. 2014. 'Arabic language text classification using dependency syntax-based feature selection', *arXiv preprint arXiv:1410.4863*.
[26]. Bogorny, Vania, Bart Kuijpers, and Luis Otávio Alvares. 2008. 'Reducing uninteresting spatial association rules in geographic databases using background knowledge: a summary of results', *International Journal of Geographical Information Science*, 22: 361-86.
[27]. Majumder, Alak, Barnali Chowdhury, Abir J Mondai, and Kunj Jain. 2015. "Investigation on Quine McCluskey method: A decimal manipulation based novel approach for the minimization of Boolean function." In *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, 18-22. IEEE.
[28]. Duşa, Adrian. 2010. 'A mathematical approach to the boolean minimization problem', *Quality & Quantity*, 44: 99-113.
[29]. Mill, John Stuart. 1869. *A System of Logic, Ratiocinative and Inductive: Being a Connected View of the Princilples of Evidence and the Methods of Scientific Investigation* (Harper and brothers).
[30]. Coverdill, James E, and William Finlay. 1995. 'Understanding Mills via Mill-type methods: An application of qualitative comparative analysis to a study of labor management in southern textile manufacturing', *Qualitative sociology*, 18: 457-78.
[31]. Staneva, Liliya Anestieva. 2019. "Minimising using the Method of Quine-McCluskey with Generalised Nets." In *2019 29th Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*, 1-3. IEEE.
[32]. El Bazzi, Mohamed Salim, Taher Zaki, Driss Mammass, and Abdelatif Ennaji. 2016. 'Automatic Indexing of Arabic Texts: State of the Art', *Electronic Journal of Information Technology*.
[33]. Ghareb, Abdullah S., Abdul Razak Hamdan, and Azuraliza Abu Bakar. 2012. "Text associative classification approach for mining Arabic data set." In, 114-20 %@ 1467327182. IEEE.