TP Data Mining

Compte rendu des algorithmes Apriori et FP-Growth

MEDJKOUNE Sofiane : 201400007781

HADJERES Yasmine : 201400007136

I. Introduction

Dans l'exploration de données, la recherche de modèles fréquents dans les grandes bases de données est très importante et a été étudiée à grande échelle au cours des dernières années. Malheureusement, cette tâche est coûteuse en termes de calcul, en particulier lorsqu'il existe un grand nombre de modèles.

Dans ce document, nous introduisons les concepts de base de modèles fréquents et d'associations et nous étendons notre discussion aux méthodes avancées d'exploration de nos jeux de données qui permettent d'accélérer le processus d'extraction de ces modèles fréquents.

Les deux algorithmes que nous verrons sont :

- L'algorithme Apriori.
- L'algorithme FP-Growth

II. Apriori

L'algorithme Apriori est donné par R. Agrawal et R. Srikant en 1994 pour rechercher des items fréquents dans un jeu de données pour une règle d'association booléenne. Le nom d'algorithme est Apriori, car il utilise des connaissances préalables sur les propriétés fréquentes des ensembles d'éléments. Nous appliquons une approche itérative ou une recherche par niveau dans laquelle les ensembles d'éléments k-fréquents sont utilisés pour trouver k + 1 éléments.
Pour améliorer l'efficacité de la génération au niveau des jeux d'éléments fréquents, une propriété importante est utilisée, appelée propriété Apriori, qui permet de réduire l'espace de recherche.
Propriété Apriori:
Tous les sous-ensembles non vides d'items fréquents doivent être fréquents. Le concept clé de l'algorithme Apriori est son anti-monotonicité de mesure de support.

Nous illustrons le fonctionnement de l'algorithme Apriori à travers un exemple et pour cela nous considérons le jeu de données suivants :

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Pour cet exemple nous prenons pour support minimum la valeur 2.

Les étapes de déroulement de l'algorithme sur le dataset sont les suivantes :

Étape 1 : Création d'une table contenant le support de chaque item. dans le cas où un item à un support inférieur à notre support minimum, celui-ci sera supprimé de notre table. Pour notre exemple tous les supports sont supérieurs au support minimum donc pas de suppression.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Étape 2 : Nous générons de nouveaux candidats en associant les items deux à deux comme suit :

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Nous vérifions par la suite les itemset dont le support est inférieur au support minimum, si tel est le cas nous supprimons l'itemset comme suit :

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Étape 3 : Nous testons à présent les items trois par trois en supprimant les combinaisons dont le support est inférieur au support minimum comme suit :

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Nous nous arrétons ici car au delà de cette étape les combinaisons d'items ne présentent pas un support suffisant.

Le résultat final de l'algorithme est l'union des différentes tables obtenus à chaque étape de l'algorithme.

III. FP-Growth

L'algorithme FP-Growth, proposé par Han dans "Data Mining, Concept and Technologies", est une méthode efficace et évolutive pour extraire l'ensemble complet d'items fréquents par croissance de fragments d'item, et ce en utilisant une structure d'arborescence étendue pour stocker des informations cruciales sur les items fréquents. Cet arbre est nommé FP-tree. Dans son étude, Han a prouvé que sa méthode surpasse d'autres méthodes populaires d'extraction de modèles fréquents, par exemple. l'algorithme Apriori que nous avons étudié plus haut.

L'algorithme FP-Growth constitue un moyen alternatif de rechercher des ensembles d'éléments fréquents sans utiliser de générations de candidats, améliorant ainsi les performances. Pour autant, il utilise la stratégie "Diviser pour régner" . à l'aide du FP-Tree, cette méthode permet de conserver les informations d'association d'éléments.

En termes simples, cet algorithme fonctionne comme suit:

- Il compresse la base de données d'entrée en créant une instance d'arborescence FP pour représenter des éléments fréquents.

- La base de données compressée est divisée en un ensemble de bases de données conditionnelles, chacune associée à un modèle fréquent.
- Enfin, chacune de ces bases de données est exploitée séparément. En utilisant cette stratégie, FP-Growth réduit les coûts de recherche en recherchant des modèles plus court de manière récursive, puis en les concaténant dans les modèles à fréquence longue et fréquente, offrant une bonne sélectivité.

Pour Illustrer le fonctionnement de cet algorithme nous prenons l'exemple d'un itemset que nous dérouleront à l'aide de FP-Growth pour le jeu de données suivants :

.

TID	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B. E
4	B, A, D
5	D
6	D,B
7	A,D,E
8	B,C

Étape 1 : On définit un support minimum, donnons à notre support la valeur 3.

Étape 2 : On donne la fréquence de chaque occurrence de chaque item et on les ordonne en ordre décroissant des fréquences :

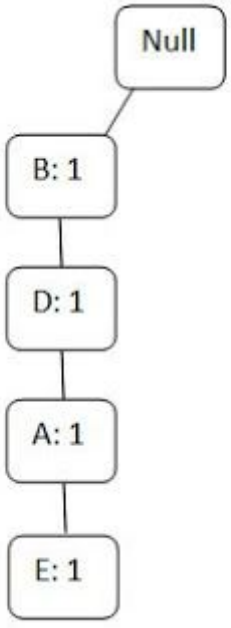
Item	Frequency	
A	5	3
B	6	1
C	3	5
D	6	2
E	4	4

Étape 3 : Réordonner la liste des Items selon l'ordre que nous avons vu dans le tableau précédent :

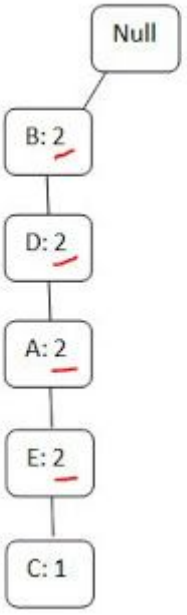
TID	Items	Ordered Items
1	E, A, D, B	B,D,A,E
2	D, A, C, E, B	B,D,A,E,C
3	C, A, B. E	B,A,E,C
4	B, A, D	B,D,A
5	D	D
6	D,B	B,D
7	A,D,E	D,A,E
8	B,C	B,C

Étape 4 : Nous passons à présent à la construction de notre FP-Tree, pour cette étape nous présenterons la construction de l'étape transaction par transaction :

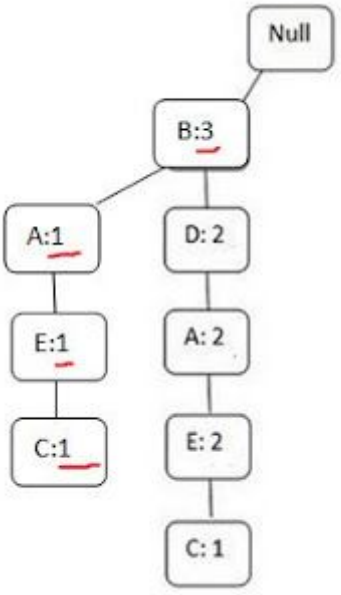
Ligne 1 : L'arbre commence toujours avec un noeud null auquel nous chaînons les items de la première transaction :



Ligne 2 : Nous passons à la seconde transaction. Si la suite d'item par laquelle commence notre transaction est présente dans l'arbre nous incrémentons l'indice associé à chaque noeud de l'arbre sinon nous créons une nouvelle branche comme suit :

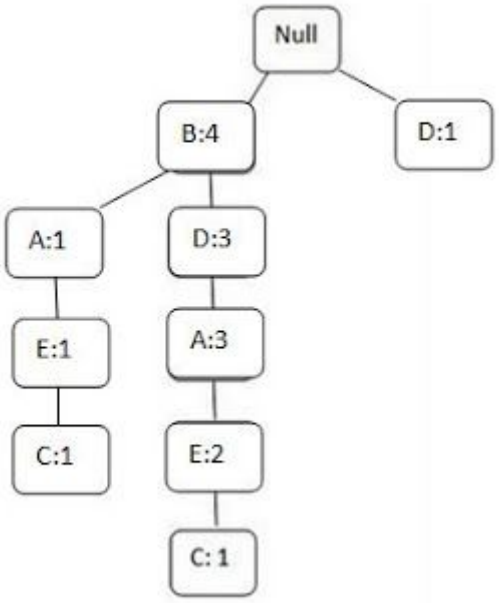


Ligne 3 : La troisième transaction présente un cas où nous devons créer un nouveau noeud de l'arbre :



Ligne 4 : les items de la quatrième transactions sont déjà présents dans l'arbre, nous effectuons l'incrémentation seulement.

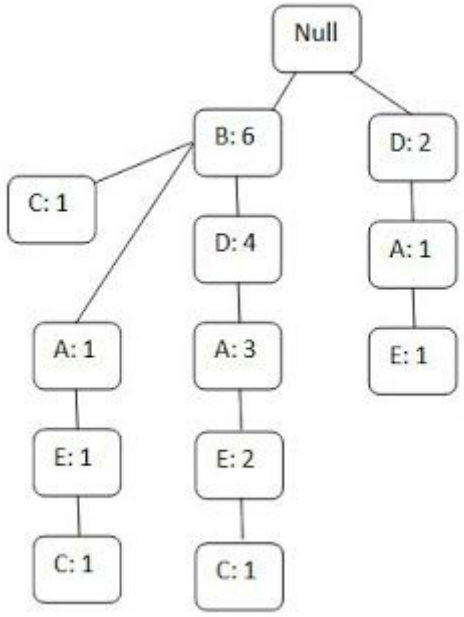
Ligne 5 : Pour cette cinquième ligne, nous connections l'item D directement au noeud racine.



Ligne 6 : B et D existent déjà, il s'agira donc d'une incrémentation.

Ligne 7 : Nous chaînons de nouveaux noeuds Au noeud D:1.

Ligne 8 : Nous chaînons l'item C directement à l'Item B :



Mettant ainsi fin à la construction du FP-Tree. Cet algorithme comme nous l'avons vu a permis de construire l'arbre et donc de trouver le modèle le plus fréquent et ce en une seule itération.