

FP-Growth

Ilyes Benarroum

Nadir Habib

Groupe 1

September 27

1 Quel est FP-growth?

FP-growth est une version améliorée de l'algorithme Apriori qui est largement utilisé pour d'exploration de données(génération des règles d'association). Il est utilisé comme processus analytique qui trouve des modèles ou des associations fréquents à partir d'ensembles de données. Les données sur les transactions des épiceries peuvent avoir un schéma fréquent selon lequel les gens achètent généralement des chips et de l'eau ensemble. L'algorithme Apriori produit des modèles fréquents en générant des ensembles d'éléments et en découvrant l'ensemble d'éléments le plus fréquent au-dessus d'un seuil de "nombre minimal de support". Cela réduit considérablement la taille de l'ensemble d'éléments dans la base de données par un principe simple:

"Si un ensemble d'éléments est fréquent, tous ses sous-ensembles doivent également être fréquents."

Cependant, l'algorithme Apriori présente un déficit majeur. L'utilisation d'Apriori nécessitait plusieurs analyses de la base de données pour vérifier le nombre de supports de chaque élément et ensembles d'éléments. Lorsque la base de données est énorme, cela coûtera une quantité importante d'E / S disque et de puissance de calcul. Par conséquent, l'algorithme FP-Growth est créé pour combler ce déficit. Il scanne seulement la base de données deux fois et utilise une structure arborescente (FP-tree) pour stocker toutes les informations. La racine représente nul, chaque noeud représente un élément. tandis que l'association des noeuds est les jeux d'éléments avec l'ordre maintenu lors de la formation de l'arbre. L'arbre FP est concis et est utilisé pour générer directement de grands ensembles d'éléments. Une fois qu'un FP-tree a été construit, il utilise une approche récursive de division et de conquête pour extraire les ensembles d'éléments fréquents.

2 Pseudocode et explication de l'arbre FP

- Étape 1: Déduire les articles fréquents commandés. Pour les éléments de même fréquence, l'ordre est donné par ordre alphabétique.
- Étape 2: Construire l'arbre FP à partir des données ci-dessus
- Étape 3: À partir de l'arbre FP ci-dessus, construire l'arbre conditionnel FP pour chaque élément (ou ensemble d'éléments).
- Étape 4: Déterminer les modèles fréquents.

Voici un exemple qui illustre la création d'un arbre FP. Trouvez tous les ensembles d'éléments fréquents avec un support ≥ 2 . Tout d'abord, recherchez tous les éléments avec un nombre de support ≥ 2 .

A Simple Example: Step 1

The diagram illustrates the first step in creating an FP-tree. It shows three tables: a transaction table, a frequency table, and an ordered frequency table. A red arrow points from the transaction table to the frequency table, indicating the process of calculating item frequencies. A yellow box labeled "Threshold = 2" highlights the support threshold used to filter items in the frequency table. Another red arrow points from the frequency table to the ordered frequency table, indicating the process of ordering items by frequency.

TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

Item	Frequency
A	4
B	3
C	3
D	3
E	2

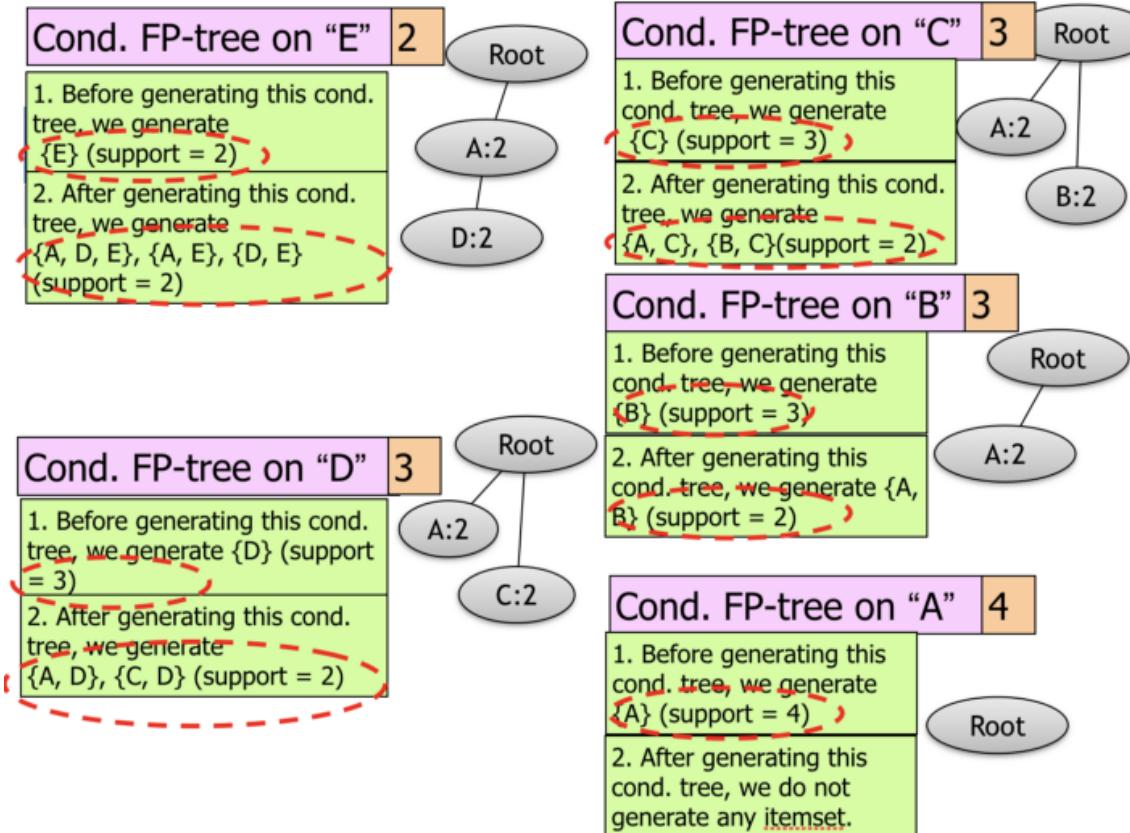
Threshold = 2

Item	Ordered Frequency
A	4
B	3
C	3
D	3
E	2

Avant de construire l'arbre FP, réorganisez la transaction en fonction de la fréquence de leurs éléments. Lorsque vous commencez à construire l'arbre FP, nous devons créer une table d'en-tête qui enregistre l'emplacement de tous les nœuds d'élément avec une liste de liens. Chaque fois qu'un nouveau nœud est ajouté à l'arborescence, il doit être lié au dernier nœud avec le même élément. La table d'en-tête est essentielle pour créer une arborescence FP conditionnelle dans les étapes ultérieures.

Maintenant, nous avons scanné la base de données deux fois et construit l'arbre FP. Toutes les informations de transaction sont contenues dans l'arborescence. La seule chose qui reste à faire est de construire récursivement l'arbre FP conditionnel pour trouver tous les éléments fréquents avec un nombre de support supérieur à 2. Utilisez le tableau d'en-tête que nous avons créé à la dernière étape et commencez à partir de l'élément avec le nombre le moins la plupart (E, D, C, B, A). .

Enfin, nous pouvons facilement générer tous les ensembles fréquents à partir de ces FP-arbres conditionnels E, A, D, E, A, E, D, E, D, A, D, C, D, C, A, C, B, A, B,



A.

3 Implémentation Python

Voici un exemple de code pour créer un FP-tree à partir de zéro et trouver tous les jeux d'éléments de fréquence dans Python 3.

En conclusion, FP-tree reste la méthode la plus efficace et la plus évolutive pour extraire l'ensemble complet des modèles fréquents dans un ensemble de données. La plupart des langages de programmation, y compris Python, R et même Pyspark, ont des bibliothèques bien prises en charge pour générer FP-tree.

For the Code Source

This is my link: [FP-Growth](#).