Haute Ecole Spécialisée
de Suisse occidentale

BDA

Rapport : Recommending Music and the Audioscrobbler Data Set

Rapport
Version 1.0, 29.05.2020

Samuel Torche
Ayrton Dumas
Marco Mattei
Groupe B

MASTER OF SCIENCE
IN ENGINEERING

Table des matières

1. Introduction

Dans le cadre du cours de Big Data Analytics, nous allons réalisé un mini-projet afin de mettre en pratique les principes vus durant le cours. Notre choix s'est porté sur le dataset Audioscrobbler et sur un mini-projet centré sur la recommandation de musiques.

2. Description du dataset

Le dataset utilisé est appelé AudioScrobbler. Ce dataset contient les artistes musicaux écoutés par plus de 140'000 utilisateurs. Il s'agit d'une archive compressée contenant plusieurs fichiers texte. L'archive fait 135 MB compressée et 500 MB une fois décompressée.

User artist data

Le fichier user_artist_data.txt contient les utilisateurs et les artistes qu'ils écoutent.Chaque ligne correspond contient l'id de l'utilisateur, l'id de l'artiste et le nombre de fois qu'il a écouté une chanson de cet artiste, donc pour chaque artiste écouté par un utilisateur, il y aura une nouvelle ligne. Il y a 140'000 utilisateurs qui ont écouté en tout 24'296'858 artistes. Ce fichier contient donc 3 colonnes: userid, artistid, playcount

Artist data

Le fichier artist_data.txt contient les mappings artistes id -> nom de l'artiste. Ce fichier contient donc 2 colonnes: artistid, artist_name. Il y a 1'848'707 artistes dans ce fichier.

Artist alias

Le fichier artist_alias.txt contient les noms d'artiste qui sont souvent mal orthographiés ou qui sont sujets à variante (Depeche Mode mal orthographié en Depeche Mood par exemple). Ce fichier contient donc 2 colonnes: badid, goodid. Il y a 193'027 alias.

3. Questions d'analyse

1. Peut-on obtenir des résultats intéressants en utilisant des technique de Market Basket Analysis telles que les règles d'association ?
2. Peut-on déduire les genres des musiques à l'aide d'un clustering des utilisateurs et leurs écoutes ?
3. Est-il possible de recommander non pas des artistes mais d'autres utilisateurs qui partagent les même goût avec ALS ?

4. Description des features / pre-processing afin d'extraires des features additionnelles

Market Basket Analysis preprocessing

Afin d'appliquer des techniques de Market Basket Analysis, les données doivent être transformées. En effet, actuellement les données se trouvent sous cette forme:

user	artist	count
1000002	1	55
1000002	1000006	33
1000002	1000007	8
1000002	1000009	144
1000002	1000010	314

On peut voir qu'un utilisateur est présent sur plusieurs lignes, une ligne pour chaque artiste qu'il a écouté. Afin de pouvoir exécuter les algorithmes de Market Basket Analysis (voir chapitre 5. Algorithmes appliqués pour plus d'informations), les données doivent être transformé en format transactionnel, c'est à dire ainsi:

user	items
1000002	[1, 18, 28, 30, 5...]
1000019	[1000010, 1000028...]

On aperçoit à présent qu'un seul utilisateur est présent par ligne, et que tous les artistes qu'il a écouté ont été regroupé sur une seule ligne, dans une colonne nommé "items" de type *Array*. Pour faire l'analogie avec un exemple plus parlant, on peut dire, qu'avant, notre dataset contenait une ligne pour chaque item que notre client avait acheté, donc une ligne pour clientA + pain, une deuxième ligne pour clientA + beurre. Maintenant ces lignes ont été regroupé par client donc clientA + pain + beurre.
Pour obtenir ce résultat, il a fallu:

- Grouper le dataframe par user
- Pivoter le champs artiste en tant que colonne
- Indiquer pour chaque artiste s'il fait partie de la playlist du user, stocker un null dans la colonne sinon
- Grouper toutes ces colonnes d'artistes un une seule colonne "items" de type *Array*

5. Algorithmes appliqués

Market Basket Analysis

L’algorithme utilisé pour miner les frequent items sets est FPGrowth. Il existe deux implémentations de cet algorithme dans la librairie Scala: org.apache.spark.ml.fpm.FPGrowth et org.apache.spark.mllib.fpm.FPGrowth. La première alternative a été utilisé car il était plus aisé de performer les différentes étapes de preprocessing décrites au chapitre 4.

6. Optimisations réalisées

7. Tests et évaluations

8. Résultats obtenus

9. Améliorations futures

10. Conclusion