UPMC 2014-2015
UFR 919 M2 BDLE

TME 1 : Utilisation de Scala sous Spark

Prérequis :
Consulter la rubrique Informations pratiques/Notes sur les TD-TME de la page de l'UE pour les instructions de démarrage.

Exerice 1 : Prise en main à travers le programme "Word Count"
Dans cet exercice, nous allons charger un fichier texte dans le but de calculer les occurences de mots. Ce fichier contient des statistiques de visite de pages wikipedia données sous forme de n-uplets dont voici un exemple

En.d updates 3 24145

Ce n-uplet indique que dans une page écrite en anglais (symbole 'En'), figure le mot 'updates' qui a été cliqué 3 fois.
L'accès au fichier se fait au moyen de la méthode *textFile()* invoquée à partir de la variable *context* comme suit :

val data = sc.textFile("/Vrac/bdle/data/wordcount-small.txt")

Ici, la variable *data* est de type RDD (cf. doc en ligne)
L'invocation de la méthode *take(n)* sur cette variable affiche les *n* premiers éléments sur une seule ligne. Pour un affichage sur plusieurs lignes (tel un head -n sous linux), utiliser plutôt *take(n)*.foreach(println) qui itère sur les éléments de la variable d'où elle est invoquée et affiche chaque élément sur une ligne séparément.
Noter également que la méthode *count* retourne la cardinalité (nombre d'éléments) de la variable d'où elle est invoquée.

1- Que contient la variable *data*?

2- Dans un premier temps, structurer le contenu de *data* de sorte à obtenir un tableau de tableaux de chaines de caracteres. Ce dernier devra être stocké dans une nouvelle variable *liste*.
Afficher ensuite les 100 premiers éléments correspondant à la 3e colonne de *data.*

3- Transformer le contenu de *liste* en une liste de paires ('mot', nb) où mot est de type String et correspond à la premiere case d'un élément de *liste* et nb est un entier qui correspond à sa troisieme case.

4- Grouper les paires par ‘mot’ et additionner leur nombre nb.

5- Reprendre les questions 3 et 4 en calculant ‘mot’ différemment : désormais, ‘mot’ doit correspondre au préfixe du premier élément de chaque élément de *liste.*

Exercice 2 : Jointure naturelle
Dans cet exercice nous allons nous intéresser à la formulation de jointures.
Comme dans les exercices précédents, nous commençons par formatter les données afin de pouvoir les traiter.
Pour cet exercice, les données sont celles du dataset *movielens* obtenu à partir de http://grouplens.org/datasets/movielens/. Nous nous intéressons au fichier ratings.dat contenant les notes des utilisateurs et le fichier users.dat contenant des informations sur ces derniers.
Pour information :
- chaque ligne de ratings.dat est de la forme UserID::MovieID::Rating::Timestamp
- chaque ligne de users.dat est de la forme UserID::Gender::Age::Occupation::Zip-code

1- Charger les fichiers ratings.dat et users.dat dans les variables f1 et f2 respectivement en verifiant qu’ils ont bien été chargés. Vérifier le type de ces variables.

2- Transformer chaque ligne des structures de f1 et f2 en tableau de chaines nommés c1 et c2 respectivement et procéder aux vérifications habituelles (à l’aide de count et take())

3- Nous voulons compter le nombre de notes réalisées par chaque utilisateur. En faisant l’hypothese que ratings.dat soit stocké dans une table RAtings, la requete equivalente en SQL est select distinct(UserID), count(*) from Ratings.
Comment effectuer ce traitement sous Scala?

4- On veut connaître les les 3 utilisateurs qui ont noté le plus de films.
Indice : la méthode takeOrdered(n) retourne les n premiers elements du RDD depuis lequel elle est invoquee suivant l’ordre descendant de leur clé

Exercice 3 : Jointure avec condition
Reprendre le schéma des relations ratings.dat et users.dat et formuler les instructions en MR permettant d’effectuer des jointures avec conditions. Pour rapper, les opéateurs considérés sont $<$, $\leq$, $>$ et $\geq$.