TP 2 sur Apache Spark

1. Objectifs

Le TP consiste à regrouper des documents textuels tels que les documents qui partagent le même thème se retrouvent dans le même groupe, et les documents qui portent sur des sujets très différents se trouvent dans des groupes différents.

2. Mise en place de l'environnement de travail

1. Créer un notebook sur Google Colab
2. Installer Spark comme vu en cours
3. Définir la valeur de la variable d'environnement `PYSPARK_SUBMIT_ARGS` à `'--packages org.apache.spark:spark-avro_2.11:2.4.5 pyspark-shell'`. Ceci permettra d'utiliser Avro. (optionnelle)
4. Créer un objet `SparkContext`
5. Créer un objet de type `SparkSession`. Cet objet servira pour utiliser Spark SQL qui offre des fonctionnalités de plus haut niveau que les RDDs.

3. Données

1. Télécharger des données textuelles à l'adresse suivante :

   http://qwone.com/~jason/20Newsgroups/20news-19997.tar.gz

2. Décompresser les données
3. Charger les données dans deux variables de type RDD tel que :
   a. voir https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.SparkContext.wholeTextFiles.html
   b. Les documents de la catégorie "alt.atheism" soient dans un RDD et ceux de "rec.sport.baseball" soient dans un autre RDD
   c. Chaque RDD représente une liste de documents (un élément du RDD = un document)
4. Séparer le corps du message de l'entête. (séparation sur "\n\n" ?)
5. Extraire quelques champs de l'entête, par exemple l'organisation et la catégorie (champ "Newsgroups").

6. Fusionner les deux RDD (union)
7. Transformer le nouveau RDD obtenu pour que chaque élément soit de type pyspark.sql.Row
8. Créer un objet de type DataFrame à partir du RDD précédent
9. Sauvegarder la DataFrame au format Avro
10. Sauvegarder la DataFrame au format Parquet

4. Analyse descriptive

Faire une analyse descriptive de la DataFrame obtenue à l'étape précédente (API Spark SQL) :

1. Vérifier qu'on a bien deux catégories différentes de documents
2. Donner le nombre d'organisations différentes
3. Suivant les champs extraits à la partie 3, donner d'autres statistiques descriptives (nombre, moyenne, etc)

5. Transformation du texte

1. Consulter la page (API MlLib) :

   https://spark.apache.org/docs/latest/ml-features.html#feature-extractors

2. Découper les documents en listes de mots à l'aide de Tokenizer
3. Créer une représentation vectorielle des documents à l'aide de HashingTF

6. Grouper les documents qui ont des représentations vectorielles proches

1. Consulter la page :

   https://spark.apache.org/docs/latest/ml-clustering.html

2. Utiliser l'algorithme KMeans avec un nombre du cluster égal à 2 (pour essayer de retrouver les 2 catégories qu'on a)

7. Pour aller plus loin (optionnel)

1. Pondérer les mots avec la formule Tf-Idf (avant KMeans)
2. Normaliser les vecteurs représentant les documents (avant KMeans).