**MS BGD: Spark TP 1 (2018-2019)**

**Spark-shell et word count**

**1. Setup (MAC et Linux): Installation de Spark**

Spark utilise des machines virtuelles java, il faut donc commencer par l’installer, puis installer Spark:

**=>** Suivre la section **TP 1** dans le [Setup](https://docs.google.com/document/d/1lGV2sEq57Jao1cFPzFNwcgWlb-4XuMY2cHlzOyVmu4U/edit?usp=sharing).

**2. Début du TP**

Pour ce premier TP, on utilisera uniquement le spark-shell, qui permet d'exécuter des commandes spark de façon interactive (à la iPython).

**=> Lancez un shell spark dans un terminal puis l’interface utilisateur dans votre navigateur.**

Le shell spark dispose de l’autocompletion (en appuyant sur la touche “Tab”) qui est utile pour voir quelles fonctions sont applicables sur l’objet que l’on manipule.

Exemple dans le spark-shell:

Tapez “sc.” puis la touche “Tab”

Dans la liste des fonctions disponibles sur l’objet “sc” (pour sparkContext), il y a en particulier “textFile” dont on va se servir dans la partie 1.

Utilisez la documentation détaillée de l’API scala :

<http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.package>

Retrouvez les objets **sparkContext** et **textFile** (Dans le package org.apache.spark.sql.sparkSession) dans la documentation pour voir à quoi servent ces fonctions et comment s’en servir.

Vous pouvez également vous reporter au document [spark\_notes](https://docs.google.com/document/d/1DFmqUELV8xnnB9lwVVz0F9KtRySf8y07-aASxLEpW8I/edit?usp=sharing), pour des ressources et quelques généralités sur Spark et sa documentation.

Utilisez Google, il y a beaucoup de ressources avec des extraits de code et des explications dans les forums, et en particulier sur StackOverflow.

1. Lire un fichier de données non structurées

Ici on veut travailler avec des **RDD**, qui sont la structure sous-jacente des dataframes et datasets. En principe vous utiliserez presque toujours des dataframes pour vos projets Spark, mais il est intéressant d’avoir vu les RDD parce que vous trouverez peut-être en entreprise des projets Spark sur d’anciennes versions qui n’utilisent pas les dataframes.

Le point d’entrée pour utiliser les RDD est le “sparkContext”, pour utiliser les datasets c’est la “sparkSession”. Le SparkContext reste accessible surtout pour des raisons de backward compatibility.

* 1. Charger le fichier README.md (se trouvant dans le dossier Spark qui a été décompressé) dans un RDD:

**val** rdd = sc.textFile("/Users/maxime/spark-2.2.0-bin-hadoop2.7/README.md")

* 1. Afficher dans le terminal les 5 premières lignes du fichier:

*println*("5 first rows of the RDD")

rdd.take(5).foreach(*println*)

* 1. Faire un word count sur le README.md du dossier spark-2.2.0-hadoop2.6, et afficher les résultats (mettre les résultats dans un dataFrame avec une colonne “word” et une colonne “count”)

**val** wordCount = rdd

.flatMap { line: String => line.split(" ") }

.map { word: String => (word, 1) }

.reduceByKey { (i: Int, j: Int) => i + j }

.toDF("word", "count")

* 1. Afficher les résultats du word count sous forme de table, on veut les mots les plus fréquents en haut, comme dans l’exemple suivant :

wordCount.orderBy($"count".desc).show()

**$ -> used to refer to a column object**

+----- + ----- +

| word|count|

+----- + ----- +

| | 68|

| the| 22|

|Spark| 15|

| to| 14|

| and| 11|

* 1. Un même mot peut être présent dans la table précédente avec des majuscules et minuscules (“The” et “the”). Modifier le dataframe précédent pour mettre les mots en lower case dans la colonne “word”.

**val** df\_lower = wordCount.withColumn("word\_lower", *lower*($"word"))

* 1. Sommer la valeur count pour un même mot mis en minuscule, et afficher les résultats sous forme de table (indication: utilisez groupBy).

**val** df\_grouped = df\_lower

.groupBy("word\_lower")

.agg(*sum*("count").as("new\_count"))

df\_grouped.orderBy($"new\_count".desc).show()

1. Équivalent en utilisant uniquement des dataframes:

**val df2 = spark**

**.read**

**.text("/Users/maxime/spark-2.2.0-bin-hadoop2.7/README.md") // read.text retourne un dataFrame avec une seule colonne, nommée "value"**

**.withColumn("words", *split*($"value", " "))**

**.select("words")**

**.withColumn("words", *explode*($"words"))**

**.withColumn("words", *lower*($"words"))**

**.groupBy("words")**

**.count**