# PLP : rapport

Antoine Lavault

7 janvier 2018

## Remarques préliminaires

Les implémentations proposées ont été testées avec Hadoop 2.8.3 sur une machine Ubuntu 17.04 : la machine virtuelle ne donnait pas satisfaction vu la configuration de la machine de test.

## 1 Résultats

### 1.1 Exploration de HDFS

#### 1.1.1 Affichage d'un CSV

L'implémentation effectuée ne prend pas en compte le header/en-tête du CSV, seules les valeurs des deux colonnes demandées sont affichées.

La commande

```
hadoop jar yht.jar cs.bigdata.Tutorial2.CompterLigneFile [input]
```

permet de lancer le jar joint.

### 1.2 Problèmes

#### 1.2.1 TF-IDF

Le code source consiste en 10 fichiers pour les 3 jobs et 1 driver principal.

Plus précisément,

— WordCount* : Correspond à la première étape

— PerDocWC* : comprendre "Per Document Word Count".

— PerWordDC* : comprendre "Per Word Document Count"

On obtient au final les 20 meilleurs résultats suivants. La forme d'une ligne est "[mot]@[nom du document] [tf-idf]".

Notons qu'il n'y a pas de traitement particulier dans la partie WordCount, en particulier des changements de casse ou la suppression de la ponctuation.

```
Buck@callwild 0,0054966672
dogs@callwild 0,0013305425
Thornton@callwild 0,0012869181
myself@defoe-robinson-103.txt 0,0012545960
Buck's@callwild 0,0009597355
Spitz@callwild 0,0009379234
Francois@callwild 0,0009161112
```

```
sled@callwild 0,0008288625
John@callwild 0,0008288625
Buck,@callwild 0,0007198017
dogs,@callwild 0,0006761773
Friday@defoe-robinson-103.txt 0,0006501088
shore,@defoe-robinson-103.txt 0,0005816763
-@defoe-robinson-103.txt 0,0005474601
Perrault@callwild 0,0005453043
However,@defoe-robinson-103.txt 0,0005360547
Hal@callwild 0,0005234921
God@defoe-robinson-103.txt 0,0005075411
me.@defoe-robinson-103.txt 0,0004961357
me;@defoe-robinson-103.txt 0,0004961357
```

Les résultats sont obtenus avec l'application de la commande shell suivante sur le dossier de sortie (dans le cas d'un fichier dans HDFS)

```
hadoop fs -cat output/part* | sort -g -k2 -r | head -n20
```

A titre de remarque, en mode "pseudo-distributed", le traitement complet dure 1 minute et 9 secondes.

### 1.2.2 PageRank

Avec 3 itérations, on obtient les utilisateurs ayant le plus grand PageRank ainsi que le dit PageRank.

```
18 330.89865
737 163.91165
790 153.64359
143 140.15887
1719 139.00761
136 131.68274
118 117.10892
4415 105.40317
1621 91.089554
1619 90.57585
```

On distinguera trois parties dans le traitement :
- Le "pré-traitement" (PageRank*) : préparer les données pour le traitement PageRank
- Le traitement (PageRankProcess*) : itérer ce dernier sur les données. L'utilisation de dossiers in/out alternés évitent une utilisation trop importante de l'espace disque.
- Le nettoyage (PageRankSort*) : renvoyer uniquement les informations pertinentes (ici le PageRank).

Pour obtenir le résultat final la commande suivante convient :

```
hadoop fs -cat output/part* | sort -g -k2 -r | head -n10
```

### 1.2.3 Notre Arbre de Paris

On considérera que le type d'un arbre est donné par son genre sans plus de précision du sujet donc en colonne 3 du CSV.

La hauteur se trouve en colonne 7.

Ainsi, le genre le plus commmun est *platanus* et tient aussi le record de hauteur parmi tous les genres avec $45m$.