# TP - Map-Reduce (Partie 2)

## Utilisation de mrjob

Mrjob est un package qui vous permet d'écrire des jobs en python (voir la doc à https://pythonhosted.org/mrjob/). Vous l'utiliserez directement dans une console sur votre terminal linux, en dehors de votre machine virtuelle.

Etudiez et exécutez le premier job donné ici dans la documentation de mrjob (https://pythonhosted.org/mrjob/guides/quickstart.html#writing-your-first-job) qui permet de faire le comptage de lettres, mots, phrases. Vous lancerez votre commande en utilisant l'option « -r local ». Pouvez vous le faire tourner avec plusieurs fichiers en entrée ?

### Statistiques de bigrams de mots

On veut réaliser un code calculant des statistiques sur le mot suivant. Le code doit afficher pour chaque paire de mots existant dans le texte la probabilité de voir le second mot de la paire succéder au premier mot de la paire.

Le code appelé avec la commande « `python mr_next_word_stat.py -r local texte_stats` » sur le fichier texte `texte_stats` suivant :

```
$ more texte_stats

a a b a c a d e a f
```

donne le résultat suivant :

```
["a", "a"] [5, 1, 20.0]

["a", "b"] [5, 1, 20.0]

["a", "c"] [5, 1, 20.0]

["a", "d"] [5, 1, 20.0]

["a", "f"] [5, 1, 20.0]

["b", "a"] [1, 1, 100.0]

["c", "a"] [1, 1, 100.0]

["d", "e"] [1, 1, 100.0]

["e", "a"] [1, 1, 100.0]
```

Complétez le code suivant pour qu'il réalise le traitement voulu :

mr_next_word_stat.py

```
from mrjob.job import MRJob
from mrjob.step import MRStep
import re
```

```
WORD_RE = re.compile(r"[\w']+")


class MRNextWordStats(MRJob):

    SORT_VALUES = True

    def steps(self):
        return [MRStep(mapper=self.m_find_words,
                       combiner=self.c_combine_counts,
                       reducer=self.r_sum_counts),
                MRStep(reducer=self.r_compute_stats)]

    def m_find_words(self, _, line):
        """Tokenize lines, and look for pairs of adjacent words.
        Yield (prev_word, word), 1 and (prev_word, '*'), 1 for each pair
        """
        prev_word = None

        for word in WORD_RE.findall(line):
            word = word.lower()

            if prev_word is not None:
                # total up the number of times prev_word appears
                # and the number of times next_word appears after it
                yield (prev_word, '*'), 1
                yield (prev_word, word), 1

            prev_word = word

    def c_combine_counts(self, key, counts):
        """Sum up all those 1s before passing data off to the reducer"""
        yield key, sum(counts)

    def r_sum_counts(self, key, counts):
        """Compute the number of times each pair of words appears, and the
        number of times the first word in a pair appears, and send it to
        a reducer that keys on the first word in the pair.
        """
        … A remplir …

     def r_compute_stats(self, prev_word, value):
        """For each pair of words, compute how many times it appears,
        how many times the first word appears in a pair, and the percentage
        of time the second word follows the first.
        This relies on values appearing in sorted order; we need the total
        number of times the first word appears before we can compute the
        percentage for each second word.
        """
        total = None

        for value_type, data in value:
            if value_type == 'A: total':
                total = data
            else:
                assert value_type == 'B: stats'
                word, count = data
                # A comes before B, so total should already be set
                percent = 100.0 * count / total
```

```
            yield (prev_word, word), (total, count, percent)


if __name__ == '__main__':
    MRNextWordStats.run()
```

## Multiplication matricielle

Identifiez un mapper et un reducer vous permettant de faire la multiplication matricielle.

Ecrivez le code permettant de faire une multiplication matricielle en lisant les valeurs d'une matrice rangées dans un fichier selon le format suivant.

Pour une matrice Matrice M1.txt, mettez une valeur par ligne précédée des indices de ligne et de colonne. Par exemple la matrice unité 2x2 serait stockée comme :

0 0 1

0 1 0

1 0 1

1 1 0

Vous aurez probablement besoin de distinguer la matrice de gauche de la matrice de droite. Vous pourrez utiliser un autre codage pour vous simplifier la vie où les lignes de la matrice de gauche commencent par 1, les lignes de la matrice de droite commencent par 2.

## K-means

Ecrivez un mapper et un reducer pour réaliser une étape de K-means, étant donné 3 centroids et 10 points dans un fichier texte comme suis. Les vecteurs sont de dimensions 2.

Les étapes seront 1. le calcul de distance de chaque point aux centroids, 2. trouver la distance minimale pour assigner les points aux centroids correspondants, 3. recalculer la position du centroid en faisant la moyenne des points assignés.

```
k 0 0.5997271982681154 0.45601314691051775
k 1 0.034624947802525896 0.9114679574841518
k 2 0.696194092994658 0.675816757952084
x 0 0.9846344192226623 0.9709162676887786
x 1 0.696194092994658 0.675816757952084
x 2 0.6287248674910642 0.8269759672255591
x 3 0.5179380850479713 0.4332178380853853
x 4 0.5997271982681154 0.45601314691051775
x 5 0.034624947802525896 0.9114679574841518
x 6 0.7624592473716646 0.9927550834531442
x 7 0.006063459321677178 0.3833451846435594
x 8 0.28624883511135724 0.17877366109616866
x 9 0.9154494682064119 0.1760196666282976
```