

# SINF 1250: Rapport de projet

Maxime Dillion

Jacques Yakoub

24 décembre 2017

# Table des matières

<b>1</b>	<b>Théorie</b>	<b>2</b>
1.1	Rappel . . . . .	2
1.1.1	Système d'équations linéaires . . . . .	2
1.1.2	Power method . . . . .	3
1.2	Calcul . . . . .	3
1.2.1	Système d'équation sous forme matriciel . . . . .	3
1.2.2	Système d'équation après résolution . . . . .	3
<b>2</b>	<b>Implémentation</b>	<b>4</b>
2.1	Matrice d'adjacence . . . . .	4
2.2	Degré entrant des noeuds . . . . .	4
2.3	Matrice de probabilité de transition . . . . .	4
2.4	Matrice Google . . . . .	4
2.5	Trois premières itérations de la power method . . . . .	5
2.5.1	Itération n°1 . . . . .	5
2.5.2	Itération n°2 . . . . .	5
2.5.3	Itération n°3 . . . . .	5
2.6	Score PageRank . . . . .	5
<b>A</b>	<b>Code complet</b>	<b>6</b>

# 1 Théorie

## 1.1 Rappel

Quelque soit la méthode que nous choisissons par la suite, il faut obtenir la matrice Google. Celle-ci s'obtient de la manière suivante :

- Obtention de la matrice de probabilité, en utilisant la matrice d'adjacence et les degrés entrants de chaque noeud :

$$p_{ij} = \frac{w_{ij}}{w_i}$$

Avec  $1 \leq i \leq n$  et  $1 \leq j \leq m$  et où  $W$  est la matrice d'adjacence,  $P$  est la matrice de probabilité et  $w_{i.} = \sum_{j=1}^m w_{ij}$

- Calculer la matrice Google en utilisant cette formule :

$$(\alpha * P) + \frac{1 - \alpha}{n} * e^T$$

Où  $\alpha$  est un paramètre compris entre 0 et 1,  $P$  la matrice de probabilité de transition,  $n$  étant le nombre de noeuds, et un vecteur colonne composé uniquement de 1.

### 1.1.1 Système d'équations linéaires

Avec cette méthode, la recherche du vecteur de score devient une question de vecteur propre. Ici, il serait fait le choix d'expliquer le vecteur de droite afin de faciliter les calculs dont voici son expression mathématique :

$$G^T * x = x$$

Où  $x$  étant le vecteur et  $G^T$  la transposée de la matrice google.

Appliquons cette formule à la situation présente :

$$\begin{bmatrix} \frac{1}{50} & \frac{187}{350} & \frac{49}{200} & \frac{1}{50} & \frac{47}{100} \\ \frac{11}{50} & \frac{1}{50} & \frac{49}{200} & \frac{1}{5} & \frac{47}{100} \\ \frac{21}{50} & \frac{1}{50} & \frac{1}{50} & \frac{37}{50} & \frac{1}{50} \\ \frac{3}{25} & \frac{97}{350} & \frac{79}{200} & \frac{1}{50} & \frac{1}{50} \\ \frac{11}{50} & \frac{26}{175} & \frac{19}{200} & \frac{1}{50} & \frac{1}{50} \end{bmatrix} * \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix} = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix}$$

En rajoutant une ligne de 1, nous nous assurons que la somme de  $x1, \dots, x5$  termes doit valoir 1. Enfin, avant de pouvoir résoudre le système par des opérations matricielles pour obtenir une

forme en escalier , il faut simplifier la forme présente en passant les inconnues du côté gauche de l'équation.

### 1.1.2 Power method

Pour obtenir les scores PageRank, il nous faut calculer le vecteur propre de gauche correspondant à la valeur propre 1 de la matrice de google :

$$x^T * G = x^T$$

Ce vecteur peut être calculé grâce à la power method, où chaque itération est définie comme telle :

$$x^T(k+1) = \alpha * x^T(k) * P + \frac{1-\alpha}{n} * e^T$$

Où  $x^T$  est le vecteur de scores ( $x^T(0)$  = degrés entrant dans chaque noeuds),  $(1-\alpha)$  est la probabilité de téléportation,  $n$  est le nombre de noeuds et  $e^T$  est la transposée d'un vecteur colonne composé de 1.

Sauf dans certains cas spécifiques, cette itération sera toujours convergente et il nous suffit donc ensuite de tester si  $||x^T(k) - x^T(k+1)|| < v$ , ou  $v$  est l'erreur ( $10^{-8}$  dans notre cas), pour approximer au mieux le vecteur propre et donc obtenir le plus précisément possible les scores PageRank de chaque noeuds.

## 1.2 Calcul

### 1.2.1 Système d'équation sous forme matriciel

$$\left( \begin{array}{ccccc|c} 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{-49}{50} & \frac{187}{350} & \frac{49}{200} & \frac{1}{50} & \frac{47}{100} & 0 \\ \frac{11}{50} & \frac{-49}{50} & \frac{49}{200} & \frac{1}{5} & \frac{47}{100} & 0 \\ \frac{21}{50} & \frac{1}{50} & \frac{-49}{50} & \frac{37}{50} & \frac{1}{50} & 0 \\ \frac{3}{25} & \frac{97}{350} & \frac{79}{200} & \frac{-49}{50} & \frac{1}{50} & 0 \\ \frac{11}{50} & \frac{26}{175} & \frac{19}{200} & \frac{1}{50} & \frac{-49}{50} & 0 \end{array} \right)$$

### 1.2.2 Système d'équation après résolution

$$\left( \begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & \frac{278338891485003}{1185309949623550} \\ 0 & 1 & 0 & 0 & 0 & \frac{1574961688097759}{7540409867060228} \\ 0 & 0 & 1 & 0 & 0 & \frac{3332146709283619}{1.32265796170871e+16} \\ 0 & 0 & 0 & 1 & 0 & \frac{534623}{2789366} \\ 0 & 0 & 0 & 0 & 1 & \frac{1572003}{13946830} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

On a donc :  $x_1 = \frac{278338891485003}{1185309949623550}$  ;  $x_2 = \frac{1574961688097759}{7540409867060228}$  ;  $x_3 = \frac{3332146709283619}{1.32265796170871e+16}$  ;  $x_4 = \frac{534623}{2789366}$  ;  $x_5 = \frac{1572003}{13946830}$

## 2 Implémentation

### 2.1 Matrice d'adjacence

$$\begin{pmatrix} 0 & 2 & 4 & 1 & 2 \\ 4 & 0 & 0 & 2 & 1 \\ 3 & 3 & 0 & 5 & 1 \\ 0 & 1 & 4 & 0 & 0 \\ 3 & 3 & 0 & 0 & 0 \end{pmatrix}$$

### 2.2 Degré entrant des noeuds

$$(10 \quad 9 \quad 8 \quad 8 \quad 4)$$

### 2.3 Matrice de probabilité de transition

$$\begin{pmatrix} 0 & \frac{2}{9} & \frac{4}{9} & \frac{1}{9} & \frac{2}{9} \\ \frac{4}{7} & 0 & 0 & \frac{2}{7} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{5}{12} & \frac{1}{12} \\ 0 & \frac{1}{5} & \frac{4}{5} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

### 2.4 Matrice Google

$$\begin{pmatrix} \frac{1}{50} & \frac{11}{50} & \frac{21}{50} & \frac{3}{25} & \frac{11}{50} \\ \frac{187}{350} & \frac{1}{50} & \frac{1}{50} & \frac{97}{350} & \frac{26}{175} \\ \frac{49}{200} & \frac{49}{200} & \frac{1}{50} & \frac{79}{200} & \frac{19}{200} \\ \frac{1}{50} & \frac{1}{5} & \frac{37}{50} & \frac{1}{50} & \frac{1}{50} \\ \frac{47}{100} & \frac{47}{100} & \frac{1}{50} & \frac{1}{50} & \frac{1}{50} \end{pmatrix}$$

## 2.5 Trois premières itérations de la power method

### 2.5.1 Itération n°1

$$\left( \frac{1051}{4550} \quad \frac{391}{1950} \quad \frac{527}{1950} \quad \frac{191}{1050} \quad \frac{794}{6825} \right)$$

### 2.5.2 Itération n°2

$$\left( \frac{43003}{182000} \quad \frac{2121}{10000} \quad \frac{27683}{113750} \quad \frac{35673}{182000} \quad \frac{20429}{182000} \right)$$

### 2.5.3 Itération n°3

$$\left( \frac{67623}{288557} \quad \frac{93767}{451224} \quad \frac{145393}{568750} \quad \frac{188765}{996486} \quad \frac{15789}{140000} \right)$$

## 2.6 Score PageRank

$$\left( \frac{139718}{594991} \quad \frac{200248}{958723} \quad \frac{50534}{200589} \quad \frac{154407}{805610} \quad \frac{112253}{995910} \right)$$

## A Code complet

```
import csv
import numpy as np
import fractions
# A more user-friendly way to print matrix as fractions
# credits to https://stackoverflow.com/a/42209716/6149867
np.set_printoptions(formatter={'all': lambda x: str(fractions.
    Fraction(x).limit_denominator())})

def pageRankScore(A: np.matrix, alpha: float = 0.9):
    # without astype : numpy thinks it is a matrix of string
    adj_matrix = A.astype(np.int)
    print("Starting_the_program:_Matrix_of_shape_%s_with_alpha_%f\n
        %s" % (A.shape, alpha, adj_matrix))
    # Vector of the sum for each column
    in_degree = adj_matrix.sum(axis=0)
    print("indegree_of_each_node\n%s" % in_degree)

    # help us to not call sum multiple time when we will modify the
        matrix
    out_degree = adj_matrix.sum(axis=1).getA1()

    probability_matrix = []
    counter = 0
    for line in adj_matrix:
        row = line.getA1() / out_degree[counter]
        probability_matrix.append(row)
        counter += 1

    probability_matrix = np.matrix(probability_matrix, np.float)
    print("Computing_the_transition-probability_matrix\n%s" %
        probability_matrix)

    print("Init_vector_(using_in_degree_and_normalize_it);")
    vector = in_degree.transpose()
    # Now time to normalize this vector by the sum
    vector = vector / vector.sum()
```

```

print(vector)

# Relative error
epsilon = pow(10, -8)
print("Power_method_iteration_(left_eigenvector)_of_the_google_
matrix_with_an_error_of_%s" % epsilon)
# Number of nodes (number of columns inside the probability
matrix)
n = probability_matrix.shape[1] # tuple shape : rows, columns
# vector
vector_google = vector.transpose()
# column vector : Full of ones line vector
et = np.ones(n)
# Vector's norms
norm = np.linalg.norm(vector_google, ord=1)
new_norm = 0
# google matrix
print("Google_matrix:_")
google = (alpha*probability_matrix)+((1-alpha)/n)*et
print(google)
print("Iterations_now_begins:_")
# counter for iteration
step = 1
while abs(new_norm-norm) / norm > epsilon:
    print("t_Iteration_n^°_%s" % step)
    norm = np.linalg.norm(vector_google, ord=1)
    vector_google = vector_google * google
    new_norm = np.linalg.norm(vector_google, ord=1)
    """ Just a way to print only the first 3 iterations """
    if step in [1, 2, 3]:
        print("t_t_Computed_PageRank_score:_\n\t\t_%s" %
vector_google)
    step = step + 1
print("The_final_PageRank_score_is:_\n\t_%s" % vector_google)

# Read the matrix from csv and transform it to numpy matrix
def main():

    matrix = []
    cr = csv.reader(open("adjacenceMatrix.csv", "r"))
    for i, val in enumerate(cr):
        matrix.append(val)

    adj_matrix_np = np.matrix(matrix)
    pageRankScore(A=adj_matrix_np)

if __name__ == "__main__": main()

```