Projet de recherche

Page-Rank Google

Auteurs : Ghiles Kemiche

Nom de l'organisme : Institut Mathématiques Orsay
Professeur Encadrant : Benjamin Graille

Licence double diplôme Mathématiques - Informatique

2022 - 2023

Table des matières

Table des figures

1

Introduction

La capacité à trouver des informations pertinentes sur le Web est cruciale pour une utilisation efficace de celui-ci. Des intérêts économiques considérables sont en jeu et différentes entreprises multinationales sont impliquées. Le leader actuel du marché, Google, utilise plusieurs algorithmes pour déterminer la pertinence des références, certains étant des secrets industriels bien protégés et d'autres étant publics. Nous allons nous intéresser ici à l'algorithme PageRank.

Le PageRank fonctionne en attribuant une note de "page rank" à chaque page Web en fonction du nombre et de la qualité des liens pointant vers cette page. Plus une page a de liens de qualité pointant vers elle, plus son "page rank" sera élevé. Le PageRank est un facteur important dans le classement des pages par Google.

On peut considérer que le Web est une collection de N pages, avec N très grand ($\sim 25 \times 10^{10}$ en octobre 2005). La plupart de ces pages incluent des liens hypertextes vers d'autres pages. On dit qu'elles pointent vers ces autres pages. L'idée de base utilisée par les moteurs de recherche pour classer les pages par ordre de pertinence décroissante consiste à considérer que plus une page est la cible de liens venant d'autres pages, c'est-à-dire plus il y a de pages qui pointent vers elle, plus elle a de chances d'être fiable et intéressante pour l'utilisateur final, et réciproquement. Il s'agit donc de quantifier cette idée, c'est-à-dire d'attribuer un rang numérique ou score de pertinence à chaque page.

1.1 Données

On se donne donc un ordre arbitraire sur l'ensemble des pages que l'on numérote de $i = 1$ à $i = N$. La structure de connectivité du Web peut alors être représentée par une matrice C de taille $N \times N$ telle que $c_{i,j} = 1$ si la page j pointe sur la page i, $c_{i,j} = 0$ sinon. Les liens d'une page sur elle-même ne sont pas significatifs, on pose donc $c_{i,i} = 0 \ \forall i \in \{1, ..., N\}$. On observe que la ligne i contient tous les liens significatifs qui pointent sur la page i, alors que la colonne j contient tous les liens significatifs présents sur la page j. On souhaite attribuer à chaque page i un score $r_i \in \mathbb{R}_+^*$ de façon à pouvoir classer l'ensemble des pages par score décroissant et présenter à l'utilisateur une liste ainsi classée des pages correspondant à sa requête.

1.2 Algorithme initial

L'algorithme PageRank part du principe qu'un lien de la page j pointant sur la page i contribue positivement au score de cette dernière, avec une pondération par le score r_j de la page dont est issu le lien, on déduit alors :

- Une page ayant un score élevé a ainsi plus de poids qu'une n'ayant qu'un score médiocre
- Le nombre total de liens présents sur ladite page $N_j = \sum_{k=1}^{N} c_{k,j}$.

On introduit donc la matrice Q définie par :

$$q_{i,j} = \begin{cases} \frac{c_{i,j}}{N_j} & \text{si } N_j \neq 0 \\ 0 & \text{sinon} \end{cases}$$

On peut remarquer que la somme des coefficients des colonnes non nulles de Q vaut toujours 1 ie pour $N_j \neq 0$ on a $\sum_{i=1}^{N} \frac{c_{i,j}}{N_j} = \frac{N_j}{N_j} = 1$. L'application des principes ci-dessus conduit donc à une équation pour le vecteur $r \in \mathbb{R}^N$ des scores des pages de la forme

$$r_i = \sum_{j=1}^{N} q_{i,j} r_j,$$

c'est-à-dire $r = Qr$.

Cette équation peut être réécrite sous la forme $r = (I - Q)^{-1}e$ où I est la matrice identité de taille $N \times N$ et e est un vecteur de taille N contenant tous ses éléments égaux à 1. La matrice $I - Q$ est appelée matrice de Google. Le vecteur r obtenu est alors le vecteur des scores de PageRank des pages. La méthode itérative consistant à calculer successivement $r^{(k+1)} = Qr^{(k)}$ à partir d'une valeur initiale $r^{(0)}$ converge généralement vers le vecteur r des scores de PageRank.

1.3 Problème posé

Il peut arriver que la matrice Q n'admette pas la valeur propre 1, ce qui invalide la stratégie précédente. En général, cette matrice contient des colonnes nulles car certaines pages du Web n'ont pas de liens sortants. Par conséquent, les colonnes associées à ces pages seront nulles.

En outre, il est très probable que la valeur propre 1 ne soit pas admise par la matrice Q car, en général, les valeurs propres de cette matrice ont un module strictement inférieur à 1. Cela est dû au fait que, pour chaque page j avec $N_j > 0$, la somme des coefficients de la colonne j de la matrice Q est égale à 1 (transposée d'une matrice stochastique). Comme la valeur propre 1 correspond à une multiplication par 1, il est peu probable que la matrice Q admette cette valeur propre.

2

Etapes de Résolution

La matrice Q peut ne pas admettre 1 comme valeur propre en général. Néanmoins, il est possible de la modifier de manière à ce qu'elle admette 1 comme sa valeur propre la plus grande et non multiple.

2.1 Existance de la valeur propre 1

Pour remédier au fait que la matrice Q ne possède pas 1 comme valeur propre, nous allons remplir les colonnes nulles de cette matrice de sorte à obtenir une matrice stochastique transposée. Nous définissons alors $e = (1, ..., 1)^T$ et d tel que $d_j = 1$ si $N_j = 0$ et $d_j = 0$ sinon. La matrice $P = Q + \frac{1}{N} e d^T$ est alors la transposée d'une matrice stochastique, c'est-à-dire que ses coefficients sont tous positifs et que la somme des coefficients de chaque colonne vaut 1. On a donc pour tout $i, j \in \{1, ..., N\}$:

$$(e^T P)_{i,j} = \begin{cases} \sum_{i=1}^{N} \frac{1}{N} = 1 & \text{si } N_j = 0 \\ \sum_{i=1}^{N} \frac{c_{i,j}}{N_j} = \frac{N_j}{N_j} = 1 & \text{sinon} \end{cases}$$

et donc $e^T P = e^T$, on voit que P admet bien 1 comme valeur propre.
De plus si $\lambda \in Sp(P)$ on pose $X = (x_1, ..., x_N)^T$ le vecteur propre associé à λ. Soit $i \in 1, \ldots, n$ tel que $|x_i| = \max\limits_{1 \leq k \leq n} |x_k|$. Comme $PX = \lambda X$, en regardant la i-ième coordonnée, on obtient :

$$\sum_{j=1}^{n} P_{i,j} x_j = \lambda x_i$$

En passant au module, on obtient donc :

$$|\lambda x_i| = |\lambda||x_i| = \left|\sum_{j=1}^{n} P_{i,j}x_j\right| \leq (P_{i,1} + \ldots + P_{i,n})|x_i| = |x_i|$$

On conclut donc que $|\lambda| \leq 1$.

La valeur propre 1 est souvent multiple pour la matrice P dans le cas du Web car il peut exister plusieurs pages qui n'ont pas de liens sortants, mais qui sont liées à toutes les autres pages du Web. Dans ce cas, chaque colonne de la matrice Q associée à ces pages sera remplie de $\frac{1}{N}$ et la somme de chaque colonne de P sera égale à 1, ce qui entraînera l'apparition de la valeur propre 1 comme valeur propre multiple de la matrice P. Si aucune page n'a de liens sortants, alors la valeur propre 1 sera unique.

2.2 Valeur propre simple

Il nous manque plus qu'à gerer le cas où la valeur propre 1 est multiple, dans ce cas on pose $0 < \alpha < 1$ et on definit :

$$A = \alpha P + (1 - \alpha)\frac{ee^T}{N}$$

On peut montrer que pour A est toujours stochastique, car on a A est toujours positive , P est stochastique et ee^T/N est une matrice de remplie de $1/N$ qui est toujours stochastique, donc leur somme est aussi une matrice stochastique pour $0 < \alpha < 1$.

Le théorème de Perron-Frobenius nous garantit alors que la valeur propre dominante de A, c'est à dire 1, qui est strictement plus grande en valeur absolue que toutes les autres valeurs propres de A. De plus, il existe un vecteur propre associé à cette valeur propre dominante, ce qui conclut notre problème.

Il en résulte que la résolution de l'équation $A\mathbf{r} = \mathbf{r}$ ne diffère pas trop de la résolution de l'équation $Q\mathbf{r} = \mathbf{r}$, car A est une combinaison linéaire convexe de Q.

Remarque : La constante $1 - \alpha$ est utilisée pour définir la nouvelle matrice A, qui est une combinaison linéaire de la matrice P et de la matrice constante $\frac{ee^T}{N}$. Le

fait de mettre un poids $(1-\alpha)$ sur la matrice constante $\frac{ee^T}{N}$ permet d'introduire une probabilité de transition aléatoire entre les pages du Web. Cette probabilité est interprétée comme la probabilité pour un internaute de "zapper" sur une autre page au hasard lors de sa navigation. Plus précisément, lorsqu'un internaute se trouve sur une page, il a la possibilité de suivre un lien sur cette page et de naviguer vers une autre page, ou de rester sur la même page ou de zapper sur une autre page au hasard. La probabilité de zapping au hasard est donnée par $1-\alpha$. Si $\alpha = 1$, la matrice A redevient P donc les probabilités de transitions sont celles données par les liens de la page, si $\alpha = 0$ c'est la probabilité de transition est égale pour toutes les pages donc l'internaute zappe au hasard.

Une correction différente est possible en utilisant un autre vecteur p qui modifierait la matrice de téléportation du facteur de zapping $(1-\alpha)$, tel que dorénavant on aura :

$$A = \alpha P + (1-\alpha)\frac{pe^T}{N} \quad \text{avec} \quad \|p\| = N \quad \text{et} \quad p \neq e$$

Ce qui gardera néanmoins les propriétés de notre matrice et l'unicité du vecteur propre, mais fournira des résultats différents.Le vecteur p étant différent de e, la matrice de téléportation ne sera pas uniquement constituée uniquement de 1, et donc la probabilité de zapping vers une page aléatoire du web ne sera pas uniformément répartie. On pourrait imaginer une modification de l'algorithme du PageRank où p changerait en fonction de l'utilisateur pour mieux coller à ses habitudes de navigation. L'algorithme serait alors plus précis et pourrait donner de meilleurs résultats.

2.3 Variations du paramètre α

Le paramètre α a un grand impact sur la vitesse de convergence de l'algorithme du PageRank. En effet, plus le paramètre α est proche de 0, plus la vitesse de convergence est élevée, ce qui est un avantage primordial dans le traitement des plusieurs millions de pages du web. Au contraire, plus elle est proche de 1, plus la convergence est lente en addition avec une erreur réduite . Notons que peut importe si α est proche de 1 ou de 0 le classement restera quant à lui inchangé.
On pourrait alors se demander la raison du choix de $\alpha = 0.85$ par les équipes de Google, et la raison est en fait due à notre modélisation. Rappelons que le paramètre α a pour signification probabiliste qu'il est la probabilité de continuer à suivre les hyperliens de page en page, en opposition avec $(1 - \alpha)$ qui est la probabilité de zapping. Plus α est proche de 1, plus la matrice de transitions initiale est privilégiée et plus il est proche de 0, plus le zapping aléatoire prime. Ainsi, pour garantir le réalisme du modèle, il est nécessaire de prendre α grand pour coller au mieux aux habitudes de la majorité et la valeur $\alpha = 0.85$ a été retenue comme le meilleur compromis entre vitesse de convergence et réalisme.

3

Aspects Algorithmiques

Il nous reste maintenant de trouver le vecteur propre associé à la valeur propre 1.

3.1 Méthode de la puissance

Dans un premier temps nous allons utiliser l'algorithme de la méthode de la puissance pour trouver ce vecteur propre.

Algorithm 1 Algorithme de la puissance

Choisir un vecteur initial r
$r = \frac{r}{\|r\|}$
while erreur $\geq$ tolerance **do**
    r_tild= A * r
    erreur = $\|r - r\_tild\|$
    r = r_tild
Return : r

On peut remarquer que le fait que la matrice A soit remplie de coefficients, rend cette multiplication très conséquente.

3.2 Modification de l'algorithme

Comme $\|r\| = 1$ on peut écrire $Ar = \alpha Qr + \beta e/N$ et comme on a aussi $\|Ar\| = 1$ on a alors $\beta = 1 - \|\alpha Qr\|$ donc :

$$Ar = \alpha Qr + (1 - \|\alpha Qr\|)e/N$$

On pourra utiliser cette formule car on utilise la multiplication sur Q qui est une matrice creuse car elle a beaucoups de colonnes nulles comme vu dans 1.3 ce qui nous économise beaucoup de temps de calcul

On pourra ainsi modifier notre algorithme comme ceci :

Algorithm 2 Algorithme de la puissance

Choisir un vecteur initial r
$r = \frac{r}{\|r\|}$
while erreur $\geq$ tolérance **do**
$r' = \alpha Q r$
$\beta = 1 - \|r'\|_1$
$\tilde{r} = r' + \frac{\beta}{N} e$
erreur $= \|\tilde{r} - r\|_1$
$r = \tilde{r}$

Sur des matrices de très grande envergure, l'avantage d'une matrice creuse n'est pas négligeable. Considérons par exemple qu'une page a en moyenne 1000 liens vers d'autres pages, pour N très grand ($N \geq 10000$), il y aurait environ 1000N éléments non nuls, ce qui ferait alors passer l'algorithme d'une complexité quadratique à linéaire.

4

Simulations numériques

On montrera dans cette section quelques simulations et conclusions tirées sur les simulations numériques faites.

4.1 Comparaison de l'algorithme Page-Rank et l'algorithme de la puissance classique

Comme prévu, lorsque les matrices deviennent plus grandes, il est constaté que les multiplications sont généralement plus lentes sur la méthode de la puissance classique que sur l'algorithme de Page-Rank.

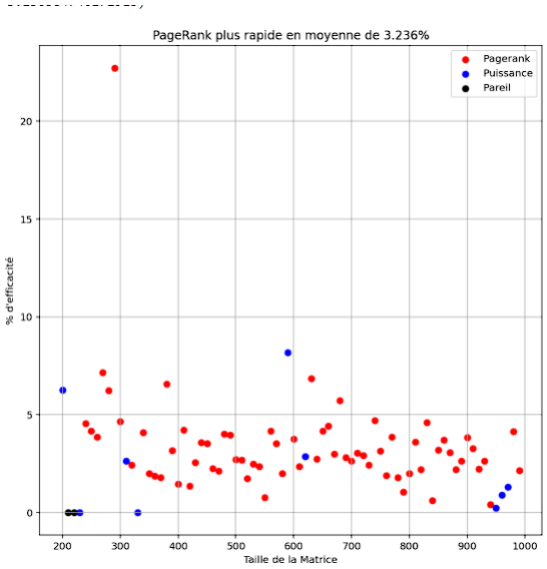

Figure 4.1 : comparaison de l'algo de la puissance sur A et le pagerank de google

4.2 Taille de la matrice et complexité

La méthode de la puissance s'utilise sur de grandes matrices de ce fait plus la matrice est grande plus l'algorithme converge plus vite, en outre plus la matrice est grande, plus la plus grande valeur propre est généralement éloignée des autres valeurs propres en termes de module, et donc plus le taux de convergence est rapide.

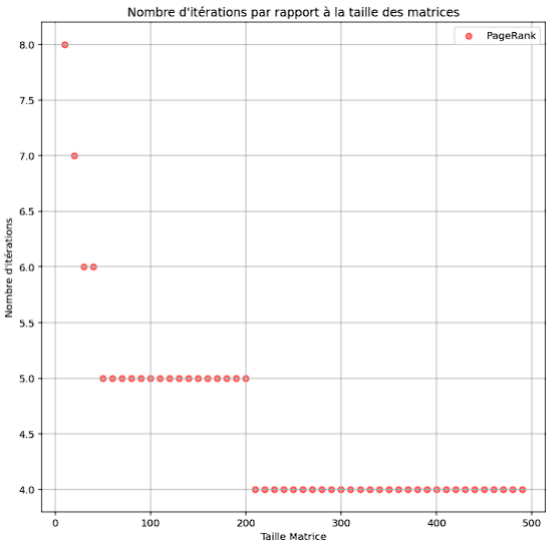

Figure 4.2 : Nombre d'iterations en fonction de la taille des matrices

4.3 Taille de la matrice et erreur

Il est probable qu'il existe une forte corrélation entre les différentes lignes et colonnes de la matrice quand elle est grande, ce qui peut entraîner une réduction de l'erreur entre les itérations successives.

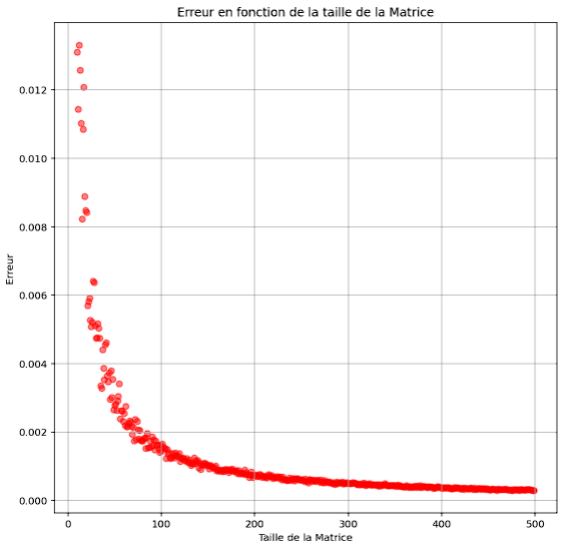

Figure 4.3 : Comportement de l'erreur en fonction de la taille des matrices

4.4 Taille de matrice et temps d'execution

Plus la matrice est grande plus la multiplication prend du temps .

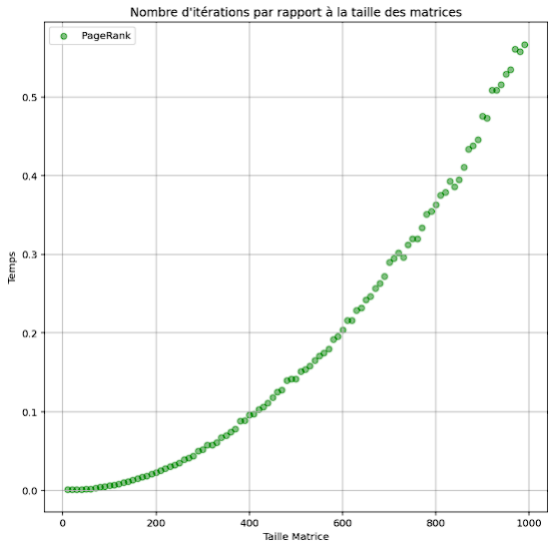

Figure 4.4 : Comportement de l'erreur en fonction de la taille des matrices

4.5 Comportement du paramètre α

Finalement, on retrouve dans ces 4 derniers graphes les résultats énoncés dans la section 2.3, notamment le fait qu'un α assez proche de 0 converge plus rapidement qu'un alpha proche de 1, avec un taux d'erreur supérieur à ce dernier.

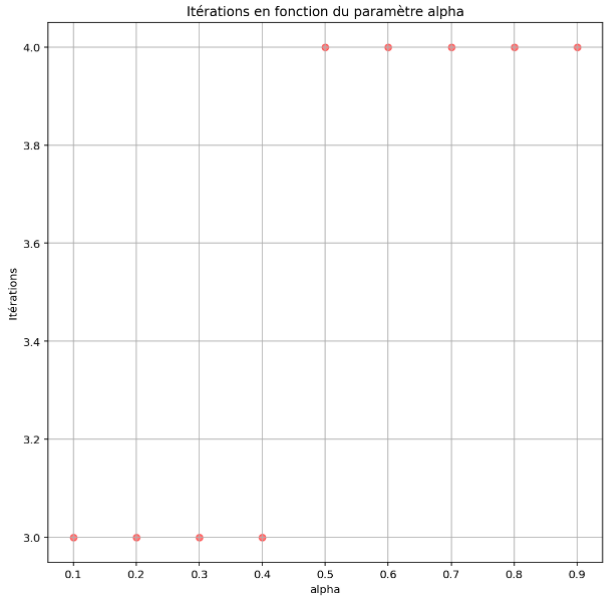

Figure 4.5 : Nombre d'itérations en fonction de α

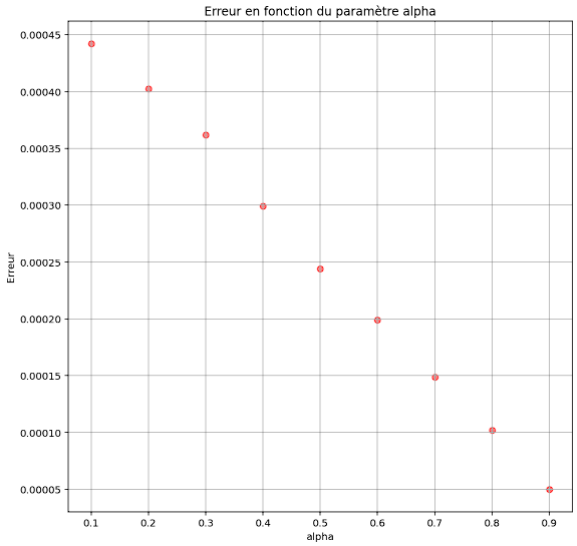

Figure 4.6 : Précision du résultat en fonction de α

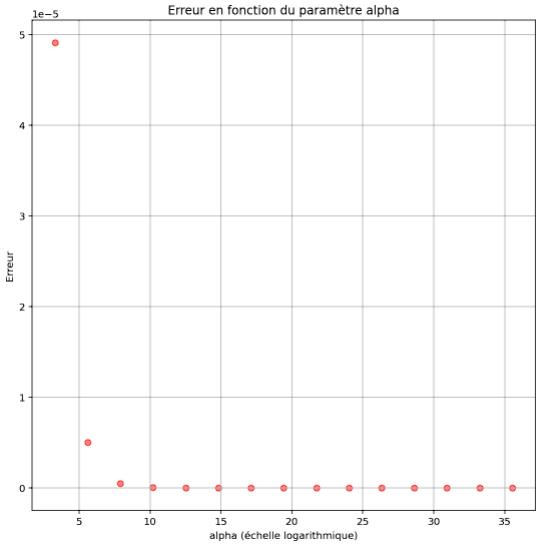

Figure 4.7 : Précision du résultat en fonction de α au voisinage de 1

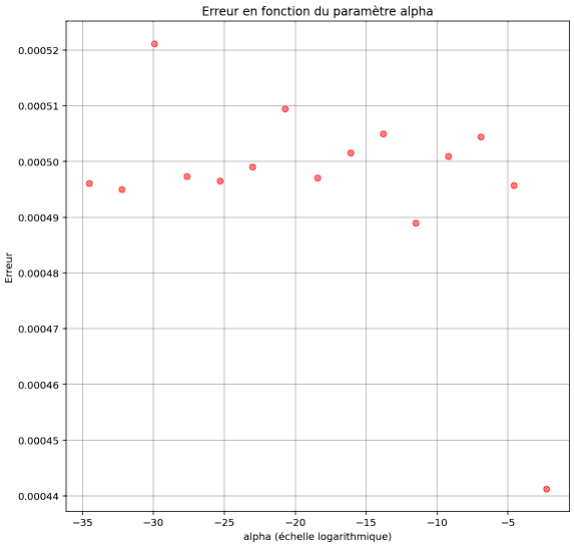

Figure 4.8 : Précision du résultat en fonction de α au voisinage de 0