ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Rapport de projet mathématique

Google PageRank

Xavier CAVARELLI et **Jonathan RYBAK**

1) Introduction

PageRank est un algorithme révolutionnaire d'analyse de lien permettant de mesurer l'importance relative d'un document dans un réseau d'hyperliens. Cet algorithme a été inventé dans les années 2000 par Google au moment où sortaient les premières générations de moteurs de recherche comme Altavista.

Google s'est rapidement imposé comme le meilleur moteur de recherche grâce à l'algorithme PageRank qui s'est révélé être beaucoup plus efficace que ceux de ses concurrents. Le principe de l'algorithme est relativement simple à comprendre, cela repose sur le fait que plus une page internet est citée depuis d'autres pages, plus cette page sera intéressante à consulter pour la personne effectuant la recherche.

Nous allons maintenant nous intéresser au fonctionnement de cet algorithme en détail.

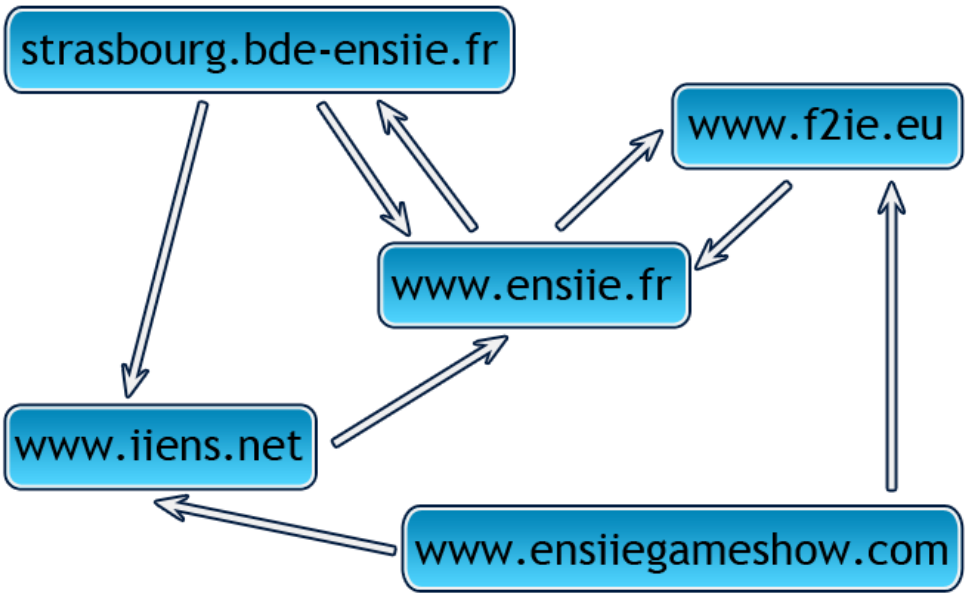
2) Problème

Afin de comprendre précisément le fonctionnement de cet algorithme, nous avons choisi de l'appliquer sur un exemple relativement simple. Nous allons montrer à travers cet exemple que plus une page est pointée par d'autres pages, plus elle sera pertinente pour l'utilisateur et sera bien classée parmi les pages analysées.

Il est possible de représenter ce réseau composé de sites internet par l'intermédiaire d'un graphe. Chaque sommet de ce graphe représente un site internet dont on souhaite connaître le rang et les arcs orientés représentent le fait qu'une page pointe vers la page ciblée.

Étant étudiants à l'ENSIIE, nous avons choisi de nous intéresser aux différents sites liés à notre école. Nous en avons identifié 5 principaux et souhaitons, à l'aide de l'algorithme du PageRank, déterminer celui qui aura le plus de chance d'apparaître en premier lors d'une recherche.

L'exemple que nous avons choisi est représenté par le graphe suivant :



À titre d'exemple, d'une part, on remarque que le site "www.iiens.net" est pointé depuis les sites "strasbourg.bde-ensiie.fr" et "www.ensiiegameshow.com". D'autre part, le site "www.iiens.net" pointe vers le site "www.ensiie.fr".

3) Modélisation du problème

La modélisation du problème consiste à utiliser les informations données dans le graphe ci-dessus. En effet, il est possible d'associer à un tel graphe une matrice d'adjacence que l'on notera X. Pour définir cette matrice, il est nécessaire de donner un ordre aux sites internet du graphe que l'on numérote de 1 à 5. Dans le cas de notre graphe, on considère l'ordre suivant : strasbourg.bde-ensiie.fr (1) , www.ensiie.fr (2), www.f2ie.eu (3), www.iiens.net (4) et www.ensiiegameshow.com (5).

La matrice X est construite de manière à ce que x[i][j] = 1 si la page i pointe vers la page j et 0 dans le cas contraire. Il n'est pas possible d'avoir une page qui pointe vers elle-même c'est pourquoi x[i][i] = 0. Après exploitation du graphe de notre exemple, on se retrouve avec la matrice X suivante :

$$X = \begin{pmatrix} 0. & 1. & 0. & 1. & 0. \\ 1. & 0. & 1. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 1. & 0. \end{pmatrix}$$

Dans cette matrice, on retrouve les mêmes informations que sur le graphe sur un site :

- Pour chaque ligne, on peut connaître les sites internet pointés par un site
- Pour chaque colonne, on peut connaître les sites internet qui pointent vers un site

4) Résolution du problème

À partir de cette matrice, nous pouvons créer la matrice des pondérations Q qui est définit très simplement par :

Q[i][j] = X[i][j] / (Somme des valeurs de la $j^{ème}$ colonne)
0 si la $j^{ème}$ colonne est nulle

Dans notre exemple, nous obtenons la matrice Q suivante :

$$Q = \begin{pmatrix} 0. & 0.3333333 & 0. & 0.5 & 0. \\ 1. & 0. & 0.5 & 0. & 0. \\ 0. & 0.3333333 & 0. & 0. & 0. \\ 0. & 0.3333333 & 0. & 0. & 0. \\ 0. & 0. & 0.5 & 0.5 & 0. \end{pmatrix}$$

Il est intéressant de noter que la matrice Q ainsi définie doit vérifier la propriété suivante : la somme des valeurs de chaque colonne non vide doit être égale à 1.

Le vecteur des scores des pages r est en fait un vecteur définit par r = Qr.

Néanmoins, il faut prendre en considération que les matrices carrées sur lesquelles cet algorithme est appliqué ne sont pas de taille 5 mais plutôt de dizaines de millions. Nous n'avons donc aucune raison de penser que celle-ci ait 1 comme valeur propre, ce qui impliquerait que nous serions incapables de trouver une valeur pour le vecteur r.

Pour cette raison, nous allons faire évoluer le problème en travaillant sur une nouvelle matrice que nous allons définir de telle manière que nous sachions qu'elle admette 1 comme valeur propre. Nous définissons donc une nouvelle matrice P avec la formule suivante :

$$P = Q + \frac{1}{N} e * {}^{t}d$$

Dans cette formule, nous utilisons 3 variables que nous n'avons pas encore défini et qui sont :

- N : la taille de l'espace dans lequel on travaille
- e : un vecteur colonne de 1
- d : un vecteur colonne dont le $i^{ème}$ coefficient est 1 si la $i^{ème}$ colonne de X est vide et 0 sinon.

Dans notre exemple concret, nous obtenons la matrice P suivante :

$$P = \begin{pmatrix} 0. & 0.3333333 & 0. & 0.5 & 0.2 \\ 1. & 0. & 0.5 & 0. & 0.2 \\ 0. & 0.3333333 & 0. & 0. & 0.2 \\ 0. & 0.3333333 & 0. & 0. & 0.2 \\ 0. & 0. & 0.5 & 0.5 & 0.2 \end{pmatrix}$$

À la différence de la matrice Q, la somme de toutes valeurs de chaque colonne de P est égale à 1. Nous pouvons rapidement remarquer que la différence entre la matrice P et la matrice Q tient entièrement au fait que les colonnes vides de Q ont été remplies par la même valeur : 1/N. Ceci se voit très facilement dans la définition de P puisque le produit e * ᵗd donne une matrice dont les colonnes remplies de 1 sont les colonnes vides de X.

De plus, nous remarquons que P admet parfois 1 comme valeur propre multiple ; en effet, d'après les propriétés que nous connaissons sur P, nous savons que : ᵗe P = ᵗe. Or nous voulons être sûr d'obtenir un seul vecteur solution en cherchant le vecteur propre de la valeur propre 1 de la matrice P.

Afin de nous assurer de ceci, nous allons définir une nouvelle matrice en nous assurant qu'elle admette 1 comme valeur propre simple. Nous définissons donc la matrice A avec l'équation suivante ($0 < \alpha < 1$) :

$$A = \alpha P + (1-\alpha)\frac{1}{N} e * {}^t e$$

Après quelques recherches sur internet, nous avons trouvé la valeur de α qui est de 0.85. Cette valeur a été trouvée de manière empirique comme étant celle qui fournissait les meilleurs résultats.

Dans notre exemple, nous obtenons donc la matrice A suivante :

$$A = \begin{pmatrix} 0.03 & 0.3133333 & 0.03 & 0.455 & 0.2 \\ 0.88 & 0.03 & 0.455 & 0.03 & 0.2 \\ 0.03 & 0.3133333 & 0.03 & 0.03 & 0.2 \\ 0.03 & 0.3133333 & 0.03 & 0.03 & 0.2 \\ 0.03 & 0.03 & 0.455 & 0.455 & 0.2 \end{pmatrix}$$

Il ne nous reste maintenant plus qu'à calculer le vecteur r tel que **r = Ar**.

Nous sommes à présent en mesure de calculer le vecteur solution r. Il existe différentes méthodes possibles et nous allons nous choisir d'utiliser la **méthode de la puissance**. Cette méthode consiste à prendre un vecteur aléatoire, par exemple x_0, et de construire une suite récurrente v_n telle que :

$$v_0 = x_0 \quad \text{et} \quad v_{n+1} = \frac{A v_n}{\|A v_n\|}$$

Lors de l'implémentation de cette méthode sur Scilab, il a fallu définir une condition d'arrêt. Nous avons alors décider d'utiliser comme condition d'arrêt, non pas le nombre d'itération, mais la convergence. En effet, lors ce que nous appliquons la méthode de la puissance, la fonction continue d'itérer tant que la différence entre v_n et v_{n+1} est supérieure à 10^{-a}, avec a un ordre de grandeur passé en paramètre de la fonction.

Une fois la méthode de la puissance implémentée, nous n'avons plus qu'à l'utiliser pour calculer le vecteur r. Dans notre cas, nous trouvons le vecteur r suivant :

$$r = \begin{pmatrix} 0.4538539 \\ 0.6542570 \\ 0.3184940 \\ 0.3184940 \\ 0.4038411 \end{pmatrix}$$

Afin de comprendre le vecteur r que nous obtenons, nous commençons tout d'abord par faire le lien entre les pages et leurs scores, puis nous déterminerons le rang de chaque page. Le rang de chaque page correspond simplement à son rang dans un classement par ordre décroissant des scores des pages. Voilà un tableau récapitulant les résultats de notre exemple :

Numéro	Page	Score	Rang
1	strasbourg.bde-ensiie.fr	0.4538539	2
2	**www.ensiie.fr**	**0.6510570**	**1**
3	www.f2ie.eu	0.3184940	4
4	www.iiens.net	0.3184940	4
5	www.ensiiegameshow.com	0.4038411	3

Nous en déduisons donc que la page susceptible d'apparaître en premier lors d'une recherche est la page **www.ensiie.fr**.

5) Améliorations apportées

En complément de cette méthode, il y avait dans le sujet, deux pistes pour améliorer la méthode de la puissance dans le cas de l'algorithme de PageRank.

La première piste à explorer était de normaliser le vecteur initial lors du calcul de la suite récurrente. Cela permet un gain en vitesse de calcul car nous ne sommes alors plus obligés de renormaliser le vecteur v_n calculé à chaque itération. Nous avons néanmoins remarqué que cette modification ne permet, en général, pas de gain, ou peu, dans le nombre d'itérations dans la fonction, quand on compare les deux méthodes avec le même vecteur initial et le même ordre de grandeur bien entendu. C'est la possibilité de pouvoir effectuer ces comparaisons qui nous a poussé à passer le vecteur initial en paramètre plutôt que de le générer aléatoirement dans la fonction ainsi que de renvoyer le nombre d'itérations effectuées dans la fonction.

La deuxième piste à explorer consistait, quant à elle, à utiliser une décomposition de Ar afin de gagner en temps de calcul. En effet, on pouvait s'apercevoir que :

$$r = Ar \Leftrightarrow r = \alpha Qr + \frac{\beta}{N} e \quad \text{avec} \quad \beta = 1 - \|\alpha Qr\|_1$$

Or $r_1 = 1$, par définition du vecteur initial : $r = x_0 / ||x_0||_1$ donc est une constante. Nous obtenons donc le vecteur r de manière relativement simple :

$$r_n = \alpha^n Q^n r_0 + \frac{\beta}{N} e \sum_{i=0}^{n-1} \alpha^i Q^i$$

Le seul souci de cette méthode est que nous avons du modifier la structure générale de la fonction de la méthode de la puissance afin de passer en paramètre des éléments non utilisés précédemment, comme par exemple, le nombre d'itérations à effectuer. Nous avons pu remarquer que cette fonction ne nécessitait pas autant d'itérations que les deux précédentes, elle devient stable plus rapidement.

6) Conclusion

L'implémentation de l'algorithme PageRank aura été l'occasion de mettre en pratique des connaissances que nous avons acquises pendant le cours d'analyse numérique. Cela nous a permis de découvrir un cas concret dans lequel on utilise les matrices pour résoudre un problème. Nous avons choisi d'inventer un exemple pour exposer le fonctionnement de l'algorithme néanmoins nous aurions souhaité tester notre algorithme sur des situations réelles. Après plusieurs recherches, nous ne sommes pas parvenus à trouver un site internet permettant de générer un graphe de dépendance entre plusieurs sites internet. Nous avons réussi à implémenter la méthode de la puissance ainsi que ces deux améliorants. Néanmoins nous avons remarqué que bien que toutes ces fonctions nous donnent le même classement, elles ne donnent jamais exactement les mêmes valeurs, les unes par rapport à autres.