Projet : Algorithmes de ranking et de recommandations

- Je procéderai à un tirage au sort pour répartir les sujets 1 à 13
- Les graphes du web sur lesquels vous devrez travailler sont sur le site de l'ENT.
- Les groupes doivent être de taille 3. En cas de groupe plus petit, voir avec moi pour réduire la taille du sujet. Si il y a un groupe de taille 2, il y a le sujet 14. Si il y a plusieurs groupes de taille 1 ou 2, ils seront agrégés.
- Les analyses doivent être effectuées sur les 6 graphes du Web, y compris les plus grands.
- Pour chaque sujet, vous devez étudiez l'influence ces paramètres cités dans le sujet.
- Rendre un rapport décrivant les résultats numériques et le nouvel algorithme, pas la peine de me refaire le cours sur l'algorithme de PageRank 98. Mettre le code en annexe (ce n'est pas un projet de programmation)

1 Utilisation du pagerank précédent pour initialiser un nouveau calcul-graphe Erdos

Comme le Pagerank est calculé chaque mois et que le graphe du web évolue lentement, on peut essayer d'initialiser le vecteur x à la valeur obtenue lors du mois précédent. Il faut simplement faire attention aux nouveaux sommets.

L'initialisation est la suivante : soit S les sommets du graphe H^{t+1} qui ne sont pas dans le graphe H^t, S c'est les sommets nouveaux.

- si i est dans le graphe H^t et H^{t+1}, alors $x_{t+1}[i] = x_t[i]$
- si i est dans le graphe H^{t+1} mais pas dans le graphe H^t, alors $x_{t+1}[i] = 0.0$

Etudier la convergence de l'algorithme des puissance lorsqu'il est initialisé comme proposé (par rapport à la version du cours où il est initialisé avec e/n). Pour cela, vous proposez des algorithmes de modification des graphes du web (ajout de sommets, et ajout d'arcs depuis les nouvelles pages vers les anciennes en utilisant un ajout d'un graphe de Erdos) Un graphe de Erdos est un graphe dont le arcs sont générés aléatoirement avec une probabilité p.

Parametres à étudier :

- le nombre de nouveau sommets
- p
- α

2 Utilisation du pagerank précédent pour initialiser un nouveau calcul-graphe attachement préférentiel

Comme le Pagerank est calculé chaque mois et que le graphe du web évolue lentement, on peut essayer d'initialiser le vecteur x à la valeur obtenue lors du mois précédent. Il faut simplement faire attention aux nouveaux sommets ou aux sommets disparus.

L'initialisation est la suivante : soit S les sommets du graphe H^{t+1} qui ne sont pas dans le graphe H^t, S c'est les sommets nouveaux.

- si i est dans le graphe H^t et H^{t+1}, alors $x_{t+1}[i] = x_t[i]$
- si i est dans le graphe H^{t+1} mais pas dans le graphe H^t, alors $x_{t+1}[i] = 0.0$

Etudier l'accélération de la convergence de l'algorithme des puissance lorsqu'il est initialisé comme proposé (par rapport à la version du cours où il est initialisé avec e/n). L'attachement préférentiel consiste à ajouter des arcs depuis les nouveaux sommets vers les sommets de degré elevé dans le graphe initial. Parametres à étudier :

- le nombre de nouveau sommets
- le nombre de nouveaux arcs
- α

3 Utilisation du pagerank précédent pour initialiser un nouveau calcul-sous graphe

Comme le Pagerank est calculé chaque mois et que le graphe du web évolue lentement, on peut essayer d'initialiser le vecteur x à la valeur obtenue lors du mois précédent. Il faut simplement faire attention aux sommets disparus.

L'initialisation est la suivante : soit R les sommets du graphe H^t qui ne sont pas dans le graphe H^{t+1}. R contient les sommets disparus.

- si i est dans le graphe H^{t+1}, alors $x_{t+1}[i] = x_t[i]$
- ensuite on renormalise car la somme des $x_{t+1}[i]$ est plus petite que 1.0

Etudier l'accélération de la convergence de l'algorithme des puissance lorsqu'il est initialisé comme proposé (par rapport à la version du cours où il est initialisé avec e/n). Pour cela, vous proposez des algorithmes de modification des graphes du web (retrait de sommets, gestion des arcs arrivant sur une page vide) dont vous disposez Parametres à étudier :

- le nombre de sommets détruits
- le nombre d'arcs détruits
- α

4 Calcul du pagerank par Gauss Seidel-ascendant

L'algorithme de Gauss Seidel consiste à mélanger les solutions à l'itération $k+1$ déjà calculées et les solutions à l'itération k pour les autres indices quand on calcule la solution à l'itération $k+1$. Pour la méthode des puissances, au lieu de

$$x^{k+1}[i] = \sum_{j=1}^{n} x^k[j]G[j,i]$$

on obtient pour l'algorithme ascendant (on fait la boucle de i=1 à N) et les valeurs de 1 à $i-1$ sont déjà modifiées quand on calcule la valeur pour l'indice i :

$$x^{k+1}[i](1-G[i,i]) = \sum_{j=1}^{i-1} x^{k+1}[j]G[j,i] + \sum_{j=i+1}^{n} x^{k}[j]G[j,i].$$

Il faut aussi renormaliser le vecteur x car il n'est plus de norme égale à 1.

Proposez une version de Gauss Seidel ascendant efficace en mémoire et en calcul et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez.

Parametres à étudier :

- α

5 Calcul du pagerank par Gauss Seidel-descendant

L'algorithme de Gauss Seidel consiste à mélanger les solutions à l'itération $k+1$ déjà calculées et les solutions à l'itération k pour les autres indices quand on calcule la solution à l'itération $k+1$. Pour la méthode des puissances, au lieu de

$$x^{k+1}[i] = \sum_{j=1}^{n} x^{k}[j]G[j,i]$$

on obtient pour l'algorithme descendant (on fait la boucle de i=N à 1) et les valeurs de N à $i+1$ sont déjà modifiées quand on calcule la valeur pour l'indice i :

$$x^{k+1}[i](1-G[i,i]) = \sum_{j=1}^{i-1} x^{k}[j]G[j,i] + \sum_{j=i+1}^{n} x^{k+1}[j]G[j,i].$$

Il faut aussi renormaliser le vecteur x car il n'est plus de norme égale à 1.

Proposez une version de Gauss Seidel descendant efficace en mémoire et en calcul et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez.

Parametres à étudier :

- α

6 Calcul du pagerank par la méthode SOR (successive over-relaxation) ascendante

La méthode SOR ascendante mélange à chaque itération le résultat de la méthode des puissances et le résultat de la méthode de Gaus Seidel ascendante. L'algorithme de Gauss Seidel consiste à mélanger les solutions à l'itération $k+1$ déjà calculées et les solutions à l'itération k pour les autres indices quand on calcule la solution à l'itération $k+1$. Pour la méthode des puissances, on a

$$y^{k+1}[i] = \sum_{j=1}^{n} x^{k}[j]G[j,i].$$

Pour SOR, on calcule y^{k+1} (ligne précédente) et z^{k+1} (ci dessous)

$$z^{k+1}[i](1-G[i,i]) = \sum_{j=1}^{i-1} x^{k+1}[j]G[j,i] + \sum_{j=i+1}^{n} x^{k}[j]G[j,i].$$

Il faut aussi renormaliser le vecteur z car il n'est plus de norme égale à 1. Dans la méthode SOR, on combine y^{k+1} et z^{k+1} comme suit:

$$x^{k+1}[i] = \omega y^{k+1}[i] + (1-\omega)z^{k+1}[i]$$

Proposez une version de SOR ascendant efficace en mémoire et en calcul et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez pour deux valeurs de ω (0.8 et 1.2).

Parametres à étudier :

- α
- ω

7 Calcul du pagerank par la méthode SOR (successive over-relaxation) descendante

La méthode SOR mélange à chaque itération le résultat de la méthode des puissances et le résultat de la méthode de Gaus Seidel descendante. L'algorithme de Gauss Seidel consiste à mélanger les solutions à l'itération $k+1$ déjà calculées et les solutions à l'itération k pour les autres indices quand on calcule la solution à l'itération $k+1$. Pour la méthode des puissances, on a

$$y^{k+1}[i] = \sum_{j=1}^{n} x^k[j]G[j,i].$$

Pour SOR, on calcule y^{k+1} (ligne précédente) et z^{k+1} (ci dessous)

$$z^{k+1}[i](1-G[i,i]) = \sum_{j=1}^{i-1} x^k[j]G[j,i] + \sum_{j=i+1}^{n} x^{k+1}[j]G[j,i].$$

Il faut aussi renormaliser le vecteur z car il n'est plus de norme égale à 1. Dans la méthode SOR, on combine y^{k+1} et z^{k+1} comme suit:

$$x^{k+1}[i] = \omega y^{k+1}[i] + (1-\omega)z^{k+1}[i]$$

Proposez une version de SOR descendant efficace en mémoire et en calcul et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez pour deux valeurs de ω (0.8 et 1.2).

Parametres à étudier :

- α
- ω

8 Calcul du pagerank par l'algorithme utilisant le vecteur ∇

Soit G la matrice stochastique. On pose $\nabla[j] = \min_i G[i,j]$. C'est un vecteur ligne. De même, on définit un second vecteur ligne par : $\Delta[j] = \max_i G[i,j]$. On construit par itération veux vecteurs lignes X^k et Y^k. L'algorithme suivante converge vers la distribution stationnaire π_G de la chaine de Markov de matrice G. De plus $X^k \leq \pi_G \leq Y^k$ pour tout k (inégalité par élément).

1. Initialiser $X^0 = \nabla$. $Y^0 = \Delta$

2. Faire une boucle sur k
   (a) $X^{k+1} = max(X^k, X^k G + \nabla(1 - ||X^k||_1)$
   (b) $Y^{k+1} = min(Y^k, Y^k G + \nabla(1 - ||Y^k||_1)$
3. jusqu'à ce que $||X^k - Y^k||_1 < \epsilon$.

Proposer une version efficace en mémoire et en calcul de l'algorithme et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez.

Parametres à étudier :

- α

9 Simulation d'un Google Bombing

A partir d'un graphe du web dont vous disposez, ajouter différentes structures de graphes composés d'attaquants et d'une cible (déjà dans le graphe) et faites varier les nombres d'attaquants et la structure du graphe du web entre les attaquants avant de calculer dans chaque cas le pagerank de la cible.

Plus précisemment, vous allez étudier l'impact sur 3 cibles potentielles que vous déterminerez grâce aux pertinences initiales : une cible de pertinence forte, une de pertinence moyenne, et une de pertinence faible. Les structures de graphes que vous ajouterez sont des graphes complets, des anneaux et des sommets isolés. Le parametre que vous ferez varier est la taille du graphe ajouté qui attaque la page cible pour changer sa pertinence.

Essayer de déduire des règles empiriques pour avoir une attaque efficace. On supposera que l'attaque est efficace si la probabilité calculée par pagerank devient significativement plus forte.

Parametres à étudier :

- α
- la taille du graphe attaquant
- la topologie du graphe attaquant
- le type de cible

10 Le backspace pour les pages sans lien de sortie

On suppose que l'on n'utilise pas le modèle du surfer aléatoire pour donner des successeurs aux pages sans liens de sortie. On va supposer que l'utilisateur qui arrive sur une page sans url de sortie utilise la touche backspace pour revenir en arrière dans sa navigation.

Mais comme il y a peut-être plusieurs pages qui pointent sur une page sans lien de sortie, il faut trouver un moyen de se souvenir par où on est arrivé sur cette page. Pour cela, on modifie comme suit le graphe du web.

Chaque page P qui est de degré sortant $Dout$ nul et de degré entrant $Din > 0$ se voit recopié en Din exemplaires.

Son nom devient (P, X) (où X a Din valeurs) et la page X pointe sur la page (P, X) et est la seule à pointer sur cette page. Il est donc facile de savoir comment revenir de (P, X) quand on a visité cette page.

Le but du projet est de comparer les ranking obtenus par l'approche de Google et par cette autre approche. Attention, on garde la seconde modification de la matrice (c'est à dire le mélange avec le coéfficient de 0.85 entre le graphe du web modifié et la matrice du surfer aléatoire).

Si les graphes du web que vous utiliserez n'ont pas de pages de degré sortant nul (on ne sait pas), vous transformerez aléatoirement ces graphes en détruisant des liens de sortie.

Parametres à étudier :

- α

11 Extrapolation dans l'algorithme des puissances en supposant que $\lambda_2 = \alpha$

L'algorithme des puissances utilise le fait que la premiere valeur propre est 1. Par contre on sait aussi que la seconde valeur propre de la matrice Google est de module α (les valeurs propres peuvent être complexes). On va étudier expérimentalement une méthodes d'accélération qui suppose que $\lambda_2 = \alpha$. On suppose que les itérés $x^{(k)}$ vérifient :

$$x^{(k)} = \pi + \beta_2 v_2$$

Donc

$$x^{(k+1)} = x^{(k)}G = \pi G + \beta_2 v_2 G = \pi + \beta_2 v_2 \lambda_2$$

et donc en supposant que $\lambda_2 = \alpha$,

$$x^{(k+1)} = \pi + \beta_2 v_2 \alpha$$

En combinant, pour éliminer le terme en β_2

$$\pi = \frac{x^{(k+1)} - \alpha x^{(k)}}{1 - \alpha}$$

Les itérations suivantes utilisent π pour redémarrer plutôt que que $x^{(k+1)}$ Cette extrapolation peut être effectuée 1 fois toutes les m itérations.

Proposer une version efficace en mémoire et en calcul de l'algorithme et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez .

Parametres à étudier :

- α
- m

12 Extrapolation dans l'algorithme des puissances en supposant que $\lambda_2 = \alpha$ et $\lambda_3 = -\alpha$

L'algorithme des puissances utilise le fait que la premiere valeur propre est 1. Par contre on sait aussi (la preuve est publiée sur le web) que la seconde valeur propre de la matrice Google est de module α (les valeurs propres peuvent être complexes). On va étudier expérimentalement une méthodes d'accélération qui suppose que $\lambda_2 = -\alpha$. On suppose que les itérés $x^{(k)}$ vérifient :

$$x^{(k-2)} = \pi + \beta_2 v_2 + \beta_3 v_3$$

Donc

$$x^{(k)} = x^{(k-2)}G^2 = \pi + \beta_2 v_2 \lambda_2^2 + \beta_3 v_3 \lambda_3^2$$

et donc en supposant que $\lambda_2 = \alpha$, et $\lambda_3 = -\alpha$,

$$x^{(k)} = \pi + (\beta_2 v_2 + \beta_3 v_3)\alpha^2$$

En combinant, pour éliminer le terme en β_2

$$\pi = \frac{x^{(k)} - \alpha^2 x^{(k-2)}}{1 - \alpha^2}$$

Les itérations suivantes utilisent π pour redémarrer plutôt que que $x^{(k+1)}$ Cette extrapolation peut être effectuée 1 fois toutes les m itérations.

Proposer une version efficace en mémoire et en calcul de l'algorithme et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez en faisant varier m. Parametres à étudier :

- α
- m

13 Accélération de Aitken quadratique dans l'algorithme des puissances

Dans la méthode des puissances, la vitesse de convergence est dominée par la seconde valeur propre λ_2. L'accélérateur a pour but d'estimer cette valeur propre de manière à corriger les estimateurs pour converger plus vite (avec la valeur propre suivante qui est plus proche de 0). On suppose que $\lambda_2 > \lambda_3$. Soit β_1, . β_n, la décomposition de $x^{(0)}$ sur la base des vecteurs propres v_1,, v_n. On a

$$x^{(0)} = \sum_{i=1}^{n} \beta_i v_i$$

Supposons que $\beta_1 \neq 0$. On a pour tout k

$$x^{(k)} = \beta_1 v_1 + \sum_{i=2}^{n} \beta_i \lambda_i^k v_i$$

On va considérer 3 termes successifs $(x^{(k)}, x^{(k+1)}, x^{(k+2)}$ pour la version quadratique que l'on calcule par la méthode des puissances. En omettant les termes d'ordre supérieur à deux (et en posant pour tout i, $u_i = \beta_i v_i$), on obtient:

$$\begin{aligned} x^{(k)} &= u_1 + u_2 \lambda_2^k \\ x^{(k+1)} &= u_1 + u_2 \lambda_2^{k+1} \\ x^{(k+2)} &= u_1 + u_2 \lambda_2^{k+2} \end{aligned}$$

On a donc un système à 3 équations et 2 équations u_2 et λ_2 que l'on peut résoudre. Supposons que l'on dispose d'un estimateur des termes λ_2 et $u_2\lambda_2$, on peut périodiquement modifier la valeur de $x^{(k)}$ pour inclure les estimations de $u_2\lambda_2^k$ et donc converger avec une vitesse dépendant de λ_3. La remarque suivante prouve que l'on peut estimer les quantités par la différence entre itérations passées une certaine valeur de k puisque le terme $u_2(\lambda_2^k)(1-\lambda_2)$ devient dominant (rappel : $1 > \lambda_2 > \lambda_3$).

$$x^{(k)} - x^{(k+1)} = u_2(\lambda_2^k)(1-\lambda_2) + \sum_{i=3}^{n} u_i(\lambda_i^k)(1-\lambda_i)$$

Proposer une version efficace en mémoire et en calcul de l'algorithme et comparez expérimentalement avec la méthode des puissances sur les graphes du web dont vous disposez.

14 Sujet pour un groupe de 2

Le projet a pour but de tester l'algorithme de Pagerank lorsque la matrice du WEB est NCD et que α s'approche de 1. Comme on est pas sûr que les graphes du WEB fournis sont NCD, on va les modifier. Le sujet se décompose comme suit:

1. Essayer de rendre les matrices du WEB décomposables en enlevant des arcs. On propose de faire deux expériences : enlever 10 pourcents puis 20 pourcents des arcs des graphes du web. Vous êtes libre de le faire de facon aléatoire (avec un tirage aléatoire) ou déterministe (on enleve un arc sur 10 ou sur 5). Si vous choisissez un tirage aléatoire, la méthode est la suivante :
   - Vous parcourez le fichier du graphe
   - Pour chaque arc, vous faites un tirage aléatoire u
   - Si $u < 0.1$ (ou 0.2), vous enlevez l'arc sinon vous le gardez
   - N'oubliez pas de mettre à jour le degré sortant du sommet en début de ligne sur le fichier.
2. Tester l'algorithme de Pagerank avec une valeur α de 0.85 (comme pour Google 98) ou une valeur plus proche de 1 (je propose 0.99). Pour chacun des graphes modifiés, donner le temps de convergence et le nombre d'itérations pour l'algorithme.