Projet : Algorithmes de ranking et de recommandations

Essayer de faire des groupes de taille 4 et de vous répartir les sujets de manière à ce que tous les sujets soient pris. En cas de blocage, je procéderai à un tirage au sort pour répartir les sujets conflictuels. Les graphes du web sur lesquels vous devrez travailler sont sur le site de l'ENT.

1 Projet 1 : Utilisation du pagerank précédent pour initialiser un nouveau calcul

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 et 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, et soit R les sommets du graphe H^t qui ne sont pas dans le graphe H^{t+1}. S c'est les sommets nouveaux et R les sommets disparus.

- 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 $x_{t+1}[i] = \frac{\sum_{j \in R} x_t[j]}{|S|}$

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 (ajout ou retrait de sommets, et ajout d'arcs depuis les nouvelles pages vers les anciennes) dont vous disposez et vous étudierez l'influence des modifications sur l'accélération.

2 Projet 2 : Calcul du pagerank par Gauss Seidel

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

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

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

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

3 Projet 3 : 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.

4 Projet 4 : 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:

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

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.

5 Projet 5 : 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.

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

La méthode SOR mélange à chaque itération le résultat de la méthode des puissances et le résultat de la méthode de Gaus Seidel. 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 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).

7 Projet 7 : 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.

8 Projet 8 : structure NCD, approche de Courtois

Ce projet utilise une structure de bloc de la matrice G associée aux éléments les plus représentatifs de la matrice. On va construire une partition des pages et se servir de cette partition pour résoudre le calcul des pertinences.

On définit G_ϵ comme suit : si un élément de la matrice G est inférieur à ϵ alors on met cette entrée à 0 dans G_ϵ, si il est supérieur à ϵ, on le laisse inchangé dans G_ϵ.

Remarque : si on prend $\epsilon > 1/n$, G_ϵ ne contient aucun arc ajouté par le surfer aléatoire et certains arcs du graphe du web ont pu aussi disparaitre. Dans la pratique on va prendre $\epsilon = 10^{-3}$.

Une fois que G_ϵ est obtenue, on recherche les composantes connexes de G_ϵ et on partitione selon ces composantes connexes. Soit B le nombre de composante connexes. Si on suppose que la matrice G a été re-ordonnée selon cette partition, on obtient (tous les blocs sont non nuls):

$$G = \left[\begin{array}{c|c|c|c|c} P_{1,1} & P_{1,2} & ... & ... & P_{1,B} \\ \hline P_{2,1} & P_{2,2} & P_{2,3} & ... & P_{2,B} \\ \hline & & & & \\ \hline ... & & P_{i,i} & & ... \\ \hline & & & & \\ \hline P_{N,1} & P_{N,2} & ... & ... & P_{B,B} \end{array}\right]$$

où les blocs $P_{i,j}$ (avec $j \neq i$) sont de norme petite devant la norme de $P_{i,i}$. Cette matrice possède la propriété NCD.

Pour ces matrices, l'algorithme de Courtois donne rapidement par décomposition une approximation du vecteur de pertinence. L'algorithme procède comme suit en supposant que chacun des blocs $P_{i,j}$ a été identifié:

1. Résoudre pour chacun des blocs diagonaux $P_{i,i}$ (donc il y a B résolutions) le problème :

   $$\lambda_i \pi_i = \pi_i P_{i,i},$$

   où λ_i est la valeur propre dominante de $P_{i,i}$ et π_1 est un vecteur de probabilités (il somme à 1). Comme $P_{i,i}$ n'est pas une matrice stochastique, λ_i ne vaut pas 1.

2. Calculer la matrice de couplage A, c'est une matrice de taille $B \times B$ défini comme suit :

   $$A[i,j] = (\sum_m \sum_k P_{i,j}[m,k]\pi_i[m])$$

3. Calculer le vecteur de probabiiltés β solution de $\beta A = \beta$ (votre algorithme initial de la méthode des puissances est suffisant)

4. L'approximation π est

   $$\pi = (\beta(1)\pi_1, \beta(2)\pi_2, ..., \beta(B)\pi_B).$$

   Dans cette formule, on concatène les petits vecteurs.

Comparez la précision et la vitesse par rapport à votre algorithme de Pagerank.

9 Projet 9 : structure NCD, approche KMS simplifiée

Ce projet utilise une structure de bloc de la matrice G associée aux éléments les plus représentatifs de la matrice. On va construire une partition des pages et se servir de cette partition pour résoudre le calcul des pertinences.

On définit G_ϵ comme suit : si un élément de la matrice G est inférieur à ϵ alors on met cette entrée à 0 dans G_ϵ, si il est supérieur à ϵ, on le laisse inchangé dans G_ϵ.

Remarque : si on prend $\epsilon > 1/n$, G_ϵ ne contient aucun arc ajouté par le surfer aléatoire et certains arcs du graphe du web ont pu aussi disparaitre. Dans la pratique on va prendre $\epsilon = 10^{-3}$.

Une fois que G_ϵ est obtenue, on recherche les composantes connexes de G_ϵ et on partitione selon ces composantes connexes. Soit B le nombre de composante connexes. Soit n_i la taille du sous ensemble S_i de la partition Si on suppose que la matrice G a été re-ordonnée selon cette partition, on obtient (tous les blocs sont non nuls):

$$G = \left[\begin{array}{c|c|c|c|c} P_{1,1} & P_{1,2} & ... & ... & P_{1,B} \\ \hline P_{2,1} & P_{2,2} & P_{2,3} & ... & P_{2,B} \\ \hline & & & & \\ \hline ... & & P_{i,i} & & ... \\ \hline & & & & \\ \hline P_{N,1} & P_{N,2} & ... & ... & P_{B,B} \end{array}\right]$$

où les blocs $P_{i,j}$ (avec $j \neq i$) sont de norme petite devant la norme de $P_{i,i}$. Cette matrice possède la propriété NCD.

Pour ces matrices, l'algorithme de KMS donne rapidement par décomposition et itération une valeur du vecteur de pertinence. Comme l'algortihme est iteratif, l'initialisation est simple. L'algorithme procède comme suit en supposant que chacun des blocs $P_{i,j}$ a été identifié:

1. Pour chacun des blocs diagonaux $P_{i,i}$ (donc il y a B initialisations différentes), initialiser

$$\pi_i[k] = 1/n_i,$$

2. Calculer la matrice de couplage A, c'est une matrice de taille $B \times B$ défini comme suit :

$$A[i,j] = (\sum_m \sum_k P_{i,j}[m,k]\pi_i[m])$$

3. Calculer le vecteur de probabiiltés β solution de $\beta A = \beta$ (votre algorithme initial de la méthode des puissances est suffisant)

4. A cette itération k $\alpha^{(k)}$ est

$$\alpha^{(k)} = (\beta(1)\pi_1, \beta(2)\pi_2, ..., \beta(B)\pi_B).$$

   Dans cette formule, on concatène les petits vecteurs.

5. puis on fait une iteration la méthode des puissances

$$\pi^{(k)} = \alpha^{(k)} P.$$

6. Faites un test de convergence entre $\pi^{(k)}$ et $\pi^{(k-1)}$. Si il n'y a pas convergence, recalculez les nouvelles valeurs de $\pi_i[m]$ à partir de $\pi^{(k)}$ et retourner en 2). La formule de calcul est pour tout sommet m qui est dans le sous enesmble i

$$\pi_i[m] = \frac{\pi^{(k)}[m]}{\sum_{p \in S_i} \pi^{(k)}[p]}$$

Comparez la précision et la vitesse par rapport à votre algorithme de Pagerank.

10 Projet 10 : Méthode d'élimination

La méthode a pour but de diminuer le nombre d'éléments non nuls dans la matrice du Web pour ensuite diminuer la complexité du produit matrice vecteur. La constatation empirique sur les graphes su web est que de nombreux sommets ont les mêmes liens de sortie et que beaucoup de sommets ont aussi les mêmes prédécesseurs. On utilise cette propriété. Si les graphes du web utilisées dans le projet ne vérifient pas cet propriété, vous pouvez modifiez les graphes en changeant les probabilités.

L'algorithme propose donc d'agréger deux colonnes égales. On peut constater que si l'on ajoute les jeux colonnes (et les deux lignes) on obtient une matrice plus petite. La pertinence des sommets qui sont agrégés est la somme des pertinences.

Il faut comparer l'algorithme sur le graphe initial au même algorithme mais sur le graphe agrégé. Pour eviter de tester toutes les couples de colonnes, on va chercher à agréger les colonnes qui contiennent d'éléments. On procède comme suit :

1. Calculer le degré entrant de chaque noeud (le nombre d'éléments dans la colonne)
2. Trier le graphe en mettant en tete les sommets de degré elevé.
3. Pour des sommets de degré élevé (valeur à choisir), vérifier si les colonnes sont égales et si oui les fusionner (et fusionner les lignes).

Comparer le temps initial (Pagerank sur le graphe initial) à l'approche consistant à agréger le graphe avant de l'analyser.

11 Projet 11 : Accélération due à la trace dans l'algorithme des puissances

L'algorithme procède comme suit (je ne vous donne pas les justifications mathématiques). C'est un algorithme de restart : on recommence les calculs en changeant la valeur initiale.

1. Prendre un vecteur initial $\pi^{(0)} = e/n$.
2. Calculez l le nombre de sommets sans successeurs dans le graphe du Web. Définir $\mu = 1+\alpha(l/n-1)$.
3. Faire la méthode des puissances pendant m itérations
4. Si il y a convergence (norme de $\pi^{(m)} - \pi^{(m-1)}$ inférieure à ϵ), l'algorithme est terminé et la solution est $\pi^{(m)}$
5. Si non, on va recommencer avec une nouvelle valeur initiale $\pi^{(0)}$ On pose $\pi^{(0)} = \pi^{(m)} - \pi^{(m-1)(\mu-1)}$ et on renormalise $\pi^{(0)}$ pour que cela soit une distribution de probabilités.
6. Si la norme de $\pi^{(m)} - \pi^{(0)}$ est inférieure à ϵ l'algorithme est terminé et la solution est $\pi^{(0)}$
7. Sinon recommencer en 3)

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. Le choix de m vous est laissé.