Recherche d'Information

M2 DAC

TME 5. Collections Structurées

Au cours de ce TME, nous nous interessons à la structure des collections considérées. Faisant le constat que le contenu des pages n'est parfois pas suffisant pour répondre de manière efficace à un besoin d'information, l'idée est de considérer les relations entre documents pour établir des modèles plus performants et plus robustes. L'idée est de se baser sur les hyperliens des pages Web pour caractériser l'importance (ou la centralité) de certaines pages par rapport à la thématique de la requête utilisateur. Le travail à réaliser est d'implémenter et d'expérimenter un modèle de recherche basé sur la structure de la collection : on implémentera des modèles basés sur *PageRank* et *HITS* vus en cours.

1 Indexation des hyperliens

Afin d'être à même de pouvoir considérer les relations entre documents du corpus considéré, les hyperliens présents dans ceux-ci doivent être extraits au moment de l'indexation de la collection. Il s'agit alors dans un premier temps de reprendre le processus d'indexation défini en TME1 pour y ajouter une prise en compte des relations entre documents.

Selon l'architecture définie, il suffit de reprendre la méthode *getDocument* du parser spécifique à nos collections pour y inclure la considération des éléments de la balise .X et d'ajouter ces informations dans l'index. Chaque ligne de la balise .X contient plusieurs éléments. Seul le premier nous interesse: il correspond à l'identifiant d'une page pointée par un lien du document courant.

2 Marche aléatoire sur les graphes

Les algorithmes *PageRank* et *Hits* sont des algorithmes à base de marche aléatoire sur les graphes. Considérant que les relations entre les documents forment un graphe $G = (V, E)$, avec V les documents de la collection et E un ensemble de liens orientés (hyperliens) entre ces documents, ces algorithmes cherchent à caractériser l'importance

de chacune des pages en fonction de leur entourage, et plus précisément en fonction de l'importance des pages qui leur sont liées.
Au cours de ce TME, il convient de mettre en place les deux algorithmes *PageRank* et *Hits* décrits ci-dessous dans deux classes distinctes étendant une classe abstraite *RandomWalk* qui définit une méthode de marche aléatoire dans un graphe de documents passé en paramètre (contenu dans une instance d'Index par exemple).

2.1 Algorithme *PageRank*

L'algorithme *PageRank* s'appuie sur l'hypothèse qu'une page A pointe vers une page B si B contient une information pertinente par rapport à la page A. Une page ayant beaucoup de liens *entrants* est donc susceptible de contenir une information plus importante qu'une page isolée. Néanmoins, cette notion d'importance doit être robuste à la *manipulation* : essayer d'augmenter artificiellement le score doit être coûteux (dans l'idéal : coût de la manipulation >> gain).
Plutôt que de considérer directement le nombre de liens entrants (qui est donc facilement manipulable), *PageRank* propose alors une définition récursive de l'importance des pages: une page A est importante si on a beaucoup de chance d'arriver sur A en partant de pages importantes. *PageRank* simule le comportement d'un surfeur aléatoire:

1. au temps t, le sufeur est sur la page $p_t \in V$,
2. avec une probabilité d, il clique aléatoirement sur un lien de p_t. p_{t+1} est alors la page pointée par ce lien.
3. avec une probabilité $1-d$, p_{t+1} est une page Web choisie aléatoirement.
4. retour à l'étape 1.

Avec $P_i = \{j \in V | (j \to i) \in E\}$ l'ensemble des pages pointant vers la page $i \in V$ et ℓ_j le nombre de liens sortant de la page j, la probabilité μ_i^t que le surfeur soit sur la page i au temps t est alors définie par :

$$\mu^{t+1} = \frac{1-d}{N}\mathbf{1} + dA\mu^t \quad \text{avec} \quad A_{ij} = \begin{cases} \frac{1}{\ell_j} \text{ si } j \in P_i \\ 0 \text{ sinon} \end{cases} \quad \text{et} \quad N : \#\text{pages Web.}$$

Si on pose $\mu_i^0 = \frac{1}{N}$ pour tout $i \in V$ et que l'on choisit $1 \geq d > 0$, l'algorithme converge vers une solution unique stable. L'importance des pages est alors caractérisée par le vecteur final μ.

2.2 Algorithme HITS

L'intuition derrière *HITS* est que si le besoin d'information est vague les mots ne sont pas suffisants pour discriminer correctement les pages Web, mais que si une page fait référence sur un domaine, il est probable que beaucoup de pages traitant du sujet aient un lien pointant vers cette page et qu'il est alors possible de se servir de cette information pour améliorer les modèles de recherche. Le principe de *HITS* repose sur l'identification de deux types de documents:

1. Les *autorités* : des pages fréquemment citées comme référence sur un sujet;
2. Les *hubs* : des pages qui ont des liens vers de nombreuses références.

L'identification de ces deux types de documents se fait au moyen de deux scores d'autorité a_i et de hub h_i se renforçant mutuellement pour les différents noeuds i du graphe G. Pour chaque noeuds i du graphe et pour chaque itération t de l'algorithme, $HITS$ met à jour ces scores a_i^t et h_i^t de la manière suivante:

- $a_i^t = \sum_{j, j \to i \in E} h_j^{t-1}$
  (Un noeud est considéré comme une bonne autorité si il est pointé par des noeuds reconnus comme de bons hubs)
- $h_i^t = \sum_{j, i \to j \in E} a_j^{t-1}$
  (Un noeud est considéré comme un bon hub si il pointe vers des noeuds reconnus comme des bonnes autorités)

avec $a_i^0 = 1$ et $h_i^0 = 1$ pour tous les noeuds i du graphe.
Afin d'assurer la convergence de l'algorithme, les vecteurs de scores a et h sont normalisés (au sens de la norme L2) à chaque itération de l'algorithme. L'importance des pages est alors caractérisée par le vecteur final a.

3 Modèle de Recherche

Les algorithmes *PageRank* et *Hits* peuvent selon les cas, être appliqués au graphe de la collection entière (dans ce cas on pourra utiliser leurs scores combinés à des scores dépendant de la requête) ou bien être utilisés directement comme modèles de recherche appliqués à un graphe résultant d'une recherche orientée requête préliminaire. Il s'agit ici de mettre en oeuvre cette deuxième approche pour définir des modèles de recherche documentaire basés sur ces algorithmes à base de marche aléatoire sur des graphes.
Pour mettre en place ce genre de modèle, la première étape est alors la détermination d'un sous-graphe de documents candidats. Il s'agit alors pour chaque requête Q formulée par l'utilisateur de déterminer le sous-graphe $G_Q = (V_Q, E_Q) \subseteq G = (V, E)$ des documents candidats sur lequel nous allons lancer un algorithme de marche aléatoire. Cela se fait en deux temps:

1. Initialisation de V_Q avec un ensemble S de documents *seeds*: les n premiers documents retournés par un modèle sur le contenu donné;
2. Pour chaque document $D \in S$, ajout dans V_Q de tous les documents pointés par D et de k documents choisis aléatoirement parmi ceux pointant vers D.

Trois paramètres sont donc à définir pour mettre en place le modèle:

- Le modèle de base permettant de récupérer les documents *seeds*;
- Le paramètre n déterminant le nombre de documents *seeds* à considérer;

- Le paramètre k déterminant le nombre de liens entrants à considérer pour chaque document *seed*.

Une fois le sous-graphe extrait pour la requête de l'utilisateur, il s'agit d'appliquer un de nos deux algorithmes de marche aléatoire et ordonner les documents selon l'importance qui leur a été attribuée dans ce contexte.