Propagation de signatures lexicales dans le graphe du Web

Propagation of lexical signatures in the Web graph.

M. Bouklit[1] M. Lafourcade[2]

[1] Algorithmique et Combinatoire.
Laboratoire d'Informatique Algorithmique, Fondements et Applications.
Université Denis Diderot (case courrier 7014).
2, place Jussieu. 75251 Paris cedex 5 - France
[2] Traitements Algorithmiques du Langage.
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier.
161, rue Ada. 34392 Montpellier Cedex 5 - France

bouklit@liafa.jussieu.fr
lafourcade@lirmm.fr

Résumé

L'analyse du graphe formé par les pages web et les liens hypertextes qui les relient, communément appelé graphe du Web, a permis d'améliorer la performance des moteurs de recherche actuels. Ainsi, lancé en 1998, le moteur de recherche Google classe les pages grâce à la combinaison de plusieurs facteurs dont le principal porte le nom de PageRank. Nous présentons dans cet article l'algorithme LexicalRank propageant deux signatures lexicales : l'une interne, l'autre externe. Une signature lexicale est un ensemble de termes pondérés décrivant une page.

Mots Clef

Recherche d'informations, graphe du Web, PageRank, moteur de recherche, signature lexicale.

Abstract

Theorical analysis of the Web graph is often used to improve the efficience of search engines. The PageRank algorithm, proposed by Brin and Page in 1998 is used by Google search engine to improve the results of requests. In this paper, we present the LexicalRank algorithm which propagates two lexical signatures : one interns, the other external one. A lexical signature is a set of weighted terms describing a page.

Keywords

Information Retrieval, Web graph, PageRank, search engine, lexical signature.

1 Introduction

Le Web (ensemble des pages hypertextes disponibles sur Internet) est devenu une partie intégrante de la vie quotidienne de millions de gens. La nature même des médias électroniques, ainsi que la volonté de ses inventeurs [BLCL+94] lui ont donné une nature *hypertexte* : les documents sont structurés en *pages*, qui se *pointent* les unes vers les autres, par un système de références.

La croissance exponentielle du Web rend problématique l'appréhension de sa structure globale. Pourtant, une connaissance du contenu et de la structure du Web est indispensable pour réaliser de nombreuses tâches essentielles à la vie de l'internaute, telles que la *recherche d'informations* (où trouver une page sur tel sujet ?) ou la *mesure d'audience* (ma page est-elle populaire ?).

C'est pourquoi les moteurs de recherche ont développé des méthodes de tri automatique des résultats. Leur but est d'afficher dans les dix à vingt premières réponses les documents répondant le mieux à la question. Dans la pratique, aucune méthode de tri n'est parfaite, d'autant plus que la question de la justesse d'un classement est en grande partie subjective. Un classement est justifié au mieux par un sondage, le plus souvent au jugement du lecteur. Cependant, la variété des méthodes offre à l'usager la possibilité de traquer l'information de différentes manières : cette variété augmente donc ses chances d'améliorer ses recherches.
La suite de l'article est organisée comme suit. La section 2 décrit tout d'abord quelques méthodes de tri automatique des résultats comme *PageRank*, une mesure de popularité des pages Web. La section 3 introduit notre modèle *LexicalRank* et l'algorithme qui en est déduit. *LexicalRank*

propage dans le Web deux signatures lexicales : l'une interne, l'autre externe. Rappelons qu'une signature lexicale est un ensemble de termes pondérés décrivant une page. Nous pensons que l'utilisation de ces signatures permettront d'améliorer la performance des moteurs de recherche. La section 4 présente enfin les résultats issus de nos expérimentations.

2 Méthodes de tri automatique des résultats

2.1 Tri par contenu

La méthode de tri la plus ancienne et la plus utilisée est la méthode de tri par contenu : on la trouvait dans les moteurs Voila, Lycos, AltaVista, Excite, InfoSeek, ... Elle est basée sur le nombre d'occurrences des termes de la recherche dans les pages, de leur proximité, de leur place dans le texte [Sal89, YLYL95].

Malheureusement, cette méthode présente l'inconvénient d'être facile à détourner par des auteurs désireux de placer leurs pages en tête de liste : pour cela, il suffit de répéter les mots importants soit dans l'en-tête, soit dans le texte en utilisant des techniques de *spamming* (écrire le texte en blanc sur fond blanc par exemple) pour modifier à son avantage le classement.

2.2 PageRank

Les limites du tri par contenu ont alors conduit à rechercher, à partir de principes tout à fait différents, d'autres méthodes complémentaires indépendantes du contenu des documents. C'est dans ce contexte que sont apparues des méthodes de tri basées sur une notion de popularité.
En 1998, Sergei Brin et Larry Page alors étudiants en thèse à l'Université Stanford mettent au point une méthode qui va révolutionner le Web [PBMW98]. Cette méthode consiste à estimer la popularité des pages web en se servant de la structure induite par les pages web et les liens hypertextes qui les relient communément appelé *graphe du Web*. Plus précisément, elle classe les pages en utilisant un indice numérique (le «rang») calculé globalement pour chaque page d'où le nom *PageRank*. Ce rang donne en fait une *bonne* estimation de la popularité de la page. C'est ce même rang qui permettra en particulier d'ordonner les résultats d'une requête d'un usager. Quelques mois plus tard, le moteur de recherche Google [Goo98] voit le jour ...

Dans la suite, nous appellerons $G = (V, E)$ le graphe formé par les page web V et les liens hypertextes qui les relient E. N représentera le nombre de pages de V. En pratique, G est principalement obtenu par une succession de parcours du Web (*crawls*). En effet, il y a en amont du processus les *robots* qui chalutent continuellement le Web dans l'intention de découvrir de nouvelles pages et à défaut de mettre à jour les anciennes. Ces pages sont stockées dans un entrepôt de données. Viennent ensuite les hyperliens[1] qui sont stockés séparément pour former un sous graphe du Web[CGMP98]. Ce graphe est alors utilisé pour le calcul des rangs de page.

Le surfeur aléatoire. L'axiome caché derrière l'algorithme de PageRank est assez étrange, voire peu flatteur pour les internautes. Il dit que les pages les plus intéressantes sont celles sur lesquelles on a le plus de chance de tomber en cliquant au hasard. Exprimé autrement, le cerveau est un outil secondaire quand il s'agit de trouver les «bonnes» pages web. Toutes les variantes de PageRank peuvent s'interpréter comme un *surfeur aléatoire*, censé modéliser un internaute lambda, dont le comportement, bien qu'aléatoire, est soumis à certaines règles qui définissent la variante.

Le plus souvent, ces règles se traduisent par un processus stochastique de type markovien. A partir d'une distribution initiale de probabilité sur l'ensemble des pages web, le processus est itéré et, sous réserves de garanties de convergence et d'unicité de la limite, tend vers une distribution de probabilité qui est par définition le PageRank de la variante en question. On comprend donc qu'il existe en réalité une multitude de PageRanks même si on parle souvent du PageRank au singulier[BP98].

Modèle initial. Le niveau zéro du *surfeur aléatoire*, proposé par [PBMW98], suppose que notre internaute, quand il est sur une page donnée, va ensuite cliquer de manière équiprobable sur un des liens sortants.
Si $R_{n+1}(p)$ représente la probabilité de présence de notre surfeur à l'instant $n+1$ sur la page p, l'équation de propagation du rang s'écrit donc :

$$R_{n+1}(p) = \sum_{q \to p} \frac{R_n(q)}{deg(q)} \tag{1}$$

où $q \to p$ désigne "q pointe sur p" et où $deg(q)$ est le degré externe de q.

Vectoriellement, si on appelle M la matrice d'adjacence de G, et $A_{i,j} = \frac{M_{i,j}}{deg(i)}$ (par convention $0 = \frac{0}{0}$), l'équation de propagation se formule ainsi :

$$R_{n+1} = A^t R_n \tag{2}$$

Rechercher une distribution de probabilité sur V vérifiant (1) revient à trouver la distribution asymptotique de la chaîne de Markov homogène dont la matrice de transition est A. Si A est apériodique et irréductible[2], il est bien connu [SC96] que le processus itératif (2) converge géométriquement vers une distribution de probabilité R vérifiant (1) quelque soit la distribution de probabilité initiale Z. Dans ce cas, la matrice A est *stochastique* car c'est une matrice positive dont la somme de chacune des lignes vaut

[1] Notons que [PBMW98] ignore les ancres pour faciliter le calcul du PageRank.

[2] Une matrice est dite *irréductible* si son graphe est fortement connexe, et *apériodique* si le p.g.c.d. des longueurs des circuits est 1.

1. C'est précisément cette distribution de probabilité R obtenue par l'algorithme 1 que l'on appelle *PageRank*.
La norme ℓ_1 d'une matrice M (notée $\| M \|_1$) désigne l'application de $\mathcal{M}_{n,p}(\mathbb{K})$, par :

$$\| M \|_1 = \sum_{\substack{1 \le i \le n \\ 1 \le j \le p}} |M(i,j)|$$

où $\mathbb{K}$ désigne $\mathbb{R}$ ou $\mathbb{C}$.

Algorithme 1: PageRank : modèle originel [PBMW98]

Données
– une matrice irréductible et apériodique A ;
– une distribution de probabilité Z ;
– un réel ϵ.
Résultat
Le vecteur propre principal de probabilité de A^T avec une précision ϵ.
début
  $R_0 = Z$
  répéter
    $R_{n+1} = A^T R_n$
    $\delta = \|R_{n+1} - R_n\|_1$
  jusqu'à $\delta < \epsilon$
fin

Le facteur *zap*. Un graphe du Web n'est en général pas fortement connexe. En appliquant l'algorithme 1, [PBMW98] ont constaté que le PageRank remonte dans les composantes fortement connexes terminales d'où le nom de *puits de rang*. En plus des feuilles, il existe donc de nombreux puits de rang dont on ne peut pas sortir en cliquant [BKM+00]. Pour échapper aux puits de rang et aux feuilles, il est nécessaire «de temp en temps» de sauter aléatoirement vers une page quelconque du Web.

Méthode du rang par défaut. Pour modéliser les sauts aléatoires, [PBMW98] proposent de doter chaque page d'un rang par défaut appelé *source de rang*. Ainsi chaque page p se voit attribuer un rang de $Z(p) \geq 0$. (1) devient alors :

$$R_{n+1}(p) = \sum_{q \to p} \frac{R_n(q)}{deg(q)} + Z(p) \qquad (3)$$

L'écriture vectorielle de cette équation est $R_{n+1} = A^T R_n + Z$. Quand $||Z||_1 = 1$, Z représente une loi de distribution sur l'ensemble des pages de V appelée distribution de *zap*. Le plus souvent, on choisit pour Z une loi de distribution uniforme : $\forall p \in V$, $Z(p) = \frac{1}{|V|}$. Mais il a été proposé que cette distribution puisse être «personnalisée» [BMPW98].

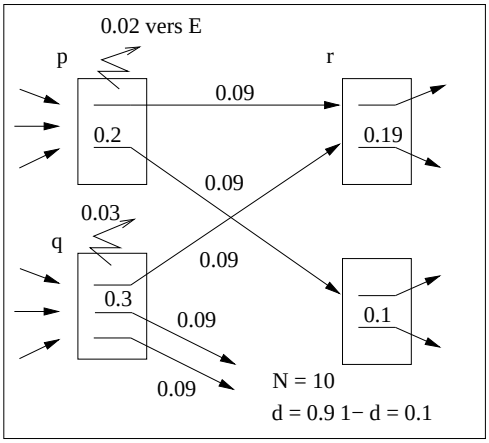

Figure 1: propagation de rang

Facteur d'amortissement. Nous supposerons ici que la matrice A est stochastique. L'équation (3) n'admet pas d'interprétation probabiliste directe. [BP98] propose une variante empirique du modèle en introduisant un facteur d'amortissement $d \in]0,1[$, ce qui donne :

$$R_{n+1}(p) = d \times \left(\sum_{q \to p} \frac{R_n(q)}{deg(q)} \right) + (1-d) Z(p) \qquad (4)$$

avec pour condition $\sum_{p \in V} Z(p) = 1$ et $\forall n$, $\sum_{p \in V} R_n(p) = 1$.

La figure 1 illustre une propagation de rang d'une paire de pages à l'autre. On y suppose $d = 0.9$ et $N = 10$. En observant la page p sur cette figure, nous remarquons que :
– d =90 % de son rang (soit 0.18) est redistribué équitablement sur ses liens sortants (soit $\frac{0.18}{2} = 0.09$) affectant ainsi le rang des pages pointés par p.
– $1 - d$ =10 % de son rang (soit 0.02) est dissipée au profit d'une répartition globale sur l'ensemble du graphe contribuant ainsi à alimenter chaque page d'un rang égal à $\frac{1-d}{N} = \frac{0.1}{10} = 0.01$.
Nous pouvons vérifier par exemple que le rang de la page r est bien 0.19 :

$$\begin{aligned} R_{n+1}(r) &= d\frac{R_n(p)}{d^+(p)} + d\frac{R_n(q)}{d^+(q)} + \frac{1-d}{N} \\ &= 0.9\frac{0.2}{2} + 0.9\frac{0.3}{3} + \frac{0.1}{10} \\ &= 0.19\,. \end{aligned}$$

L'écriture vectorielle de cette équation est :

$$R_{n+1} = dA^t R_n + (1-d)Z \qquad (5)$$

On espère approcher asymptotiquement la valeur de R vérifiant l'équation de conservation :

$$R = dA^T R + (1-d)Z \qquad (6)$$

Algorithme 2: PageRank : modèle par ajout d'un facteur *zap* [BP98]

Données
- une matrice stochastique A ;
- une distribution de *zap* recouvrante Z ;
- un coefficient de *zap* $d \in]0,1[$;
- un réel ϵ.

Résultat
Le vecteur propre de probabilité R de $\hat{A^T}$ associé à la valeur propre maximale à une précision ϵ.

début
  $R_0 = Z$
  répéter
    $R_{n+1} = d.A^T R_n + (1-d).Z$
    $\delta = \|R_{n+1} - R_n\|_1$
  jusqu'à $\delta < \epsilon$
fin

Soit la matrice $\hat{A} = dA + (1-d)\mathbf{1}^T \times Z^T$ où **1** désigne le vecteur ligne ne contenant que des 1. Remarquons que l'équation (6) peut être reformulée comme suit : $R = \hat{A}^T R$. En effet, comme $\sum_{p \in V} R(p) = 1$, on obtient alors : $\mathbf{1} \times R = I_1$.
Il en découle que : $Z = Z \times \mathbf{1} \times R$
Puisque nous avons supposé que A est stochastique alors dans ce cas R est un vecteur propre de la matrice $\hat{A^T}$ pour la valeur propre 1.
On obtient ainsi l'algorithme 2. Observons que l'initialisation R_0 du processus est égal à la distribution Z "par défaut" mais peut être choisie autrement. Par exemple, prendre le résultat d'un calcul précédent peut souvent accélerer la convergence. Cette remarque s'applique à la méthode précédente dite de rang par défaut, même si Z y a l'interprétation de «rang par défaut», voir (3). Le plus souvent, on imposera à toute distribution de *zap* d'être *recouvrante* : on garantit ainsi que toutes les pages connues sont potentiellement accessibles après un *zap*. Par exemple, la distribution uniforme est bien évidemment uniforme. Il est en de même de la distribution sur les pages d'acceuil à la seule condition que les pages connues d'un site soient accessibles à partir de la page d'acceuil.

Modèle du surfeur aléatoire. La définition de l'équation (5) peut s'interpréter dans un modèle de surfeur aléatoire plus évolué comme suit. A chaque étape, notre internaute lambda a la possibilité sur une page :

1. soit de *cliquer* sur l'un des liens sortants ($A^T R$) avec la probabilité *d*.
2. soit de *zapper*, avec la probabilité $1-d$ cette fois, sur une page choisie aléatoirement selon la distribution de Z.

Choix de d. Depuis l'article originel [PBMW98], 0.85, a toujours été une valeur de référence. Selon [Hav99], l'introduction du paramètre d est destinée à améliorer la «qualité» du PageRank en garantissant la convergence vers un unique vecteur rang. La matrice A est explicitement supposée stochastique en éliminant les pages sans liens. Le modèle probabiliste de l'internaute que nous décrivons comme support (implicite) au modèle PageRank prédit que le nombre de «clics» consécutifs suit une distribution géométrique de raison *d*. En particulier, la longueur moyenne d'un chemin entre deux *zap* vaut

$$\sum_{k=0}^{\infty} k d^k (1-d) = \sum_{k=1}^{\infty} d^k = \frac{d}{1-d}$$

Cela pourrait donner une façon empirique de trouver *d*. Pour $d = 0.85$, on en déduit une longueur moyenne entre deux zaps successifs d'environ 5.67. A titre de comparaison, différentes études donnent suivant l'époque et la méthode employée, des nombres variant entre 3 et 10 [CP95, WM04]. Plus supprenant, [MFJR+04] estime cette moyenne à 5,6 ...

3 Description du modèle

Nous présentons dans cette section l'algorithme *LexicalRank* qui propage dans le graphe G deux signatures lexicales : une interne et une externe. Nous espérons ainsi ouvrir la voie à une nouvelle famille d'algorithmes PageRank fondés sur la propagation de signatures lexicales.

Définition 1 (signature lexicale). *Une* signature lexicale *d'une page est un ensemble de termes décrivant une page.*

Formellement, la signature lexicale $S(p)$ d'une page p est un ensemble de termes pondérés permettant de caractériser thématiquement cette page : $S(p) = \{(t_1; p_1), (t_2; p_2), \ldots, (t_k; p_k)\}$
Par exemple, on peut avoir comme signature lexicale (d'une page hypothétique p parlant des différents sens de bottes) :

$S(p)$ = {(‹*chaussure*› ;0,9), (‹*assemblage*› ;0,68), (‹*végétaux*› ;0,4), (‹*coup*› ;0,35), (‹*réunion*› ;0,32), (‹*escrime*› ;0,2), (‹*fleuret*› ;0,14), (‹*épée*› ;0,13), (‹*balle*› ;0,09), (‹*bottine*› ;0,8), (‹*pied*› ;0,78), (‹*touffe*› ;0,05), (‹*attaque*› ;0,04), (‹*jambe*› ;0,04)}

On remarquera que cette signature ne comporte que des termes proches thématiquement des trois principales acceptions de ‹*botte*› : la chaussure, le coup et la réunion de végétaux.
Nous associons à chaque page deux signatures lexicales : l'une interne, l'autre externe.

Définition 2 (signature interne). *La* signature interne *d'une page p (notée $I(p)$) est la signature lexicale que souhaite donner l'auteur de la page p.*

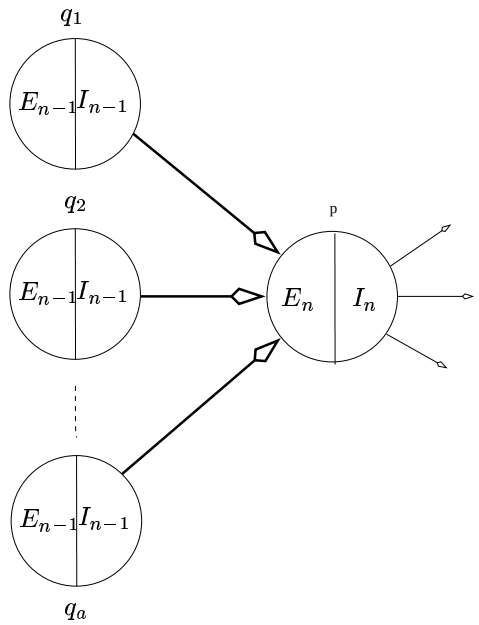

Figure 2: propagation avant

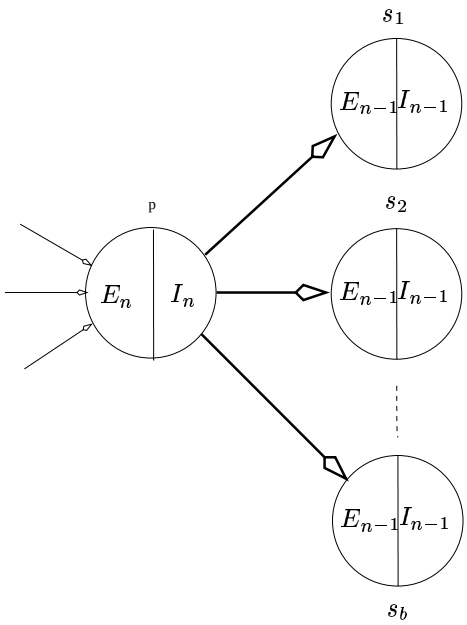

Figure 3: propagation arrière

Définition 3 (signature externe). *La* signature externe *d'une page p (notée $E(p)$) est la signature lexicale perçue par les auteurs des pages qui la pointent.*

Définition 4 (contenu). *Le contenu d'une page p (notée $C(p)$) est la signature lexicale caractérisant le contenu de la page p en dehors des liens hypertextes (contenu brut).*

Le contenu d'une page est une information que l'on peut compléter à l'aide des signatures interne et externe. Nous pensons que l'utilisation de telles signatures permettront d'améliorer la performance des moteurs de recherche. En effet, ce modèle peut nous permettre de réduire le spamming en otant les pages dont les signatures internes et externes sont éloignées. Ces signatures sont obtenues en appliquant itérativement les équations de *propagation avant et arrière*.

Définition 5 (Equation de *propagation avant*). *La signature externe d'une page à l'instant n est obtenue à partir des précedentes signatures internes des pages qui la pointent :*

$$E_0(p) \leftarrow \emptyset \tag{7}$$

et

$$E_n(p) \leftarrow f(I_{n-1}(q_1), \cdots, I_{n-1}(q_a)) \tag{8}$$

où $q_1, \ldots$ et q_a désignent les predécesseurs de p dans G (figure 2).

Définition 6 (Equation de *propagation arriere*). *La signature interne d'une page p à l'instant n est obtenue à partir de son contenu et des précedentes signatures externes des pages qu'elle pointe :*

$$I_0(p) \leftarrow C(p) \tag{9}$$

et

$$I_n(p) \leftarrow g(E_{n-1}(s_1), \cdots, E_{n-1}(s_b), C(p)) \tag{10}$$

où $s_1, \ldots$ et s_b représentent les successeurs de p dans G (figure 3).

De notre point de vue, la signature interne d'une page ne doit pas reposer uniquement sur le contenu de cette page. Elle doit aussi s'exprimer à partir des signatures externes des pages pointées. En effet, rappelons qu'un auteur est au moment de la création d'une page d'abord un internaute. Ce dernier construit sa page en fonction de la perception des pages qu'il estime nécessaires (signatures externes).

3.1 L'algorithme

L'algorithme 3 consiste à appliquer itérativement les deux équations de propagation des signatures lexicales (8) et (10). Initialement, pour chaque page p, sa signature interne $I_0(p)$ est égale à son contenu $C(p)$ et sa signature externe $E_0(p)$ est vide.

3.2 Calcul de $C(p)$

Pour calculer la signature $C(p)$ d'une page p nous utilisons les techniques classiques en usage en recherche d'informations. En particulier, nous nous basons sur le modèle TF×IDF. Le facteur *TF* correspond à la fréquence relative d'un terme donné dans une page. Le facteur *IDF* correspond à la fréquence inverse de ce terme sur l'ensemble du corpus. Formellement, nous avons pour chaque terme t :

$$\mathit{TFIDF}(t) = \mathit{TF}(t) \times \log(\frac{N}{DF(t)}) \tag{11}$$

Algorithme 3: LexicalRank

Données
– un graphe du Web $G = (V, E)$;
– un entier k.
Résultat
Signatures internes et externes des pages de V.
début
  pour $p \in V$ **faire**
    $I_0(p) = C(p)$
    $E_0(p) = \emptyset$
  fin
  pour $n = 1$ *à* k **faire**
    pour $p \in V$ **faire**
      Appliquer l'équation de propagation avant (8) aux predécesseurs de p dans G pour obtenir $E'_n(p)$
      Appliquer l'équation de propagation arrière (10) aux successeurs de p dans G pour obtenir $I'_n(p)$
    fin
    Normaliser le vecteur signature externe E'_n pour obtenir E_n
    Normaliser le vecteur signature interne I'_n pour obtenir I_n
  fin
  Retourner I_k et E_k les signatures interne et externe de V
fin

N est le nombre total de documents du corpus et $DF(t)$ est le nombre de pages contenant le terme t.

Généralement, la valeur de fréquence d'un terme correspond à son nombre d'occurences. En pratique, une occurence d'un terme voit son poids (par défaut égal à 1) augmenté ou diminué en fonction de sa postion dans la page. Par exemple, un terme présent dans le titre de la page verra son poids fortement augmenté. D'une façon générale, une heuristique satisfaisante consiste à privilégier plutôt les termes en début de page.

Notre approche est incrémentale, toutefois nous ne désirons pas recalculer les facteurs N et $DF(t)$ à chaque ajout de document. un telle approche sera impraticable. Nous adoptons donc une solution alternative approchée, en nous basant sur les informations fournies par le moteur de recherche Google. Le facteur N correspond au nombre de pages contenant le terme le plus fréquemment rencontré sur le Web jusque là. Le facteur $DF(t)$ lui correspond au nombre de pages contenant le terme t tel que renvoyé par Google. Par exemple, Google indique qu'il y a 3000000 de pages contenant le terme *chien*.

3.3 Fonction f

La fonction f est celle qui combine les signatures lexicales internes pour le calcul d'une signature externe. Il s'agit simplement, dans l'expérience que nous avons menée, d'une somme normalisée des termes pondérés.
Soit $I_{n-1}(q_i) = \{(t_1; p_1), (t_2; p_2), \ldots (t_k; p_k)\}$ la signature interne d'une page q_i, nous définissons la somme normalisée de signatures lexicales comme suit :

$$f(I_{n-1}(q_1), I_{n-1}(q_2) \ldots I_{n-1}(q_a)) = \frac{I_{n-1}(q_1) \oplus I_{n-1}(q_2) \oplus \ldots \oplus I_{n-1}(q_a)}{||I_{n-1}(q_1) \oplus I_{n-1}(q_2) \oplus \ldots \oplus I_{n-1}(q_a)||} \quad (12)$$

où l'opérateur $\oplus$ désigne l'union *ensembliste* de deux ensembles de termes pondérés.
Par exemple, soient :

$I(p_1)$ = {(‹*chaussure*› ;0,9), (‹*assemblage*› ;0,68), (‹*végétaux*› ;0,4), (‹*coup*› ;0,35), (‹*réunion*› ;0,31), (‹*escrime*› ;0,2), (‹*fleuret*› ;0,14), (‹*épée*› ;0,13)}

et

$I(p_2)$ = {(‹*chaussure*› ; 0,5), (‹*végétaux*› ; 0,68), (‹*bottine*› ; 0,6), (‹*coup*› ;0,35), (‹*pied*› ;0,32), (‹*viande*› ;0,2), (‹*fleur*› ;0,14), (‹*épée*› ;0,13)}

alors

$I(p_1) \oplus I(p_2)$ = {(‹*chaussure*› ;1.4), (‹*assemblage*› ;0,68), (‹*végétaux*› ;1.08), (‹*bottine*› ;0,6), (‹*coup*› ;0,7), (‹*pied*› ;0,32), (‹*réunion*› ;0,31), (‹*escrime*› ;0,2), (‹*fleuret*› ;0,14), (‹*épée*› ;0,26), (‹*viande*› ;0,2), (‹*fleur*› ;0,14)}

3.4 Fonction g

La fonction g est celle qui combine les signatures lexicales externes et le contenu de la page pour le calcul d'une signature interne. Il s'agit simplement, dans l'expérience que nous avons menée, d'une somme normalisée des termes pondérés des signatures externes (correspondant aux URL) d'une part, et de la signature du contenu d'autre part. En l'absence d'une étude plus poussée, le poids global des signatures externes est équivalent au poids du contenu.

$$g(E_{n-1}(s_1), E_{n-1}(s_2) \ldots E_{n-1}(s_b)) = \frac{E_{n-1}(s_1) \oplus E_{n-1}(s_2) \oplus \ldots \oplus E_{n-1}(s_b)}{||E_{n-1}(s_1) \oplus E_{n-1}(s_2) \oplus \ldots \oplus E_s(s_b)||} \oplus C(p) \quad (13)$$

Cette formule est simplifiée dans la mesure où à chaque URL (contenue dans la page) on associe le même poids. En pratique, le poids de chaque URL dépend de sa position dans le document (de la même manière que pour le calcul de $C(p)$).

4 Résultats

Un logiciel implémentant *LexicalRank* nous permet de nous promener dans le graphe de page en page. Notre graphe de travail est un site web consacré à l'étude des réseaux pairs à pairs. Les arcs du graphe ne reflètent donc que la structure du site (qui présente une cohérence assez forte).

Le logiciel permet de suivre l'évolution au cours des itérations de l'algorithme des signatures interne et externe de la page courante . Les tables 2 à 4 illustrent l'évolution des signatures lexicales d'une page Web introduisant les réseaux pairs à pairs. Certaines pages web dans le site y font référence. En quelques itérations, nous avons constaté sur certaines pages l'émergence dans leurs signatures lexicales de termes liés implicitement à la page. En effet, dès les premières itérations les termes *exploration* et *identification* apparaissent en tête de la signature externe de la page dans la mesure un certain nombre de pages dans le site traite de la difficile problématique de l'exploration de ces réseaux ou de l'identication des pairs et font référence à la première page.

Par ailleurs, on constate dans certains cas que les termes associés à des liens navigationnels (*monter*, *suivant* ou *précédent*) se retrouvent en bas des signatures lexicales.

5 Conclusion et perspectives

Nous avons présenté dans cet article l'algorithme *LexicalRank* propageant deux signatures lexicales : l'une interne, l'autre externe. Nous espérons ainsi ouvrir la voie à une nouvelle famille d'algorithmes PageRank fondés sur la propagation de signatures lexicales. Nous avons obtenu des résultats très prometteurs. Nous avons constaté au cours de l'exécution de l'algorithme l'emergence de termes précis caractérisant implicitement la page.

Dans nos expérimentations, nous avions choisi des fonctions f et g qui soient dans un premier temps combinaison linéaire des signatures lexicales. D'autres fonctions plus élaborées sont actuellement à l'étude. Nous projetons enfin une validation complète de *LexicalRank* en l'incorporant dans un moteur de recherche et en effectuant une série de tests de satisfaction sur une population témoin.

rang	terme	poids
1	‹*réseau*›	58,20
2	‹*clients*›	39,62
3	‹*précédent*›	37,64
4	‹*matières*›	37,64
5	‹*participant*›	31,46
6	‹*rapport*›	29,77
7	‹*fichier*›	29,51
8	‹*interconnectés*›	24,67
9	‹*généralement*›	24,67
10	‹*utilisé*›	24,67
11	‹*manière*›	24,67
12	‹*décentralisé*›	24,67
13	‹*possédant*›	24,67
14	‹*répondant*›	24,67
15	‹*constitué*›	24,67
16	‹*échanger*›	24,57
17	‹*adresse*›	22,11
18	‹*IP*›	21,21
19	‹*principale*›	20,98
20	‹*fonction*›	20,98
21	‹*appelé*›	20,67
22	‹*chaque*›	20,38
23	‹*Internet*›	18,49
24	‹*certains*›	18,49
25	‹*trouver*›	18,47
26	‹*client*›	17,21
27	‹*connecte*›	16,54
28	‹*programme*›	13,51
29	‹*type*›	12,28
30	‹*pair*›	12,17
31	‹*fichiers*›	3,21
32	‹*monter*›	0,76
33	‹*suivant*›	0,05

Table 1: signature interne à l'itération 2.

References

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *Proc. 9th International World Wide Web Conference*, pages 309–320, 2000.

[BLCL+94] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world wide web. *Communications of ACM*, 37(8) :76–82, 1994.

[BMPW98] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a Web in your Pocket ? *Data Engineering Bulletin*, 21(2) :37–47, 1998.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7) :107–117, 1998.

[CGMP98] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7) :161–172, 1998.

[CP95] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world wide web. *Computer Networks and ISDN Systems*, 27(6) :1065–1073, 1995.

[Goo98] Google. http ://www.google.com/, 1998.

[Hav99] T. Haveliwala. Efficient computation of PageRank. Technical report, Computer Science

rang	terme	poids
1	‹*réseau*›	88,25
2	‹*clients*›	79,90
3	‹*précédent*›	78,78
4	‹*matières*›	78,78
5	‹*participant*›	74,89
6	‹*rapport*›	73,6
7	‹*fichier*›	73,50
8	‹*constitué*›	69,61
9	‹*interconnectés*›	69,61
10	‹*appelé*›	69,61
11	‹*généralement*›	69,61
12	‹*utilisé*›	69,61
13	‹*échanger*›	69,61
14	‹*décentralisé*›	69,61
15	‹*possédant*›	69,61
16	‹*critères*›	69,61
17	‹*manière*›	69,51
18	‹*fonction*›	68,09
19	‹*adresse*›	67,22
20	‹*IP*›	66,33
21	‹*principale*›	66,09
22	‹*chaque*›	65,46
23	‹*Internet*›	63,34
24	‹*trouver*›	63,34
25	‹*certains*›	63,34
26	‹*connecte*›	60,93
27	‹*répondant*›	59,51
28	‹*programme*›	58,53
29	‹*type*›	54,42
30	‹*pair*›	54,26
31	‹*client*›	51,79
32	‹*fichiers*›	25,33
33	‹*monter*›	0,91
34	‹*suivant*›	0,42

Table 2: signature interne à l'itération 3.

rang	terme	poids
1	‹*exploration*›	94,44
2	‹*protocole*›	93,57
3	‹*identification*›	93,55
4	‹*matières*›	89,78
5	‹*diagramme*›	87,22
6	‹*problématique*›	83,86
7	‹*fonctionnalités*›	83,66
8	‹*tête*›	80,02
9	‹*implémentation*›	79,97
10	‹*noeuds*›	79,24
11	‹*introduction*›	79,23
12	‹*decentralisé*›	79,19
13	‹*hybride*›	79,17
14	‹*modèle*›	78,45
15	‹*architectures*›	77,68
16	‹*centralisé*›	76,68
17	‹*architecture*›	75,99
18	‹*rapport*›	74,43
19	‹*système*›	74,14
20	‹*differentes*›	73,78
21	‹*table*›	71,81
22	‹*applications*›	71,36
23	‹*super*›	70,77
24	‹*définition*›	70,62
25	‹*réseaux*›	69,54
26	‹*réseau*›	68,85
27	‹*historique*›	68,25
28	‹*index*›	66,67
29	‹*pair*›	65,72
30	‹*suivant*›	54,33
31	‹*précédent*›	39,25
32	‹*monter*›	38,72

Table 3: signature externe à l'itération 2.

Department, Stanford University, 1999.

[MFJR+04] Natasa Milic-Frayling, Rachel Jones, Kerry Rodden, Gavin Smyth, Alan Blackwell, and Ralph Sommerer. Smartback : supporting users in back navigation. In *Proceedings of the 13th international conference on World Wide Web*, pages 63–71. ACM Press, 2004.

[PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking : Bringing Order to the Web. Technical report, Computer Science Department, Stanford University, 1998.

[Sal89] G. Salton. Automatic text processing. Massachusetts, 1989.

[SC96] L. Saloff-Coste. Lectures on finite Markov chains. In G.R. Grimmet E. Giné and L. Saloff-Coste, editors, *Lecture Notes on Probability Theory and Statistics*, number 1665 in LNM, pages 301–413. Springer Verlag, 1996.

[WM04] Long Wang and Christoph Meinel. Behaviour recovery and complicated pattern definition in web usage mining. In *ICWE*, pages 531–544. LNCS, 2004.

[YLYL95] Budi Yuwono, Savio L. Lam, Jerry H. Ying, and Dik L. Lee. A world wide web resource discovery system. In *Proc. 4th International World Wide Web Conference*, December 1995.

Rang	terme	poids
1	‹*exploration*›	94,59
2	‹*protocole*›	83,97
3	‹*identification*›	83,96
4	‹*introduction*›	79,23
5	‹*decentralisé*›	79,23
6	‹*hybride*›	79,23
7	‹*noeuds*›	79,21
8	‹*architecture*›	77,71
9	‹*architectures*›	77,44
10	‹*matières*›	77,37
11	‹*table*›	77,00
12	‹*rapport*›	76,97
13	‹*suivant*›	76,92
14	‹*centralisé*›	76,71
15	‹*des*›	76,64
16	‹*index*›	75,60
17	‹*système*›	74,17
18	‹*differentes*›	73,11
19	‹*définition*›	70,55
20	‹*super*›	70,51
21	‹*applications*›	70,47
22	‹*réseau*›	69,98
23	‹*réseaux*›	68,19
24	‹*historique*›	66,12
25	‹*pair*›	64,50
26	‹*précédent*›	42,69
27	‹*monter*›	42,68

Table 4: signature externe à l’itération 3.