# Construction et utilisation de contextes autour des nœuds d'un hypertexte pour la recherche d'information

**Fernando Aguiar*** — **Michel Beigbeder****

* *Netbooster*
*11, rue Dieu*
*F-75010 Paris*
*faguiar@netbooster.com*

** *École Nationale Supérieure des Mines de Saint-Etienne*
*158, cours Fauriel*
*F-42023 Saint-Etienne cedex 2*
*michel.beigbeder@emse.fr*

RÉSUMÉ. *Nous faisons l'hypothèse que la mise sous forme hypertexte d'un document atomise l'information dans le sens où les nœuds de l'hypertexte qui sont créés ne sont pas auto-suffisants pour pouvoir être appréhendés. Sous cette hypothèse, le contenu seul du nœud n'est pas suffisant pour l'indexer dans un but de l'insérer dans un système de recherche d'information. Nous avons implémenté et testé une méthode de construction de contextes autour des nœuds d'un hypertexte en utilisant une méthode de classification automatique. Cette dernière est basée sur une mesure de similarité entre les nœuds prenant en compte à la fois les aspects structurels de l'hypertexte, à savoir les liens entre les nœuds, et le contenu textuel des nœuds. Notre système de recherche d'information indexe à la fois les nœuds et leurs contextes. Le modèle de requête que nous utilisons est à deux niveaux : niveau sujet et niveau contexte.*

ABSTRACT. *We assume that the nodes of an hypertext are not always self-contained because when some information is made available in a hypertextual form, it is split in many nodes. Under this hypothesis the sole content of a node is not sufficient to index it in order to insert it in an information retrieval system. Using a clustering method, we have implemented and tested a context building method around the nodes of an hypertext. The similarity function between the nodes is based both on hypertext structural clues – the hyperlinks – and the textual content of the nodes. Our information retrieval system indexes both the nodes and their contexts. We use two level queries: the subject level and the context level.*

MOTS-CLÉS : *hypertexte, contexte, recherche d'information, informatique.*

KEYWORDS: *hypertext, context, information retrieval, computer science.*

DN – 8/2004. Fouille de textes, pages 71 à 82

## 1. Introduction

L'utilisation massive de la Toile[1] depuis le milieu des années 1990 nous a familiarisé avec les documents hypertextes. Tout un chacun connaît par ailleurs le besoin d'outils de recherche d'information performant sur ce médium, au vu justement de son succès qui a entrainé l'accessibilité potentielle à une quantité d'informations sans précédent concernant tous les domaines de la connaissance humaine. Pour améliorer leur efficacité[2], les outils de recherche d'information doivent prendre en compte les particularités de ce médium.

Une caractéristique des systèmes hypertextes des plus importantes est le découpage de l'information en nœuds[3]. Lorsqu'un lecteur lit ou consulte un document hypertexte, il ne lit normalement pas un seul nœud, mais suit un parcours en partant d'une page particulière (la page d'accueil). De ce fait, il ne lit pas les nœuds indépendamment les uns des autres mais chacun des nœuds qu'il voit construit un contexte pour les autres. C'est la présence des liens hypertexte, autrement dit la *structure* qui permet au lecteur de passer d'un nœud à un autre.

Différentes approches ont récemment été proposées pour prendre en compte la structure de la Toile dans la modélisation de deux principaux types d'outils de recherche d'information : les moteurs de recherche et les annuaires thématiques. Ces approches portent sur des problématiques spécifiques dont l'application la plus évidente est celle d'améliorer les outils de recherche. Nous pouvons citer comme exemple la découverte de communautés d'intérêt, le calcul de la réputation d'un site[4] ou encore la recherche d'unités logiques d'information. La méthode que nous proposons dans cette étude s'attaque à cette dernière problématique.

Les méthodes de classification automatique ont une longue histoire dans le domaine de la recherche d'information. Leur principe fondateur énoncé par Rijsbergen (1979) fait l'hypothèse que des documents qui se ressemblent (donc qui seront *in fine* dans la même classe) seront pertinents aux mêmes requêtes. D'abord invoqué pour des questions d'efficience, ce principe peut l'être aussi pour des raisons d'efficacité. C'est dans cet esprit que nous l'utilisons dans notre modèle présenté en section 3. Nous présenterons ensuite dans les sections suivantes sa mise en œuvre et les résultats obtenus. Dans la section 5, nous décrivons les expériences que nous avons menées sur 523 sites de la collection WT10g de la conférence TREC[5]. Nous terminerons dans la section 7 par les conclusions que l'on peut tirer des expériences menées.

---

1. Le *web*, ou encore le *World Wide Web*.
2. Nous distinguons *efficacité* : qui est lié à la qualité du résultat, de *efficience* : qui est lié au rendement (rapidité et/ou quantité de ressources utilisées).
3. *Page* dans le jargon de la Toile, nous utiliserons indifféremment l'un ou l'autre terme.
4. http://www.cs.toronto.edu/db/topic
5. http://trec.nist.gov/

## 2. Etat de l'art

### 2.1. *Utilisation de la structure hypertexte dans les moteurs de recherche*

Le premier et le plus connu des moteurs de recherche de grande diffusion à avoir utilisé les liens de la Toile pour classer les résultats de ses recherches est *Google*[6] (Brin and Page, 1998). Le classement qu'il fait avec l'algorithme *Page Rank* est en grande partie fait selon la popularité des pages, c'est-à-dire selon la quantité d'autres pages qui les référencent. L'algorithme HITS de Kleinberg (1999) s'apparente à l'algorithme sous-jacent à *Google*, il a été utilisé dans le cadre de la Toile par Chakrabarti *et al.* (1998) dans le projet *Clever*. Dans HITS la structure du graphe de la Toile est utilisée pour noter les nœuds selon deux critères, leur qualité de « page pivot » (*hub*, en anglais) et leur qualité de « page référence » (*authority*, en anglais). Une différence notable toutefois entre ces deux méthodes est que celle de *Google* note les pages indépendamment de la requête, et que celle de *Clever* prend en compte la requête dans son évaluation des scores des documents.

Ces deux algorithmes ont été utilisés pour différents buts, nous pouvons citer la collecte sélective de pages par les robots, la découverte de pages similaires à une (ou plusieurs) pages données, la découverte de la réputation[7] de ressources de la Toile, la découverte de communautés.

Marchiori (1998) a proposé de propager des informations d'indexation (des *méta-informations*) le long des liens, ces informations étant elles-mêmes issues d'une indexation manuelle réalisée par l'auteur d'une page et matérialisées grâce aux balises `<META>` du langage HTML. Cette proposition vient du fait que ces méta-informations sont disponibles sur peu de pages. L'algorithme de cette méthode est basé sur un parcours en largeur. D'autres utilisations des voisinages concernent la catégorisation de pages et la découverte de la portée géographique de ressources de la Toile.

Enfin, plusieurs moteurs prennent en compte le texte des ancres[8] qui pointent sur une page pour l'indexer. L'idée dans ce cas est que ces ancres décrivent d'une façon précise et concise le contenu de la page pointée.

Dans les travaux de Marchiori et dans l'utilisation du texte des ancres par les moteurs, l'idée implicite est que les pages ne sont pas auto-suffisantes, et que leur indexation nécessite des données issues de pages voisines. Ce problème est plus explicitement abordé par d'autres travaux que nous allons passer en revue.

---

6. http://www.google.com/
7. Une ressource de la Toile, par exemple un site, peut véhiculer de l'information concernant plusieurs thèmes différents. La réputation d'une ressource est ce qui indique « par quoi » une ressource est connue. Pour estimer la réputation d'une ressource on analyse, en général, les thèmes des ressources qui pointent vers la ressource considérée.
8. Et de leur voisinage.

### 2.2. *Découverte de pages complémentaires*

Les trois méthodes suivantes s'appliquent dans un cadre d'augmentation du rappel, dans l'idée que tous les concepts d'une requête conjonctive ne se trouvent pas nécessairement dans une page, mais certains sont dans des pages voisines.

Une requête dans le système *Jumping Spider* de Dyreson (1998) est une conjonction $Q = K_1 \rightarrow K_2 \rightarrow \cdots \rightarrow K_n$ où les termes sont ordonnés par ordre décroissant de généralité. La fonction de correspondance consiste à trouver les pages contenant le dernier terme $K_n$ (le plus spécifique) puis à remonter aux pages voisines contenant $K_{n-1}$, puis $K_{n-2}$, etc. selon un graphe construit à partir du graphe des liens hypertextes. Ce graphe ne doit contenir que des liens de généralisation/spécialisation. Pour cela, à partir des chemins codées dans les URL, les liens sont classés en liens i) vers le bas, ii) vers le haut, iii) de côté. Les liens vers le haut sont supprimés. De nouveaux liens sont créés entre le premier et le dernier nœud de tous les chemins composés uniquement de liens vers le bas et se terminant par zéro ou un lien de côté.

Tajima *et al.* (1999) traitent avec une méthode dans le même esprit les requêtes conjonctives $Q = K_1 \wedge K_2 \wedge \cdots \wedge K_n$. Connaissant les ensembles $R_i$ de nœuds pertinents[9] au sujet $K_i$, des graphes sont construits par dilatations autour des pages de $\bigcup R_i$ jusqu'à ce qu'ils contiennent tous les termes de la requête. Une méthode analogue proposée par Li *et al.* (2001) exécute un post-traitement toujours pour des requêtes conjonctives. Après la récupération des $R_i$, le problème consiste à trouver des arbres de longueur minimale avec au moins un nœud dans chaque $R_i$ – ce problème est connu sous le nom de problème de Steiner.

Ces trois méthodes ont l'inconvénient de se placer dans un cadre booléen dont on connaît les limites. De plus, la sémantique du modèle de requête de la première de ces méthodes nous semble difficilement appréhendable par l'utilisateur final. Enfin, ces trois méthodes interviennent en post-traitement des réponses d'un moteur classique, et sont donc susceptibles d'obérer l'efficience. Les quatre propositions que nous citons maintenant découvrent les unités logiques d'information avant la requête.

La première, proposée par Doan et Beigbeder (1999), suggère que les pages de la Toile ne sont pas nécessairement autonomes, et que les ressources doivent être regroupées récursivement. Le résultat obtenu par ces regroupements se rapproche de la structure logique bien connue dans les documents structurés. Par contre, le handicap de la proposition est que ces ULI[10] doivent être construites manuellement, ce qui dans l'état actuel de la Toile est rédhibitoire.

Géry (2002) propose un modèle des sites de la Toile avec deux structurations : une structure logique et une structure de chemins de lecture. Ces deux structures sont complétées par les contextes qui sont les pages externes pointées et les pages externes pointant-sur. C'est l'indexation de tous les chemins de lecture qui est proposée ; la complexité algorithmique est donc très élevée.

9. En fait, les nœuds contenant le terme $K_i$.
10. Unité logique d'information.

Tajima *et al.* (1998) proposent de découvrir des unités logiques d'informations en appliquant une classification automatique selon le lien moyen[11]. Pour prendre en compte la structure, la similarité entre deux classes est nulle s'il n'y a pas de lien hypertexte entre au moins une page de l'une et une page de l'autre, autrement elle est calculée selon un cosinus avec un modèle vectoriel de Salton.

Dans une autre étude, Mizuuchi et Tajima (1999) proposent cette fois d'utiliser la structure de répertoires codée dans les URLs, pour découvrir pour chaque page $P$ *une* page d'entrée, c'est-à-dire la page par laquelle un lecteur est supposé être passé pour arriver sur $P$, ainsi un arbre de pages est obtenu. Pour cela, de nombreuses hypothèses traduites en règles sont faites. L'indexation proposée, mais non testée, prend en compte le contenu de certaines balises des ancêtres de $P$ sur le chemin vers la racine.

Li *et al.* (2000) proposent de découvrir des domaines logiques – par rapport à des domaines physiques liés à l'adresse des machines hôtes. Toutefois, ces domaines d'une granularité plus grande que celle qui nous intéresse n'ont pas pour but d'améliorer l'indexation, mais de regrouper les résultats d'une recherche. Pour constituer ces domaines, la première étape consiste à trouver $k$ (paramètre de l'algorithme) points d'entrée selon des critères prenant en compte le contenu de la balise `title`, le contenu textuel de l'URL[12], les degrés entrant et sortant dans le graphe de la Toile, etc. Dans la seconde étape une page restante est rattachée au point d'entrée le plus proche qui est au dessus selon son chemin d'URL (de ce fait, certaines pages peuvent rester orphelines). De plus d'autres conditions – taille minimale d'un domaine, rayon – peuvent conditionner la constitution des domaines.

De ces trois dernières propositions, les deux dernières reposent sur de nombreuses hypothèses difficiles à vérifier, en particulier sur l'organisation des pages dans les systèmes de fichiers. Notre proposition se rapproche donc de la première, mais nous évaluons la nôtre par une implémentation et une mise en œuvre dans un moteur de recherche.

## 3. Modèle

Notre proposition de système de recherche d'information rend comme réponse à une requête des nœuds triés selon leur pertinence calculée par le système. Nous restons donc dans le cadre classique des moteurs de recherche. Nous avons choisi cette option plutôt que de rendre des ULI, ce qui pourrait se justifier parce que les ULI sont auto-suffisantes, alors que nous supposons que les nœuds ne le sont pas toujours. Toutefois, ce choix permet une présentation des résultats habituelle pour l'utilisateur, alors qu'il faudrait développer de nouveaux modes de présentation de résultats pour montrer les ensembles de pages. De plus, avec ce choix, nous pouvons utiliser les

11. Dans les méthodes de classification automatique hiérarchique ascendante, on a besoin de calculer la distance entre deux classes. Dans la méthode du lien moyen, on pose que cette distance inter-classe est la moyenne des distances entre les éléments des deux classes.
12. Pour ces deux critères, l'hypothèse est anglophone (*welcome*, *home*, *people*, *user*, etc.).

méthodes traditionnelles et les collections de tests existantes pour l'évaluation des systèmes de recherche d'information.

Notre but principal est donc de mieux indexer les nœuds d'un hypertexte et pour cela nous suggérons donc que soient pris en compte pour l'indexation d'un nœud non seulement le contenu de ce nœud, mais aussi les contenus des nœuds qui le contextualisent. Ceci impose donc une première étape dans laquelle nous découvrons ces contextes, qui sont des ensembles de nœuds.

Dans la découverte des contextes des pages, nous ne considérons que les pages appartenant au même « site ». La notion de « site » ne se définit pas très précisemment, l'idée est que c'est un ensemble de pages formant un ensemble cohérent d'information, géré et maintenu par une seule personne ou un seul groupe de personnes. Cependant, selon cette définition les frontières des sites sur la Toile ne sont pas aisées à découvrir. Nous nous basons donc comme d'autres sur la notion approchée qui définit un site comme l'ensemble des pages disponibles sur la Toile qui partagent le même préfixe `http://<nom de machine>:<numéro de port>/`. C'est bien évidemment une approximation de la définition floue précédente, il suffit en effet de considérer les domaines gérés par des fournisseurs d'accès à internet : ceux-ci contiennent de nombreux sites, ces derniers étant gérés par leurs abonnés. L'hypothèse sous-jacente à cette approximation est que ces cas sont marginaux sur la Toile.

Après cette étape, nous avons le choix entre deux possibilités, soit indexer chaque page avec son contenu et celui de son contexte, ou garder dans la base d'index i) l'index de chaque nœud, ii) l'index de chaque contexte, iii) le lien entre un nœud et son contexte. Nous avons choisi la deuxième option qui nous permet d'introduire des requêtes à deux niveaux : un niveau *sujet* et un niveau *contexte*. Ainsi, une requête $q$ est un couple $(q_s, q_c)$, où $q_s$ est la partie *sujet*, et $q_c$ est la partie *contexte*.

Dans les différentes étapes, des choix d'algorithmes et/ou de paramètres de ces algorithmes sont possibles. Nous ne détaillons pas dans cet article tous ces choix, mais seulement ceux que nous avons finalement retenus après une phase de test non décrite ici.

## 4. Méthode

### 4.1. *Construction des contextes*

Notre méthode se base sur une similarité $S(N, N')$ entre deux nœuds $N$ et $N'$ qui combine :

– une similarité de contenu, $S_c(N, N')$, calculée selon le modèle vectoriel de Salton ;

– une proximité structurelle, $P_s(N, N')$. Dans nos expériences, nous avons choisi :

$$P_s(N, N') = \beta \cdot S^{anc}(N, N') + \gamma \cdot S^{desc}(N, N') + \lambda \cdot S^{spl}(N, N')$$

avec :

$$S^{spl}(N,N') = \frac{1}{2^{spl(N,N')}} + \frac{1}{2^{spl(N',N)}}$$

$$S^{anc}(N,N') = \sum_{x \in Anc(N,N')} \frac{1}{2^{(spl^{\overline{N'}}(x,N) + spl^{\overline{N}}(x,N'))}}$$

$$S^{desc}(N,N') = \sum_{x \in Desc(N,N')} \frac{1}{2^{(spl^{\overline{N'}}(N,x) + spl^{\overline{N}}(N',x))}}$$

où i) $\mathrm{spl}(x,y)$ est la longueur d'un plus court chemin de $x$ vers $y$ s'il en existe, et l'infini sinon ; ii) $\mathrm{spl}^{\overline{z}}(x,y)$ est la longueur d'un plus court chemin de $x$ vers $y$ sans passer par $z$ ; iii) $Anc(x,y)$ est l'ensemble des ascendants communs aux deux nœuds $x$ et $y$ et $Desc(x,y)$ est l'ensemble des descendants communs à $x$ et $y$. L'utilisation de ces mesures à base de plus courts chemins a été proposée par Weiss *et al.* (1996) pour leur système *HyPursuit* .

Ces deux mesures sont combinées selon un coefficient $\alpha$ : $S(N,N') = (1-\alpha) \cdot S_c(N,N') + \alpha \cdot P_s(N,N')$. Nous obtenons ainsi une matrice symétrique.

Différentes méthodes de classification automatique peuvent alors être mises en œuvre. Nous avons retenu l'algorithme du lien complet après avoir essayé aussi les méthodes du simple-lien, du lien moyen, l'algorithme de Ward et l'algorithme de l'étoile (Anderberg, 1973).

Comme dans tous les algorithmes de classification hiérarchique, il faut choisir un niveau de la hiérarchie pour en obtenir une partition en classes. Nous avons choisi pour cela le niveau qui donne moitié moins de classes que de pages. Les classes résultats de cette classification sont les contextes.

### 4.2. *Indexation*

L'indexation des nœuds est faite selon le modèle vectoriel de Salton. Pour les contextes nous ne conservons que les composantes de poids les plus forts dans l'idée que seule l'information répétée dans les différentes pages est du contexte. Plus précisemment, l'indexation des contextes est obtenue à partir de la moyenne des vecteurs représentant ses éléments dans lesquels ne sont conservées que les composantes de poids supérieur à la moyenne des composantes non nulles.

Enfin, pour modéliser le fait que certaines pages sont plus centrales dans un contexte, nous introduisons la notion d'*attachement* entre un nœud et le contexte auquel il appartient. Cet attachement est la moyenne des similarités entre la page considérée et les autres pages du contexte :

$$Att(p, C(p)) = \frac{1}{|C(p)|} \cdot \sum_{p' \in C(p)} S(p,p')$$

REMARQUE. — La fonction de classification que nous avons choisie fait que les classes engendrées sont disjointes. De ce fait, une page $p$ appartient à un et un seul contexte, que nous noterons $C(p)$. Avec d'autres méthodes de classification où les classes ne sont pas nécessairement disjointes, nous aurions à considérer tous les contextes d'une page.

### 4.3. *Fonctions de correspondance*

Dans la fonction de correspondance, nous devons combiner ce qui concerne la page et son contexte du côté des documents, et les deux parties de la requête de son côté. Vu du côté de la requête nous devons faire un appariement qui satisfait la partie sujet $q_s$ ET la partie contexte $q_c$.

Nous avons envisagé et testé plusieurs hypothèses pour la fonction de correspondance. D'abord, nous avons deux façons de faire intervenir la partie contexte de la requête, soit trois fois dans la formule (choix A), soit deux fois (choix B). Indépendamment de ce premier choix, nous avons d'abord fait intervenir la valeur d'attachement entre une page et son contexte (choix a) ou de remplacer cette valeur par 1 (choix b).

Ainsi dans le choix Aa, nous définissons la fonction de correspondance comme suit :

$$Sim_{Aa}(p,q) = Sim(p, q_s + q_c) \cdot (Sim(C(p), q_c) \cdot Att(p, C(p)) + Sim(p, q_c))$$

Ici nous avons un produit entre deux facteurs. Le premier prend en compte la pertinence, $Sim(p, q_s + q_c)$ de la page $p$ contre la requête dans sa totalité. A noter que toutes les similarités sont calculées selon le modèle vectoriel comme le cosinus entre les vecteurs représentant la page et la requête.

Pour prendre en compte le contexte de la page, nous faisons intervenir deux termes pour prendre en compte i) le cas où c'est effectivement le contexte que nous avons calculé qui apparie la partie contexte de la requête ($Sim(C(p), q_c) \cdot Att(p, C(p))$) ii) le cas où la page est auto-suffisante et apparie elle-même la partie contexte de la requête ($Sim(p, q_c)$).

Dans le choix Ab par rapport au choix Aa, nous remplaçons le facteur $Att(p, C(p))$ par la valeur 1, supprimant ainsi un effet d'atténuation du contexte :

$$Sim_{Ab}(p,q) = Sim(p, q_s + q_c) \cdot (Sim(C(p), q_c) + Sim(p, q_c))$$

Dans le choix Ba, nous n'utilisons que la partie sujet de la requête pour le premier facteur. En effet, revenant à notre hypothèse sur la requête qui doit être appariée dans sa partie $q_c$ ET dans sa partie $q_s$, nous faisons le choix de représenter explicitement cette conjonction comme dans le modèle des ensemble flous avec un produit entre ce qui concerne $q_s$ d'une part et ce qui concerne $q_c$ d'autre part. Ainsi, dans le choix Ba, nous n'utilisons que la partie sujet de la requête pour le premier facteur. En

effet, nous avions introduit $q_c$ dans ce facteur pour prendre en compte les pages auto-suffisantes, cependant $q_c$ est aussi pris en compte dans le deuxième facteur. D'autre part, le premier choix risque de favoriser les pages n'appariant que la partie contexte de la requête. Ainsi, la formule pour le choix Ba est la suivante :

$$Sim_{Ba}(p,q) = Sim(p,q_s) \cdot (Sim(C(p),q_c) \cdot Att(p,C(p) + Sim(p,q_c))$$

Enfin, nous dérivons le choix Bb du choix Ba en remplaçant encore la valeur d'attachement par 1 :

$$Sim_{Bb}(p,q) = Sim(p,q_s) \cdot (Sim(C(p),q_c) + Sim(p,q_c))$$

## 5. Expériences

Le modèle et la méthode décrits précédemment sont à la base d'un moteur de recherche d'information complet que nous avons implémenté. Avec une collection de test, nous pouvons donc évaluer notre modèle selon les critères classiques du domaine de la recherche d'informations, en particulier le rappel et la précision. Notre choix s'est porté sur la collection WT10g utilisée dans la *Web Track* de TREC en 2000 (Hawking, 2000) et les topiques 451–500[13].

Dans nos expériences nous avons considéré les sites indépendamment les uns des autres, ce qui correspondrait à une configuration où chaque site aurait son propre moteur de recherche restreint aux documents qu'il fournit.

Parmi les sites de WT10g, nous n'avons considéré que ceux contenant au moins un document pertinent à l'un des 50 topiques. 871 sites parmi les 11 673 de la collection répondent à ce premier critère. Parmi ces sites, les questions de complexité du modèle qui prend en compte les descendants communs et les ascendants communs (cf. 4.1) et donc de temps de calcul sur l'évaluation des indices structurels nous ont menés à n'utiliser ce moteur que sur des hypertextes avec au plus 700 nœuds. L'application de cette seconde contrainte réduit le nombre de sites à 641. Enfin, nous n'avons pas pris en compte les topiques 456, 481 et 495 qui ne pouvaient pas avoir de réponse avec notre système parce qu'un élément essentiel de leur besoin d'information est composé de chiffres et que notre analyse lexicale les élimine. Ceci nous a conduit à encore écarter 118 sites. Nous avons donc travaillé sur 523 sites pour cette expérience.

Les topiques fournis par TREC pour la tâche d'évaluation d'un système de recherche d'information sont composés de trois formes. La première (et la plus courte) `<title>` est la requête telle qu'elle a été soumise par un internaute au moteur *eXcite*[14], la deuxième, plus longue, est `<desc>` (description)[15], et la troisième, un petit paragraphe, est dans le champ `<narr>` (forme narrative). Ces deux formes plus longues ont été construites manuellement par le personnel du NIST[16] à partir de la

13. http://trec.nist.gov/data/topics_eng/topics.451-500.gz
14. http://www.excite.com/
15. Nous ne nous sommes pas servi de cette forme dans nos expériences.
16. *National Institute of Standards and Technology*, cf. http://www.nist.gov/

| | | meilleure | équivalente | moins bonne |
|---|---|---|---|---|
| $Sim_{Aa}$ | $q$ =(<*title*>, <*title*>) | 75 s. 84 c. | 331 s. 392 c. | 117 s. 144 c. |
| | $q$ =(<*title*>, <*narr*>) | 72 s. 84 c. | 261 s. 313 c. | 190 s. 223 c. |
| $Sim_{Ab}$ | $q$ =(<*title*>, <*title*>) | 85 s. 102 c. | 362 s. 425 c. | 76 s. 93 c. |
| | $q$ =(<*title*>, <*narr*>) | 79 s. 97 c. | 284 s. 338 c. | 160 s. 185 c. |
| $Sim_{Ba}$ | $q$ =(<*title*>, <*title*>) | 75 s. 84 c. | 331 s. 392 c. | 117 s. 144 c. |
| | $q$ =(<*title*>, <*narr*>) | 128 s. 144 c. | 296 s. 356 c. | 99 s. 129 c. |
| $Sim_{Bb}$ | $q$ =(<*title*>, <*title*>) | 85 s. 102 c. | 362 s. 425 c. | 76 s. 93 c. |
| | $q$ =(<*title*>, <*narr*>) | 136 s. 163 c. | 313 s. 371 c. | 74 s. 86 c. |

**Tableau 1.** *Comparaison des résultat obtenus avec nos quatre fonctions de correspondance par rapport à une méthode vectorielle classique*

première. Pour construire les requêtes $q = (q_s, q_c)$ à partir des topiques tels qu'ils sont fournis, nous avons utilisé deux méthodes. Dans la première, nous associons le champ `<title>` à la fois à $q_s$ et à $q_c$. Dans la deuxième, nous associons le champ `<title>` à $q_s$ et le champ `<narr>` à $q_c$.

Pour chaque site sélectionné, nous comparons la précision moyenne de notre méthode et de la méthode vectorielle classique où n'est pris en compte que le contenu du nœud. Notre évaluation consiste à compter le nombre de sites pour lesquels i) la précision moyenne de notre méthode est supérieure d'au moins 5 % à celle de la méthode vectorielle classique, ii) elle est la même à 5 % près, iii) ou elle est inférieure d'au moins 5 %.

## 6. Résultats

Nous avons fait de nombreuses expérimentations en faisant varier i) le paramètre $\alpha$ de pondération entre la similarité de contenu et la proximité structurelle, ii) les coefficients $\beta$, $\gamma$, et $\lambda$ qui règlent les importances données aux mesures intervenant dans la proximité structurelle, iii) la méthode de classification. Nous ne présentons ici qu'une partie de ces résultats où nous avons retenu $\alpha = 0.75$, $\beta = 0$, $\gamma = 0$, $\lambda = 1$, et la méthode du lien complet.

Nous indiquons dans le tableau 1 le nombre de sites (s.) et le nombre de couples site-topique (c.) pour lesquels notre méthode est successivement i) meilleure, ii) équivalente, iii) moins bonne, pour les deux constructions de requêtes à partir des topiques que nous avons citées et les quatre fonctions de correspondance.

Dans le cas $q$ =(<*title*>, <*title*>) la variation entre $Sim_A$ et $Sim_B$ n'introduit pas de variations comme on pouvait s'y attendre, puisque la différence de formule ne peut exister que si $q_s$ et $q_c$ sont différents. De plus dans ce cas, ne pas prendre en compte la valeur d'attachement améliore les résultats.

Cependant notre système est conçu pour des requêtes à deux niveaux, il est donc plus pertinent d'étudier les résultats du cas $q$ =(<*title*>, <*narr*>). Ce passage de

$q$ =(<*title*>, <*title*>) à $q$ =(<*title*>, <*narr*>) a des effets contraires selon les formules $Sim_A$ ou $Sim_B$. Dans le premier cas, ce passage apporte une légère détérioration des résultats et dans le deuxième cas une nette amélioration ; et les résultats obtenues avec les formules $Sim_B$ sont nettement meilleurs que ceux obtenus avec les formules $Sim_A$. Ces résultats sont un indice appuyant l'hypothèse conjonctive sur les requêtes à deux niveaux. Enfin, il y a là aussi amélioration en ne prenant pas en compte la valeur d'attachement ; ce qui nous questionne sur la pertinence de cet indicateur.

## 7. Conclusion et perspectives

Les résultats de cette première expérience montrent une amélioration sensible lorsqu'une requête où les deux niveaux *sujet* et *contexte* de la requête sont vraiment utilisés, puisque pour les requêtes (<*title*>, <*title*>) il n'y a pas vraiment indication d'un contexte. Par ailleurs, le nombre de sites (resp. de couples) pour lesquels il y a une amélioration est lui aussi statistiquement significatif. Ces résultats sont donc un indice pour valider notre hypothèse et le modèle de recherche d'information que nous proposons. Une analyse plus fine (requête par requête) des résultats des expériences resterait nécessaire pour essayer de comprendre quelles caractéristiques des hypertextes et/ou des requêtes font que nous obtenons dans certains cas une amélioration et dans d'autres une détérioration.

Nous envisageons une nouvelle série d'expériences dans lesquelles nous ferons un unique moteur de recherche pour l'ensemble de la collection. En effet, ce sont des problèmes de complexité algorithmique, en particulier pour trouver les ascendants et descendants communs qui nous avaient fait faire le choix d'étudier notre moteur site par site. L'influence de ces critères liés à l'ascendance et à la descendance est minime[17], nous pensons donc revenir sur la définition de la formule de similarité et de ne pas utiliser ces critères.

Par ailleurs, nous avons utilisé des requêtes à deux niveaux en utilisant simplement deux champs différents des topiques de TREC. Nous pensons définir manuellement les deux niveaux de la requête. en effet le champ `<narr>` contient des mots qui peuvent contextualiser le sujet, mais aussi de nombreux mots vides.

Il serait enfin intéressant de comparer les différentes méthodes de découverte des unités logiques d'information décrites dans la section 2.2 dans l'environnement de test que nous avons décrit, c'est-à-dire y remplacer uniquement la fabrication des contextes par classification automatique par ces différentes méthodes et y utiliser les mêmes conditions expérimentales (collection WT10g, même implémentation du modèle vectoriel de base, même modèle de requête et mêmes requêtes, etc.).

17. En fait, nous n'avons présenté dans cet article que des résultats où les paramètres $\beta$ et $\gamma$ sont nuls, donc l'influence de ces critères dans la formule de similarité retenue est nulle. Cependant nous avons fait d'autres expériences avec des valeurs non nulles pour ces paramètres, et les résultats en termes de précision moyenne étaient quasiment identiques.

## 8. Bibliographie


Anderberg M. R., *Cluster Analysis for Application*. Academic Press, 1973.

Brin S., Page L., The anatomy of a large-scale hypertextual web search engine. In WWW7.

Chakrabarti S., Dom B., Raghavan P., Rajagopalan S., Gibson D., Kleinberg J., Automatic resource compilation by analyzing hyperlink structure and associated text. In WWW7.

Doan B., Beigbeder M., Virtual WWW documents : a concept to explicit the structure of WWW sites. In Landoni M., editor, *proceedings of the 21st Annual BCS-IRSG Colloquium on IR Research*, pages 185–204, 1999.

Dyreson C. E., A jumping spider : Restructuring the WWW graph to index concepts that span pages. In Vercoustre A.-M., Milosavljevi M., Wilkinson R., editors, *Proceedings of the Workshop on Reuse of Web Information, held in conjunction with the 7th WWW Conference*, pages 9–20, 1998.

Géry M., Chemins de lecture en contexte, indexation sémantique de la structure du Web. In Sèdes F., editor, *actes de INFORSID 2002, 20e congrès informatique des organisations et des systèmes d'information et de décision*, pages 123–138, juin 2002.

Hawking D., Overview of the TREC-9 web track. In Voorhees E. M., Harman D. K., editors, *Proceedings of The Ninth Text REtrieval Conference (TREC-9)*, 2000.

Kleinberg J., Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5) : 604–632, 1999.

Li W.-S., Candan K. S., Vu Q., Agrawal D., Retrieving and organizing web pages by "information unit". In Lyu M. R., Zurko M. E., editors, *Proceedings of the Tenth International World Wide Web Conference*, 2001.

Li W.-S., Kolak O., Vu Q., Takano H., Defining logical domains in a web site. In *HYPERTEXT '00, Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 123–132, 2000.

Marchiori M., The limits of web metadata and beyond. In WWW7.

Mizuuchi Y., Tajima K., Finding context paths for web pages. In *HYPERTEXT '99, Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*, pages 13–22, 1999.

Rijsbergen C. J. v., *Information Retrieval*. Butterworth (London), 1979.

Tajima K., Hatano K., Matsukura T., Sano R., Tanaka K., Discovery and retrieval of logical information units in web. In Wilensky R., Tanaka K., Hara Y., editors, *Proc. of Workshop of Organizing Web Space (in conjunction with ACM Conference on Digital Ligraries '99)*, pages 13–23, 1999.

Tajima K., Mizuuchi Y., Kitagawa M., Tanaka K., Cut as a querying unit for WWW, netnews, e-mail. In *HYPERTEXT '98, Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space–structure in hypermedia systems*, pages 235–244, 1998.

Weiss R., Velez B., Sheldon M. A., Nemprempre C., Szilagyi P., Duda A., Gifford D. K., HyPursuit : A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the Seventh ACM Conference on Hypertext*, pages 180–193, 1996.

WWW7, *Proceedings of the Seventh International World Wide Web Conference*, 1998.