Restitution des fragments pertinents pour la recherche d'information dans des documents XML

Ben aouicha Mohamed (*)(***), Tmar Mohamed **, Boughanem Mohand *, Abid Mohamed ***

* Institut de Recherche en Informatique de Toulouse
118, route de Narbonne 31000 Toulouse Cedex 9
mohamed.benaouicha@irit.fr
bougha@irit.fr

** Institut Supérieur d'Informatique et du Multimédia de Sfax
4, route de tunis
3018 Sfax

mohamed.tmar@isimsf.rnu.tn
** Ecole Nationale d'Ingénieurs de Sfax
4, route de Soukra
3038 Sfax
mohamed.abid@enis.rnu.tn

RÉSUMÉ. Cet article présente un modèle pour la recherche d'information dans les documents XML basée sur la comparaison d'arbres, nous proposons une méthode permettant de restituer les fragments XML pertienents. Les requêtes et les documents sont représentés par des arbres étendus. Un arbre étendu est construit à partir de l'arbre original, avec la pondération des liens virtuels entre chaque nœud et ses descendants indirects, permettant à chacun d'atteindre directement ses descendants. Nous proposons également un algorithme pour comparer aisément et avec flexibilité les contraintes structurelles de la requête de l'utilisateur et la structure du document. Ainsi, les fragments d'un document (éléments) retournés en réponse à la requête lui sont similaires en terme de contenu et de structure. Ce modèle supporte l'interrogation de corpus XML par des requêtes orientées contenu et structre (type COS). Quelques expérimentations ont été menées dans le cadre de la compagne d'évaluation INEX [1] pour montrer l'efficacité de notre approche.

ABSTRACT. This paper presents an information retrieval model in XML documents based on tree matching, we propose a retrieval method for XML fragments. Queries and documents are represented by extended trees. An extended tree is built starting from the original tree, with additional weighted virtual links between each node and its indirect descendants allowing to directly reach each descendant. This allows to compare easily structural constraints of the user query and the document structure with flexibility. Thus document fragments (elements) returned in response to the query are the most similar ones to it. This model allows XML corpus interrogation with content and structure queries (COS). Some experiments have been undertaken into a dataset provided by INEX to show the effectiveness of our approach.

MOTS-CLÉS : Recherche d'information XML, COS, arc virtuel, Comparaison d'arbres

KEYWORDS : XML retrieval, COS, virtual link, Tree to tree correction

1. INitiative for the Evaluation of XML retrieval, un forum d'évaluation dont l'objectif est de promouvoir la recherche d'information structurée.

1. Introduction

L'objectif principal d'un SRI classique est de retrouver les documents dont le contenu est conforme à une requête donnée. Dans cette optique, les documents sont représentés par un ensemble de mots clés décrivant leurs contenus. La structure du document n'est pas prise en considération ni au niveau de la requête, ni au niveau de la réponse pour retourner les parties pertinentes : la réponse à une requête reste le document tout entier. Aujourd'hui, l'utilisation de l'information apportée par la structure devient une nécessité dans le domaine d'accès à l'information. Cette nécessité provient de deux communautés : la communauté de la recherche d'information qui s'intéresse à la structure présentée dans les documents pour apporter des réponses plus précises à une requête d'un utilisateur, et la communauté des bases de données qui s'intéresse à la structure comme un moyen plus souple pour stocker des données [1]. L'objectif principal sur lesquelles se focalisent ses deux communautés est un type de document très répondu sur Internet et utilisé comme un standard d'échange sur le Web [2] : Le langage XML (eXtensible Markup Langage) [4] [3] est utilisé comme format de données structurées sur le Web. Son intégration à la problématique de la recherche d'information impose au système de recherche d'information de retrouver des unités d'information qui ne sont pas nécessairement le document entier. En effet, dans le même document, l'utilisation peut juger que quelques nœuds ou fragments sont pertinents alors que d'autres ne le sont pas.
Dans cet article, nous proposons une méthode qui traite la structure du document d'une manière flexible, en se basant sur la comparaison d'arbres. Nous utilisons les liens de descendance indirecte comme moyen de relaxation et comparaison vague.
Cet article est organisé comme suit : la deuxième section présente un état de l'art sur les méthodes de comparaison d'arbres. La troisième section présente notre approche pour la manipulation de la structure. La quatrième section s'intresse à notre mécanisme de propagation de texte utilisé pour la recherche sur le contenu. La cinquième section donne un apperçu sur les mesures utilisées dans INEX'2005 ainsi que les expérimentations réalisées. La sixième section conclut.

2. Approches de comparaison d'arbres

Plusieurs techniques ont été mises en place pour la comparaison d'arbres. Ces techniques se basent sur la construction de plans de reconstitution d'un arbre à partir d'un autre, en effectuant des opérations de base (suppression, ajout, permutation,...) sur les nœuds, et d'estimer le coût de reconstitution.

2.1. Algorithme de Selkow

Cet algorithme est basé sur les opérations élémentaires traditionnelles : insérer nœud, supprimer nœud et modifier nœud, il ne permet la suppression et l'insertion que sur les feuilles de l'arbre de départ [6]. De ce fait, un nœud interne ne peut être supprimé ou inséré que si tout sous-arbre subordonné a été supprimé afin qu'il devienne une feuille. Pour calculer la distance entre deux arbres donnés A et B, l'algorithme commence par calculer le coût de transformer la racine de A en la racine de B auquel il ajoute le coût de comparer par le même algorithme les sous arbres $A_1, A_2, ..., A_n$ de A aux sous arbres $B_1, B_2, ..., B_m$. L'algorithme est donc invoqué récursivement jusqu'à arriver aux feuilles des arbres. La complexité de cet algorithme est de l'ordre de $(n.m.d)$ où n et m sont les

nombres de nœuds respectifs de A et B et d est le maximum des profondeurs des deux arbres.

2.2. Algorithme de Tai

Tai utilise une approche de programmation dynamique non récursive basée sur les opérations changer nœud par un autre, insérer nœud et supprimer nœud. Cet algorithme parcourt l'arbre en pre-ordre. Le premier nœud visité est la racine puis, récursivement, tous les sous arbres sont traversés en pré-ordre de gauche à droite jusqu'à atteindre les feuilles.

Pour comparer les arbres, Tai utilise des associations entre les nœuds. La paire de nœuds dans une association peut signifier une opération de changement du premier nœud par le deuxième ou une simple connexion si les nœuds ont la même étiquette. Si pour un nœud n du premier arbre il n'existe aucune association avec un nœud du deuxième arbre, ceci est équivalent au fait que n va être supprimé. De même si pour un nœud m du deuxième arbre il n'existe aucune association, ceci est équivalent au fait que m va être inséré. L'important dans l'algorithme de Tai est qu'il faut prendre en considération la structure (la hiérarchie) de l'arbre. En effet, chaque nœud de l'arbre est en relation avec les autres nœuds du même arbre. Cette relation doit être prise en compte lors de l'élaboration des associations. La figure 1 montre un exemple d'association illégale entre les deux nœuds d'étiquettes D.

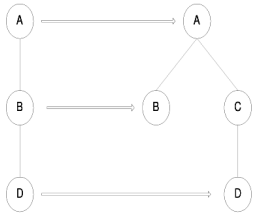

Figure 1. *Exemple d'une association illégale*

Les résultats de cet algorithme au niveau de la liste des opérations et du coût sont exacts. La complexité de cet algorithme est de l'ordre de $o(n.m.d^2.d^{'})$ où n et m sont respectivement le nombre de nœuds du premier et du deuxième arbres et d et $d^{'}$ leurs profondeurs respectives.

2.3. Algorithme de Zhang & Shasha

Cet algorithme est très similaire à celui de Tai au niveau des définitions, des associations et des opérations élémentaires [7]. La différence se situe au niveau du parcours et de l'énumération des nœuds.

Le calcul des différents coûts est basé sur l'utilisation de nœuds spécifiques dans les arbres appelés " keyroots ", c'est l'ensemble des nœuds qui ont un frère gauche en plus de la racine. Cette approche n'a pas besoin de vérifier les associations illégales puisque le coût de la transformation d'un nœud n'est calculé qu'après avoir déterminé les différents coûts des opérations sur tous ses descendants. La complexité de cet algorithme est de l'ordre de $o(n.m.d.d^{'})$ où n et m sont respectivement le nombre de nœuds du premier et du deuxième arbres, et d et $d^{'}$ leurs profondeurs respectives.

3. Traitement de la structure des documents XML

Définitions Un document structuré au format XML est modélisé par un arbre composé d'un nœud racine, de nœuds avec descendance, et d'arcs qui expriment les liens hiérarchiques entre ces nœuds. Un arbre D est un triplet $< R, N, \Omega >$ Où $R \in N$ est le nœud racine, N est un ensemble fini de nœuds et Ω est une relation qui modélise les arcs de l'arbre : $n_1 \Omega\, n_2 \Leftrightarrow n_1$ est le père de n_2.
Ω est une relation non réflexive, anti-symétrique et non transitive vérifiant :

– $\forall\, n \in N/\{R\}, \exists!\, n^{'} \in N/\{n\}, n^{'} \Omega\, n$:
– Si $N \neq \{R\}, \exists n \in N/\{R\}, R\, \Omega\, n$:
– $\{n \in N, n\, \Omega\, R\} = \emptyset$: Un arbre XML admet une seule racine.

Pour naviguer facilement dans l'arbre, on caractérise chaque nœud de l'arbre par une valeur de pré-ordre, un ou plusieurs attributs et un type (nœud élément ou nœud texte). Les attributs sont caractérisés par leurs valeurs. Un nœud est alors un quintuplet : $n = (p, t, A, w, i)$ où p est le nom de la balise, i est l'index du nœud selon le parcours en pré-ordre, t est le type de nœud, A est un ensemble d'attributs et w est un poids, il mesure la relation entre le nœud et son père.

Transformation et assignement des poids aux arcs Cette étape consiste à associer un index à chaque nœud et de propager les liens entre les différents niveaux de chaque arbre. Chaque lien recevra un poids selon la profondeur du lien comme illustre la figure 2.

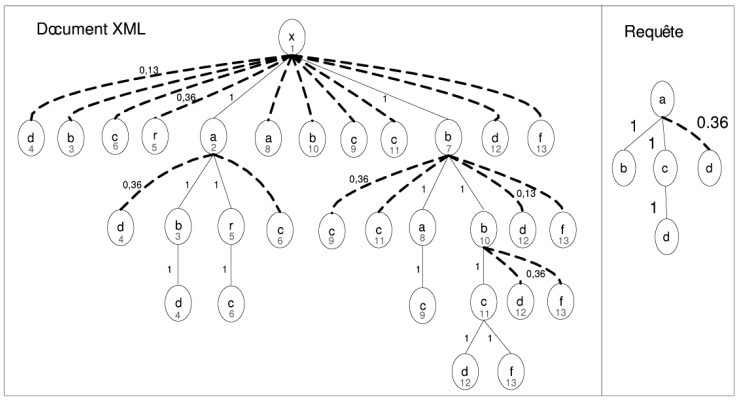

Figure 2. *Arbres étendus et poids des arcs*

Les arcs virtuels sont appliqués seulement sur les nœuds répondant à la structure (nœuds de type élément). Ils expriment les relations (nœud-descendant) en de nouvelles relations où les descendants d'un nœud seront définis comme des fils directement liés à ce nœud.
En partant de la racine, la transformation consiste à relier chaque nœud à ses descendants. A Chaque arc virtuel est assignée une valeur de poids. La valeur du poids d'un arc virtuel est définie selon la distance (nombres de descendants) qui sépare le nœud de son descen-

dant. Le poids d'un arc virtuel est défini par la fonction f :

$$\begin{array}{rccl} f: & C & \rightarrow &]0,\ 1] \\ & (n,\ n^{'}) & \mapsto & exp(1-d(n,\ n^{'})) \end{array}$$

où C est l'ensemble des couples $(n, n^{'})$ tels que n est un descendant (direct ou indirect) de $n^{'}$ et $d(n, n^{'})$ mesure la distance en profondeur entre n et $n^{'}$.

Identification de fragments pertinents Nous avons mis en œuvre un algorithme permettant de rechercher dans l'arbre relatif au document des fragments ayant une structure similaire à celle de la requête. Cet algorithme part de l'élément racine de la requête, restitue tous les éléments dans le document ayant le même nom et on construit pour chacun un arbre résultat, puis pour chaque fils d'un nœud d'un arbre résultat ayant le même nom nom, il alimente la structure de celui-ci et récursivement cette opération est itérée jusqu'aux nœuds feuilles. Les fils considérés sont ceux ayant un arc virtuel dans l'arbre étendu de chaque représentation (arbre et requête). Au fûr et à mesure de la construction, un score sur la structure est calculé (rsv_s). Chaque alimentation de la structure d'un arbre résultat engendre l'ajout du produit du poids de l'arc relatif à la requête par celui relatif au document. Plus le poids d'un arc est élevé, plus la relation est forte et par conséquent le score de la structure est élevé. L'algorithme 1 illustre la recherche de fragments pertinents selon la structure.

4. Manipulation du contenu des documents XML

Le contenu textuel n'apparait que dans les nœuds feuilles. Cependant, un nœud interne peut être pertinent même s'il ne contient aucun terme d'indexation : la pertinence ne provient pas seulement de la structure. Afin de rétablir l'ordre entre les nœuds pour que les feuilles ne soient pas favorisés aux nœuds internes, ceux-ci doivent avoir un score relatif au contenu.

On distingue dans la littérature deux principes courants, la plupart des modèles de recherche d'information utilisent la propagation des scores : on propage le score d'un nœud pertinent à une requête (en terme de contenu) à ses ancêtres [8] [9] [10]. Les autres modèles se basent sur la propagation du contenu [11] [12] au lieu de la propagation du score, on propage le contenu d'un nœud à un autre via les liens de descendance et on calcule son score indépendamment. Nous traduisons le contenu de chaque nœud feuille par un ensemble de termes pondérés, ceux-ci seront propagés vers les ancêtres de ce nœud tout en diminuant leurs poids en fonction de la distance parcourue au moment de la propagation.
Par ailleurs, un nœud interne peut recevoir du contenu à partir de plusieurs nœuds feuilles, ceux-ci étant dans ce cas ses descendants, en particulier, l'élément racine qui reçoit le contenu de toutes les feuilles de l'arbre XML. Or le même terme peut appartenir à plusieurs nœuds, le poids final associé à un nœud interne sera alors la somme de tous les poids reçus par les nœuds feuilles descendants de celui-ci :

Algorithm 1 Recherche par la structure

1: $q \in \tau$ {L'arbre relatif à la requête}
2: $d = (E_o, r_o, R_o) \in \tau$ {L'arbre relatif au document}
3: $Q = (E_q, r_q, R_q) \leftarrow f(q)$
4: $D = (E_d, r_d, R_d) \leftarrow f(d)$
5: $F \leftarrow \emptyset$ {Ensemble de résultats initialement vide}
6: **pour chaque** $n_q = (b_q, A_q, t_q) \in E_q$ **faire**
7: **pour chaque** $n'_q \in E_q, \exists x \in R^+ (n'_q, n_q, x) \in R_q$ {n'_q est un ancêtre de n_q} **faire**
8: **pour chaque** $(f_q = (E'_q, r'_q, R'_q), f_d = (E'_d, r'_d, R'_d), C, p) \in F, n'_q \in E'_q$ {les fragments qui peuvent être enrichis par le nouveau nœud n, C est un ensemble de couple de nœuds, p est la valeur de pertinence calculé au fûr et à mesure de la recherche} **faire**
9: soit $n'_d \in E'_d, (n'_q, n'_d) \in C$
10: $F \leftarrow F - \{(f_q, f_d, C, p)\}$
11: $f \leftarrow (f_q, f_d, C, p)$
12: **pour chaque** $n_d = (b_q, A_d, t_d) \in E_d, \exists y \in R^+$ où $(n'_d, n_d, y) \in R_d$ et $n_d \notin \{n = (b_q, A, t) \in E'_q, \exists z \in R^+ (n'_q, n, z) \in R_q$ et $b = b_q\}$ **faire**
13: $C \leftarrow C \cup \{(n_d, n_q)\}$ {Le nouveau couple ajoutés au fragment f}
14: $E'_q \leftarrow E'_q \cup \{n_q\}$
15: $E'_d \leftarrow E'_d \cup \{n_d\}$
16: $p \leftarrow p + x \times y$ {Le score de la structure p est mise à jour}
17: $R'_q \leftarrow R'_q \cup \{(n'_q, n_q, x)\}$
18: $R'_d \leftarrow R'_d \cup \{(n'_d, n_d, y)\}$
19: $F \leftarrow F \cup \{(f_q, f_d, C, p)\}$
20: **fin pour**
21: **fin pour**
22: **fin pour**
23: **pour chaque** $n = (b_q, A, t) \in E_d - \bigcup_{(f_q, f_d = (E'_d, r'_d, R'_d), C, p)} E'_d$ {nouveaux fragements, contenant seulement le nœud n} **faire**
24: $F \leftarrow F \cup \{((\{n_q\}, n_q, \emptyset), (\{n\}, n, \emptyset), (n_q, n), 0\}$
25: **fin pour**
26: **fin pour**

$$w(p,n) = idf_p \times \sum_{n\,\Omega\,c_1\,\Omega\,c_2 \ldots c_k} \frac{tf(p,c_k)}{d(n,c_k)}$$

où $w(p,\ n)$ est le poids du terme p dans le nœud n, $tf(p,\ c_k)$ est la fréquence du terme p dans le nœud c_k (descendant de n) et idf_p est inversement proportionnel au nombre de nœuds de la collection contenant le terme p :

$$idf_p = \frac{N}{N_p}$$

où N est le nombre de documents dans la collection et N_p est le nombre de documents contenant le terme p. $n\Omega c_1\Omega c_2 \ldots c_k$ est une branche de l'arbre en question ($n_i\Omega c_j$ signifie que n_i est le père de n_j).

Notre approche sépare entre la recherche sur le contenu et la structure, nous utilisons une combinaison linéaire pour calculer le score final r où r_c (resp. r_s) est le score sur le contenu (resp. score sur la structure) le score final est donné par :

$$r = \alpha \times r_c + (1 - \alpha) \times r_s$$

La valeur de $\alpha \in [0,\ 1]$ est un paramètre permettant de renforcer la recherche selon le contenu ou la structure. Cette valeur est définie par l'expérimentation mais dans des conditions réelles, elle peut être définie par l'utilisateur.

5. Expérimentations

Les mesures d'évaluation utilisées dans la campagne INEX'2005 sont basées sur les mesures **XCG** et **ep/gr** [13]. Pour un rang donné i, le gain cumulé $nxCG[i]$ reflète le gain relatif de l'utilisateur accumulé jusqu'à ce rang, comparé à ce qu'il aurait du atteindre si le système avait produit une liste triée optimale.
L'effort-précision (ep) à une valeur donnée de gain-rappel (gr) mesure l'effort d'un utilisateur pour atteindre un gain relatif au gain total qu'il peut obtenir. La moyenne non interpolée $MAep$ (Mean Average Effort Precision) d'effort-precision est utilisée pour moyenner les valeurs d'effort-précision pour chaque rang auquel un élément pertinent est renvoyé.

Les tableaux 1 et 2 illustrent les résultats obtenus pour la tâche COS avec et sans chevauchement. Les requêtes de cette tâche prennent en compte les contraintes structurelles, pour cette raison nous avons testé plusieurs valeurs de α, la meilleure valeur est 0.6. Nos résultats (NA) ont été comparés avec la moyenne (Moy) des résultats officiels des participants d'INEX.

Quantification	nxCG[10]		nxCG[25]		nxCG[50]		MAep	
	NA	Moy	NA	Moy	NA	Moy	NA	Moy
f_{strict}	0.0500	0.0268	0.0493	0.0418	0.0835	0.0491	0.0176	0.0113
f_{gen}	0.1593	0.2098	0.1632	0.1917	0.1688	0.1724	0.0591	0.0462
$f_{genLifted}$	0.1720	0.2275	0.1757	0.2049	0.1760	0.1792	0.0363	0.0337

Tableau 1. *Résultats comparatifs avec la moyenne des résultats officiels d'INEX'2005, pour la tâche COS (sans chevauchement)*

Quantification	nxCG[10]		nxCG[25]		nxCG[50]		MAep	
	NA	Moy	NA	Moy	NA	Moy	NA	Moy
f_{strict}	0.0363	0.0471	0.0441	0.0590	0.0535	0.0713	0.0093	0.0197
f_{gen}	0.1005	0.1777	0.0950	0.1508	0.1011	0.1444	0.0361	0.0576
$f_{genLifted}$	0.1060	0.1910	0.0953	0.1559	0.0971	0.1473	0.0380	0.0651

Tableau 2. *Résultats comparatifs avec la moyenne des résultats officiels d'INEX'2005, pour la tâche COS (avec chevauchement)*

6. Conclusion

Nous avons présenté dans cet article une approche de recherche d'information strcturée. L'approche consiste à comparer d'une manière flexible chacune des représentations de la requête et du document. Le flexibilité dans notre approche est mise en relief au moment de l'indexation des documents XML du corpus. Nous avons proposé une méthode de propagation de termes d'un nœud à ses ancêtres. Nous avons effectué des expérimentations pour évaluer notre approche sur un corpus issu de la campagne d'évaluation INEX'2005, pour des requêtes de type COS. Nous avons obtenu les meilleurs résultats pour les quels ces derniers ne tiennent pas compte du chevauchement où nous étions toujours au dessus de la moyenne par rapport aux participants d'INEX, dans la mesure des moyennes de précision (MAep). Notre modèle permet grâce au paramètre α de renforcer la recherche sur le contenu en l'augmentant ou sur la structure en le diminuant. A travers les résultats, nous enregistrons une nette amélioration de la qualité de recherche en tenant compte de la structure, contrairement à ce qui a été constaté dans la littérature.
Nous envisageons de tester notre modèle pour le corpus Wikipédia issu d'INEX, et de tenir compte des nombres de descendants au moment de la propagation du texte dans un document XML en plus de la distance entre les nœuds.

7. Bibliographie

[1] N. FUHR, T. ROLLEKE, « A Probabilistic Relational Algebra fort he Integration of Information Retrieval and Database Systems », *ACM TOISe*, vol. 15, n° 1, 1997.

[2] R. W. LUK, H. LEONG, T. S. DILOON, W. B. CROFT, J. ALLAN, « A survey in indexing and searching XML documents », *Journal of the American Society for Information Science and Technology*, 2002.

[3] N. BRADELEY, « The XML Comanion », *Addison-Weseley Professional Publisher*, 2001.

[4] W3C, « EXtensible Markup Language (XML) 1.0 », *Technical report, World Wide Web Consortium (W3C), TEchnical report*, 1998.

[5] WORD WIDE WEB CONSORTIUM (W3C) RECOMMANDATION, « Extensible Markup Language (XML) », *https ://www.w3.org/TR/2004/REC-xml-20040204/*, 2004.

[6] S.M. SELKOW, « The tree-to-tree edition problem », *Information processing letters/*, 1977.

[7] D. SHASHA, K. ZHANG, « Fast parallel algorithms for the unit cost editing distance between trees », *ACM Symposium on Parallel Algoritms and Architecture*, 1989.

[8] N. FUHR, K. GROSSJOHANN, « XIRQL : A query language for information retrieval in xml documents », *ACM SIGIR conference on research and development in Information Retrieval*, 2001.

[9] N. GOVERT, M. ABOLHASSANI , N. FUHR, « Content-oriented XML retrieval with HyRex », *Proceedings of INEX2002, Dagstuhl, Germany*, 2002.

[10] K. SAUVAGNAT, M. BOUGHANEM, « XFIRM : Flexible Information Retrieval Model for Indexing and Searching XML documents », *Proceedings of ECIR*, 2004.

[11] T. SCHILEDER, H. MEUSS, « Querying and ranking XML documents », *Journal of the American Society for Information Science and Technology*, 2002.

[12] Y. MASS, M. MANDELBROD, « Retrieving the most relevant XML compnents », *Proceedings of INEX2003, Dagstuhl, Germany*, 2003.

[13] INEX, « INEX - INitiative for the Evaluation of XML Retrieval », *http ://inex.is.informatik.uni-duisburg.de*, 2005.