Traitement des requêtes co (content only) sur un corpus de documents xml

Samia Berchiche-Fellag[1], Mohand Boughanem[2]

[1] Université Mouloud Mammeri de Tizi-Ouzou, 15000 Tizi-Ouzou, Algérie.
samfellag@yahoo.fr
[2] IRIT - SIG-RI, 118 Route de Narbonne, 31 062 Toulouse Cedex 4. bougha@irit.fr

Résumé. L'interrogation des documents XML peut se faire de deux manières : avec des requêtes portant sur le contenu (CO Content Only) ou avec des requêtes portant sur le contenu et la structure (CAS Content And Structure). Les requêtes CO sont celles qui sont le plus facilement formulée par l'utilisateur mais restent les plus difficilement traitées par le système de recherche d'information. En effet l'utilisateur ne donnant aucune indication au système de recherche sur la taille de l'unité d'information qu'il désire voir retourner, c'est à ce dernier de décider de la granularité de l'information appropriée à la requête formulée. La problématique qui en découle est : Comment retourner l'unité d'information pertinente ? C'est précisément à cette question que nous allons donner des réponses dans le présent article en proposant une méthode de propagation de termes des nœuds feuilles vers la racine du document XML. Pour ce faire nous considérons qu'un document XML est un arbre composé de nœuds et donc l'unité d'information considérée est un sous arbre.

Mots clés : XML, propagation de termes, requêtes CO, unité d'information

KeyWords : XML, word propagation, CO request, information unit

1. Introduction

L'apparition de nouveaux standards comme XML, permet de séparer la structure logique d'un document de son contenu. Un document est ainsi caractérisé par un contenu informationnel (du texte) et des contraintes structurelles (balises). Ce type de documents ne peut être, cependant, exploité efficacement par les techniques classiques de recherche d'information. En effet, ces dernières traitent le document comme un granule d'information indivisible, alors que le format standard XML permet d'envisager des granules d'information plus fins, tant du point de vue de la représentation que de l'accès. Ceci nécessite alors, la création de nouveaux outils et de nouveaux modèles capables de restituer des *unités d'information* (non plus le document) pertinentes à une requête utilisateur.

Différents modèles de RI ont été proposés et adaptés pour le traitement des documents XML, et ont tenté de répondre efficacement aux requêtes utilisateurs en renvoyant les *unités d'information* (pas le document entier) les plus pertinentes. Ces unités dépendent fortement des requêtes utilisateurs qui peuvent être formulées de

deux manières : par de simples mots clés on parle alors de requêtes orientée contenu (CO pour **C**ontent **O**nly), ou avec en plus des contraintes sur la structure on parle alors de requêtes orientées contenu et structure (CAS pour **C**ontent **A**nd **S**tructure).
L'interrogation des documents XML avec des requêtes CAS suppose une connaissance au moins partielle de la structure des documents à considérer puisque les requêtes formulées possèdent des conditions sur la structure des documents qu'on veut voir retourner par le système. Ces conditions permettent d'aider le système à localiser de manière exacte l'unité d'information recherchée. Dans le cas des requêtes CO ceci n'est pas envisageable puisque aucune condition n'est considérée, c'est donc au système de recherche d'information (SRI) de décider de la granularité d'information appropriée à retourner à l'utilisateur. En effet, le SRI doit pouvoir identifier les *unités d'informations* les plus pertinentes à une requête utilisateur.
La pertinence dans le cadre de la recherche d'information structurée est estimée selon deux grandeurs : l'*exhaustivité* et *la spécificité* [1]. La *spécificité* mesure si tout le contenu de l'unité d'information concerne la requête. L'*exhaustivité* mesure si toutes les informations requises dans la requête sont présentes dans l'unité d'information.

Le problème crucial dans la recherche avec des requêtes CO est : *comment identifier les unités d'information pertinentes ?* Sachant que ces unités doivent être de taille appropriée ni trop grande sous peine que l'information recherchée soit noyée au milieu d'autres sujets ni trop petite sous peine de ne pas être informative. Nous nous intéressons dans cet article à cette problématique que nous allons tenter de résoudre en proposant une méthode de propagation de termes avec leurs poids associés des nœuds feuilles vers la racine du document XML afin d'identifier les unités d'information pertinentes recherchées.

2. Etat de l'art

Nous allons présenter brièvement dans cette section quelques travaux se rapportant à la recherche d'information avec des requêtes orientée contenu dans un corpus XML.

Une approche basée sur la correspondance d'arbres *« tree matching »*, l'arbre du document et l'arbre de la requête est proposée par Schlieder, Meuss et Naumann [2], [3]. Cette correspondance traduit la distance entre les arbres. Grabs et schek [4] proposent quand à eux de mesurer l'importance d'un terme dans un élément en fonction de son importance dans les éléments de même type. Dans le but de retourner le fragment de document répondant de manière pertinente à la requête utilisateur, le score de pertinence d'un nœud (élément) est calculé. Pour ce faire, certaines approches, adoptent la méthode de propagation des termes [5], d'autres, la méthode de propagation de pertinence (score) [6] en utilisant une combinaison linéaire des scores des enfants appelées « *maximum-by-category* » et « *summation* ». Sauvagnat [7], propose dans le même ordre d'idées une méthode de propagation de score des nœuds feuilles vers la racine du document dans son modèle XFIRM. Fuhr &al [8][9]utilise une méthode de propagation de poids des nœuds les plus spécifiques dans l'arbre du document en utilisant un facteur d'*augmentation* (multiplication par un facteur donné) basée sur le modèle probabiliste. Il en découle que, quelque soit l'approche adoptée, le score de pertinence d'un nœud dépend fortement du score de ses descendants.

3. Traitement des requêtes

Nous considérons *qu'un document XML est un arbre, composé de nœuds interne, nœuds feuilles et d'attributs* (voir figure suivante).

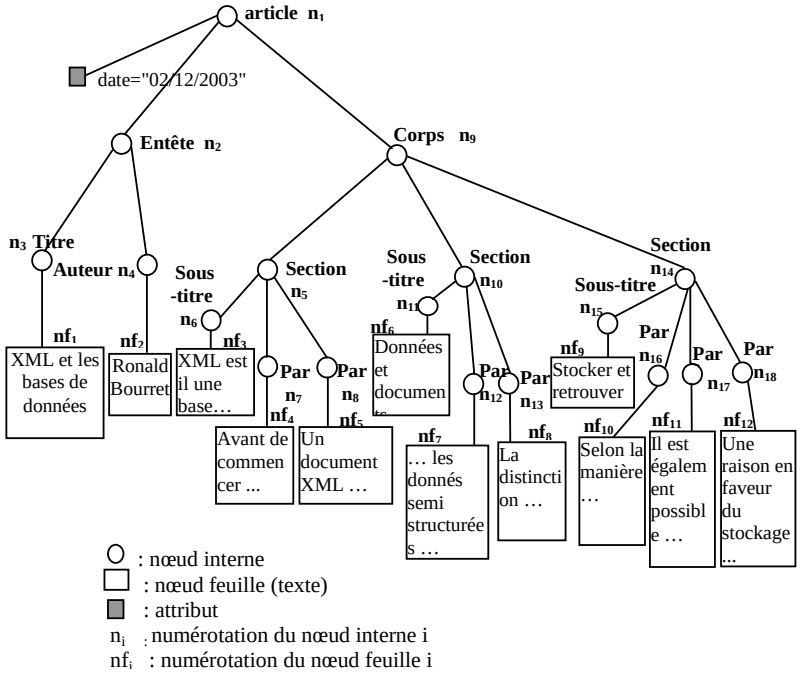

Fig. 1. document article.xml

Le traitement des requêtes que nous effectuons a pour but de renvoyer les unités d'informations les plus *spécifiques* et les plus *exhaustives* à l'utilisateur.
Les requêtes que l'utilisateur peut formuler, sont composées de *simples mots clés sans précision aucune sur la structure*, et c'est au système, de décider de la granularité appropriée de l'information à renvoyer.

Le traitement de requêtes CO que nous proposons de faire, s'effectue comme suit :

- la première phase, consiste à pondérer les termes des nœuds feuilles,
- la seconde phase concerne, l'élagage de l'arbre du document, en ne conservant que les nœuds informatifs,
- la troisième phase, consiste à propager les termes bien distribués dans les nœuds feuilles, vers les nœuds ascendants. Afin de pouvoir identifier les unités d'information, pertinentes et informatives,

- enfin, la dernière phase consiste, à calculer le score de pertinence des unités d'information ainsi identifiées, avec la requête et à présenter les résultats obtenus, par ordre décroissant des scores.

Le calcul du score d'un nœud et la pondération des termes dans les nœuds sont des éléments prépondérants dans la phase d'évaluation de la pertinence d'un nœud vis-à-vis d'une requête. Avant de présenter le processus de propagation que nous avons adopté, nous commençons par décrire ces deux points.

3.1. Calcul du score des nœuds

La requête utilisateur est représentée par des mots clés éventuellement pondérés. On peut avoir la représentation suivante de la requête :
$q = \{(t_1,p_{q1}), \ldots\ldots ,(t_M,p_{qM})\}$ avec t_k un terme de la requête et p_{qk} le poids de t_k dans la requête q et M le nombre de termes dans la requête.
Le score d'un nœud n_i identifié dans l'arbre, est calculé avec une fonction de similarité $RSV(q,n_i)$ du modèle vectoriel (Inner product) comme suit :

$$RSV(q, n_i) = \sum_{k=1}^{M} p_{qk} \times p_{nik} \ . \qquad (0)$$

ou p_{qk} et p_{nik} sont respectivement le poids du terme k dans la requête q et le nœud n_i.
Le poids P_k d'un terme k dans les nœuds feuilles, et dans la requête est calculé grâce à la formule de pondération que nous avons proposée :

$$P_k = tf_k \times Idf_k \times ief_k \ . \qquad (1)$$

Où :
tf_k : term frequency (fréquence d'un terme k dans le nœud feuille ou dans la requête)
Idf_k : Inverted document frequency = fréquence inverse de document = $\log\left(\frac{N}{n}\right)$
Ief_k: Inverted element frequency = fréquence inverse d'éléments
= $\log\left(\frac{Ne}{ne} + \alpha\right)$ avec $0.5 \le \alpha \le 1$ est une adaptation de Idf à la granularité de l'information à traiter qui n'est plus le document mais l'élément.
N : Nombre total de documents dans la collection
n : Nombre de documents contenant le terme k
Ne : Nombre total de nœuds feuilles dans le document
ne : Nombre de nœuds feuilles contenant le terme k dans le document
Les facteurs, tf, idf et ief et leur combinaison peuvent prendre plusieurs formes. Nous avons fait une série d'expérimentation afin d'évaluer et mesurer les performances des différentes combinaisons.

3.2. Propagation des termes

Afin d'identifier la partie du document qui répond le mieux à la requête utilisateur, nous proposons, une méthode de propagation des termes et des poids, en partant des nœuds feuilles jusqu'à la racine du document.

La question qui en découle est : *quels termes propager et comment ?*
Dans ce but, nous introduisons une notion fondamentale, qui est *l'informativité d'un nœud*.
Un nœud est dit informatif s'il est porteur de suffisamment d'informations pour répondre efficacement à une requête utilisateur. S'il aisé de définir l'informativité, encore faut il la mesurer ! L'intuition guidant cette mesure étant la *taille* d'un nœud (c'est-à-dire le nombre de termes qu'il contient). En effet, un nœud qui ne contient *que* les termes de la requête, est s*pécifique* à cette requête.
Il est cependant, *non informatif* car il n'apporte pas l'information requise à l'utilisateur (un nœud *titre* par exemple peut être pertinent pour une requête mais pas informatif). Nous définissons, à cette fin, un *seuil* qui consiste en, le *nombre de termes minimal* qu'un nœud doit avoir pour être considéré comme informatif.

Deux cas dans la propagation des termes, sont à considérer : *le premier*, consiste à traiter les nœuds dont le nombre de termes est inférieur au seuil, et *le second*, consiste à prendre en compte les nœuds dont le nombre de termes est supérieur au seuil. Il est clair que nous n'avons aucun moyen théorique pour déterminer ce seuil. Dans notre cas, nous proposons de le fixer par expérimentation.

A. Cas ou le nombre de termes des nœuds est inférieur à un seuil

L'arbre du document est parcouru en commençant par les nœuds feuilles. Durant le parcours, lorsqu'un nœud visité a un nombre de termes inférieur à un seuil (à définir), ce nœud est supprimé de l'arbre et son contenu remonté vers son nœud parent. Ce procédé se fait de manière récursive jusqu'à atteindre (et éventuellement dépasser) le seuil, ou atteindre le nœud racine du document, ou atteindre un nœud interne, dont le nombre de termes d'au moins un de ses fils est supérieur ou égal au seuil. L'algorithme illustratif est comme suit :

Algorithme de propagation cas A

1. Commencer par les nœuds feuilles
2. Visiter un nœud de l'arbre
3. Si le nombre de termes du nœud est inférieur au seuil alors
3.1 remonter les termes vers le nœud parent avec leurs poids respectifs,
3.2 supprimer le nœud
4. reprendre les étapes à partir de 2 jusqu'à atteindre ou dépasser le seuil, ou atteindre le nœud racine ou atteindre un nœud interne dont le nombre de termes, d'au moins un de ses fils est supérieure ou égal au seuil.

B. Cas ou le nombre de termes des nœuds est supérieur au seuil

L'idée sous-jacente est la suivante : *« des termes bien distribués dans les éléments enfants d'un élément peuvent être représentatifs pour cet élément »*.
Deux cas peuvent se présenter, un nœud peut avoir plusieurs nœuds fils, ou n'en posséder qu'un seul (seul un nœud feuille n'a pas de fils).

1. **Cas ou un nœud possède plusieurs nœuds fils:** de manière intuitive, on peut penser qu'un terme d'un nœud peut être représentatif pour son parent, s'il existe au moins, dans un de ses frères. Cette intuition à elle seule ne suffit pas, car il faudrait tenir compte d'un facteur très important, qui est la pondération des termes dans les nœuds. En effet, un terme peut appartenir à tous les nœuds fils d'un élément, mais si son poids est faible, par rapport à l'ensemble des termes des nœuds fils. Il ne pourra pas être discriminant pour ces nœuds. C'est dans cet ordre d'idée, que nous avons adjoins une autre intuition (en plus de la précédente) qui consiste à ne prendre en considération, que les termes dont le poids moyen au niveau des nœuds fils ou ils apparaissent, est compris entre, le poids moyen des termes des nœuds fils et leurs poids maximal.

2. **Cas ou un nœud possède un seul nœud fils :** L'intuition utilisée dans ce cas, est de considérer qu'un terme d'un nœud fils, ne peut être discriminant pour son parent, que si son poids (du terme), est compris entre le poids moyen des termes du nœud fils et leurs poids maximal.

Le terme vérifiant les intuitions de l'un des cas, 1 ou 2, sera supprimé de son nœud d'origine et remonté vers son nœud parent considéré. Son poids dans le nœud parent, est égal à son poids moyen au niveau de tous les nœuds fils, dans le cas 1, ou à son poids dans le nœud fils considéré, dans le cas 2.
Ces intuitions sont formalisées de la manière suivante :

1. Soit *e* un nœud possédant *plusieurs* nœuds fils *e'*. Soit *t* un terme d'un nœuds fils *e'* de *e*. *P(t,e')* le poids du terme *t* dans le nœud *e'*, calculé avec la formule (1). *t* peut être remonté vers *e*, si *t* existe dans au moins un nœud frère de *e'* et si la moyenne du poids de *t* dans les nœuds fils de *e* ou il apparaît, vérifie la condition suivante :

$$P_{moy} \leq \underset{e' \in enf(e)}{moy}(P(t,e')) \leq P_{max} \qquad (2)$$

Avec

$$P_{moy} = \frac{\sum_{e' \in enf(e)} \sum_{i=1}^{Nte'} p(t_i, e')}{Nt}. \qquad (3)$$

P_{moy} : le poids moyen des termes dans les nœuds *e'* fils de *e*

$$\text{moy}_{e' \in enf(e)}\left(P(t,e')\right) = \frac{\sum_{e' \in enf(e) / t \in e'} P(t,e')}{Ne'} . \quad (4)$$

Nte' : nombre de termes dans le nœud *e'*
Nt : nombre de termes dans tous les nœuds *e'* enfants de *e*
Ne' : nombre de nœuds *e'* contenant le terme *t*
P_{max} : le poids maximum des termes dans tous les nœuds *e'* enfants de *e*
Le terme *t* sera supprimé des nœuds fils *e'*, et remonté vers le nœud père *e*, et son poids dans *e* sera :

$$P(t,e) = \underset{e' \in enf(e)}{\text{moy}}(P(t,e')) . \quad (5)$$

2. Soit *e* un nœud possédant *un seul* nœuds fils *e'*. Soit *t* un terme du nœud *e'*. *P(t,e')* le poids du terme *t* dans le nœud *e'*. *t* peut être remonté vers *e*, s'il vérifie la condition (6) suivante :

$$P_{moy} \leq P(t,e') \leq P_{max} \quad (6)$$

avec

$$P_{moy} = \frac{\sum_{i=1}^{Nte'} P(t_i,e')}{Nte'} \quad (7)$$

P_{moy} : le poids moyen des termes dans *e'*
P_{max} : le poids maximum des termes dans *e'*
Nte' : nombre de termes dans le nœud *e'*

Le terme *t* sera supprimé du nœud fils *e'* et remonté vers le nœud père *e* en conservant son poids, donc :

$$P(t,e) = P(t, e') \quad (8)$$

Indiquons que, durant la remontée d'un terme *t* du nœud fils *e'* vers son parent *e*, celui ci peut s'y trouver déjà. Dans ce cas, le terme *t* sera supprimé du (ou des) nœud(s) fils *e'*, et son poids dans le nœud *e* sera égal, à la moyenne de son poids, dans le(s) nœud(s) fils *e'* et le nœud parent *e*, c'est-à-dire

$$P(t,e) = \frac{P_{(5)/(8)}(t,e) + P_0(t,e)}{2} \quad (9)$$

avec :

$P_0(t,e)$: poids initial du terme *t* dans le nœud *e*
$P_{(5)/(8)}(t,e)$: poids que devait avoir (s'il n'y existait pas) le terme *t* dans *e* calculé avec la formule (5) ou (8) selon le cas considéré.
Le processus de propagation des termes, se déroule de manière récursive des feuilles de l'arbre jusqu'à la racine.

Nous résumons ces différents cas dans l'algorithme suivant :

Algorithme propagation cas B

1. Commencer par les feuilles
2. visiter un nœud
3. lire un terme
4. si le nœud possède des frères alors
   4.1 si le terme existe dans au moins un nœud frère alors vérifier la formule (2)
4. sinon vérifier la formule (6)
5. si la formule (2 ou 6) vraie alors supprimer le terme du (des) nœud(s) fils et le remonter vers son parent
   5.1 si le terme existe dans le nœud parent alors le poids du terme se calcule avec la formule (9)
   1.1 sinon si la formule (2) vraie alors le poids du terme se calcule avec (5) sinon avec (8)
6. reprendre à partir de l'étape 3 jusqu'à lire tous les termes du nœud considéré
7. reprendre à partir de l'étape 1 jusqu'à atteindre le nœud racine du document

A l'issue de ce traitement, le score de similarité des termes de la requête avec les nœuds représentés par ces termes sera calculé, les résultats seront présentés par ordre décroissant des scores. Les sous arbres pertinents et informatifs seront ainsi présentés à l'utilisateur.

Exemple de traitement de requêtes

Afin de bien illustrer la méthode de propagation de termes que nous avons proposée, nous déroulons dans ce qui suit, un exemple de requête sur un exemple de document.
Reprenons le document précédant extrait de l'article de *Ronald Bourret* sur *"XML et les bases de données"*. Soit la requête suivante *"base XML native"*, composée de trois termes *base*, *XML* et *native*. Les nœuds feuilles contenant les termes de la requête sont : nf_1, nf_3, nf_4, nf_5, $\mathbf{nf_7}$, $\mathbf{nf_{10}}$, $\mathbf{nf_{11}}$, $\mathbf{nf_{12}}$, les quatre derniers nœuds cités contiennent *tous* les termes de la requête.

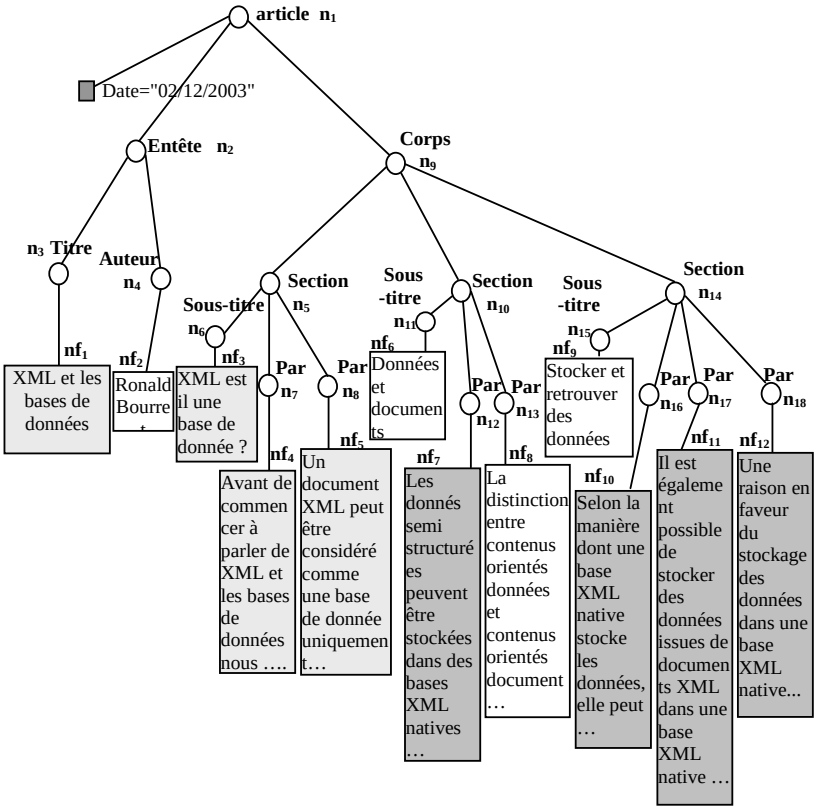

Fig. 2. Représentation en arbre du document article.xml après localisation des nœuds contenant les termes de la requête

Après application de l'algorithme A (suppression des nœuds non informatifs en considérant un seuil de 20 termes) sur le document XML considéré, on obtient un arbre ne contenant que des nœuds informatifs comme l'illustre la figure 3. Nous appliquons sur ce dernier sur l'arbre l'algorithme B et on obtient l'arbre illustré par la figure 4 dans lequel on retrouve :

Les unités d'information représentée par les termes *base*, *XML* et *native* sont :

- le nœud feuille nf_5,
- le sous arbre de racine n_5,
- le sous arbre de racine n_{12},
- le sous arbre de racine n_{14},
- et enfin l'arbre de racine n_1.

Après calcul du score de similarité des sous arbres ainsi localisés avec les termes de la requête, les unités d'informations exhaustives, spécifiques et informatives à retourner par ordre décroissant des scores sont :

- le sous arbre de racine n_{12},
- le sous arbre de racine n_{14} ,
- le nœud feuille nf_5,
- le sous arbre de racine n_5,
- et enfin plus général l'arbre de racine n_1.

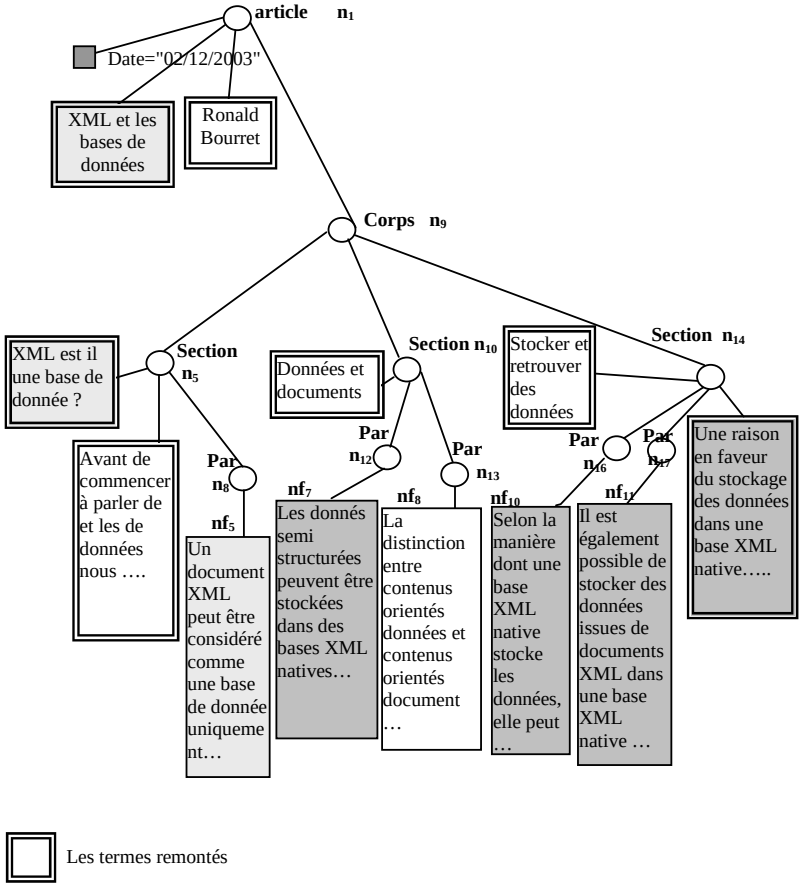

Fig. 3. Le document article.xml après suppression des nœuds non informatifs

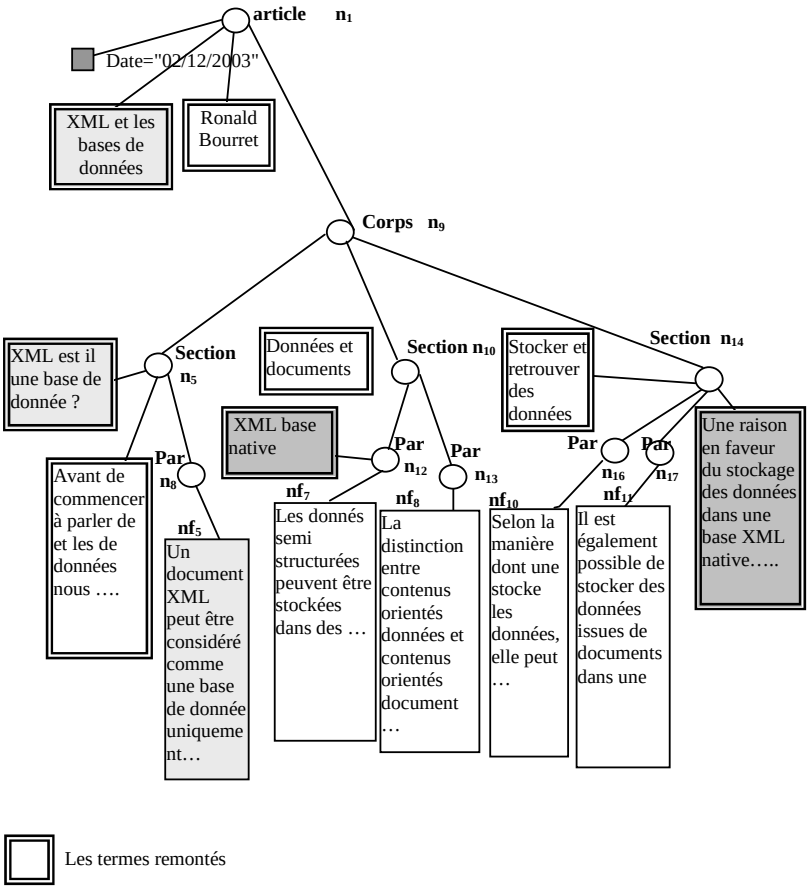

Fig. 4. Le document article.xml après application de l'algorithme B

Il est essentiel de noter que le traitement que nous venons d'effectuer peut se faire (est préférable d'être fait) de manière *statique* indépendamment de la requête. A l'arrivée d'une requête, seuls les scores seront calculés. Précisons aussi, que le traitement statique permet un gain de temps considérable, et un temps de réponse du système à une requête très appréciable, ce qui constitue l'un des atouts d'un SRI.

4. Conclusion

Dans cet article nous avons présenté notre contribution au problème de RI dans les documents semi structurés XML avec des requêtes CO en proposant une méthode de propagation des termes des nœuds feuilles vers la racine du document. En effet, cette méthode a pour but non seulement de retourner les unités d'information les plus *spécifiques* et *exhaustives* à une requête utilisateur mais surtout de renvoyer des unités *informatives* grâce à la contrainte sur le nombre de termes d'un nœud, que nous avons imposé. De plus cette méthode peut être traitée de manière *statique* c'est-à-dire avant la formulation d'une requête, et de ce fait, le traitement des requêtes devient aisé et leur temps d'exécution considérablement réduit.
Les résultats expérimentaux sur cette méthode de propagation sont en cours, néanmoins des tests manuels ont été faits et des résultats probants ont été obtenus.

5. Bibliograhie

1. M. Lalmas.: Dempster-Shafer's theory of evidence applied to structured documents: Modeling uncertainty. In: Proceedings of ACM-SIGIR, pp. 110-118. Philadelphia, 1997.
2. T.Schlieder and H.Meuss.: Querying and ranking XML documents. journal of the American society for information Science and Technology, 53(6) :489-503, 2002.
3. T.Schlieder and F.Naumann.: Approximate tree embedding for querying XML data. In: proceedings of the first annual workshop of INEX, Dagstuhl, Germany, December 2002.
4. T.Grabs and H .J.Scheck.: Flexible information retrieval from XML with Power DB XML. In : proceedings of the first annual workshop of INEX, pages 141-148, December 2002
5. H.cui, J-R.Wen, J-R.Chua.: Hierarchical indexing and flexible element retrieval for structured document. april 2003.
6. Vo Ngoc Anh, Alistair Moffat.: Compression and an IR approach to XML Retrieval. In: INEX 2002 Workshop Proceedings, p. 100-104, Germany, 2002.
7. Karen Sauvagnat. : Modèle flexible pour la recherche d'information dans des corpus de documents semi-structurés. Thèse Doctorat, Université Paul Sabatier de Toulouse, 2005
8. N. Fuhr, K. Grossjohann. : XIRQL, a query language for information retrieval in XML documents. In: proceedings of SIGIR 2001, Toronto Canada 2001.
9. N. Gövert, M. Abolhassanni, N. Fuhr, K. Grossjohann,.: Content-Oriented XML Retrieval with HyreX. In: INEX 2002 Workshop Proceedings, p. 26-32, Germany, 2002.