Modèle de recherche contextuelle orientée contenu pour un corpus de documents XML

Ounas ASFARI

SUPÉLEC, Département Informatique
Plateau de Moulon
3 rue Joliot-Curie
91192 GIF SUR YVETTE CEDEX
France
Ounas.asfari@supelec.fr

RÉSUMÉ. *Dans le cadre de corpus de documents XML, la recherche par mots-clés reste le moyen le plus utilisé pour un utilisateur dont le besoin d'information est vague, ou encore parce qu'il ne connaît pas précisément la structure des documents. Dans cet article nous présentons notre approche de recherche de nœuds pertinents à une requête orientée contenu "Content Only" composée de simples mots clés dans un corpus de documents XML en prenant en compte la pertinence contextuelle. Le processus de recherche que nous proposons repose sur une méthode de propagation de pertinence.*

ABSTRACT. *As a part of a corpus of XML documents, the keyword search is still the most widely used for a user in need of vague information, or when he does not know the precise structure of the documents. In this article, we present our approach which consists of searching the nodes relevant to a "Content Only" query composed of simple keywords in a large set of XML documents and taking into account the contextual relevance. The search process that we propose is based on a method of propagation of relevance.*

MOTS-CLÉS: *Recherche d'Information, documents semi-structurés, XML, XPath, pertinence contextuelle, propagation de pertinence, recherche orientée contenu.*

KEYWORDS: *information retrieval, semi-structured Documents, XML, XPath, contextual relevance, Propagation of relevance, content only.*

1. Introduction

Depuis quelques années l'évolution du web est accompagnée de l'importance récente d'XML comme format de représentation et de modélisation de l'information semi-structurée. De plus, l'émergence de nouveaux besoins dans la contextualisation de l'information dans les bibliothèques numériques, l'intranet, et le Web a engendré une évolution des Systèmes de Recherche d'Information (SRI). La croissance et l'importance des données semi-structurées ont incité les chercheurs à proposer des méthodes pour interroger des documents XML. Nous nous intéressons plus particulièrement aux Systèmes de Recherche d'Information SRI pour des corpus de documents semi-structurés et à améliorer la pertinence des réponses par rapport à une requête.

Cet article s'organise comme suit : dans la section 2, nous étudions un état de l'art, nous consacrons la section 3 à la présentation de notre approche de recherche des nœuds pertinents dans une collection de documents XML par rapport à une requête CO. La section 4, quant à elle, présente les étapes de réalisation et de mise au point de notre système de recherche et son évaluation. Enfin nous concluons cet article et donnons quelques perspectives à notre travail.

2. Etat de l'art

2.1. *La pertinence contextuelle*

La pertinence contextuelle consiste à prendre en compte le poids global du document dans le calcul de la pertinence d'un nœud, donc un nœud appartenant à un document fortement pertinent doit être mieux classé qu'un noeud se trouvant dans un document de pertinence moindre. Donc la pertinence d'un sous-arbre est liée à la pertinence du document dans lequel il se trouve.

2.2. *Les travaux existants*

Dans cette section nous nous somme intéressés en particulier à deux systèmes que nous avons estimé les plus importants pour le calcul de pertinence.

2.2.1. *Système XFIRM (XML Flexible Information Retrieval Model)*

L'objectif est de proposer un modèle (XFIRM) basée sur une méthode de propagation de pertinence permettant d'effectuer des recherches flexibles dans des corpus de document XML et de modéliser des documents possédant des structures différentes pour but de trouver les unités d'information les plus exhaustives et spécifiques répondant à une requête de l'utilisateur.(Boughanem et al., 2006)

2.2.2. *Système "Path-augmented Keyword Search for XML Documents"*

Ce système (Hsu *et al.*, 2004) décrit une méthode de recherche des parties de l'arbre d'un document XML qui répond le mieux à une requête CO. Cette méthode combine deux techniques de recherche de mots clés dans des documents : la proximité entre nœuds et l'expansion de requête, de plus elle introduit une technique de mesure pour déterminer la distance entre deux éléments nœuds dans le document XML, le but est de donner un score à chaque nœud du document en se basant sur une recherche sémantique des mots clés qui existent dans la requête et de la distance de ce nœud par rapport aux autres nœuds qui contiennent ou pas un des mots clés.

3. Modélisation de notre approche

3.1. *La pertinence contextuelle*

Pour réaliser la pertinence contextuelle nous proposons d'appliquer un tri sur la pertinence du document : on peut étudier l'impact du contexte de la manière suivante: (i) on calcule un score de pertinence pour tous les documents de la collection, grâce à un moteur de recherche, (ii) on calcule un score de pertinence pour tous les éléments de la collection, (iii) on trie les documents par ordre décroissant de pertinence, et (iv) pour chaque document, on trie par ordre décroissant de pertinence les éléments qu'il contient. De cette façon, les éléments sont d'abord triés en fonction de la pertinence du document auquel ils appartiennent puis en fonction de leur propre pertinence.

3.2. *Modélisation de la recherche contextuelle orientée contenu*

Notre hypothèse de travail est de considérer que l'utilisateur n'a pas de connaissances, à priori, sur la structure des documents XML qu'il veut interroger.par conséquent, nous nous attachons aux requêtes CO. Donc le challenge principal est d'identifier les parties de documents les plus pertinentes par rapport à une requête CO composée de simples mots clés et de définir une approche qui nous permet de choisir les fragments de documents qui seront retournés comme résultats pour cette requête. Nous nous sommes intéressés, à savoir comment trouver les sous-arbres pertinents pour une requête, la principale problématique soulevée est de savoir comment calculer la pertinence de ces sous-arbres. Nous nous sommes modélisés un système de traitement des requêtes CO qui comprend deux étapes :

- Une première étape consiste à évaluer la similarité des noeuds feuilles de l'index par rapport à la requête (on parle alors de calcul des poids des mots clés dans les noeuds feuilles).

- Une seconde étape consiste à rechercher les sous-arbres pertinents. La pertinence des sous-arbres est évaluée en propageant le poids des feuilles dans l'arbre du document.

3.3 *Algorithme*

Pour décrire ces deux étapes on considère qu'un document structuré ds_i est un arbre, composé de noeuds simples n_j, et de noeuds feuilles nf_j. On remarque que dans la recherche de mots clés, les noeuds feuilles ont une importance spéciale puisque ils sont porteurs de contenu, alors que les autres noeuds donnent simplement des indications de structure. Cette réflexion nous mène à choisir la manière d'indexer nos documents XML.

3.3.1 *Calcul les poids des mots clés dans les nœuds feuilles*

Chaque nœud d'un document XML a un identifiant unique dans le document qui est son chemin absolu. On calcule le nombre d'occurrences de chaque mot clé dans chaque document du corpus, et on donne un identifiant à chaque document.

On indexe les nœuds feuilles du document qui contiennent au moins un des mots clés, avec le poids de chaque mots clés de la requête, initialisé à 0, et si un mot clé est contenu dans un nœud feuille, la valeur du poids qui correspond à ce mot clé devient 1. Notre choix pour cette pondération binaire des poids des mots clés dans un nœud feuille est justifié par la simplification de calcul. Donc pour une requête CO composée de simples mots clés x_1, x_2,…x_i, on définit le poids d'un mot clé x_i dans un nœud feuille nf_k noté Poids comme suite:

$$Poids \quad (nf_k, x_i) = \begin{cases} 0 \\ ou \ 1 \end{cases}$$

Une fois le contenu d'un nœud feuille contient un des mots clés recherchés par la requête, la valeur de pertinence de ce noeud par rapport à ce terme devient non nul, et cette valeur doit propager d'un nœud descendant vers les noeuds ancêtres. Cette propagation permet de tenir compte du fait que le mot contenu dans un noeud feuille est également contenu dans ses ancêtres.

3.3.2 *Propagation de pertinence*

Pour construire l'index des nœuds ancêtres de ces nœuds feuilles, on calcule le score des poids des mots dans ces nœuds ancêtres par la propagation de pertinence depuis un noeud feuille à son père et ainsi de suite. L'équation de propagation de pertinence depuis un nœud feuille nf_k à son nœud ancêtre n_j, par rapport à un mot clé x_i, est définie par l'équation suivante notée propagation qui doit être une fonction de Poids du mot x_i dans le nœud feuille enfant et la distance entre les deux noeuds:

$$Propagation_i\ (nf_k\ ,\ n_j) = fonction(Poids\ (nf_k, x_i), Dist(nf_k, n_j), P_i)$$

Où P_i est l'ancien Poids du mot clé x_i dans le nœud n_j.
Sachant que la fonction qui va déterminer la distance entre deux nœuds jouera un rôle essentiel dans le calcul de pertinence des nœuds, alors que la distance augmente, la pertinence du nœud ancêtre doit diminuer pour que le nœud racine ne soit pas celui qui aura toujours le score le plus élevé. Donc logiquement dans cette équation la distance sera un paramètre dans le dénominateur (diviseur), donc le poids d'un

mot clé x_i dans un noeud n_j doit être propagé par un nœud feuille nf_k descendant comme suite: $\frac{1}{\max(\ Dist\ (n_j, nf_k))}$

Où nf_k est le nœud feuille descendant de ce nœud n_j qui contient le mot clé x_i. Mais cette équation donne le même poids pour un nœud simple qui contient un seul fils direct qui contient le mot clé car la distance sera considérée 1 et l'inverse de 1 est égal à 1. Donc pour s'assurer de la diminution de la valeur de pertinence lors de la propagation on modifie la relation précédente par :

$$\frac{1}{\max(\ Dist\ (n_j, nf_k) + 1)}$$

Donc l'équation de la propagation depuis un nœud feuille nf_k vers un de ces nœuds ancêtres n_j par rapport à un mot clé recherché *xi* s'écrira comme suite :

$$\Pr opagation_i(nfk_i, nj_i) = Poids(nfk_i, x_i) * \frac{1}{\max(Dist(nj, nfk) + 1)}$$

Une fois tous les poids des mots clés sont calculés pour tous les nœuds ancêtres des nœuds feuilles qui contiennent ces mots clés, le score de pertinence par rapport à la requête est calculé pour chacun des nœuds indexés selon que le document contient totalement ou partiellement les mots clés recherchés par la requête. Le score de pertinence d'un nœud n_j par rapport à une requête *Q=(x1, x2,…, xi)* est noté N :

$$N = Pertinence(n_j, Q) = \prod_{x_i \in Document} [Poids(n_j, x_i)]_i$$

Où *poids* (n_j,x_i) : poids de mot clé x_i dans le nœud n_j.

Ensuite les nœuds qui ont un score de pertinence non nul, seront trié par ordre décroissent, après, le système retournera soit le nœud le plus pertinent par document soit l'ensemble de tout les nœuds pertinent trié par ordre décroissent groupé par document, et les documents seront triés à leur tour par leur pertinence contextuelle par rapport à l'ensemble des mots clés de la requête.

4. Les études expérimentales

Pour la réalisation de notre système de recherche des nœuds pertinents dans un corpus de documents XML, à une requête CO, nous avons choisit de travailler avec XPath, Ce langage est bien supporté par les langages orientés objets comme java. Nous utilisons aussi DOM4J qui est API Open Source permet le travail avec XML, XPath et XSLT et il est plus complet, plus simple. Nous avons aussi développé une interface graphique avec la technologie Java/Swing.

L'efficacité et l'efficience:

Pour l'évaluation de notre approche il faudra faire des expérimentations sur un corpus qui contient une collection de fichiers XML où l'ensemble des documents pertinents à la requête dans la collection est |R|, les résultats retournés par notre système |A|, et |Ra| le nombre de documents pertinents renvoyés par le système, il faudra ensuite calculer la précision et le rappel comme suit:

Précision = |Ra| / |A| Rappel = |Ra| / |R|

Nous avons effectué des expérimentations sur un 1,73 GHz PC avec 1 Go de RAM. En comparaison la précision / rappel et le temps d'exécution de notre système à celui de "path augmented keyword search" (Hsu et al., 2004),qui combine deux techniques de recherche comme nous avons indiqué :la proximité entre nœuds "keyword proximity rank for XML" et l'expansion de requête "keyword expansion" (Hristidis et al., 2003).

La figure2 montre la moyenne de la précision pour 40 requêtes sous différents taux de rappels. La précision de notre système est plus élevée par rapport aux trois autres systèmes, c'est à notre avis est dû au nœud racine (root), dans le système "Path-augmented Keyword Search", est toujours sélectionnée dans l'arbre formant le résultat retourné à l'utilisateur, cela signifie que plus l'arbre du document est profond plus la réponse contiendra des nœuds inutiles à l'utilisateur. Malgré tout la précision du système "Path-augmented Keyword Search" est plus élevée par rapport aux deux autres systèmes "keyword proximity searche" et "keyword expansion".

Ensuite, nous varions la taille des données en supprimant certains documents XML pour chaque requête. Sachant que le nombre de données d'accès doit être identique à toutes les méthodes. La figure3 montre que notre système a le meilleur temps de réponse lorsque la taille des données est inférieure à 500 MB. Alors que le système "Path-augmented Keyword Search " est légèrement lent par rapport aux deux autres méthodes, Cela est dû au temps requis pour calculer la pertinence dans cette approche. Toutes les trois méthodes ont du temps de réponse pareil quand la taille de données dépasse 700MB, dans ce cas, notre système sera plus lent que les autres. En conséquence, l'amélioration de l'efficacité de notre système a été réalisée avec un coût raisonnable en efficience (performance).

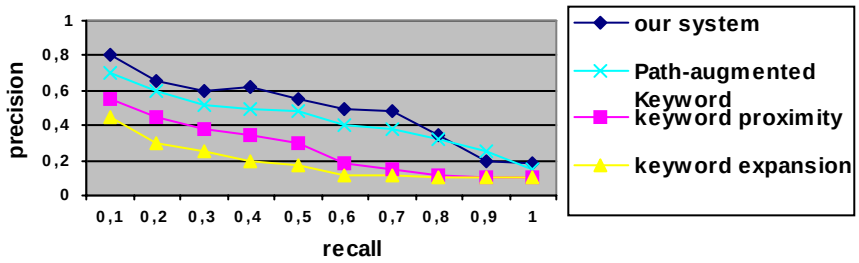

Figure1. *Moyenne précision rappel pour 40 requêtes*

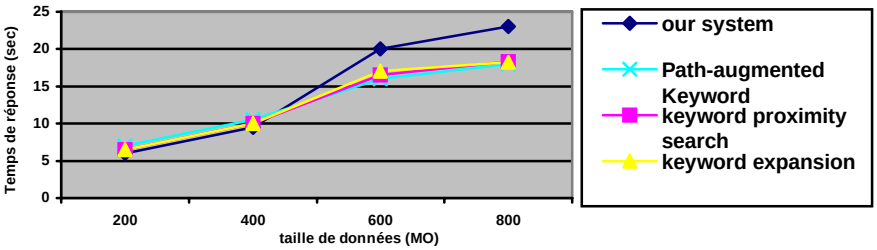

Figure2. *Effet de la taille de données sur les temps de réponse*

5. Conclusion et perspectives

Nous nous sommes consacrés aux Systèmes de Recherche d'Information dans une collection de fichiers XML en prenant en compte la pertinence contextuelle, nous avons décrit les travaux antérieurs les plus intéressants dans ce domaine de recherche, et nous nous somme basés sur les principes fondamentaux contenus dans ces travaux pour présenter une nouvelle approche de calcul de pertinence des nœuds des fichier XML à une requête CO (content only) contenant de simples mots clés. Nous allons dans un deuxième temps évoluer notre modèle pour la recherche CAS (content and structure) et évaluer l'efficience et l'efficacité de cette approche par un système d'évaluation INEX, pour évaluer notre système.

6. Références

Balmin A., et al. « A system for keyword proximity search on XML databases ». *In VLDB 2003*, pages 1069–1072, 2003.

Hristidis V., Papakonstantinou Y., and Balmin, «A.Keyword Proximity Search on XML Graphs». *IEEE ICDE 2003.*

Hsu W., and Lee.M L., and Wu X., «Path-augmented keyword search for XML documents», *Tools with Artificial Intelligence, 2004. ICTAI 2004*. 16th IEEE International Conference on Volume, Issue, 15-17 Nov. 2004 Page(s): 526 – 530.

Hlaoua L., Boughanem M .,«Towards Contextual and Structural Relevance Feedback in XML Retrieval », Dans: *workshop on Open Source Web Information Retrieval, compiègne,* 19/09/05, Michel Beigbeder, Wai Gen Yee (Eds.), ISBN:2-913923-19-4, p. 35-38.

Rocchio, « Relevance feedback in information retrieval », *Prentice Hall Inc.,Englewood Clif.*

Sauvagnat K., Boughanem M., Chrisment C., « Answering content and structure-based queries on XML documents using relevance propagation », Dans : *Information Systems, HElsevierH, Numéro spécial Special Issue SPIRE 2004*, V. 31, p. 621-635, janvier 2006.

Schenkel R., and Theobald M.,« Structural feedback for keyword-based XML retrieval», In *28PthP European Conference on Information Retrieval (ECIR 2006)*, London, UK, Apr.

Schenkel R., and Theobald M., « Feedback-driven structural query expansion for ranked retrieval of XML data», *(EDBT 2006),* Munich, Germany, Mar. 2006.