Classement des fragments de documents XML par une méthode d'aide à la décision

Faïza Abbaci*, Pascal Francq**

*
Departement de sciences de l'information et de la communication,
Université Libre de Bruxelles.
50, Av. F. D. Roosevelt, CP 123,B-1050 Bruxelles, Belgique
fabbaci@ulb.ac.be http://homepages.ulb.ac.be/ fabbaci

**
Departement de sciences de l'information et de la communication,
Université Libre de Bruxelles.
50, Av. F. D. Roosevelt, CP 123,B-1050 Bruxelles, Belgique
pfrancq@ulb.ac.be

Résumé. Vu l'accroissement constant du volume d'information accessible en ligne sous format XML, il devient primordial de proposer des modèles adaptés à la recherche d'information dans les documents XML. Tandis que la recherche d'information classique repose sur l'indexation du contenu des documents, la recherche d'information dans les documents XML tente d'améliorer la qualité des résultats en tirant profit de la sémantique véhiculée par la structure des documents. Dans cet article, nous présentons une méthode de classement des items (éléments XML) retournés lors d'une recherche dans une collection de documents XML. Le classement repose sur la prise en compte d'un ensemble de critères discriminants. La particularité de notre approche réside dans la façon dont nous les utilisons : Nous employons une méthode décisionnelle pour classer les items en les comparant deux-à-deux là où en général une fonction de scoring globale est utilisée.

1 Introduction

L'une des conséquences de la prolifération de l'information en ligne de nos jours est la diversité des données. XML se distingue comme le format par excellence pour la représentation, le stockage et l'échange de données sur Internet.

Les systèmes de recherche d'information dans les documents XML (RI-XML) utilisent soit le paradigme de l'*appariement exact* soit celui de l'*appariement approximatif* (ou *appariement par classement*). Dans le premier cas, la requête doit vérifier les contraintes sur le contenu et la structure spécifiées dans la requête, ainsi chaque item (document, fragment de document ou élément XML) sur lesquels la recherche est effectuée et jugé pertinent ou non. Dans le second cas, les items sont classés selon leur pertinence à la requête. Dans le contexte du Web, l'appariement approximatif est plus approprié. En effet, l'appariement exact nécessite un langage

d'interrogation structuré et une connaissance a priori de la structure des documents recherchés. Cependant, dans un environnement ouvert comme le Web, les utilisateurs ne sont pas nécessairement aptes à exprimer leur besoin d'information avec un langage d'interrogation complexe. En outre, la structure des documents XML recherchés n'est pas toujours disponible. L'appariement exact naturellement nécessite le classement des résultats dans le but de présenter les items les plus pertinents en premiers.

La plupart des méthodes de classement existantes proposent d'étendre les modèles traditionnels de classement utilisés dans la RI classique. Ces méthodes définissent une fonction globale qui calcule un score (RSV : Relevance Status Value) pour chaque item. Les documents sont alors triés d'une façon globale selon leurs scores et présentés à l'utilisateur. Le score d'un document est calculé en fonction d'un ensemble de critères ayant un impact sur la pertinence du document, comme par exemple la fréquence d'apparition des termes de la requête. La moyenne pondérée des valeurs des critères est par exemple souvent utilisée pour le calcul du score. Notre approche diffère dans ce sens que nous ne trions pas les résultats d'une façon globale mais partielle. Les items sont comparés deux-à-deux pour déterminer un classement partiel les résultats de ce classement sont combinés pour obtenir un classement global. Dans ce but nous avons choisi d'utiliser une méthode d'aide à la décision et notre choix s'est porté sur la méthode PROMETHEE.

La suite de cet article est organisée comme suit : Dans la section 2 nous clarifions certaines différences primordiales entre la recherche d'information classique et la recherche d'informations dans les documents XML. Ceci est important pour la suite de cet article. Nous présentons dans la section 3 l'architecture générale de notre système de recherche d'information XML. La section 4 est un état de l'art des approches de classement dans XML. Dans la section 5 nous détaillons notre approche de classement ainsi que la méthode d'aide à la décision utilisée. Et nous terminons enfin par une conclusion

2 La recherche d'information dans les documents XML (RI-XML) vs. la recherche d'information classique (RI)

Nous présentons dans cette section les différences principales entre les techniques de la RI et celles de la RI-XML.

2.1 L'interrogation

Une première différence réside dans le processus d'interrogation. En effet, alors que dans la RI les requêtes sont majoritairement des mots clés, dans la RI-XML il existe un grand nombre de langages d'interrogation (comme par exemple XPATH Consortium (1999) ou XQUERY Consortium (2003)) qui permettent d'inclure dans la requête des contraintes structurelles. Mais nous sommes forcés de constater que dans un environnement aussi vaste qu'Internet l'utilisation d'un langage représente un frein car d'une part la structure des documents n'est pas toujours connue par l'utilisateur et d'autre part, les langages d'interrogation sont souvent complexes pour des utilisateurs lambda. D'où l'existence de plus en plus de méthodes de RI-XML qui proposent l'interrogation par mots clés.

2.2 L'indexation

Deuxièmement, l'indexation d'un document XML, en plus du contenu, doit prendre en compte aussi la structure du document. Il est fréquent de considérer un document XML comme un arbre où les noeuds sont les éléments ou les noms d'attributs, où les arêtes représentent l'appartenance du noeud fils au noeud père et où les feuilles sont les contenus des éléments où les valeurs des attributs. Ainsi, l'indexation de

la structure revient à garder une trace des relations *père-fils* et *frère de* entre les noeuds. Pour indexer le contenu des documents XML, les technique de la RI sont reprises dans la RI-XML, mais souvent avec des adaptation liées à l'existence de la structure dans les documents. En effet, l'unité d'indexation en RI-XML pouvant être l'élément au lieu du document entier, des statistiques comme par exemple la fréquence d'apparition des termes représentera dans la RI-XML la fréquence d'apparition d'un terme dans un élément non pas dans le document entier. Idem pour la fréquence documentaire (DF : Document Frequency) qui représentera en RI-XML le nombre d'éléments et non pas de documents contenant un terme donné.

2.3 Les résultats

La problématique de la RI-XML est similaire à la "recherche de segments de documents" (*passage retrieval*). En effet, les documents structurés peuvent contenir un large spectre d'information hétérogènes, il est donc préférable de retrouver une partie (ou des parties) de document qui répond à la requête de l'utilisateur plutôt que le document entier Wilkinson (1994). Survient alors une difficulté supplémentaire qui est de déterminer le niveau de granularité de l'élément retourné. Lorsque l'utilisateur exprime sa requête dans un langage structuré, il a la possibilité de définir la garnularité des éléments qu'il désire. Par exemple, avec la requête XPATH suivante : *//poisson[milieu="eaux douces"]/nom* l'utilisateur spécifie qu'il désire seulement le contenu du noeud *nom*. Lorsque en revanche nous sommes en situation où l'utilisateur s'exprime uniquement à l'aide de mots clés, le système devra calculer la granularité adéquate.

3 Une vue globale de notre système de RI-XML

Lors de la conception de notre système, notre but principal était son accessibilité à l'utilisateur. Ainsi, l'interrogation du système devait être adaptable au niveau de l'utilisateur. Les requêtes peuvent varier en complexité en partant d'un simple ensemble de mots clés à de complexes expressions booléennes. Le système analyse les requêtes, et détermine les éléments les plus susceptibles de correspondre au besoin de l'utilisateur et les retourne à l'utilisateur triés par ordre de pertinence. Les résultats sont présentés de façon à ce que l'utilisateur puisse naviguer dans la structure du document d'où l'élément est extrait.

Notre système est constitué de deux parties principales (figure 1. Dans la première partie, un parseur de documents XML analyse et indexe la structure des documents et leurs contenus. Le résultats de cette analyse est sauvegardé dans une base de données. La deuxième partie est constituée de trois composants l'*analyseur de requête*, le *module d'appariement* et le *module de classement des résultats*. L'*analyseur de requête* décompose la requête lorsqu'elle contient des opérateurs booléens et crée une structure qui permet son appariement avec les documents. Le *module d'appariement* retrouve dans la base de données les éléments les plus appropriés à la requête. Enfin le *module de classement des résultats* se charge de trier selon l'ordre de pertinence les éléments issus du *module d'appariement*.

Dans Abbaci et al. (2006) nous avons détaillé l'indexation des documents XML dans notre système, l'analyse d'une requête ainsi que le processus d'appariement d'une requête aux documents de la base de données. Dans cet article nous présentons le fonctionnement du *module de classement des éléments*.

4 Les approches de classement dans la RI-XML

Dans la RI un certain nombre de critères ont été décelés importants dans le jugement de pertinence d'un document à une requête donnée. Quelque uns de ces critères sont devenus classiques tels la fréquence d'apparition des termes (TF) ainsi que leur pouvoir de discrimination (IDF Inverse Document Frequency). Il existe d'autres critères comme par exemple la proximité des termes de la requête dans le

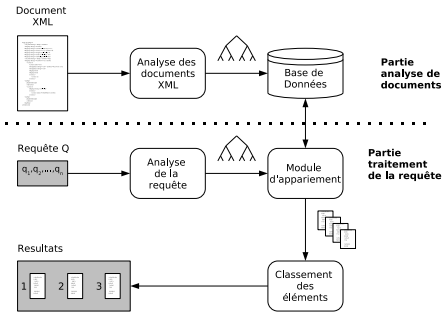

FIG. 1 – *L'architecture globale de notre système de RI-XML.*

document, leur emplacement dans le document (début, fin, titre, résumé etc...) la prise en compte de la synonymie etc...

Lorsque nous étudions les méthodes de classement des documents (ou fragments) XML proposées dans la littérature, nous constatons qu'elles utilisent à peu près les mêmes critères mais en les adaptant au format XML. Ainsi, dans Ben-Aharon et al. (2003), Cohen et al. (2003), Yosi et Mandelbrod (2003), Theobald et Weikum (2002) et Liu et al. (2004) TF représente la fréquence d'apparition des termes dans un élément XML, dans Sigurbjörnsson et al. (2004); Sigurbjornsson et al. (2003), Yosi et Mandelbrod (2003), Theobald et Weikum (2002), Cohen et al. (2003) et Liu et al. (2004) IDF (ou IEF pour Inverse élément Frequency) représente le pouvoir de discrimination entre les éléments XML de la collection . Dans Guo et al. (2003), Yosi et Mandelbrod (2003) and Theobald et Weikum (2002) les auteurs utilisent le nombre d'éléments XML au lieu du nombre de documents. Dans Ben-Aharon et al. (2003) la proximité entre deux termes est la moyenne sur tous les documents du corpus des distances entre ces deux termes lorsqu'ils apparaissent dans le même élément XML.

Certains critères sont utilisés tels qu'ils sont connus dans la RI dans ce cas le document XML est considéré comme un simple document texte, comme par exemple la proximité entre les termes de la requête. Ainsi, dans Sauvagnat et al. (2003) les auteurs considèrent la proximité entre deux termes comme étant le nombre de mots séparant ces termes dans une fenêtre de x termes et Kotsakis (2002) et Sauvagnat et al. (2003) calculent TF et IDF de la même manière qu'en RI. Les auteurs de Theobald et Weikum (2002) quant à eux, intègrent la proximité sémantique (calculée sur la base d'une ontologie) entre les termes pour augmenter la performance de la fonction de calcul du score.

D'autres critères liés à la structure des documents XML sont utilisés. Nous citons la distance entre les noeuds, dans Guo et al. (2003) les auteurs proposent d'utiliser la distance à deux dimensions. D'une part la distance classique entre termes (distance à l'horizontal dans l'arbre d'un document XML) et la distance en terme de noeuds (distance à la verticale dans l'arbre d'un document XML). Dans d'autres travaux Ben-Aharon et al. (2003), Liu et al. (2004), Kotsakis (2002) et Cohen et al. (2003) des poids sont attribués aux noms d'élément afin de favoriser les scores des élément qu'on juge plus intéressant à retourner (par exemple préférer retourner le résumé d'un livre plutôt que son titre). Un autre critère lié à la structure des documents XML et qui rentre en compte dans le calcul du score des éléments est la spécificité des éléments à classer Guo et al. (2003) Sigurbjörnsson et al. (2004), plus un élément est profond plus il est spécifique.

5 Notre approche de classement

La majorité des méthodes de classement dans la RI adoptent une approche global de classement. En d'autres termes, les critères de tri sont rassemblés dans une fonction unique qui doit être maximisée. L'inconvénient majeur de cette approche globale réside dans le fait que les critères peuvent mutuellement se compenser. Ainsi une solution dont l'un des critères présente une valeur faible peut ne pas être pénalisée si un de ses critères restants présente une valeur élevée.

Par conséquent nous avons opté pour une approche de classement partiel qui classe les items en les comparant deux-à-deux. Dans ce qui suit, nous décrivons la méthode d'aide à la décision que nous avons choisie. Le principe de cette méthode est qu'elle compare les solutions paire-par-paire. Ainsi, une solution a surclasse une autre solution b si au vue de la plupart des critères, a est meilleure que b. Les résultats de la comparaison par paire sont alors combinés afin d'établir un classement total des solutions.

5.1 PROMETHEE une méthode d'aide à la décision

Une méthode d'aide à la décision permet d'effectuer un classement d'un ensemble de solutions possibles à un problème donné en commençant par la solution la plus adéquate au vue d'un ensemble de critères et de l'importance relative accordée à chacun de ces derniers.

Nous décrivons dans ce qui suit la méthode PROMETHEE Vincke (1989) que nous avons utilisée.

Soit A un ensemble de solutions (dans notre cas les éléments XML), pour chaque $a \in A$, $f_j(a)$ représente l'évaluation de la solution a au vue du critère f_j. La table 1 représente une table générique d'évaluation.

	$f_1(.)$	$f_2(.)$	$\cdots$	$f_j(.)$	$\cdots$	$f_k(.)$
a_1	$f_1(a_1)$	$f_2(a_1)$	$\cdots$	$f_j(a_1)$	$\cdots$	$f_k(a_1)$
a_2	$f_1(a_2)$	$f_2(a_2)$	$\cdots$	$f_j(a_2)$	$\cdots$	$f_k(a_2)$
$\vdots$						
a_i	$f_1(a_i)$	$f_2(a_i)$	$\cdots$	$f_j(a_i)$	$\cdots$	$f_k(a_i)$
$\vdots$						
a_n	$f_1(a_n)$	$f_2(a_n)$	$\cdots$	$f_j(a_n)$	$\cdots$	$f_k(a_n)$

TAB. 1 – *Table d'évaluation.*

Une fonction de préférence $P_j(a,b)$ est définie afin d'attribuer un degré de préférence d'une solution a à une solution b au vue d'un critère f_j. En général, $P_j(a,b)$ modélise les différences des valeurs des solutions pour un critère donné $d = f_j(a) - f_j(b)$. La fonction $P_j(a,b)$ est normalisée comme suit :

$$\begin{cases} 0 \leq P_j(a,b) \leq 1; & & \\ P(a,b) = 0, & sid \leq 0, & \text{pas de préférence entre } a \text{ et } b; \\ P(a,b) \approx 0, & sid > 0, & \text{faible préférence de } a \text{ à } b; \\ P(a,b) \approx 1, & sid >> 0, & \text{forte préférence de } a \text{ à } b; \\ P(a,b) = 1, & sid >>> 0, & \text{préférence stricte de } a \text{ à } b. \end{cases}$$

Deux paramètres q et p de seuil d'indifférence et de préférence respectivement sont définis. Lorsque la différence entre les évaluations de a et b est inférieure à q alors elle n'est pas significative. La fonction de préférence est donc égale à 0. Lorsque cette différence est supérieure à p elle est considérée comme trés significative et la fonction de préférence est dans ce cas égale à 1. Un classement des deux solutions a et b est construits en prenant en compte tous les critères et ce selon l'expression suivante (1) :

$$\pi(a,b) = \sum_{j=1}^{k} w_j P_j(a,b) \quad (1)$$

où $w_j > 0$ sont les poids associés aux critères. Ces poids son des nombres naturels positifs qui ne dépendent pas des échelles des critères. $\pi(a,b)$ exprime le degré de préférence de la solution a à b au vue de tous les critères.

Les valeurs de $\pi(a,b)$ et $\pi(b,a)$ sont calculées pour chaque paire de solutions $a, b \in A$. De cette façon, une relation de surclassement est définie dans A.

Deux flux de surclassement sont définies :

– Flux de surclassement positif, qui représente la puissance d'une solution par rapport à toutes les autres. Plus $\Phi^+(a)$ est grand plus la solution a est mieux que les autres :

$$\Phi^+(a) = \frac{1}{n-1} \sum_{b \in A, b \neq a} \pi(a,b)$$

– Flux de surclassement négatif, qui représente la faiblesse d'une solution par rapport à toutes les autres. Plus $\Phi^-(a)$ est petit plus les autres solutions sont mieux que a :

$$\Phi^-(a) = \frac{1}{n-1} \sum_{b \in A, b \neq a} \pi(b,a)$$

L'ordre total d'une solution a est déduit du flux net $\Phi(a) = \Phi^+(a) - \Phi^-(a)$ comme suit :

$$\begin{cases} apb & ssi \quad \Phi(a) > \Phi(b) \\ aIb & ssi \quad \Phi(a) = \Phi(b) \end{cases}$$

P pour "préférable à" et I pour "indifférent à".

Un exemple Nous illustrons dans ce qui suit la méthode PROMETHEE par l'exemple du choix d'une voiture. Dans la décision de l'achat d'une voiture plusieurs critères rentrent en jeux : le prix (aussi bas que possible), la consommation en carburant (aussi basse que possible), le confort donné comme étant un nombre entre 0 et 3 (aussi haut que possible) et la puissance (aussi haute que possible). La table 2 montre les valeurs de ces critères pour quatre voitures différentes. Dans cet exemple, aucune voiture ne semble être meilleure que les autres. Cependant chaque personne doit avoir sa propre préférence, par exemple que la puissance est le critère le plus important.

	Prix (k€)	Consommation ($1/100$km)	Confort	Puissance (CV)
Voiture 1	**8.75**	**6.2**	Bon (1)	30
Voiture 2	13.75	7.5	Bon (1)	50
Voiture 3	25	8	**Excellent (3)**	80
Voiture 4	62.5	20	Très bon (2)	**120**

TAB. 2 – *Exemple d'un problème de décision multicritère.*

Le seuil de préférence p est fixé à 0.2 pour chaque critère i.e. si deux solutions ont une différence de plus de 20% pour un critère donné, l'une des deux solutions est préférable à l'autre pour le critère en question. En outre, le seuil d'indifférence q est fixé à 0.05 pour chaque critère i.e. si deux solutions ont une différence de moins de 0.5% pour un critère donné, les deux solutions doivent être considérées comme égale pour le critère en question.

	Φ^+	Φ^-	Φ
Voiture 3	0.583	0.417	0.166
Voiture 1	0.417	0.417	0.000
Voiture 2	0.417	0.417	0.000
Voiture 4	0.417	0.583	−0.166

TAB. 3 – *Le classement par la méthode PROMETHEE lorsque les critères sont à poids égaux à* 1.

Calculons le classement de ces voitures par la méthode PROMETHEE lorsque les poids de tous les critères sont fixés à 1 (Table 3).

Les résultats montrent que la solution idéale est la voiture numéro 3 et que la voiture la moins intéressante est la numéro 4. La méthode ne peut classer les deux voitures numéros 1 et 2.

Calculons à présent le classement des ces voitures par la méthode PROMETHEE lorsque le poids du critère *Consommation* est fixé à 4 (Table 4).

	Φ^+	Φ^-	Φ
Voiture 4	0.666	0.333	0.333
Voiture 3	0.619	0.381	0.238
Voiture 2	0.381	0.524	−0.143
Voiture 1	0.238	0.667	−0.429

TAB. 4 – *Le classement par la méthode PROMETHEE lorsque le critère Consommation a un poids égal à* 4.

L'effet de ce changement est clair, la voiture numéro 4 devient la solution idéale.

5.2 PROMETHEE pour classer les fragments de documents XML

Dans cette section, nous présentons les différentes étapes de l'adaptation de PROMETHEE à notre problématique de classement des fragments de documents XML, ces derniers étant le résultat d'une recherche d'information dans une collection de documents XML. D'abord nous définissons l'ensemble des critères que nous souhaitons prendre en compte dans le processus du classement. Ensuite, nous associons à chaque critère son ensemble des valeurs possibles ainsi que la valeur optimale souhaitée. Lorsqu'une requête est soumise à notre système, le tableau des critères est rempli avec les fragments des documents de la collection en guise de solutions.

Les critères de classement des fragments XML

- *La pertinence du contexte* ($Doc.$) : Nous partons du principe que la pertinence d'un élément est liée à la pertinence du document qui le contient. Ainsi le score global du document "père" constitue un critère pour la décision concernant le classement des éléments qu'il contient. Pour l'instant nous proposons que $Doc.$ représente le score classique $TF * IDF$ en ignorant la structure du document XML. Ainsi, pour une requête Q et un élément e du document D :

$$Doc.(e, Q) = TF * IDF(Q, D)$$

- *Le type des mots clés* ($Type.$) : Le critère $Type.$ représente le type d'un mots clés i.e. *nom d'élément, contenu d'élément, nom d'attribut ou valeur d'attribut*. Nous proposons d'impliquer ce critère dans le classement des éléments comme suit : lorsque dans un élément e les deux termes q_1 et q_2 ont une relation (nom élément, contenu élément) ou (nom d'attribut, valeur d'attribut) alors

e doit être avantagé dans le classement par rapport aux autres éléments où la relation est autre.

$$Type.(q_1, q_2, e) = \begin{cases} 2 & si \quad T(q_1) + T(q_2) = 5 \\ 1 & si \quad T(q_1) = T(q_2) \\ 0 & sinon \end{cases}$$

$T(q)$ représente le type du mot clé q.

$$T(q) = \begin{cases} 1 & si \quad \text{q est nom d'élément} \\ 2 & si \quad \text{q est nom d'attribut} \\ 3 & si \quad \text{q est valeur d'attribut} \\ 4 & si \quad \text{q est contenu d'élément} \end{cases}$$

– *La distance entre les mots clés* ($Dis.$) : Nous adaptons la notion de distance entre les mots clés au contexte de XML. Ainsi, $Dis.(q_1, q_2, e)$, la distance entre les noeuds qui contiennent respectivement les deux mots clés q_1 et q_2 dans l'élément e est représentée par la longueur du chemin le plus court entre les deux noeuds. Cette distance est nulle lorsque les deux mots clés appartiennent au même élément.

$$Dis.(q_1, q_2, e) = \begin{cases} 0 & si \quad ID(q_1) = ID(q_2) \\ |Cpc(q_1, q_2, e)| & sinon \end{cases}$$

$Cpc(q_1, q_2, e)$ représente le chemin le plus court entre les deux noeuds où apparaissent respectivement q_1 et q_2.

– *La fréquence d'apparition des mots clés* ($Freq.$) : Comme dans la RI classique, nous supposons que les termes fréquents dans un élément contribuent fortement à la description de ce dernier et par conséquent les éléments qui contiennent plusieurs occurrences des mots clés ont plus de chance d'être pertinents à la requête.

$$Freq.(e, Q) = \sum_i Occ(q_i)$$

$Occ(q, e)$: représente le nombre d'occurrences du terme q dans l'élément e.

– *La spécificité de l'élément* ($Specif.$) : Intuitivement nous supposons que si un élément e contient un autre élément $e^{'}$ (e et $e^{'} \in E$) alors $e^{'}$ doit être classé plus haut que e.

$$Specif.(e, Q) = \frac{1}{|e|}$$

$| e |$ est le nombre de noeuds de l'élément e.

– *Les liens structurels entre les mots clés* ($Rel.$) : Nous faisons référence ici à la relation de parenté dans la structure d'un élément XML. Nous supposons que la relation "ancêtre-descendant" entre deux noeuds contenant respectivement deux mots clés de la requête est un bon indicateur de pertinence. De ce fait, nous calculons $Rel.$ comme suit :

$$Rel.(e, Q) = \sum_{i<j}^{n} Rel.^{'}(q_i, q_j, e)$$

où

$$Rel.' = \begin{cases} \frac{1}{DisV(q_i,q_j,e)} & si \quad ID(q_i) \in Ansestors(ID(q_j),e) \\ 0 & sinon \end{cases}$$

$DisV(q_i, q_j, e)$ représente la distance verticale (en profondeur dans l'arbre XML) dans l'élément e entre les deux noeuds qui contiennent q_i et q_j respectivement.
$ID(q)$ étant l'identifiant du noeuds où se trouve le terme q.
$Ansestors(ID(q), e)$ est l'ensemble des noeuds ancêtres du terme q dans l'élément e.

Les valeurs optimales des critères de classement Le tableau 5 illustre les valeurs optimales souhaitées aux critères de classement présentées ci-dessus.

Critère	Échelle	Valeur optimale
$Doc.$	$\mathbb{R}^+$	Le plus grand possible
$Type.$	$\{0, 1, 2\}$	2
$Dis.$	$\mathbb{N}^+$	Le plus petit possible
$Freq.$	$\mathbb{N}^+$	Le plus grand possible
$Specif.$	$\mathbb{R}^+$	Le plus grand possible
$Rel.$	$\mathbb{R}^+$	Le plus grand possible

TAB. 5 – *Les échelles des valeurs des critères de classement.*

Le tableau d'évaluation Lors de l'application de la méthode PROMETHEE au classement des éléments issus de l'étape d'appariement 3 les valeurs des critères de classements sont calculés pour chacun de ces éléments, et un tableau d'évaluation est construit 6. Concernant les poids à attribuer aux critères

Éléments	$Doc.$	$Type.$	$Dis.$	$Freq.$	$Specif.$	$Rel.$
e_1	.	.	.	.	.	.
e_2	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
e_n	.	.	.	.	.	.

TAB. 6 – *Les échelles des valeurs des critères de classement.*

nous envisageons d'effectuer plusieurs tests afin de les déterminer.

6 Conclusion

Nous avons présenté dans cette article notre approche pour le classement des éléments de documents XML pertinents à une requête donnée. Nous avons montré l'originalité de notre approche qui se distingue par la façon de combiner les différents critères qui rentrent en jeux dans le jugement de pertinence dans

le but d'un classement par ordre de pertinence. En effet, la majorité des approches de la littérature combinent ces critères dans une fonction de calcul de score unique et tentent d'optimiser cette dernière. Notre approche est d'éviter cette méthode globale dont l'inconvénient principal est de permettre aux critères de se compenser les uns avec les autres. Nous appliquons une méthode d'aide à la décision qui effectue un tri partiel paire-par-paire des éléments et combine le résultat de ce tri partiel pour obtenir un tri global des éléments. Des tests de validité de notre approche sont en phase d'implémentation.

Références

Abbaci, F., J. B. Valsamis, et P. Francq (2006). Index and search xml documents by combining content and structure. In *International Conference on Internet Computing, Las Vegas, Nevada, USA. June 26-29, 2006*.

Ben-Aharon, Y., S. Cohen, Y. Grumbach, Y. Kanza, J. Mamou, Y. Sagiv, B. Sznajder, et E. Twito (2003). Searching in an XML corpus using content and structure. In *Initiative of the Evaluation of XML Retrieval, INEX'03. Germany*, Germany.

Cohen, S., J. Mamou, Y. Kanza, et Y. Sagiv (2003). XSearch : A Semantic Search Engine for XML . In *29th VLDB Conference, berlin, Germany*. http ://www.vldb.org/conf/2003/papers/S03P02.pdf.

Consortium, W. W. W. (1999). Xml path language (xpath) version 1.0. Technical report. http ://www.w3.org/TR/xpath.

Consortium, W. W. W. (2003). Xquery 1.0 : an xml query language. Technical report. http ://www.w3.org/TR/xquery.

Guo, L., F. Shao, C. Botev, et J. Shanmugasundaram (2003). Xrank : Ranked keyword search over xml documents. In *International Conference on Management of Data, 2003. San Diego, California*, San Diego, California.

Kotsakis, E. (2002). Structured information retrieval in xml documents. In *SAC '02 : Proceedings of the 2002 ACM symposium on Applied computing*, New York, NY, USA, pp. 663–667. ACM Press.

Liu, S., Q. Zou, et W. W. Chu (2004). Configurable indexing and ranking for xml information retrieval. In *SIGIR '04 : Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 88–95. ACM Press.

Sauvagnat, K., G. Hubert, M. Boughanem, et J. Mothe (2003). Irit at inex 2003. In *INitiative for the Evaluation of XML Retrieval. Dagstuhl, Germany. December 15-17, 2003*, pp. 142–148. Fuhr Norbert, Lalmas Mounia, Malik Saadia.

Sigurbjornsson, B., J. Kamps, et M. de Rijke (2003). An element-based approach to xml retrieval. In *Initiative of the Evaluation of XML Retrieval, INEX'03. Germany*, Germany, pp. 19–26.

Sigurbjörnsson, B., J. Kamps, et M. de Rijke (2004). Processing con tent-and-structure queries for xml retrieval. In *TDM'04 : the first Twente Data Management Workshop on XML Databases and Information Retrieval. Enschede, The Netherlands*.

Theobald, A. et G. Weikum (2002). The index-based xxl search engine for querying xml data with relevance ranking. In *EDBT '02 : Proceedings of the 8th International Conference on Extending Database Technology*, London, UK, pp. 477–495. Springer-Verlag.

Vincke, P. (1989). *Multicriteria decision aid*. John Wiley and sons.

Wilkinson, R. (1994). Effective retrieval of structured documents. In *SIGIR '94 : Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 311–317. Springer-Verlag New York, Inc.

Yosi, M. et M. Mandelbrod (2003). Retrieving the most relevant xml components. In *INitiative for the Evaluation of XML Retrieval. Dagstuhl, Germany. December 15-17, 2003*, pp. 142–148. Fuhr Norbert, Lalmas Mounia, Malik Saadia.

Summary

With the exponential growth of online data in XML format it becomes important to investigate retrieval models for XML documents. When classical information retrieval systems index the content of documents, XML retrieval models take advantage of document structure to greatly improve the retrieval precision. Existing XML retrieval systems support either *exact-matching* paradigm or *approximate matching* paradigm. In the first case results are exactly matching the content and structure conditions in the query. In the second case, results are ranked accordingly to their relevance to the query. In this paper, we present a XML results ranking technique which takes into account both content and structure of the XML documents frangments to be ranked. To do so, we define a set of criterions which we think important in computing results ranks. Then we use a multi-criterion decision support method to rank these results.