Classification de Structures Arborescentes : Cas de Documents XML

Ali Aïtelhadj*, **, Mohamed Mezghiche** et **Fatiha Souam*****

* *Université Mouloud Mammeri de Tizi-Ouzou (UMMTO) Algérie*
aaitelhadj@ymail.com

** *LIFAB (Labo d'Informatique Fondamentale et Appliquée de Boumerdes) Université M'hamed Bouguera de Boumerdes (UMBB) Algérie*
mohamed.mezghiche@yahoo.fr

*** *Université Mouloud Mammeri de Tizi-Ouzou (UMMTO) Algérie*
souam_fat@yahoo.fr

Chercheur

RÉSUMÉ. *Cet article présente une méthode de classification structurelle de documents XML. Notre approche consiste d'abord à extraire automatiquement la structure arborescente de chaque document XML à classer, et ensuite à utiliser cette structure comme modèle de représentation pour la classification du document XML correspondant. L'appariement de ces structures est fondé sur un calcul de leurs similarités. Pour l'expérimentation nous avons utilisé un corpus INEX.*

ABSTRACT. *In this paper we present a clustering method for XML documents. Our step is two-phase based: we first automatically extract the structure from the document; we then use it as model of representation to classify the document that it represents. The matching of the documents' structures is based on the calculation of their similarities. For the experimentation we used the INEX.*

MOTS-CLÉS: *Clustering, classification structurelle, structure, arbre, similarité, contexte, nœud, index, seuil.*

KEYWORDS: *Clustering, structural classification, structure, tree, similarity, context, node, index, threshold.*

1. Introduction

En très peu de temps, XML est devenu un standard incontournable pour la représentation et l'échange de données sur le Web. De plus, non seulement les collections de documents XML sont réutilisées, mais le volume de leurs échanges s'accroît continuellement. Cependant, avec les outils actuellement disponibles, la recherche d'informations dans ce type de documents est une tâche non triviale, notamment si elle concerne plusieurs domaines. Aujourd'hui les techniques d'accès aux données XML sont pour la plupart issues des travaux de la communauté des BD (Bases de Données) sur des données semi structurées.

Un document XML est caractérisé par un contenu (du texte) et une structure (balises). Ce type de documents ne peut cependant être exploité efficacement par les méthodes classiques de RI. En effet, ces dernières ne traitent un document que du point de vue de son contenu, alors que le format XML permet d'ajouter des contraintes structurelles (balises). Ceci impose alors d'adapter les méthodes classiques de RI ou d'introduire de nouveaux mécanismes afin d'exploiter au mieux les informations disponibles sous format XML. Notre travail rentre dans cette perspective et a pour motivation d'étendre les méthodes actuelles (classiques) de RI par un système de classification. Pour cela nous proposons une approche qui consiste d'abord à représenter chaque document XML par sa structure générique (ou résumé d'arbre), et ensuite à utiliser cette structure comme modèle de représentation pour la classification structurelle du document XML correspondant. Par conséquent, la classification d'une telle structure équivaut à la classification structurelle du document XML qu'elle représente.

La classification structurelle a pour rôle de regrouper des documents XML structurellement similaires dans des clusters (ou classes), afin de réduire le temps de réponse et augmenter la précision des moteurs de recherche. L'idée est que, si les documents partagent des structures similaires, ils sont plus à même de correspondre à la partie structurelle d'une requête. Notons que dans le cas d'XML les requêtes peuvent avoir une partie contenu et une partie structure.

Cet article est organisé de la manière suivante: outre l'introduction, la section 2 met en valeur la problématique et donne un état de l'art sur la classification structurelle de documents XML, la section 3 présente l'approche de classification structurelle que nous proposons, la section 4 est consacrée à une expérimentation de notre approche de classification, enfin la section 5, situe notre proposition de classification par rapport aux méthodes évoquées dans l'état de l'art.

2. Problématique et état de l'art

2.1. *Problématique*

La question primordiale est : *Comment doit-on indexer les informations des documents XML afin de pouvoir y accéder efficacement ultérieurement?*

En se focalisant sur la structure, on peut identifier certaines problématiques spécifiques liées au format XML. On peut citer *l'hétérogénéité sémantique* de la structure des documents XML (Denoyer, 2004). En effet, deux documents XML aux *contenus identiques* ou très proches thématiquement, peuvent avoir des *structures complètement différentes*. Le problème est : *Comment dans ce cas définir le formalisme que l'on va pouvoir utiliser pour formuler les requêtes utilisateur (besoin en information) composées de contenus et de structures?* Mais la problématique centrale qui nous incombe ici est *Comment peut-on mesurer la similarité structurelle de documents XML ?* Cette similarité soulève d'autres questions liées notamment au degré de similarité des structures, en particulier : *A partir de quel seuil de similarité doit-on considérer que deux documents peuvent être structurellement similaires?* C'est à ces questions que nous tentons de répondre dans cet article.

2.2. *Aperçu sur l'état de l'art*

Les approches de classification, se déclinent en deux variantes : la classification supervisée et la classification non supervisée (ou clustering). Dans la première, on dispose d'un ensemble d'objets classés, servant d'é chantillon d'apprentissage. Le problème est alors d'associer à tout nouvel objet sa classe la plus appropriée, en se servant des exemples déjà étiquetés. Dans la seconde par contre (classification non supervisée), les classes possibles ne sont pas connues à priori, et les exemples disponibles sont non étiquetés. Le but est donc de regrouper dans un même cluster (ou classe) les objets considérés comme similaires, pour constituer les classes.

Les travaux sur la classification de documents XML existants se distinguent par la manière de représenter les documents, mais aussi par les méthodes de classification utilisées. Dans la *classification structurelle*, on distingue deux courants principaux. Dans le premier, on classe directement des documents XML. Dans le second par contre, au lieu de classer directement des documents XML, on classe leurs DTDs (Document Type Definition).

2.2.1. *Classification structurelle de documents XML*

Nous nous intéressons ici qu'aux approches qui représentent les documents XML par des arbres *étiquetés (ordonnés ou non)*. Les étiquettes correspondent à des tags XML. Un arbre est dit *ordonné* si les fils de chacun de ses nœuds (non feuilles), sont liés par une relation d'ordre totale de gauche à droite.

–Les approches de (Nierman *et al.*, 2002) et (Francesca *et al.*, 2003) utilisent des arbres *étiquetés* et *ordonnés*. La similarité entre deux arbres est basée sur leur *distance d'édition*. Une *distance d'édition* mesure le nombre d'opérations élémentaires (insertions et suppressions de nœuds) à effectuer pour transformer un

arbre en un autre. La *distance d'édition* permet d'effectuer un clustering de ces arbres en appliquant une méthode de classification hiérarchique ascendante.

–(Dalamagas *et al.*, 2004) utilisent des "résumés d'arbres" *étiquetés* et *ordonnés*. Comme illustré sur la figure 1, un "résumé d'arbre" est obtenu par deux transformations : la première réduit la *profondeur* de l'arbre de sorte que tout nœud (non feuille) ayant la même étiquette que l'un de ses ancêtres devienne un descendant direct (fils) de cet ancêtre ; la seconde élimine les *répétitions* des nœuds frères. L'arbre obtenu, peut ne pas être *exact*, mais il est *enraciné* (sa racine est conservée). Cette approche effectue aussi un clustering hiérarchique ascendant basé sur un calcul de *distance d'édition*.

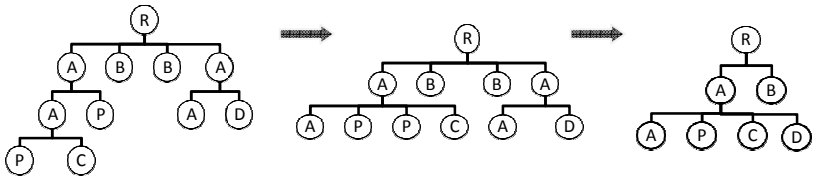

Figure 1. *Extraction de "résumés d'arbres"(Dalamagas et al, 2004)*

–(Termier *et al.*, 2002), (Costa *et al.*, 2004) et (Del Razo Lopez *et al.*, 2006), utilisent le formalisme des arbres fréquents. Ces approches sont fondées sur la découverte de sous-arbres qui apparaissent suffisamment fréquemment dans des collections d'arbres étiquetés. Dans (Termier *et al.*, 2002), les sous-arbres fréquents, peuvent être *ordonnés ou non*, ils peuvent aussi être des sous-arbres *exacts* ou bien conserver uniquement la notion d'*ancêtre*. Les sous-arbres de droite sur la figure 2 sont tous fréquents selon (Termier *et al.*, 2002). Dans les deux autres approches, les sous-arbres sont *ordonnés* et *enracinés*, comme les deux sous-arbres les plus à droite de la figure 2. (Costa *et al.*, 2004) s'appuient sur la *distance d'édition* pour effectuer un clustering hiérarchique. (Termier *et al.*, 2002) caractérisent un cluster par le sous arbre fréquent maximal commun aux sous arbres du cluster. Quant à (Del Razo López *et al.*, 2006), leur travail consiste à intégrer des schémas XML en vue de construire un schéma médiateur pour l'interrogation de documents XML.

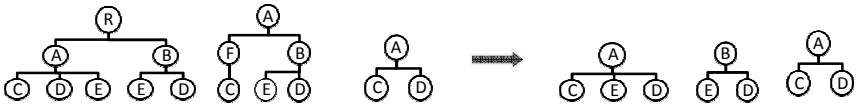

Figure 2. *Extraction de sous-arbres fréquents*

2.2.2. *Classification structurelle basée sur des DTDs*

Par définition, une DTD est considérée comme une grammaire à contexte libre qui engendre un nombre potentiellement infini de documents XML. De ce fait, au

lieu de classer directement des documents XML, l'approche préconise d'établir la classification de leurs DTDs. Ainsi chaque cluster devient le représentant d'un ensemble de documents XML structurellement similaires.

Parmi les travaux qui ont été rapportés sur la classification de DTDs, nous citons ci-après quelques uns des plus connus :

–Dans XClust (Mong *et al.*, 2002) les auteurs proposent un modèle de clustering de DTDs. Chaque DTD est représentée par un arbre. La similarité entre deux nœuds de deux arbres est calculée en exploitant plusieurs niveaux : la similarité ontologique des nœuds en utilisant un dictionnaire, la similarité de leurs ensembles de descendants immédiats (fils), la similarité de leurs ancêtres et enfin, la similarité des ensembles de feuilles des sous arbres dont ils sont respectivement les racines.

– Les auteurs de SPL (Schema Probabilistic Learning) (Su *et al.*, 2001) proposent un mécanisme qui identifie syntaxiquement la similarité entre DTDs en adoptant une stratégie de classification ascendante. Comparativement à XClust de (Mong *et al.*, 2002), SPL exploite uniquement les contextes des descendants immédiats (fils) pour deux nœuds.

–LSD (Learning Schema Document) (Doan *et al.*, 2001) est une approche basée sur l'apprentissage et l'inférence, combinés avec une instance de DTD. C'est une classification supervisée, c'est-à-dire, qu'on connaît les classes à l'avance.

– Enfin, dans Cupid (Madhavan *et al.*, 2001), les auteurs proposent un algorithme qui s'appuie sur un schéma générique de matching de DTDs. Le but du matching est de cibler un schéma médian vers lequel doivent converger toutes les DTDs ayant des structures similaires. Tout comme XClust (Mong *et al.*, 2002), pour calculer la similarité des arbres DTDs, l'approche Cupid de (Madhavan *et al.*, 2001), s'appuie sur un dictionnaire, mais elle exploite uniquement les contextes feuilles.

3. Approche pour la classification structurelle de documents XML

3.1. *Présentation générale de l'approche*

Notre approche se situe dans le premier courant, à savoir, dans les approches qui consistent à classer directement des documents XML. Plus précisément, dans notre cas, on classe directement des documents XML selon leurs structures arborescentes (en ignorant leurs contenus). Comme illustré sur la figure 3, la structure arborescente (résumé d'arbre) est extraite par un *extracteur*, elle est ensuite utilisée par un *classifieur* pour classer le document XML correspondant.

Figure 3. *Approche de classification structurelle de documents XML*

3.2. *Extraction de la structure générique*

Cette structure est générique car dans notre approche ce n'est pas toute l'information structurelle qui est utilisée pour représenter un document XML. En effet, lorsqu'une balise est dupliquée, il est inutile de répercuter cette duplication dans la structure que nous voudrions extraire. En outre, tout autre type de balises (commentaires, instructions.), sera ignoré. Cependant, les attributs seront considérés comme les fils (sous balises) des balises dans lesquelles ils apparaissent.

L'algorithme d'extraction de cette structure est composé de deux parseurs. Le premier s'appuie sur SAX (Simple API for XML), API (Application Programming Interface), qui renvoie tous les tags (balises et attributs), rencontrés sur un document XML. Ces derniers sont interceptés, filtrés puis transformés par un second parseur en le "résumé d'arbre" correspondant, conformément aux prévisions de notre approche. L'essentiel de la tâche d'extraction est accompli par ce deuxième parseur, car c'est lui qui permet de passer de la forme linéaire (source) du document à sa représentation hiérarchique (résumé d'arbre). Une illustration de cette extraction est schématiquement présentée par la figure 4.

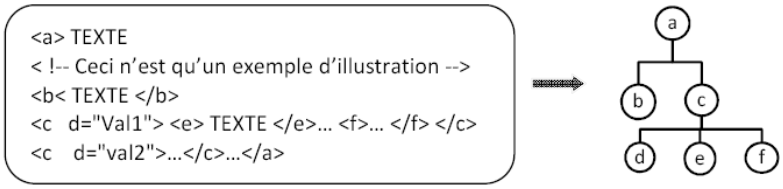

Figure 4. *Document XML et son "résumé d'arbre" généré par l'extracteur*

On remarque que le "résume d'arbre" est constitué uniquement de balises et d'attributs (les contenus du document ont disparu). On remarque sur le "résumé d'arbre", une seule occurrence du nœud c, alors qu'il y'en avait deux, sur le document XML en entrée. En outre, l'attribut d de la balise c apparait comme le fils du nœud c sur le "résumé d'arbre".

3.3. *Classification structurelle de documents XML*

3.3.1. *Rôles du cluster*

Dans notre modèle, un cluster remplit plusieurs rôles :

– Il peut être consulté à chaque fois lors du processus de classification pour accepter ou non de loger un document XML nouvellement traité.

–Il permet à un utilisateur d'interroger les documents XML qui s'y trouvent.

– Il peut servir ultérieurement pour créer un index basé conjointement sur les contenus et la structure des documents XML qui s'y trouvent.

3.3.2. *Structure d'un cluster*

On définit un cluster C_i par un ensemble d'arbres comme suit (formule [1]) :

$$C_i = \{T_{ik}/k = 1..p\} \text{ tel que } \lambda \leq Sim_{k=2..p}(T_{i1}, T_{ik}) \leq 1 \text{ et } 0 < \lambda \leq 1 \quad [1]$$

– $\{T_{ik}\}$ est l'ensemble des arbres du cluster C_i , et p son cardinal ;

– T_{i1} est le "résumé d'arbre" le plus pertinent à représenter l'ensemble des arbres du cluster C_i, on le nomme centroïde de C_i ;

– Sim est la fonction de similarité structurelle. Par définition, sa valeur est un nombre réel de l'intervalle [0,1].

– λ est le seuil à partir duquel les autres arbres $\{T_{ik}/k = 2..p\}$ de C_i sont similaires au centroïde T_{i1}.

REMARQUE. — NOUS VERRONS DANS L'ALGORITHME DE CLUSTERING QUI SUIT QUE LE CENTROÏDE EST MOBILE. CETTE « MOBILITE » EST FONDEE SUR LE PRINCIPE D'AUGMENTER AU MAXIMUM LA SIMILARITE ENTRE MEMBRES D'UN MEME CLUSTER, TOUT EN REDUISANT LA SIMILARITE ENTRE CLUSTERS.

3.3.3. *Algorithme de clustering*

```
/*  n étant le nombre de clusters et p = #Ci la taille du cluster Ci          */
Pour chaque arbre T /*  Représentant d'un document XML à classer              */
Faire Si Ω = ϕ   /*  L'ensemble des clusters Ω est vide                        */
      Alors      /*  Création du premier cluster                               */
            C1 ← T ; /* Placer T comme centroïde dans le cluster C1            */
            Ω = Ω ∪ {C1}  / * Ajouter ce premier cluster C1 dans Ω             */
      Sinon   /*  Cas où il existe déjà au moins un cluster                     */
              /* On parcourt Ω pour chercher le cluster à associer à T         */
            Si arg max_{i=1..n} (λ ≤{Sim(T_i1, T)} ≤ 1)  existe
            Alors   /* On a trouvé le cluster Ci le plus approprié pour T     */
                    Ci ← {T} / * Affecter l'arbre T au cluster Ci              */
                    /*  Calcul du nouveau centroïde pour le cluster Ci        */
                    Pour k=1..p /* Parcourir le cluster Ci en considérant     */
                                /* tous ses arbres, y compris le centroïde    */
                    Faire Importance(T_ik) = Σ_{T_ij ∈ Ci}  Sim (T_ik,T_ij) ;
                    T_i1= arg max_{k=1..p}(Importance(T_ik))
                    /* T_i1 est le nouveau centroïde calculé pour Ci          */
            Sinon  /*  On crée un nouveau cluster C pour l'arbre T            */
                    C ← {T} ; /* T est le centroïde du cluster créé C         */
                    Ω = Ω ∪ {C}
Fait
```

REMARQUE. — COMME ON PEUT LE CONSTATER, NOTRE METHODE DE CLASSIFICATION N'EST PAS SUPERVISEE. DE PLUS, LE NOMBRE DE CLUSTERS S'INCREMENTE AU FUR ET A MESURE QUE SE DEROULE LE PROCESSUS DE CLUSTERING.

Pour classer un arbre T (représentant un document XML à classer), l'algorithme précédent doit fonctionner comme suit :

– S'il n'existe aucun cluster dans la base Ω, il faut tout simplement en créer un en y mettant le premier arbre T qui se présente. On fixe initialement ce dernier comme centroïde de C_1.

– S'il existe déjà au moins un cluster, on parcourt la base Ω pour chercher s'il existe un cluster C_i qui permet de loger l'arbre T. Là aussi, deux cas peuvent être envisagés :

- Dans le premier, on est supposé avoir trouvé le cluster C_i le plus approprié pour l'arbre T. En l'occurrence, C_i est celui dont le centroïde T_{i1} donne le meilleur seuil de similarité avec l'arbre T. Cette question est réglée par la formule $argmax_{i=1..n}(\lambda \leq \{Sim(T_{i1}, T)\} \leq 1)$. Cependant, pour avoir le centroïde le plus représentatif pour C_i, il est primordial de calculer le poids de chacun de ses arbres. Pour cela, on utilise la fonction $Importance(T_{ik})$ qui nous permet d'identifier quel est l'arbre le plus important, et à ce titre, il sera désigné comme nouveau centroïde de C_i.

- Dans le second, il va falloir créer un nouveau cluster et l'intégrer dans la base Ω, ceci est réalisé par les deux dernières instructions de l'algorithme précédent.

3.4 *Calcul de la similarité structurelle de deux arbres*

De manière générale pour comparer deux termes on utilise un dictionnaire. Mais dans un contexte hiérarchique, il est nécessaire de considérer chaque terme (nœud) dans son étendue contextuelle. Ainsi, la similarité de deux nœuds va dépendre non seulement de leur similarité ontologique (en utilisant un dictionnaire), mais aussi, de celle de leurs contextes hiérarchiques. La figure 5 met en relief trois contextes : le *contexte ancêtres*, le *contexte des descendants immédiats* (fils) et le *contexte feuilles.*

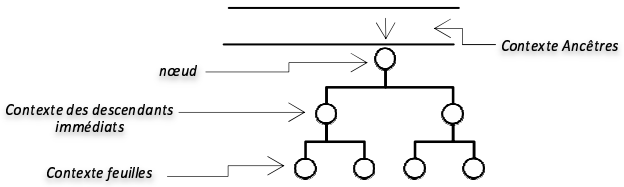

Figure 5. *Contextes utilisés pour un nœud*

A l'origine l'approche, utilisant ces trois context es, a été appliquée pour comparer des arbres représentant des DTDs dans (Mong *et al.*, 2002). Nous l'adaptons pour comparer des arbres représentant de s documents XML. Nous expliquons ci-après le formalisme utilisé.

La similarité de deux arbres T_1 et T_2 est calculée par la formule [2] suivante :

$$Similarité(T_1, T_2) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} sim(e_{1i}, e_{2j})}{Max(|T_1|,|T_2|)} \text{ où } \lambda \leq sim(e_{1i}, e_{2j}) \leq 1 \qquad [2]$$

$sim(e_{1i}, e_{2j})$ représente la similarité des nœuds e_{1i} et e_{2j}, qui est égale, au moins à λ et au plus à 1. Les nœuds e_{1i} et e_{2j} appartiennent respectivement aux arbres T_1 et T_2. $|T_1|$ et $|T_2|$ étant respectivement les tailles (nombres de nœuds) des arbres T_1 et T_2. La division par $Max\ (|T_1|, |T_2|)$ permet de normaliser le résultat de la sommation. $|T_1| = n$ et $|T_2| = m$. Le résultat sera la similarité des arbres T_1 et T_2.

Mais le calcul proprement dit, de la similarité de deux nœuds e_1 et e_2, tel qu'il est préconisé dans l'approche adoptée, nécessite d' introduire de nouvelles considérations qui tiennent compte du contexte hiérarchique de chaque nœud. Ainsi, pour synthétiser le calcul de la similarité de deux nœuds e_1 et e_2, la formule [3] agrège trois types de similarités comme suit :

$$sim(e_1, e_2) = w_1 * S_1(e_1, e_2) + w_2 * S_2(e_1, e_2) + w_3 * S_3(e_1, e_2) \qquad [3]$$

$$o\text{ù } w_1 \geq 0, w_2 \geq 0 \text{ et } w_3 \geq 0 \text{ sont des poids tels que } w_1 + w_2 + w_3 = 1$$

– $S_1(e_1, e_2)$ représente la *similarité sémantique* de e_1 et e_2. Cette dernière dépend de la similarité ontologique de e_1 et e_2 (déduite à partir d'un dictionnaire), et de la similarité de leurs ancêtres respectifs (calculée par la formule [5]).

La *similarité sémantique* de e_1 et e_2 est calculée par la formule [4] suivante :

$$S_1\ (e_1, e_2) = PCC(r_1.e_1,\ r_2.e_2) * S_0(e_1,\ e_2) \qquad [4]$$

$S_0\ (e_1, e_2)$ étant la similarité ontologique des nœuds e_1 et e_2. Un simple algorithme permet de calculer cette dernière. En effet, si les nœuds sont représentés par un même terme, leur similarité est évidemment égale à 1; s'ils sont synonymes, elle peut être fixée par exemple à 0.8 ou 0.9, sinon elle est égale à 0. Sur le tableau 1 on a un exemple de similarité ontologique de deux arbres. Les lignes sont représentées par les nœuds du premier arbre, les colonnes par ceux du second.

	facture	article	q	pu	design
facture	1	0	0	0	0
produit	0	0.9	0	0	0
code	0	0	0	0	0
qte	0	0	0.9	0	0
prixunit	0	0	0	0.9	0
num	0	0	0	0	0

Tableau 1. *Matrice de similarité ontologique de deux arbres*

$PCC(r_1.e_1,\ r_2.e_2)$ (Path Context Coefficient) représente la similarité des chemins $r_1.e_1$ et $r_2.e_2$ allant respectivement des racines r_1 et r_2 aux nœuds e_1 et e_2. En fait, c'est la similarité des ancêtres respectivement des nœuds e_1 et e_2. Les contextes ancêtres jouent un rôle prépondérant dans le calcul de la similarité. L'idée est que même si deux nœuds sont ontologiquement identiques ou similaires, ils ne le resteront pas forcément dans les contextes de leurs ancêtres respectifs qui peuvent être complètement différents. Pour calculer $PCC(r_1.e_1,\ r_2.e_2)$, on applique la formule [5] suivante :

$$PCC(r_1.e_1,\ r_2.e_2) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{q} sim_path(n_{1i}, n_{2j})}{Max(|r_1.e_1|,\ |r_2.e_2|)}$$

$$avec\ \lambda \leq sim_path(n_{i1}, n_{2j}) \leq 1 \qquad [5]$$

$sim_path(n_{1i},\ n_{2j})$ représente la similarité ontologique des nœuds n_{1i} et n_{2j}, qui est au moins égale à λ et au plus égale à 1. n_{1i} et n_{2j} appartiennent respectivement aux chemins (paths) $r_1.e_1$ et $r_2.e_2$. On normalise la sommation par le nombre de nœuds du plus long des chemins $r_1.e_1$ et $r_2.e_2$, c'est-à-dire, $Max(|r_1.e_1|,\ |r_2.e_2|)$. $|r_1.e_1| = p$ et $|r_2.e_2| = q$.

– $S_2(e_1, e_2)$ correspond à la *similarité des descendants immédiats* (fils) respectivement de e_1 et e_2. Autant que les contextes ancêtres, les contextes des descendants jouent également un rôle capital dans le calcul de la similarité. En effet, deux nœuds similaires ou identiques peuvent devenir complètement dissimilaires sous la contrainte de leurs descendants. La formule [6] permet de calculer la *similarité des descendants immédiats* (fils) respectivement de e_1 et e_2.

$$S_2(e_1,\ e_2) = \frac{\sum_{i=1}^{r}\sum_{j=1}^{s} sim_desc(n_{1i},\ n_{2j})}{Max(\ |desc_1|,\ |desc_2|)}$$

$$avec\ \lambda \leq sim_desc(n_{1i}, n_{2j}) \leq 1 \qquad [6]$$

$sim_desc(n_{1i},\ n_{2j})$ représente la similarité ontologique, des nœuds n_{1i} et n_{2j}, qui est au minimum égale à λ et au maximum égale à 1. n_{1i} et n_{2j} appartiennent respectivement aux ensembles des fils de e_1 et e_2. On normalise la sommation par le plus grand des cardinaux des deux ensembles des descendants immédiats $desc_1$ et $desc_2$, c'est-à-dire, $Max(\ |desc_1|,\ |desc_2|)$. $|desc_1| = r$ et $|desc_2| = s$.

– $S_3(e_1, e_2)$ correspond à la *similarité des contextes feuilles* des sous arbres respectivement de racines e_1 et e_2

REMARQUE. — POUR LEVER TOUTE EQUIVOQUE, LE TERME *feuilles* ICI DESIGNE NON PAS DES CONTENUS, MAIS TOUT SIMPLEMENT DES NŒUDS TERMINAUX D'UN ARBRE.

Le contexte feuilles a lui aussi son importance dans le calcul de la similarité. En effet, intuitivement deux nœuds similaires dans les contextes de leurs ancêtres et fils respectifs, ont plus de chance de le rester s'ils présentent des feuillages similaires que ceux qui n'en présentent pas. La formule [7] permet de calculer la *similarité des contextes feuilles* des sous arbres respectivement de racines e_1 et e_2.

$$S_3(e_1,\ e_2) = \frac{\sum_{i=1}^{t}\sum_{j=1}^{u} sim_leaf(n_{1i}\ ,\ n_{2j})}{Max(\ |leave(e_1)|,\ \ |leave(e_2)|)}$$

$$avec\ \lambda\ \leq sim_leaf(n_{1i}, n_{2j}) \leq 1 \qquad [7]$$

$sim_leaf(n_{1i}\ ,\ n_{2j})$ représente la similarité des feuilles n_{1i} et n_{2j} qui est au minimum égale à λ et au maximum égale 1. n_{1i} et n_{2j} appartiennent respectivement aux ensembles des feuilles des sous-arbres de racines e_1 et e_2. On note ces derniers respectivement par $leave(e_1)$ et $leave(e_2)$. On normalise la sommation par le plus grand des cardinaux de ces deux ensembles, c'est-à-dire, $Max(\ |leave(e_1)|, |leave(e_2)|)$. $|leave(e_1)| = t$ et $|leave(e_2)| = u$.

Mais pour calculer la similarité $sim_leaf(n_{1i}\ ,\ n_{2j})$ proprement dite, on reconduit la logique de la formule [4] pour chaque paire de nœuds feuilles (n_{1i}, n_{2j}), comme indiqué par la formule [8] suivante :

$$sim_leaf\ (n_{1i},\ n_{2j}) = PCC\big(e_1.n_{1i},\ e_2.n_{2j}\big) * S_0\big(n_{1i},\ n_{2j}\big) \qquad [8]$$

Tout comme avec la formule [4], la formule [8] cherche à calculer la similarité des nœuds n_{1i} et n_{2j}, qui sont des feuilles appartenant respectivement aux sous arbres de racines e_1 et e_2. Ce calcul implique la similarité ontologique $S_0\big(n_{1i},\ n_{2j}\big)$ (dictionnaire), des nœuds feuilles n_{1i} et n_{2j} et la similarité $PCC\big(e_1.n_{1i},\ e_2.n_{2j}\big)$ des chemins (similarité des ancêtres), allant de e_1 et e_2 respectivement à n_{1i} et n_{2j}.

Ce système de formules est applicable de manière générale à des nœuds internes ou racines. Cependant, si au moins un des nœuds e_1 ou e_2 est une feuille, il est évident qu'il(s) ne possède(nt) pas de descendants (ni fils, ni feuilles), et par conséquent, le calcul de leur similarité s'en trouve biaisé, car $S_2(e_1, e_2) = 0$ et $S_3(e_1, e_2) = 0$, alors qu'ils peuvent être ontologiquement identiques ou synonymes. Alors pour y remédier, le calcul des *similarités des descendants immédiats* et des *contextes feuilles* de ces nœuds sera remplacé par celui de leur *similarité sémantique*, comme indiqué ci-dessous par [9] et [10] :

$$S_2(e_1, e_2) = S_1(e_1, e_2) \qquad [9]$$

$$S_3(e_1, e_2) = S_1(e_1, e_2) \qquad [10]$$

4. Expérimentation

4.1 *Implémentation du système de classification*

Nous avons développé un premier programme en Java sous l'environnement JCreator. Le programme développé consiste en deux modules : le premier s'appuie sur SAX (Simple API for XML), API (Application Programming Interface), pour effectuer un premier parsing comme annoncé en section 3 (sous section 3.1). Ce module fournit un fichier intermédiaire intercepté par un deuxième module pour finaliser l'extraction du "résumé d'arbre".

Nous avons ensuite écrit un deuxième programme en C++ qui utilise les fichiers (résumés d'arbres) générés par le premier programme pour les clustériser.

Ces deux programmes correspondent respectivement à l'extracteur et au classifieur évoqués dans la sous section 3.1 à travers la figure 3.

4.2 *Evaluation des performances et Résultats*

Nous avons utilisé 2 collections de documents XML :

– Un corpus réel de documents XML ACM SIGMOD téléchargé à partir de www.acm.org/sigmod/record/xml. Ce corpus concerne les articles publiés par ACM SIGMOD.

– Une collection de documents XML fabriqués à partir d'un échantillon issu d'un corpus INEX 2007 (Initiative for the Evaluation of XML Retrieval). *INEX est une campagne d'évaluation des Systèmes de Recherche d'Information sur des documents XML. Son objectif principal est de promouvoir l'évaluation de la recherche sur des documents XML.*

Pour cette expérimentation nous n'avons pas eu besoin d'utiliser un dictionnaire (les vocabulaires utilisés pour les structures dans ces corpus sont standards). Nous avons alors inhibé la procédure de recherche des synonymes dans notre programme de classification (classifieur).

Le corpus ACM SIGMOD choisi est composé de 48 documents XML et de 4 DTDs qu'on peut considérer comme des classes cibles par rapport auxquelles on peut évaluer notre clustering. Par contre, les documents du corpus INEX sont hétérogènes (leurs DTDs ne sont pas connues à priori). Mais pour pouvoir effectuer notre test de clustering avec un tel corpus, nous avons fabriqué une collection comportant 250 documents XML avec l'outil Altova XMLSpy v2009. L'outil en question a été téléchargé à partir du site www.altova.com. Il permet de générer et d'associer une DTD pour un document XML et vice versa.

Pour préparer cette collection, nous avons choisi 10 documents XML INEX très hétérogènes pour lesquelles nous avons générés 10 DTDs par Altova XMLspy. Mais pour s'assurer que ces DTDs sont effectivement très différentes, nous avons utilisé un autre outil Altova DiffDog (téléchargé aussi de www.Altova.com), qui permet de

comparer deux documents et visualiser leurs différences. Ensuite, pour chacune de ces 10 DTDs ainsi fabriquées, nous avons construit un ensemble de 25 documents XML validés par Altova XMLspy. Chacun de ces ensembles contient des documents tirés du corpus INEX et des documents XML artificiels générés automatiquement par Altova XMLspy. Notons que pour confectionner un document XML à partir d'une DTD, Altova XMLspy offre plusieurs options. C es options consistent en des choix multiples sur le nombre de nœuds à garder, les répétitions de ces nœuds, les attributs associés à ces nœuds, les contenus, et même les tailles des tags.

Dans la phase suivante nous avons extrait d'abord d es deux collections précédentes, les "résumés d'arbres" correspondants, ensuite nous avons procédé respectivement à leur clustering. Pour évaluer la pertinence de notre approche, nous avons utilisé les mesures P (Précision) et R (Rappel), exprimées respectivement par les formules [11] et [12] suivantes :

$$P = \frac{Nombre\ d'arbres\ \ bien\ classés}{Nombre\ d'arbres\ du\ cluster} \qquad [11]$$

$$R = \frac{Nombre\ d'arbres\ \ bien\ classés}{Nombre\ de\ documents\ de\ la\ DTD} \qquad [12]$$

Nous avons effectué deux tests, en fixant les poids d'agrégation aux valeurs $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$ et $w_3 = \frac{1}{3}$ et le seuil de similarité respectivement à λ =0.5 et λ =0.7. Les résultats expérimentaux sont donnés par les tableaux 2 et 3. Les abréviations utilisées dans ces tableaux sont : Doc désigne le nombre de documents XML ; CL : Cluster ; S : similarité ; N : désigne le nombre d'arbres d'un cluster; et enfin X-Y dans la colonne S, signifie X ≤ valeur(S) ≤ Y.

Corpus réel							
		λ =0.5			λ =0.7		
Classes cibles	Doc	CL	N	S	CL	N	S
DTD1	30	C1	30	0.57-1	C1	19	0.94-1
DTD2	16	C2	16	0.86-1	C2	16	0.86-1
DTD3	1	C3	1	-	C3	1	-
DTD4	1	C4	1	-	C4	1	-
					C5	11	0.94-1
		P =1, R =1			P =1, R =0.76		

Tableau 2. *Résultats de classification avec le corpus réel*

Corpus synthétique							
		λ =0.5			λ =0.7		
Classes cibles	Doc	CL	N	S	CL	N	S
DTD1	25	C1	25	0.67-1	C1	20	0.83-1
DTD2	25	C2	25	0.86-0.94	C2	25	0.86-0.94
DTD3	25	C3	25	0.81-0.96	C3	25	0.81-0.96
DTD4	25	C4	25	0.60-0.91	C4	19	0.78-0.91
DTD5	25	C5	25	0.65-1	C5	23	0.81-1
DTD6	25	C6	25	0.83-1	C6	25	0.83-1
DTD7	25	C7	25	0.86-1	C7	25	0.86-1
DTD8	25	C8	25	0.78-0.94	C8	25	0.78-0.94
DTD9	25	C9	25	0.91-1	C9	25	0.91-1
DTD10	25	C10	25	0.94-0.98	C10	25	0.94-0.98
					C11	5	0.96-1
					C12	6	0.88-1
					C13	2	1
		P =1, *R* =1			*P* =1, *R* =0.948		

Tableau 3. *Résultats de classification avec le corpus synthétique*

REMARQUE. ~~S~~ ACHANT QUE LES SIMILARITES OBTENUES AVEC λ =0.5, SONT TOUTES SITUEES ENTRE 0.57 ET 1, NOUS N'AVONS REFAIT LE TEST (λ =0.7) QUE POUR CERTAINS ARBRES, A SAVOIR, CEUX DU CLUSTER5 (1ER CORPUS) ET CEUX DES CLUSTERS (CLUSTER11, CLUSTER12 ET CLUSTER13 DU 2EME CORPUS). LES SIMILARITES DANS CES CLUSTERS SONT TRES ELEVEES (ENTRE 0.88 ET 1).

Les valeurs des précisions *P* pour les deux corpus sont toutes à 1. Ceci s'explique par notre approche de représentation des documents XML. En effet, dans notre cas, un document XML est représenté par un "résumé d'arbre" très proche d'un arbre qu'on aurait dérivé à partir de la DTD a ssociée au document. Mais avec λ =0.7, les valeurs moyennes des rappels *R* sont de 0.76 et 0.948 respectivement pour les deux collections. En effet, dans certains cas, certains arbres présentent des similarités (par rapport au centroïde) insuffisantes (0.57 et 0.67) respectivement pour les deux corpus, qui ne leur permettent pas de rester regroupés dans les mêmes clusters comme dans le cas de λ =0.5. Ce décalage est dû à une perte d'information

provoquée par l'élimination de sous-arbres dont les racines sont des répétitions de nœuds frères. En outre, pour calculer la similarité de deux arbres, le système de formules ([4], [5], ..,.[10]), tient compte de l'age ncement hiérarchique de leurs nœuds, si bien qu'on ne trouve dans un même cluster que les documents ayant des structures hiérarchiquement très proches.

Nous avons vérifié manuellement les arbres ayant donné les plus petites similarités (0.57, 0.60, 0.65 et 067), et nous avons constaté qu'ils ne présentaient en fait, que de petites différences avec leurs centroïdes respectifs. Nous avons trouvé que ces différences sont en moyenne de l'ordre de 35%. Donc un taux moyen de 65% de nœuds en commun a suffi pour obtenir des similarités nettement supérieures à λ =0.5. De ce fait, et conformément aux résultats obtenus avec λ =0.7, il semble que λ =0.5 est une valeur moyenne satisfaisante pour notre approche avec le deux corpus testés.

5. Caractéristiques de la méthode proposée vis-à-vis de l'état de l'art

Par rapport à certaines méthodes évoquées dans l'état de l'art, notre proposition se démarque par un certain nombre de points importants.

– En effet, tout comme dans (Nierman *et al.*, 2002), (Francesca *et al.*, 2003) et (Dalamagas et al., 2004), dans notre approche un document XML est représenté par un "résumé d'arbre" *exact ou non* et *ordonné*, alors que dans (Termier *et al.*, 2002), un document XML peut être représenté par plusieurs sous-arbres fréquents (un document peut apparaitre dans plusieurs clusters), qui peuvent être *exacts ou non* et *ordonnés ou non*.

– Le clustering de (Termier *et al*, 2002) est basé sur le calcul du sous-arbre fréquent maximal commun aux documents XML de chaque cluster. (Nierman *et al.*, 2002), (Francesca *et al.*, 2003), (Costa *et al.*, 2004) et (Dalamagas et al., 2004), effectuent un clustering *hiérarchique*, en s'appuyant sur une mesure de *distance* (voir sous section 2.2.1). Notre approche effectue un clustering *incrémental* (voir algorithme 3.3.3), fondé sur un calcul de *similarité* (voir sous section 3.4).

–Nous avons adapté, le système de formules dévelop pé dans (Mong *et al*, 2002), mais dans notre cas, les calculs sont relativement allégés du fait que nous manipulons des arbres étiquetés simples. L'approche de (Mong *et al.*, 2002) établit un *clustering hiérarchique ascendant* sur des arbres représentant des DTDs. Les nœuds d'un arbre DTD dénotent souvent des expressio ns régulières dont le traitement n'est pas toujours trivial.

Mais, le caractère novateur de cette nouvelle approche par rapport à (Aïtelhadj *et al.*, 2006), concerne la « mobilité » des centroïdes. En effet, même si initialement à la création d'un cluster, c'est le p remier arbre qui se présente qui possède le statut de centroïde, il doit être remplacé, si c'est nécessaire, par un autre arbre plus représentatif. L'algorithme de clusterin g de la sous section 3.3.3 met bien l'accent sur ce point. En effet, la fonction *importance* qui mesure le *poids* de chaque

arbre du cluster dans cet algorithme, garantit la pertinence du nouveau centroïde calculé.

6. Conclusion et perspectives

Nous avons proposé une méthode de classification structurelle de documents XML. Pour cela, nous avons proposé une manière de représenter structurellement un document XML et un algorithme de clustering basé sur une mesure de similarité structurelle entre documents XML. La classification de telles structures équivaut à la classification structurelle des documents XML qu'elles représentent. Les clusters sont créés au fur et à mesure que les documents se présentent. Notre modèle touche à deux aspects fondamentaux intéressants de la RI. En effet, d'une part, la classification permet de réduire le nombre de documents traités et du coup, augmenter le nombre de documents pertinents qui peuvent être retournés par un moteur de recherche. D'autre part, les clusters constituent un index permettant aux utilisateurs d'accéder aux groupes de documents XML qu'ils souhaitent interroger et atteindre les "unités d'information" spécifiques qui les intéressent.

L'expérimentation menée n'est qu'une petite esquisse qui permet de tester la faisabilité et quelque peu la fiabilité de l'approche proposée. Cependant, pour un passage à l'échelle, il est judicieux d'établir des tests sur de plus grandes collections de documents avec des formules de calcul de la *précision* plus appropriées telles que la pureté, l'entropie et autres.

En perspective, on peut rebâtir ou étendre cette classification en y adjoignant la composante correspondant aux contenus des documents, afin que le système de recherche puisse traiter ultérieurement des requêtes constituées de combinaisons de mots appartenant aux vocabulaires de la structure et/ou du contenu.

7. Bibliographie

Aïtelhadj A., Boughanem M., « CSDX: Classification Structurelle de documents XML », *VSST'2006*, Télécom Lille1, 16-17 Janvier 2006.

Costa G., Manco G., Ortale R., Tagarelli. A., « A Tree-Based Approach to Clustering XML Documents by Structure », *In PKDD*, 2004, pp 137-148.

Dalamagas T., Cheng T., Winkel K.-J. Sellis T. K., « Clustering XML Documents Using Structural Summaries », *In EDBT Workshops*, 2004, pp 547-556.

Del Razo Lopez F., Laurent A., Poncelet P., Teisseire M., « Recherche de sous-structures fréquentes pour l'intégration de schémas XML », *In Conférence Extraction et Gestion des Connaissances (EGC 2006)*, Lille, Janvier 2006, volume II, p 487-498.

Denoyer L., Apprentissage et Inférence statistique dans les bases de documents structurés: Application aux corpus de documents textuels, Thèse de Doctorat Paris 6, 2004.

Doan A., Domingos P., Halevy A Y., « Reconciling schemas of disparate data source : a machine-Learning approach », *In proceeding of ACM SIGMOD international conference on Management data*, ACM Press 2001, p 509-520.

Francesca F. D., Gordano G., Ortale R., Tagarelli A « Distance-based Clustering of XML Documents », *In Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, 2003, pp 75-78.

Madhavan J., Bernstein P A., Rahm E., « Generic Schema Matching with Cupid », *VLDB* 2001.

Mong L., Huai L., Hsu W. et Yang X., « XClust: Clustering XML Schemas for Effective Integration », *School of Computing, National University of Singapore, In proceedings of 11th CIKM*, 2002, p 292-299.

Nierman A., Jagadish H. V., « Evaluating Structural Similarity in XML Documents », *In Proceedings of the Fifth International Workshop on the Web and Databases*, WebDB, 2002, Madison,Wisconsin, USA.

Su H., Padmanabhan S., Lo M., « Identification of Syntactically Similar DTD Elements in Schema Matching across DTDs », *WAIM*, 2001.

Termier A., Rousset M.C. et Sebag., « Treefinder : a First Step towards XML Data Mining », *In Proceedings of ICMD* 2002, p 450-457.