Utilisation de la théorie des graphes et de la distance d'édition pour la recherche d'information sur documents XML

Cyril Laitang — Karen Pinel-Sauvagnat

Institut de Recherche en Informatique de Toulouse, Equipe SIG-RFI, France
{ cyril.laitang, karen.sauvagnat }@irit.fr

RESUME. *La recherche d'information sur documents semi-structurés de type XML (RIS) permet de renvoyer à l'utilisateur des granules documentaires se focalisant sur les besoins exprimés. La requête et les documents structurés pouvant être vus comme des hiérarchies d'éléments imbriqués, nous considérons que leur proximité structurelle peut être évaluée au travers de la similarité entre leurs arborescences respectives. Dans ce cadre, nous proposons un modèle de RIS combinant au calcul de score sur le contenu une mesure de similarité structurelle basée sur la distance d'édition (le coût minimal d'opérations pour transformer un arbre en un autre). Nous propageons et combinons les scores ainsi obtenus en prenant en compte le voisinage de chacun des nœuds dans l'arborescence de nos arbres document. Nous avons évalué notre approche au travers de la tâche SSCAS de la campagne d'évaluation INEX 2005 et nos premiers résultats montrent l'intérêt d'une telle approche.*

ABSTRACT. *Information retrieval on semi-structured documents like XML (SIR) allows the user to narrow his search down to the document element level. Queries and semi-structured documents could be seen as hierarchically nested elements. We consider that their structural proximity could be evaluated over their trees similarity. Our SIR approach combines both content and structure scores, the latter being based on tree edit distance (minimal cost of operations to turn one tree to another). We then propagate and combine our score based on the neighbourhood of each of our nodes in the tree document structure. Our approach was evaluated over the SSCAS INEX's 2005 task and our first results show the interest of such an approach.*

MOTS-CLES : *Recherche d'information, graphes, XML, distance d'édition.*

KEYWORDS: *Information retrieval, graphs, XML, Tree Edit distance.*

Ce travail a bénéficié d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-08-CORD-009

1. Introduction

La recherche d'information sur documents semi-structurés de type XML (RIS) se différencie de la recherche d'information classique par sa prise en compte des informations structurelles des documents. L'utilisation de requêtes contenant des conditions de contenu et de structure permet de renvoyer à l'utilisateur des granules documentaires (éléments XML) plus précis et focalisés sur ses besoins.

Les documents XML sont composés de balises imbriquées les unes dans les autres. Cette organisation hiérarchique des éléments structurels nous permet de représenter un document XML sous la forme d'un graphe, et plus particulièrement d'un arbre. De la même façon, la spécification de contraintes structurelles comme moyen de localiser et de discriminer les mots clefs spécifiés dans la requête permet d'en d'établir une représentation arborescente. On définit *l'élément cible* d'une requête XML comme l'élément XML du type de celui que l'on veut obtenir au final. Le reste des conditions structurelles est nommé *support* ou *environnement*.
La figure 1 donne un exemple de traduction d'un document XML et d'une requête au format NEXI (Trotman, 2004). Il est à noter que dans notre exemple l'élément cible, soit le type d'élément que l'on cherche à retourner est "p". Pour des raisons de clarté nous avons ici simplifié le label des balises. Dans les types de collections sur lesquelles s'applique notre approche les balises sont porteuses de sémantique : on aura par exemple $article$ et non pas a.

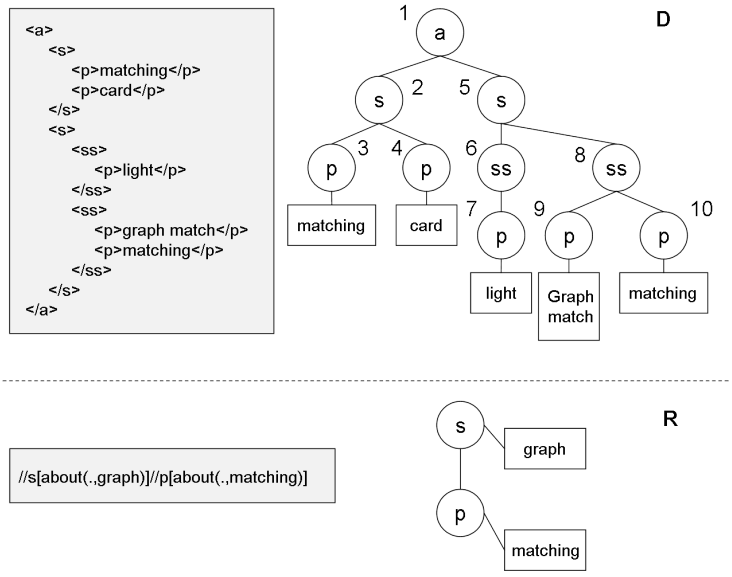

Figure 1. *Conversion d'un document XML et d'une requête en graphes. On peut traduire cette requête par : on recherche un élément "p" traitant de "matching" contenu dans un élément "s" portant sur "graph".*

Au vu de ces représentations, il apparaît que la mesure de pertinence structurelle des éléments des documents doit pouvoir être évaluée au travers d'algorithmes issus de la théorie des graphes ; et ce en mettant en rapport l'arborescence des documents avec celle de la requête. Les solutions proposées dans la littérature et utilisant explicitement la théorie des graphes sont relativement peu nombreuses. De plus, la majorité d'entre elles simplifient l'architecture arborescente des documents sous la forme de chemins, d'ensembles de nœuds ou d'arcs. Certains auteurs tels que (Alilaouar, 2007) (Aouicha, 2009) (Saito *et al.,* 2006) (Le *et al.,* 2010)) (Popovici *et al.,* 2005) sacrifient en général un niveau de détail important pour revenir à des approches de référence de leur domaine. Notre modèle est donc, à notre connaissance, une des rares tentatives d'utiliser des algorithmes propres à l'appariement approximatif de graphes dans un système de RIS.

Nous avons sélectionné et adapté un algorithme de la famille de la distance d'édition que nous avons intégré à notre modèle alliant le calcul et la propagation d'un score de contenu à celui de structure. Notre solution permet de tenir compte de l'intégralité de l'arborescence des documents dans la sélection des éléments pertinents.

La suite de cet article est organisée comme suit. La section 2 aborde les approches existantes et introduit les formalismes de la théorie des graphes ; la section 3 présente notre modèle et enfin la section 4 contient les résultats obtenus par notre approche sur la tâche SSCAS de la campagne d'évaluation INEX 2005. Enfin, dans la section 5 nous envisageons les orientations futures de nos recherches sur ce modèle.

2. Théorie des graphes et Recherche d'information sur documents semi-structurés

Nous présentons ici quelques solutions prenant en compte explicitement la structure arborescente des documents avant de poser les formalismes terminologiques propres à la théorie des graphes utilisée dans notre solution.

2.1. *Recherche d'information sur documents semi-structurés*

L'architecture arborescente des documents semi-structurés les désigne comme candidats naturels à l'application des propriétés et algorithmes issus de la théorie des graphes dans lesquels le problème d'appariement document-requête est ramené à une mise en correspondance de graphes. Toutefois, si les graphes et plus particulièrement les arbres sont couramment utilisés pour modéliser documents et requêtes, leur appariement en RIS ne se fait que rarement explicitement au travers d'algorithmes proposés par la théorie des graphes (Lalmas *et al.,* 2006) (Fuhr *et al.,* 2008) (Demartin *et al.,* 2009) (Geva *et al.,* 2010).

Les approches générales proposées par la théorie des graphes pour l'appariement d'arbres sont de deux types :
- *l'appariement exact* dans lequel on cherche à retrouver des sections des graphes

identiques.
- *l'appariement approximatif* dans lequel on souhaite obtenir une évaluation de la similarité entre deux graphes. Les algorithmes de ce domaine peuvent être des algorithmes de distance d'édition, d'inclusion, d'alignement ou de sous graphes communs maximum. Notre problématique étant liée au domaine de la recherche d'information (et non des bases de données), les approches qui peuvent nous être utiles se situent dans ce domaine.

On trouve dans la littérature trois grandes manières d'appréhender la structure dans un système de RIS au travers de l'utilisation de la théorie des graphes.
La première consiste en l'utilisation de la *relaxation*, soit une réduction du niveau de restriction sur les liens ou des contraintes. La relaxation est selon (Amer-Yahia *et al.,* 2002) un "processus d'élargissement de l'espace de recherche".
L'idée est de ramener le problème de la gestion de l'arborescence de l'arbre en un ensemble de groupes de nœuds liés par des arêtes pondérées en fonction de leur distance hiérarchique originelle. Cette simplification de la représentation permet de se rapporter à des approches connues du domaine tel que l'espace vectoriel. A titre d'exemple, dans sa thèse, (Alilaouar, 2007) utilise un système de relaxation couplé à un arbre couvrant minimum pour effectuer l'appariement de documents semi-structurés hétérogènes. (Aouicha, 2009) quant à lui relaxe l'ensemble des contraintes hiérarchiques. Pour ce faire, il ajoute des arcs pondérés par leur distance puis transpose ces arcs sur un espace vectoriel. (Saito *et al.,* 2006) enfin regroupe des ensembles de n-uplet correspondant à la requête et reliés par un ancêtre commun minimum.
Une autre famille d'approches utilise la *fermeture floue* (Damiani *et al.,* 2000) (Damiani *et al.,* 2001). De la même manière que pour la relaxation, il s'agit ici de créer un ensemble d'arcs virtuels représentatifs du niveau hiérarchique. Pour ce faire les auteurs utilisent ici la propriété de fermeture pour retrouver toutes les relations transitives présentes dans le graphe.
On trouve également quelques rares applications des algorithmes de *distance d'édition* dans la littérature. (Popovici *et al.,* 2005) a utilisé la distance d'édition des chaines de caractères de (Levenshtein, 1966) en ramenant le graphe sous la forme d'un chemin. Enfin dans leur article les auteurs de (Le *et al.,* 2010) posent les bases d'un modèle de recherche d'information utilisant la distance d'édition proposée par (Tai, 1979). Notre modèle est d'ailleurs une évolution de ce dernier.

D'autres travaux proches de notre modèle se situent dans les domaines connexes à la RIS adhoc tels que la détection d'erreurs et la catégorisation. Dans leurs articles, les auteurs de (Boobna *et al.,* 2004) (Rougemont *et al.,* 2008) vérifient la conformité de documents XML à leur DTD au travers du calcul de la distance d'édition entre leurs deux arborescences. Les auteurs de (Dalamagas *et al.,* 2004) et de (Dalamagas *et al.,* 2006) quant à eux utilisent la distance d'édition sur des résumés d'arbres afin d'en réduire l'espace structurel de calcul, et ce à des fins de catégorisation.

Nous venons de le voir, il n'y a, à notre connaissance, que peu de modèles de RIS se basant directement sur la théorie des graphes. Ceci peut-être expliqué par une complexité qui telle qu'elle reste relativement élevée. L'ensemble des approches cherchent

toujours d'une manière ou d'une autre à ramener le problème sur un espace réduit en traduisant les arbres. Cette réduction élimine malheureusement une partie importante des informations sur l'environnement des balises ; environnement dont la prise en compte nous semble pouvoir apporter un réel gain de pertinence dans l'évaluation des éléments à retourner.

2.2. *Définitions*

Un *graphe* G est un ensemble de nœuds connectés entre eux par des arcs appelés arêtes. Un graphe *orienté* est un graphe dont les arêtes indiquent un sens de navigation, autrement dit dont l'arc porte comme information le sens de parcours de nœud à nœud. Une suite d'arêtes consécutives (on utilise également l'appellation *chaîne*), dont les deux sommets aux extrémités sont identiques, est appelé *cycle*. Dans un cycle, le nœud initial est également le nœud terminal de la chaîne.

L'ensemble de ces définitions nous permet de définir un *arbre* $T()$ comme un graphe étant *acyclique* et *orienté*. Les arbres sont organisés de manière hiérarchique à partir d'un nœud appelé la *racine*. On notera $T(v)$ l'arbre de nœud racine v ou enraciné en v.
Aux différents niveaux de la hiérarchie correspondent des appellations différentes. Ainsi le nœud unique de départ ou du sommet (premier niveau) est appelé nœud *racine*. Les nœuds de niveaux hiérarchiquement inférieurs d'un nœud sont appelés *descendants* tandis que ses prédécesseurs dans la hiérarchie sont ses *ancêtres*. Le descendant immédiat d'un nœud sera son *fils* et réciproquement son ascendant immédiat sera son *père*. Les nœuds partageant le même père sont appelés *frères* et enfin, les nœuds sans descendant donc situés à l'extrémité inférieure de l'arbre sont appelés nœuds *feuilles*. Ce sont ces nœuds qui sont porteurs des informations de contenu des documents XML (texte).

Enfin, tout comme pour son usage dans la langue courante le terme *forêt désigne un ensemble d'arbres*, soit une *union disjointe d'arbres*. Ici nous utiliserons donc forêt dans le sens d'*un ensemble de graphes connexes acycliques orientés* enraciné artificiellement par un nœud parent commun et ayant pour fils l'ensemble des nœuds racines des arbres, et donc comme descendants l'ensemble de tous les nœuds des arbres.

2.3. *Similarités structurelles entre les graphes : Distance d'édition*

Deux graphes isomorphes sont identiques : ils contiennent les mêmes nœuds liés par les mêmes arêtes. Rechercher le degré d'isomorphisme entre deux graphes, ou les apparier, revient à mesurer la similarité entre la structure de ces deux arbres. On distinguera appariement approximatif d'arbres d'appariement exact. Le premier calcule un degré de correspondance entre deux arbres, tandis que, par opposition, le second cherche à valider ou non, la correspondance entre les graphes.
On relèvera quatre grandes familles d'appariement approximatif que sont la recherche

d'un sous graphe-commun maximal, la distance d'édition, la distance d'alignement et l'approche par inclusion.

Rechercher le *sous-graphe commun maximal* de deux graphes consiste en la découverte du sous-graphe partageant le plus grand nombre de nœuds commun aux deux.

La *distance d'édition* est une méthode inspirée par les travaux sur les distances de chaînes de caractères (Levenshtein, 1966) généralisée aux arbres par (Tai, 1979). La distance d'édition est la combinaison de trois opérations élémentaires que sont l'ajout ou la suppression d'un nœud dans l'arbre source et la substitution des labels des nœuds source et cible. La similarité se mesure par le nombre minimal d'opérations permettant de rendre le graphe source isomorphe au graphe cible.
Soient deux forêts F et G, Γ_F et Γ_G leurs nœuds les plus à droite, $c_{del}()$ la fonction de calcul du coût de suppression ou d'insertion, et $c_{match}()$ la fonction de calcul du coût de substitution ; le lemme sur lequel se base la distance d'édition est de la forme :

$$
\begin{array}{l}
d(F,\emptyset) = d(F-\Gamma_F,\emptyset) + c_{del}(\Gamma_f) \\
d(\emptyset,G) = d(\emptyset,G-\Gamma_G) + c_{del}(\Gamma_G) \\
d(F,G) = min \left\{ \begin{array}{ll}
\textit{(a)} & d(F-\Gamma_F,G) + c_{del}(\Gamma_F) \\
\textit{(b)} & d(F,G-\Gamma_G) + c_{del}(\Gamma_G) \\
\textit{(c)} & \begin{array}{l} d(T(\Gamma_F)-\Gamma_F, T(\Gamma_G)-\Gamma_G) \\ +d(F-T(\Gamma_F), G-T(\Gamma_G)) + c_{match}(\Gamma_F,\Gamma_G) \end{array}
\end{array} \right.
\end{array}
$$

[1]

On parle des trois opérations de la distance d'édition pour *(a)*, *(b)* et *(c)*. Il s'agit respectivement de la suppression du nœud le plus à droite de F, du nœud le plus à droite de G et de la substitution du noeud le plus à droite de F par le plus à droite de G.
Sasha et Zhang (Zhang *et al.,* 1989) ont par la suite réduit la complexité en temps et en espace de calcul de cet algorithme en proposant entre autre de calculer les sous chaînes des arbres. (Klein, 1998) puis (Demaine *et al.,* 2009) l'ont réduit grâce à une stratégie de décomposition récursive au travers du calcul du chemin lourd (le choix du sens gauche ou droite du nœud sur lequel appliquer l'opération d'édition). (Dulucq *et al.,* 2003), enfin, utilisent une stratégie de couverture pour être certains de toujours choisir le sens le plus avantageux dans la décomposition.

La *distance d'alignement* dont le représentant principal est (Jiang *et al.,* 1994) correspond à une restriction de la distance d'édition dans laquelle toutes les insertions doivent être effectuées avant les suppressions. Soit T_1 et T_2 deux arbres de labels, γ est une fonction de coût sur un couple de labels. Le coût d'alignement optimal (minimal) sera de la forme $\delta(T_1,T_2) = min(\gamma(S))$ avec S séquence d'opérations de transformation de T_1 en T_2 et γ la fonction de calcul de coût. Par nature les algorithmes à base de distance d'alignement semblent coûteux en espace mémoire.
On leur préfère donc, en général, ceux basés sur les distances d'édition pour les appariement de moins de trois graphes.

L'*inclusion d'arbres*, telle que développée par (Chen, 1998), est un cas particulier du système de calcul de la distance d'édition, où les coûts de suppression sont égaux à 0. Soit deux arbres T_1 et T_2, on dit que T_1 est inclus dans T_2 s'il existe une séquence d'opérations de suppression effectuées sur T_2 qui le rendent isomorphe à T_1.
On réserve en général l'inclusion aux problèmes d'appariements exacts ou liés aux bases de données.

3. Un modèle de recherche d'information par la distance d'édition

La distance d'édition, méthode très utilisée dans de nombreux domaines allant de l'analyse de texte à la bioinformatique, n'a pratiquement jamais été utilisée dans le cadre de la RIS.
Dans ce contexte nous proposons un modèle basé sur les propriétés de la théorie des graphes et alliant deux mesures, une pour le contenu et une pour la structure.

3.1. *Calcul du score de contenu*

La mesure du score de contenu des éléments se base, dans notre approche, sur trois opérations successives. Dans un premier temps, il s'agit de détecter les nœuds feuilles pertinents. Dans un second temps, on procède à l'extraction des sous arbres formés par les ancêtres des nœuds feuilles pertinents. Enfin, on effectue la propagation des scores de poids des éléments pour obtenir le score de contenu final des nœuds.

3.1.1. *Extraction et évaluation des éléments textes pertinents*

La première étape de notre modèle consiste en la recherche des nœuds feuilles pertinents. Pour ce faire, nous utilisons un algorithme de recherche d'information adhoc de la forme $tf \times idf$ qui récupère un ensemble de nœuds feuilles considérés comme pertinents par rapport à la concaténation des conditions de contenu de la requête. Les scores obtenus sont appelés "poids de contenu" $p(x)$ pour le nœud feuille courant x.

3.1.2. *Détection des sous-graphes candidats*

Nous cherchons ensuite à obtenir tous les sous graphes possibles depuis les nœuds feuilles de l'étape précédente. Pour ce faire, nous remontons la hiérarchie des ancêtres des nœuds feuilles pertinents (au score non nul) jusqu'à la racine du document et nous retournons les sous-arbres enracinés.
Soit x un nœud feuille pertinent, $\forall n \in Anc(x)$ nous retournons donc tous les $T(n)$.
Dans la figure 2 notre document XML et la requête ont été traduits sous forme arborescente et un poids a été attribué aux nœuds feuilles pertinents. On retourne sept sous arbres.

Nos nœuds n racines des sous-arbres candidats pour la distance d'édition serviront également à la propagation des scores de contenu, nous leur attribuons un score de poids de contenu égal à la somme du poids de tous leurs descendants. Nous conservons

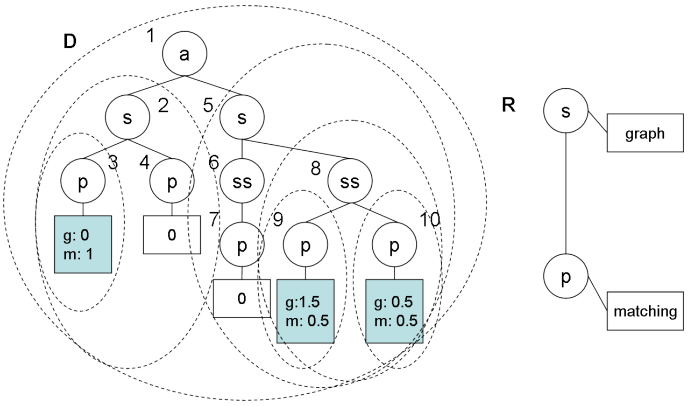

Figure 2. *Extraction des sous graphes pertinents à apparier basée sur les nœuds feuilles possédant un score de contenu non nul.*

toutefois la trace du nombre de nœuds feuilles $\mid leafs(n) \mid$ qu'ils contiennent.
Ainsi, dans notre figure 2 le nœud $n = 2$ possède un poids de contenu de 1 avec deux nœuds feuilles.

3.1.3. *Propagation et calcul du score de contenu*

Dans le principe, la propagation et la combinaison finale du contenu permet d'évaluer le score d'un nœud hiérarchiquement éloigné des nœuds feuilles. En d'autres termes, on cherche à tenir compte de l'environnement du nœud au niveau de ses ancêtres et de ses frères.
Pour ce faire, notre score de contenu $c(n)$ d'un nœud n combine le poids de l'élément au poids obtenu par ses ascendants. Dans le principe, puisqu'un nœud cumule la valeur de ses descendants il peut transmettre par son poids à ses fils une partie du score de leurs frères. L'intuition générale est qu'un nœud placé à proximité d'un élément pertinent dans une arborescence, doit pouvoir être favorisé par rapport à un nœud isolé.
Notre formule de calcul du score de contenu du nœud comporte trois éléments :
- La partie consacrée au *calcul du score du nœud* en question $\frac{p(n)}{|leafs(n)|}$, avec $p(n)$ le poids cumulé de tous les descendants textes du nœud $p(n) = \sum_{x \in leaf(n)} p(x)$ et $\mid leafs(n) \mid$ le nombre de descendants feuille du nœud n.
- La partie consacrée à la prise en compte de son *environnement immédiat* au travers de son père $\frac{p(a_1)-p(n)}{|leafs(a_1)|}$, avec $a \in Anc(n)$ donc a_1 le premier ancêtre du nœud n (son père) et par extension $p(a_1)$ le poids cumulé pour son père et $\mid leafs(a_1) \mid$ le nombre de nœuds feuilles dans les descendants de son père. Cette formule nous permet d'ajouter une partie du poids du parent immédiat au nœud courant à condition que ce dernier soit d'un poids plus important (qu'il y ait bien du poids ailleurs dans la hiérarchie à transmettre).

- La partie consacrée à la transmission des scores de *propagation des ancêtres* les plus éloignés obtenu plus haut dans la hiérarchie $\frac{c(a_1)-\frac{p(a_1)}{|leaf(a_1)|}}{|childs(a_1)|}$ Cette transmission est effectivement effectuée par $c(a_1)$ (score de contenu global du nœud père a_1) auquel on retire son poids grâce à $\frac{p(a_1)}{|leaf(a_1)|}$, avec $\mid childs(a_1) \mid$ le nombre descendant directs du nœud père. Globalement cette fonction permet de repasser le poids qui a été gagné à un plus haut niveau pendant la propagation tout en diminuant lentement ses effets en fonction du nombre de fils de chaque nœud. Ainsi plus on sera proche d'une racine et plus le poids effectif sera égal au poids des feuilles.

Le score de contenu $c(n)$ de l'élément n est ainsi obtenu en additionnant *le score de poids de contenu de l'élément auquel on ajoute celui de son père puis enfin celui de tout ses ascendants.* Formellement on aura donc :

$$c(n) = \frac{p(n)}{\mid leafs(n) \mid} + \frac{p(a_1) - p(n)}{\mid leafs(a_1) \mid} + \frac{c(a_1) - \frac{p(a_1)}{|leaf(a_1)|}}{\mid childs(a_1) \mid} \quad [2]$$

3.2. *Calcul du score de structure*

Le calcul des scores de structure des différents sous arbres extraits précédemment se fait au travers d'un algorithme de distance d'édition. Ces scores sont ensuite combinés afin d'obtenir un score moyen de l'élément candidat à la sélection.

3.2.1. *Couverture optimale de la distance d'édition*

Parmi tous les algorithmes de distance d'édition que nous avons explorés notre choix s'est porté sur une stratégie de couverture optimale à la manière de (Dulucq *et al.,* 2003). Nous réalisons cette dernière en nous basant sur la distance d'édition de (Demaine *et al.,* 2009) au travers du chemin lourd introduit par (Klein, 1998).

Comme nous l'avons vu dans la section 2.3, la distance d'édition est une mesure de similarité qui s'appuie sur le nombre minimal d'opérations de suppression ou de substitution, pour transformer un graphe en un autre (dans notre cas un document par une requête). Étant récursive, la complexité en espace et en temps diminue en fonction du sens de parcours de l'arbre sur lequel on applique ces opérations. L'ensemble des choix de direction de récursion constitue la *stratégie de couverture*.
La recherche du *chemin lourd* se justifie dans la mesure où en choisissant toujours le nœud le plus éloigné de ce chemin le nombre de sous arbres en mémoire est minimal. Le *chemin lourd* est formé par la suite des nœuds ayant le plus de descendants. On l'obtient donc en calculant le chemin de la racine jusqu'aux feuilles passant par les nœuds dont les sous arbres enracinés possèdent la cardinalité la plus élevée.

Pour un nœud n du chemin lourd et la fonction $childs()$ retournant les descendants directs du nœud ; un chemin lourd $lourd = [n_1, n_2, ..., n_i, ..., n_z]$ est :

$$\forall (n_i, n_{i+1}) \in lourd \begin{cases} n_{i+1} \in childs(n_i) \\ \forall x \in childs(n_i), x \notin n_{i+1}, \mid T(n) \mid \geq \mid T(x) \mid \end{cases}$$

Les deux graphes sont par la suite envoyés sur l'algorithme de distance d'édition pour les coûts suivants :
- c_{del} le coût de suppression vaut 0.5 pour un élément présent dans la requête et 1 pour un élément non présent.
- c_{match} le coût de substitution est fixé à 0 entre deux éléments semblables [1] et à 1 pour les éléments dissemblables.
Ces choix de coûts s'expliquent par la volonté de ne pas fausser l'algorithme de distance d'édition tout en valorisant les nœuds présents de la requête de manière plus importante ; et ce étant donné la dissymétrie de taille entre document et requête.

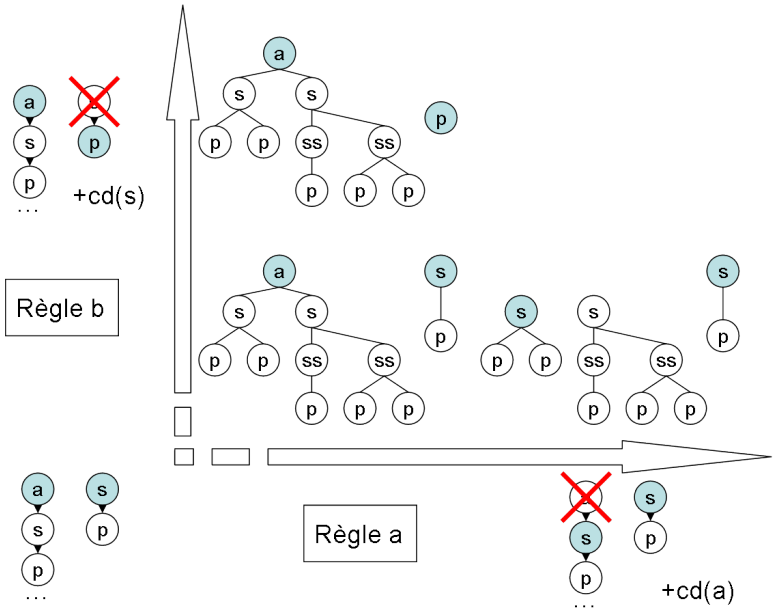

Figure 3. *Application des règles a et de la règle b de la distance d'édition, respectivement suppression d'un nœud du document et d'un nœud de la requête.*

L'originalité de notre approche tient principalement au fait que nous ne modifions pas l'algorithme de distance d'édition, mais pré-calculons le parcours avant l'injection. Autrement dit, notre adaptation du chemin lourd nous permet d'obtenir la stratégie de parcours optimale sans refaire les calculs de direction.
Nous injectons les positions de ce chemin dans notre algorithme récursif correspondant au lemme de la distance d'édition vue en section 2.3 (formule [1]) dont les trois opérations *(a)*, *(b)* et *(c)* sont illustrées dans les figures 3 et 4. Dans ces figures les chaines sont les chemins de parcours optimaux en mémoire et les arbres la correspondance virtuelle sur les arbres documents et requêtes.

1. A la sémantique proche ou identique, par exemple $section$ et $sous-section$.

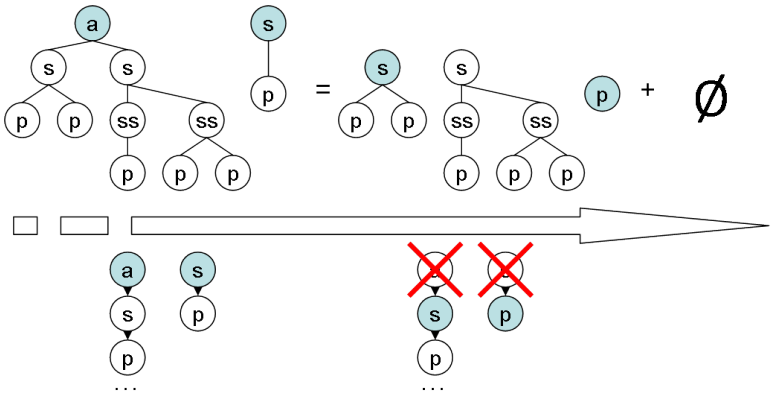

Figure 4. *Application de la règle c de la distance d'édition soit la substitution d'un nœud du document à un nœud de la requête.*

3.2.2. *Combinaison de la structure*

Le score de structure final est obtenu par la combinaison de l'ensemble des scores d'édition des sous arbres enracinés extraits comme indiqué dans la section 3.1.2.

Il s'agit ici d'avantage d'une combinaison que d'une propagation. Plus le sous-arbre est éloigné du nœud feuille (ou plus il est proche de la racine) et plus il englobe une large portion du graphe originel. En ce sens, combiner le score d'édition des sous arbres successifs revient à propager la similarité de l'environnement de plus en plus lointain du nœud.

Notre fonction de propagation de la structure cherche à moyenner l'ensemble des distances d'édition elles-même pondérées par la cardinalité des arbres en entrée, ceci afin de réduire les écarts de taille entre les sous arbres successifs. Le score de structure $s(n)$ est donc le suivant

$$s(n) = \frac{\sum_{a \in Anc(n)} (1 - \frac{d(T(a),R)}{|T(a)|})}{\mid Anc(n) \mid} \quad [3]$$

Avec $Anc(n)$: l'ensemble des ancêtres de n ; a : un nœud sur l'ensemble des ancêtres de n tel que $a \in Anc(n)$; $T(a)$: le sous arbre enraciné en a ; $d(T(a), R)$: la distance d'édition entre le sous arbre $T(a)$ et la requête R.

3.3. *Combinaison*

Au terme des étapes précédentes nous avons obtenus un score de contenu $c(n)$ ainsi qu'un score de structure $s(n)$ pour chaque arbre candidat extrait comme indiqué en section 3.1.2.

Le score final $score(n)$ du nœud n est obtenu par la combinaison des scores préalablement normalisés par 1 - $\frac{score}{max}$ avec max le score d'édition le plus élevé afin d'obtenir un score $\in [0, 1]$. On aura la formule de combinaison finale avec $\lambda \in [0, 1]$

$$score(n) = \lambda \times c(n) + (1 - \lambda) \times s(n). \quad [4]$$

Les éléments sont filtrés pour correspondre aux nœuds cibles puis ordonnés selon leur score et renvoyés en réponse à la requête de l'utilisateur.

4. Évaluations et expérimentations

Afin d'évaluer notre modèle nous avons implémenté notre solution au sein d'un prototype et comparé nos résultats obtenus avec ceux des participants de la campagne d'évaluation INEX 2005 pour la tache SSCAS.

4.1. *La collection INEX*

4.1.1. *Collection et tâches*

INEX (*Initiative for the Evaluation of XML Retrieval*) est la campagne d'évaluation de référence des différents systèmes de RIS appliqués aux documents XML.
Pour nos expérimentations nous avons utilisés la collection INEX 2005. Cette collection est composée d'articles scientifiques provenant de la IEEE Computer Society, balisées au format XML. La collection contient environ 12000 articles.
Un article moyen est composé d'environ 1500 éléments, et la profondeur moyenne des documents est de 6.9. Au total, la collection contient 8 millions de nœuds et 192 balises différentes.
La campagne INEX 2005 contient deux grands types de requêtes :
Les requêtes de la tâche CO (*Content Only*) qui ne s'intéressent qu'au contenu des documents et les requêtes de la tâche CAS (*Content And Structure*) qui contiennent des contraintes sur la structure des documents, comme par exemple des conditions de contenu de tel ou tel élément. Nous avons choisi cette campagne pour ses tâches d'évaluations centrées sur la structure, que l'on ne retrouve pas dans les campagnes postérieures [2].
La tâche CAS compte en fait quatre sous-tâches :
- *VVCAS* : les conditions de structure doivent être considérées comme vagues sur l'élément cible et son environnement.

2. Il est à noter qu'en 2010 une nouvelle tâche *Datacentric* est apparue. Elle est centrée sur des structures riches dans lesquelles les modèles de recherche d'information classiques ne suffisent pas, en d'autres termes, lorsque les labels des éléments structurels portent une information essentielle que le contenu textuel ne spécifie pas. Cette nouvelle tâche se base sur un corpus extrait du site internet *IMDB* qui regroupe 1 590 000 films et plusieurs millions d'acteurs, producteurs, réalisateurs, etc. Nos prochaines expérimentations utiliseront bien sûr cette collection, dès que les requêtes et jugements de pertinence seront disponibles.

- *SVCAS* les conditions de structure sont strictes sur l'élément cible et vagues sur son environnement.
- *VSCAS* : les conditions de structure doivent être considérées comme vagues sur l'élément cible et strictes sur son environnement.
- *SSCAS* : les conditions de structure doivent être considérées comme strictes sur l'élément cible tout comme sur son environnement.
C'est sur cette dernière tâche que nous avons porté notre choix. Elle comporte les contraintes permettant d'évaluer les requêtes dans lesquels la sémantique de la structure est porteuse d'information sur la pertinence des éléments.

4.1.2. *Mesures de pertinence*

Les mesures de référence des sous-tâches CAS sont au nombre de deux (Kazai *et al.,* 2006) :
- *nxCG* pour gain cumulé normalisé (*normalized cumulated gain*). Cette mesure représente la valeur cumulée jusqu'à un seuil (par exemple *nxCG10* pour les dix premiers résultats) par rapport à un classement idéal.
Pour i le rang, xCI le vecteur de classement idéal et $xCG(i)$ la somme des scores jusqu'au rang i :

$$nxCG(i) = \frac{xCG(i)}{xCI(i)} \qquad [5]$$

- *MAep* pour Moyenne Effort Précision (*Non-interpolated mean average effort-precision*). Cette mesure est utilisée pour moyenner les valeurs d'effort-précision pour chaque rang auquel un élément pertinent est renvoyé.
Nous utilisons la quantification "stricte" qui n'augmente la pertinence que lorsque l'élément retourné est exactement l'élément recherché.

4.2. *Résultats*

Afin d'évaluer les performances globales de notre solution et de mesurer l'importance de la structure nous avons fait varier notre paramètre λ de l'équation [4] de 0.1 pour le maximum de structure à 1.0 pour un score basé uniquement sur le contenu. Les résultats sont sont présentés dans le tableau 1.

Notre première observation est que les résultats optimaux toutes mesures confondues sont obtenus avec $\lambda \in \{0.4; 0.5\}$ pour le λ de l'équation [4], soit pour une part équivalente de structure et de contenu. Nous observons également que pour une part excessive de structure($\lambda < 0.3$), les résultats s'effondrent tandis que pour une part majoritaire de contenu ils régressent jusqu'à un seuil de 0.32 pour les 50 premiers résultats et de 0.15 pour la *MAep*. Enfin lorsque les λ est élevé les premiers résultats *nxCG 10, 25 et 50* augmente tandis-ce que la moyenne diminue. On en conclut que la structure fait remonter le score des premiers éléments pertinents.

λ	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
nxCG10	0.0	0.15	0.40	**0.42**	**0.42**	0.37	0.35	0.32	0.30	0.30
nxCG25	0.0	0.14	0.39	0.41	**0.42**	0.39	0.37	0.36	0.36	0.35
nxCG50	0.0	0.37	0.40	**0.41**	0.39	0.38	0.37	0.36	0.35	0.34
MAep	0.0	0.11	0.13	0.14	**0.19**	0.18	0.18	0.17	0.16	0.15

Tableau 1. *Présentation de nos résultats selon les différentes mesures pour la tâche SSCAS en quantification stricte.*

4.3. *Comparaison à l'état de l'art*

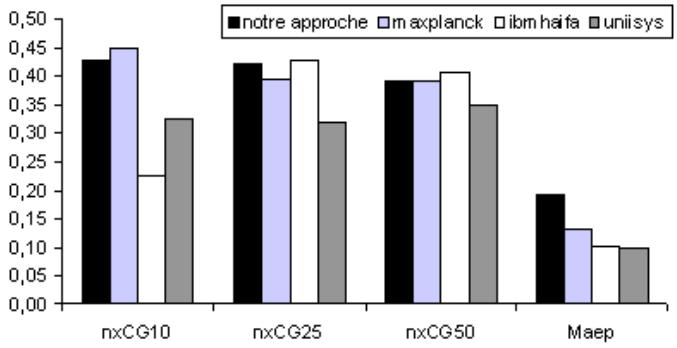

Figure 5. *Évaluation de notre modèle par rapport aux premiers "runs" de la tâche SSCAS pour la campagne d'évaluation INEX2005*

La figure 5 nous permet de placer notre meilleur "run" par rapport aux meilleurs participants de l'époque : l'institut Max Planck avec son système TopX (Theobald *et al.,* 2005) centré sur une approche base de donnée, IBM Haifa Research Lab qui propose une adaptation du modèle vectoriel (Mass *et al.,* 2004) et l'University of Klagenfurt qui utilise un modèle vectoriel (Hassler *et al.,* 2005). On observe que nos résultats pour les mesures *nxCG* sont sensiblement les mêmes que les meilleurs de l'époque ; en moyenne 5% en dessous du premier et 13% au dessus du second.
Nos meilleurs résultats se portent sur la *MAep* ou nous réalisons une amélioration d'environ 45 % par rapport au meilleur "run" et de plus de 90% par rapport au second.

5. Conclusion et perspectives

Nous avons, au cours de cet article, présenté un modèle de RIS sur documents XML permettant la détection et l'évaluation de la pertinence d'un élément au travers du score de son contenu et de sa structure. La principale originalité de notre contribution porte sur le choix de la distance d'édition, algorithme propre à la théorie des graphes. Les différentes mesures de propagation que nous avons mis en place afin

de combiner et de renforcer les scores en fonction de l'organisation structurelle des arbres nous ont permis d'obtenir des résultats qui s'annoncent très encourageants tout particulièrement sur la mesure MAep.

Nos futurs travaux s'organiseront autour de quatre axes. Nous souhaitons renforcer les scores de la partie contenu en remplaçant le $tf \times idf$ par un modèle de langue plus fin et plus performant. Nous projetons également d'augmenter la vitesse d'exécution de notre prototype en proposant un système de résumé de l'arborescence des documents. Nous travaillons également sur des coûts de suppression et de substitution plus fins afin d'améliorer la prise en compte de la sémantique des éléments. Nous gagnerions sans doute à remplacer la linéarité de notre combinaison des score de structure par une approche plus progressive. Enfin nous allons procéder à l'évaluation de notre solution sur la tache *Datacentric* d' *INEX* 2010, qui se trouve être sémantiquement riche sur les contraintes structurelles et donc parfaitement adaptée à l'expérimentation de notre modèle.

6. Bibliographie

Alilaouar A., Contribution à l'interrogation flexible de données semi-structurées, PhD thesis, Université Paul Sabatier, Toulouse, 2007.

Amer-Yahia S., Cho S., Srivastava D., « Tree Pattern Relaxation », *Proceedings of the 8th International Conference on Extending Database Technology : Advances in Database Technology*, EDBT '02, Springer-Verlag, London, UK, p. 496-513, 2002.

Aouicha M. B., Une approche algébrique pour la recherche d'information structurée, PhD thesis, Université Paul Sabatier, Toulouse, 2009.

Boobna U., de Rougemont M., « Correctors for XML Data », *XSym*, p. 97-111, 2004.

Chen W., « More efficient algorithm for ordered tree inclusion », *J. Algorithms*, vol. 26, p. 370-385, February, 1998.

Dalamagas T., Cheng T., jan Winkel K., Sellis T., « A methodology for clustering XML documents by structure », *Information Systems*, vol. 31, p. 187-228, 2006.

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

Damiani E., Oliboni B., Tanca L., « Fuzzy Techniques for XML Data Smushing », *Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications*, Springer-Verlag, London, UK, p. 637-652, 2001.

Damiani E., Tanca L., Arcelli F., « Fuzzy XML queries via context-based choice of aggregation », *Kybernetika*, vol. 36, p. 635-655, 2000.

Demaine E. D., Mozes S., Rossman B., Weimann O., « An optimal decomposition algorithm for tree edit distance », *ACM Trans. Algorithms*, vol. 6, p. 2 :1-2 :19, December, 2009.

Demartin G., Denoye L., al., « Report on INEX 2008 », *SIGIR Forum*, vol. 43, p. 17-36, June, 2009.

Dulucq S., Touzet H., « Analysis of tree edit distance algorithms », *Proceedings of the 14th annual symposium of combinatorial pattern matching*, p. 83-95, 2003.

Fuhr N., Kamps J., Lalmas M., Malik S., Trotman A., « Focused Access to XML Documents », Springer-Verlag, Berlin, Heidelberg, chapter Overview of the INEX 2007 Ad Hoc Track, p. 1-23, 2008.

Geva S., Kamps J., Lethonen M., Schenkel R., Thom J. A., Trotman A., « Overview of the INEX 2009 ad hoc track », *Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval*, INEX'09, Springer-Verlag, Berlin, Heidelberg, p. 4-25, 2010.

Hassler M., Bouchachia A., « Searching XML Documents - Preliminary Work », *INEX*, p. 119-133, 2005.

Jiang T., Wang L., Zhang K., « Alignment of Trees - An Alternative to Tree Edit », *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, CPM '94, Springer-Verlag, London, UK, p. 75-86, 1994.

Kazai G., Lalmas M., « INEX 2005 evaluation measures », *In Advances in XML Information Retrieval and Evaluation : Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005), volume 3977 of Lecture Notes in Computer Science*, Springer Verlag, p. 16-29, 2006.

Klein P. N., « Computing the Edit-Distance between Unrooted Ordered Trees », *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, Springer-Verlag, London, UK, p. 91-102, 1998.

Lalmas M., Kazai G., « Report on the ad-hoc track of the INEX 2005 workshop », *SIGIR Forum*, vol. 40, p. 49-57, June, 2006.

Le M. D., Sauvagnat K. P., « Utilisation de la distance d'édition pour l'appariement sémantique de documents XML », *Proceedings of GAOC, Workshop*, 2010.

Levenshtein V., « Binary Codes Capable of Correcting Deletions, Insertions and Reversals », *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

Mass Y., Mandelbrod M., « Component Ranking and Automatic Query Refinement for XML Retrieval », *INEX*, p. 73-84, 2004.

Popovici E., Menier G., Marteau P.-F., « SIRIUS : A Lightweight XML Indexing and Approximate Search System at INEX 2005 », *INEX*, p. 321-335, 2005.

Rougemont M., Vieilleribière A., « Approximate schemas, source-consistency and query answering », *J. Intell. Inf. Syst.*, vol. 31, p. 127-146, October, 2008.

Saito T. L., Morishita S., « Amoeba Join : Overcoming Structural Fluctuations in XML Data », *WebDB*, 2006.

Tai K.-C., « The Tree-to-Tree Correction Problem », *J. ACM*, vol. 26, p. 422-433, July, 1979.

Theobald M., Schenkel R., Weikum G., « Topx xxl at inex 2005 », *In Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX*, p. 201-214, 2005.

Trotman A., « Narrowed Extended XPath I (NEXI », *In Proceedings of the INEX 2004 Workshop*, Springer-Verlag GmbH, p. 16-40, 2004.

Zhang K., Shasha D., « Simple fast algorithms for the editing distance between trees and related problems », *SIAM J. Comput.*, vol. 18, p. 1245-1262, December, 1989.