

Chapitre 4

XML et recherche d'information

4.1. Introduction

La nature des collections de documents électroniques évolue. Elles intègrent de plus en plus des meta-informations et notamment des informations structurales : de simples documents texte « plat », on dispose aujourd'hui de documents structurés ou semi-structurés. Les informations structurales sont liées à l'utilisation de formats tels que SGML (*Standard Generalized Markup Language*) [GOL 90] ou encore XML (*eXtensible Markup Language*) [W3C 98], [BRA 01]. Ces derniers, conçus à l'origine pour faciliter l'échange et la standardisation des données, voient leur importance augmenter grâce à l'expansion d'Internet [MIG 03].

Du point de vue de la recherche d'information, l'accès à ce type de documents soulève de nouvelles problématiques liées à la co-existence de l'information structurale et de l'information de contenu. La prise en compte de la dimension structurale devrait permettre de mieux répondre aux différentes attentes des utilisateurs. Elle réactualise la réflexion sur le problème de la granularité de l'information à retourner.

Ce chapitre a pour objectif de présenter les différentes problématiques soulevées par la RI sur des collections d'informations semi-structurées, ainsi que les différentes solutions proposées dans la littérature.

Chapitre rédigé par Karen PINEL-SAUVAGNAT et Claude CHRISMENT.

4.1.1. *Historique des langages de balisage et des approches de recherche d'information liées*

En recherche d'information traditionnelle, les systèmes de recherche d'information, tant dans leur modèle de représentation des données que dans les résultats qu'ils renvoient, traitent les granules des collections (c'est-à-dire les documents) dans leur globalité. Cependant, un document possède souvent des contenus hétérogènes, et l'utilisateur doit alors aller chercher l'unité d'information pertinente à sa requête au milieu des autres thèmes abordés par le document. Une solution à ce problème est de dissocier l'unité d'information *logique* renvoyée à l'utilisateur de l'unité d'information physique de la collection.

La problématique de la granularité de l'information à renvoyer à l'utilisateur n'est pas récente. Des travaux ont cherché à proposer une démarche permettant de caractériser des granules d'information plus fins que les granules de la collection explorée : il s'agit des travaux de recherche par passage. Les premières approches de recherche de passages proposaient de renvoyer à l'utilisateur des parties de documents de taille fixe (limitées en nombre de caractères) [MOF 93].

L'avènement du marquage électronique, dans lequel il s'agit d'insérer dans un fichier électronique (que l'on peut considérer comme linéaire) des informations liées au texte lui-même mais n'en faisant pas directement partie, a renouvelé la problématique de la recherche de parties de documents. Les premiers marquages électroniques concernaient des commandes typographiques (passer en gras, en italique, changement de police de caractères, etc.) et ont abouti à des formats tels que RTF (*Rich Text Format*) ou MIF (*Maker Interchange Format*). Ces marquages sont faits à partir de *balises*, qui permettent d'indiquer des changements d'états.

Dans [LAM 95], les auteurs proposent de rechercher au sein des documents des structures connues *a priori* pour la réécriture de documents. Les approches de recherche par passage proposent quant à elles de renvoyer à l'utilisateur des passages de taille non fixe, en se basant sur la structure des documents [SAL 93], [HEA 97].

En 1981, Charles Goldfarb, Edward Mosher et Raymond Lorie inventent le premier langage de *balisage générique*, GML (*Generalized Markup Language*). GML doit permettre aux sous-systèmes d'édition, de formatage et de repérage de partager les mêmes documents et non pas de créer un fichier destiné à l'édition, un autre au repérage, etc. *Pour la première fois, un langage rend les contenus indépendants des outils manipulant ces contenus.* Le créateur de documents GML peut définir ses propres balises, selon ses besoins et les besoins de ses applications. De plus, GML introduit le concept de types de documents formellement définis comprenant une structure d'*éléments imbriqués*.

En 1986, au travers de la norme ISO 8879, GML évolue en SGML (*Standard Generalized Markup Language*) [GOL 90]. On voit alors apparaître des concepts importants comme la possibilité de prévoir l'ordre d'apparition d'éléments dans la structure d'un document. Les documents sont balisés conformément à une grammaire, la DTD (*Document Type Definition*) : ceci implique la notion de validité d'un document. De plus, la conception ou le choix d'une DTD permettent d'ajouter un balisage sémantique du fond du document. SGML, malgré ses nombreux avantages, souffre cependant de quelques inconvénients : la mise en oeuvre de documents respectant ce format est lourde et complexe, et la création de liens hypertextes est possible, mais reste aussi complexe.

En 1992, le W3C (World Wide Web Consortium) propose HTML (*HyperText Markup Language*), un langage de balisage pour le *Web*. HTML est en fait une DTD de SGML, qui ne cesse d'évoluer depuis sa création. HTML est un langage simple, possédant des balises standardisées et permettant la mise en forme d'un texte. C'est un standard reconnu par tous les navigateurs, et par conséquent très populaire sur le *web*. HTML mélange cependant le fond et la forme des documents, ce qui rend les mises à jours difficiles et qui surtout, mêle les données utiles (possédant une sémantique) et les données de mise en forme. De plus, il n'est pas un outil idéal pour l'échange de données.

En 1998, le W3C publie une recommandation officielle concernant le langage XML (*eXtensible Markup Language*) [W3C 98]. XML est un langage séparant le balisage d'auteur du balisage de présentation. XML est en passe de devenir le format le plus répandu : sa présence et son utilisation sur le *web* se font de plus en plus importantes, tant dans les domaines génériques que dans les noms de domaines dans lesquels il apparaît [MIG 03].

4.1.2. Documents semi-structurés et XML

La structure des documents est définie par des balises encadrant les fragments d'informations. Une *balise* (ou *tag* ou *label*) est une suite de caractères encadrés par « < » et « > », comme par exemple <nombalise>. Un *élément* est une unité syntaxique identifiée, délimitée par des balises de début et de fin , comme par exemple <mabalise> mon texte </mabalise>. Les éléments peuvent être imbriqués comme le montre le document exemple du tableau 4.1, mais ne doivent pas se recouvrir. Les *attributs* des éléments sont intégrés à la balise de début en utilisant la syntaxe *nomattribut*=valeur. Par exemple, <mabalise monattribut="mavaleur">texte </mabalise>.

La DTD (*Document Type Definition*) associée au document décrit la structure générique du document : elle contient l'ensemble des balises qu'il est possible d'inclure, ainsi que des relations de composition entre ces balises.

Contrairement à SGML, il n'est pas obligatoire d'associer une DTD à un document XML. Notons que l'on assiste aujourd'hui au développement d'une nouvelle forme de grammaire, qui permet de définir des éléments plus complexes et possède un typage des données plus riche, les *XML-schémas* [FAL 01].

Une classe de document possède donc une structure *générique* définie par la DTD (ou le schéma XML) alors qu'un document instance de cette classe possède une structure *spécifique*, exprimée par l'imbrication des éléments *via* leurs balises.

```

< ?xml version="1.0" ?>
<!--Exemple de fichier XML décrivant un article scientifique -->
<article annee="2003">
<en-tete>
  <titre>Recherche d'information sur le web : la grande révolution</titre>
  <auteur>André Dupont</auteur>
</en-tete>
<corps>
  <section>
    <sous-titre>Histoire de l'hypertexte : des pères fondateurs au World
    Wide Web</sous-titre>
    <par>Afin de maîtriser les enjeux des systèmes hypertexte,
    il convient, même si c'est une tâche ardue, de d'essayer de les définir...
    </par>
  </section>
  <section>
    <sous-titre>Moteurs de recherche</sous-titre>
    <par>On distingue plusieurs types de moteurs de recherche...</par>
    <par>Les annuaires...</par>
    <par>Les moteurs de recherche plein-texte...</par>
    <par>Les meta moteurs...</par>
  </section>
  <section>
    <sous-titre>L'analyse des liens</sous-titre>
    <par>...</par>
  </section>
</corps>
</article>

```

Tableau 4.1. Exemple de fichier XML *article.xml*

Les formats SGML et XML permettent de produire des documents *structurés* ou *semi-structurés*. Les documents *structurés* possèdent une structure régulière, ne contiennent pas d'éléments mixtes (c'est-à-dire d'éléments contenant du texte *et* d'autres éléments) et l'ordre des différents éléments qu'ils contiennent est généralement non significatif.

Les documents *semi-structurés* quant à eux sont des documents qui possèdent une structure flexible et des contenus hétérogènes. La modification, l'ajout ou la suppression d'une donnée entraîne une modification de la structure de l'ensemble.

Dans notre contexte, nous nous intéressons plus particulièrement à la recherche d'information dans des documents semi-structurés, les documents structurés servant plutôt à conserver des données au sens bases de données. Par abus de langage, on parlera cependant de *RI structurée*. Le format XML nous permettra d'illustrer nos propos.

Une galaxie de standards ou de recommandations a émergé conjointement à XML afin de définir des outils et des applications autour du langage. Parmi les plus connus et les plus susceptibles d'aider à la recherche d'information, citons les mécanismes de base pour adresser des éléments dans des documents XML (XPath), pour traiter les documents XML (DOM et SAX), pour présenter et transformer les contenus XML (XSL), les espaces de nom, XLink et XPointer pour la gestion des liens, RDF, etc. Nous nous proposons de détailler ici DOM et XPath, dont la compréhension est nécessaire dans le cadre de la RI structurée.

DOM (*Document Object Model*) permet d'effectuer une représentation en arbre des documents XML. DOM génère un arbre d'objets reliés entre eux, chaque objet représentant un atome du document XML. On trouvera un exemple d'arbre DOM sur la figure 4.1. Un tel arbre se compose d'une *racine* Document, de *nœuds internes* représentant les éléments ou les attributs, et de *nœuds feuilles* contenant les valeurs d'éléments ou d'attributs. Dans la suite du chapitre, nous représenterons les documents XML sous cette forme, et utiliserons indifféremment les termes *éléments* ou *nœuds* pour désigner des sous-arbres de documents XML.

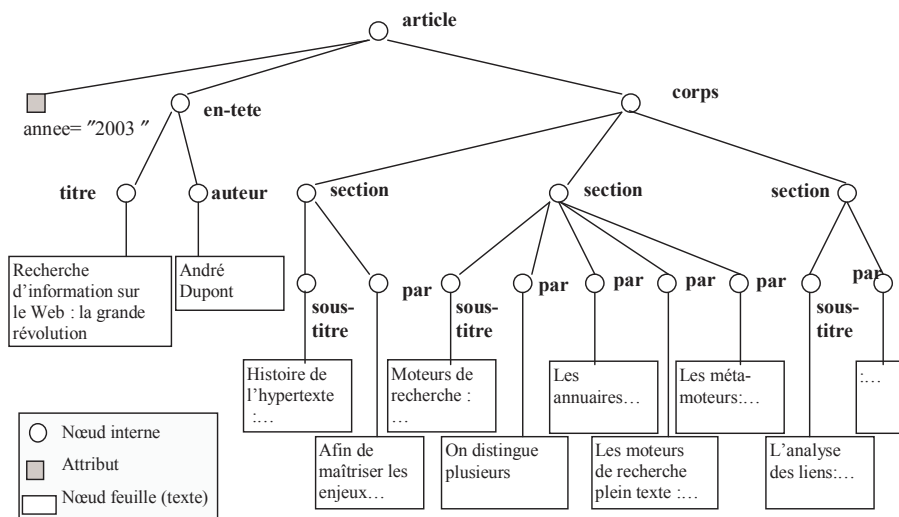


Figure 4.1. Exemple d'arbre DOM correspondant au document du tableau 4.1

Xpath est un langage de spécification d'expressions régulières décrivant un chemin ou une famille de chemins dans une arborescence XML. Xpath 1.0 est une recommandation du W3C [CLA 99]. Il s'agit d'un langage permettant de sélectionner des sous-arbres d'un document XML. Il possède une syntaxe simple et non ambiguë et implémente les types usuels (chaînes, nombres, booléens, variables, fonctions). Il permet aussi de manipuler des nœuds et des ensembles de nœuds.

XPath permet d'effectuer des expressions de chemins :

- recherche d'éléments : *sous-titre*
- parent-enfant : *section/sous-titre*
- ancêtre-descendant : *article//section*
- racine du document : */article/**
- filtre sur la structure : *//article[section]*
- filtre sur le contenu : */article[@annee="2002" and auteur="Tim Bray"]*

XPath permet aussi d'effectuer la navigation dans un document XML selon différents axes (enfants, descendants, parents, ancêtres, nœuds suivants, nœuds précédents, etc.).

4.1.3. Recherche d'information structurée : problèmes et enjeux

4.1.3.1. Granularité du résultat d'une recherche

Les documents semi-structurés réactualisent la problématique de la granularité de l'information à renvoyer, et permettent de traiter l'information avec une granularité modulable qui peut être identifiée en faisant intervenir le balisage [LAM 94]. L'information doit maintenant être traitée sans connaissance *a priori* de la structure des documents, et en tenant compte en plus des liens d'adjacence, de la présence de liens d'inclusions entre les balises.

Dans le cadre des documents semi-structurés, l'unité d'information pouvant être renvoyée à l'utilisateur correspond à un nœud de l'arbre du document, c'est-à-dire à un *sous-arbre*. La pertinence d'un nœud vis-à-vis d'une requête est évaluée selon les deux notions suivantes : l'*exhaustivité* et la *spécificité* [CHI 96], [LAL 97].

On dit qu'une unité d'information est exhaustive à une requête si elle contient toutes les informations requises par la requête et qu'elle est spécifique si tout son contenu concerne la requête.

Le principe de recherche dans les documents structurés pourrait donc être défini ainsi : *un système devrait toujours retrouver l'élément le plus exhaustif et spécifique répondant à une requête*. Dans des corpus de documents XML, chercher les nœuds les

plus exhaustifs et spécifiques pour une requête revient donc à trouver les sous-arbres de taille minimale pertinents à la requête.

De par leur structure, l'utilisateur interrogeant des corpus de documents XML peut formuler deux types de requêtes, selon sa connaissance du corpus :

- des requêtes portant sur le *contenu* seul des unités d'information : ces requêtes sont composées de simples mots-clés, et l'utilisateur laisse le SRI décider de la granularité de l'information à renvoyer ;
- des requêtes portant sur la *structure* et le *contenu* des unités d'information, dans lesquelles l'utilisateur spécifie des besoins précis sur certains éléments de structure. Dans ce type de requêtes, l'utilisateur peut utiliser les conditions de structure pour indiquer le type des éléments qu'il désire voir renvoyer, mais aussi plus simplement pour préciser son besoin.

Afin de permettre ces différentes recherches, les techniques de la recherche d'information traditionnelle doivent être adaptées ou de nouvelles méthodes doivent être proposées pour l'*indexation*, l'*interrogation* ou encore la *recherche* et le *tri* des unités d'information. Nous nous proposons de détailler ces différentes problématiques dans la section suivante.

4.1.3.2. *Les approches spécifiques*

Les méthodes pour l'indexation, l'interrogation, la recherche et le tri des documents XML peuvent être divisées en deux courants principaux [FUH 03a] :

- *l'approche orientée données* voit les documents XML comme des collections de données, typées et relativement homogènes. Elle utilise des techniques développées par la communauté des *bases de données* (BD) ;
- *l'approche orientée documents* se focalise sur des applications considérant les documents structurés d'une manière traditionnelle, c'est-à-dire que les balises servent uniquement à décrire la structure logique spécifique des documents. Cette approche a quant à elle été prise en charge par la communauté de la *recherche d'information*.

Alors que les deux communautés sont historiquement à l'origine de méthodes bien dissociées, la frontière entre les différentes approches pour la recherche dans des documents XML tend aujourd'hui à s'estomper.

En ce qui concerne l'*indexation*, la problématique réside essentiellement en l'extraction des clés de recherche, à savoir les termes les plus représentatifs des documents ainsi que l'information structurelle qu'ils contiennent. Dans le cas des documents textes « plats », le contenu textuel des documents est traité afin de trouver et de pondérer les termes les plus représentatifs des documents. Dans le cas des documents

semi-structurés, la dimension structurelle s'ajoute au contenu, et les questions suivantes se posent alors : que doit-on indexer de la structure des documents ? Comment relier cette structure au contenu même du document ? En fonction de quelle dimension (niveau éléments, documents, collection) doit-on pondérer les termes d'indexation ?

Les approches orientées BD confondent les notions d'indexation et de stockage. *Toute* l'information textuelle et structurelle des documents est ainsi stockée au sein de tables dans des bases de données. Ceci pose particulièrement problème pour les recherches sur le contenu textuel des documents, puisque ce dernier est indexé en tant que chaîne de caractères, et non sous forme de termes indépendants. Ces approches proposent néanmoins des schémas de stockage optimaux pour la structure des documents. Cette dernière peut être reflétée dans le schéma de la base de données, ou bien être stockée de manière générique dans des tables particulières.

Les approches orientées RI utilisent des techniques traditionnelles pour l'extraction des termes d'indexation, mais de nouvelles problématiques sont soulevées concernant la structure.

Considérons maintenant les *langages d'interrogation*, dont la grande majorité a été proposée par la communauté des bases de données. Ces langages d'interrogation doivent permettre à l'utilisateur d'exprimer des conditions sur le contenu et/ou la structure des documents.

La communauté BD a été historiquement la première à proposer des langages pour l'interrogation des documents XML. Ces langages, presque exclusivement basés sur des syntaxes proches de SQL, permettent à l'utilisateur d'exprimer des conditions très précises sur la structure des documents. L'expression de conditions sur les chemins est par exemple permise. Des prédicats de type *contains* sont aussi proposés pour effectuer des recherches sur le contenu textuel. Cependant, ces dernières conditions doivent toujours porter sur des conditions de structure bien définies, et l'utilisateur doit de plus spécifier le type d'élément qu'il désire voir retourné par le système, alors qu'il n'a pas forcément d'idée précise sur la question.

Les approches orientées RI cherchent quant à elles à simplifier ces langages en ce qui concerne les conditions de structure, tout en proposant de nouvelles fonctionnalités concernant la recherche sur le contenu (utilisation d'un prédicat *about* pour remplacer le prédicat *contains*, ou bien encore d'opérateurs booléens dans les conditions de contenu).

La dernière problématique concerne les *modèles de recherche* et de *tri des unités d'information*. La problématique traditionnelle liée à l'évaluation de la pertinence d'une information vis-à-vis d'une requête reste d'actualité, mais elle se complique et implique d'autres questions dans le cadre des documents XML, notamment en ce qui concerne la structure. Les requêtes orientées contenu, qui sont de loin les plus simples

pour l'utilisateur, imposent au SRI de décider la granularité appropriée de l'information à renvoyer.

Dans le cadre des requêtes orientées contenu et structure, deux cas sont possibles. Tout d'abord, l'utilisateur peut spécifier le type des éléments que le système doit renvoyer. D'autres notions de pertinence entrent alors en jeu. La dimension de spécificité n'a plus réellement de sens, puisque l'utilisateur précise la granularité de l'information qu'il désire. Cependant, le contenu des éléments de structure ainsi que les expressions de chemin présentes dans la requête doivent pouvoir être traitées de manière vague. En d'autres termes, la pertinence des informations structurelles doit pouvoir être évaluée, et l'arbre de la requête et l'arbre du document doivent pouvoir être comparés de façon non stricte. Le second cas concerne les requêtes pour lesquelles l'utilisateur exprime des conditions sur la structure des documents, mais sans préciser ce qu'il recherche exactement. Si le problème de l'évaluation de la pertinence des informations structurelles se pose de nouveau, vient s'y ajouter, comme dans les requêtes orientées contenu, celui de la granularité de l'information à renvoyer.

Les approches orientées BD évaluent de façon *exacte* des expressions du type *attribut = valeur*. Le traitement des requêtes est donc fait de manière booléenne et il n'est pas possible de renvoyer à l'utilisateur une liste de résultats triés en fonction de leur pertinence.

Les approches orientées RI cherchent, quant à elles, à évaluer le degré de similarité entre la requête et les unités d'informations et attribuent à ces dernières un score de *pertinence*. L'intérêt est double : tout d'abord sélectionner les unités d'informations qui répondent *au mieux* au besoin de l'utilisateur, et lui proposer ensuite une liste triée de résultats.

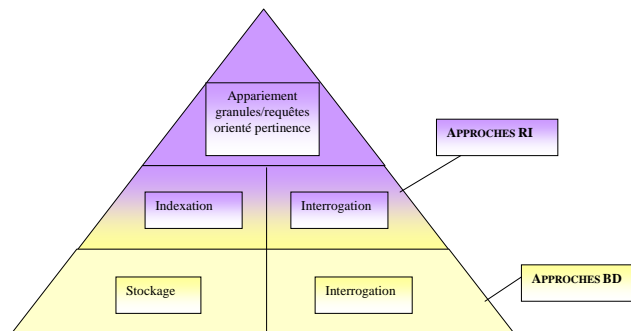


Figure 4.2. Domaines de compétence de la BD et de la RI

D'une manière générale et comme nous allons le voir dans la suite du chapitre, les solutions proposées par la communauté RI peuvent être utilisées comme « sur-couche » aux solutions orientées BD (figure 4.2). Cette sur-couche sert essentiellement à intégrer la notion de pertinence dans la recherche, en complétant les approches proposées par la communauté BD pour le stockage et l'interrogation des documents.

La notion de pertinence étant la plus importante à nos yeux, les problématiques détaillées dans la suite du document (à savoir l'indexation, l'interrogation et le traitement des requêtes) sont, dans le domaine du possible, abordées sous l'angle de la recherche d'information.

4.2. Techniques d'indexation des documents semi-structurés

Dans cette section, nous présentons les différentes approches proposées dans la littérature pour répondre à la problématique de l'indexation.

La façon la plus simple d'indexer des documents XML est bien sûr de les considérer comme des fichiers plats, et le processus d'indexation dans ce cas-là est similaire à celui utilisé en RI traditionnelle, c'est-à-dire qu'il consiste à sélectionner les termes importants du contenu textuel des documents. Cependant, aucune recherche sur la structure n'est plus possible, et les documents existent uniquement dans leur intégralité.

Un schéma d'indexation de documents XML devrait couvrir les aspects suivants :

- 1) permettre la reconstruction du document XML décomposé dans les structures de stockage ;
- 2) permettre le traitement des expressions de chemin sur la structure XML ;
- 3) accélérer la navigation dans des documents XML ;
- 4) autoriser le traitement de prédicats vagues et précis sur le contenu de documents XML ;
- 5) permettre la recherche par mots-clés.

Par conséquent, la plupart des schémas d'indexation proposés dans la littérature redéfinissent la granularité du stockage et utilisent la structure des documents XML.

Même si l'indexation des informations de contenu et des informations structurelles sont étroitement liées, nous nous proposons de les décrire séparément, afin de mieux comprendre les différents enjeux soulevés par l'une et l'autre.

4.2.1. Indexation de l'information textuelle

Le processus d'indexation de la recherche d'information traditionnelle consiste à extraire les termes importants des documents. Cette problématique reste bien entendue d'actualité dans le cadre des documents structurés.

Pour les approches orientées BD, l'unité textuelle d'indexation est le texte complet des nœuds feuilles. Pour les approches orientées RI, il s'agit au contraire du terme, qui sera de plus pondéré afin de refléter son importance.

Quelle que soit l'unité textuelle d'information choisie, le problème de la portée des termes d'indexation se pose, et nous nous proposons de le détailler dans la section suivante.

4.2.1.1. Portée des termes d'indexation

Le problème de la portée des termes d'indexation est le suivant : comment rattacher les termes à l'information structurelle ? Doit-on chercher à agréger le contenu des nœuds ou au contraire à indexer tous les contenus des nœuds séparément ? Ces deux solutions correspondent aux approches d'indexation dites *des sous-arbres imbriqués* et des *unités disjointes*.

4.2.1.1.1. Sous-arbres imbriqués

Les approches de ce premier groupe considèrent que le texte complet de chaque nœud de l'index est un document atomique [ABO 04], [SIG 03], [KAM 04] et propagent donc les termes des nœuds feuilles dans l'arbre des documents. En d'autres termes, ces approches *indexent tous les sous-arbres* (jugés potentiellement pertinents) des documents. Comme les documents XML possèdent une structure hiérarchique, les nœuds de l'index sont imbriqués les uns dans les autres et l'index contient de nombreuses informations redondantes. On trouvera une illustration de l'indexation de sous-arbres imbriqués sur la figure 4.3.

Les termes « *andré dupont* » sont par exemple reliés aux nœuds */article*, */article/en-tête*, et */article/en-tête/auteur*.

4.2.1.1.2. Unités disjointes

Dans ces approches, le document XML est décomposé en unités disjointes, de telle façon que le texte de chaque nœud de l'index est l'union d'une ou plus de ces parties disjointes [OGI 03], [FUH 03a], [GOV 02], [KAZ 02], [ROE 02], [ANH 02]. Les termes des nœuds feuilles sont uniquement reliés au nœud parent qui les contient. Si on reprend en exemple l'arbre de la figure 4.3, les termes « *recherche d'information enjeux* » seront uniquement reliés au nœud */article/en-tête/titre*, les termes « *andré dupont* » au nœud */article/en-tête/auteur* et les termes « *la recherche d'information* »

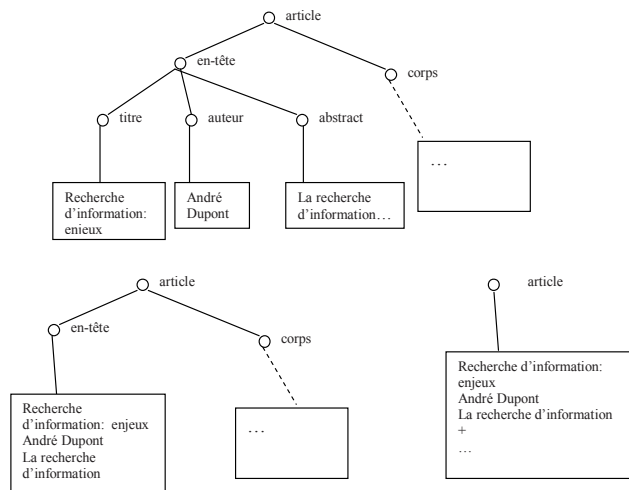


Figure 4.3. Indexation de sous-arbres imbriqués

au nœud `/article/en-tête/abstract`. Le nœud `/article/en-tête` n'est quant à lui relié à aucun terme.

L'approche utilisée pour indexer le contenu des documents semi-structurés implique l'utilisation de méthodes différentes pour la recherche dans les documents. Nous reviendrons sur ces différentes méthodes dans la section 4.4.

4.2.1.2. Pondération des termes d'indexation

Les approches orientées BD se contentent de stocker le texte des documents comme un tout, c'est-à-dire sous forme de chaînes de caractères. Ce type d'approche pour l'information textuelle montre peu d'intérêt dans le cadre de la RI, puisque la pondération des termes n'est pas permise, et que par conséquent, seules des mesures de pertinence très simples pourront être calculées par le système (comme le nombre de termes communs entre la requête et l'élément).

Les approches orientées RI extraient les termes d'indexation selon des processus similaires à ceux utilisés en RI traditionnelle. La pondération de ces termes doit cependant être vue sous un nouvel angle. Alors qu'en RI traditionnelle, le poids d'un terme cherche à rendre compte de son importance de manière locale au sein du document et de manière globale au sein de la collection, s'ajoute en RI structurée l'importance du terme au niveau de l'élément qui le contient.

Les occurrences des termes ne suivent plus forcément une loi de Zipf [ZIP 49], [GRA 03]. Le nombre de répétitions des termes peut être (très) réduit dans les documents XML et l'utilisation d'*idf* (*Inverse Document Frequency*) n'est pas forcément appropriée.

L'utilisation d'*ief* (*Inverse Element Frequency*) a été proposée par de nombreux auteurs [WOL 00], [GRA 02], [PIN 06]. On trouvera des exemples d'adaptation des formules de pondération traditionnellement utilisées en RI à la RI structurée dans [TRO 05].

Dans [ZAR 04], le calcul du poids des termes est influencé par le contexte (l'unité d'indexation) dans lequel ils apparaissent. Ce calcul de poids s'inspire de la méthode *tf-idf* qu'on applique aux balises. Ainsi, les auteurs définissent le *tf-idf* (*Term Frequency - Inverse Tag and Document Frequency*), qui permet de calculer la force discriminatoire d'un terme *t* pour une balise *b* relative à un document *d*.

D'autres paramètres permettant d'évaluer l'importance des termes peuvent être pris en compte : la fréquence du terme au sein de l'élément bien sûr, mais aussi la fréquence du terme au sein du document, ou encore la longueur de l'élément et la longueur moyenne des éléments de la collection [PIN 06].

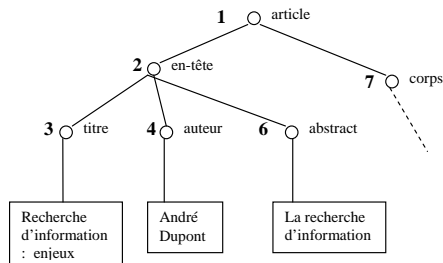
4.2.2. *Indexation de l'information structurelle*

La structure peut être indexée selon des granularités variées [LUK 02], c'est-à-dire que toute l'information structurelle n'est pas forcément utilisée dans le processus d'indexation. Parmi les approches proposées dans la littérature, on distingue trois types d'approches pour l'indexation de l'information structurelle : l'indexation basée sur des champs, l'indexation basée sur des chemins, et enfin l'indexation basée sur des arbres. Nous nous proposons de les détailler ici, par ordre croissant de quantité d'information stockée.

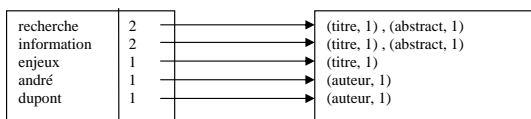
Les schémas d'indexation de la structure que nous présentons dans la suite sont indépendants de l'*unité textuelle* d'indexation (*terme* ou bien *texte entier des feuilles*) choisie. En d'autres termes, les exemples que nous utilisons pour étayer nos propos peuvent être utilisés indifféremment pour traiter l'information textuelle selon des approches orientées RI ou bien orientées BD.

4.2.2.1. *Indexation basée sur des champs*

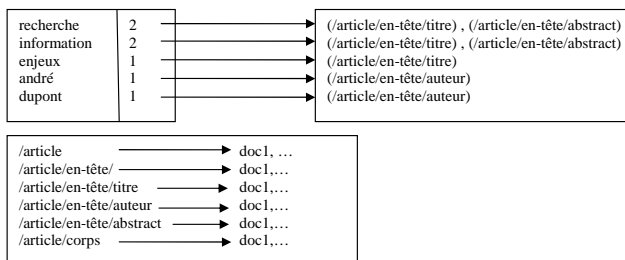
Il s'agit certainement de la méthode d'indexation semi-structurée prenant en compte la structure la plus simple. Un document est représenté comme un ensemble de champs (par exemple *titre*, *auteur*, *abstract*, etc.) et de contenu associé à ces champs. Pour permettre une recherche restreinte à certains champs, les termes de l'index sont construits en combinant le nom du champ avec les termes du contenu, comme l'illustre le point a- de la figure 4.4.



a - Exemple d'indexation basée sur des champs



b - Exemple d'indexation basée sur des chemins



c - Exemple d'indexation basée sur des arbres

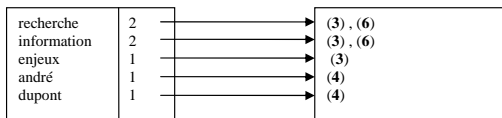


Figure 4.4. Exemples d'indexation de l'information structurelle

4.2.2.2. Indexation basée sur des chemins

Les techniques d'indexation basées sur des chemins ont pour but de retrouver rapidement des documents ayant des valeurs connues pour certains éléments ou attributs. Il s'agit aussi de faciliter la navigation de façon à résoudre efficacement des expressions Xpath et utiliser des index plein texte sur les contenus. En conséquence, les solutions proposées utilisent des index de chemins, c'est-à-dire des index donnant pour chaque valeur répertoriée d'un chemin de balises (de type Xpath) la liste des documents répondants contenant un élément atteignable par ce chemin et ayant cette valeur. On trouvera une illustration de l'indexation basée sur des chemins sur le point b de la figure 4.4.

Parmi les approches utilisant une indexation basée sur des chemins, on peut citer Natix [KAN 00] ou bien encore InfonyteDB [HUC 00]. Dans ces approches cependant, il devient difficile de retrouver les relations ancêtres-descendants entre les différents nœuds des documents. Les approches d'indexation basées sur des arbres le permettent quant à elles.

4.2.2.3. Indexation basée sur des arbres

Le point c- de la figure 4.4 donne un exemple d'indexation basée sur des arbres. Les nœuds de l'arbre sont numérotés dans les index de façon à pouvoir reconstruire la structure arborescente des documents.

De nombreux modèles de numérotation ont été proposés dans la littérature. On peut par exemple citer [LEE 96] qui propose une structure d'index ANOR (*inverted index for All NOdes without Replication*), [JAN 99] avec une architecture BUS (*Bottom Up Schema*), ou encore [FLO 99] avec les approches EDGE et BINARY.

La structure d'index du Xpath Accelerator [GRU 02] a été conçue pour l'évaluation des expressions de chemin. L'intuition guidant le Xpath Accelerator est la suivante : en chargeant un nouveau document XML, le Xpath Accelerator exécute une traversée de la représentation en arbre du document. Durant ce parcours, des valeurs croissantes de pré-ordre ou post-ordre sont assignées aux nœuds visités, comme le montre la figure 4.5.

En stockant de plus la dimension de prédécesseur du nœud parent, un champ indiquant la présence d'attributs et le nom de balise de chaque nœud, une navigation efficace devient possible. Xpath Accelerator est particulièrement intéressant pour une navigation dans des documents XML et pour le traitement d'expressions Xpath. Contrairement à d'autres approches basées sur des index de structure, Xpath Accelerator permet de répondre à des expressions Xpath qui n'ont pas pour origine la racine du document.

XISS (XML Indexing and Storage System) [LI 01] a pour but de créer un index efficace pour la recherche de Xpath. Contrairement à l'approche Xpath Accelerator, XISS ne permet de traiter efficacement que des relations ancêtre-descendant. Cependant, les insertions sont facilitées.

De nouvelles approches cherchent à combiner l'*approche orientée données* à l'*approche orientée documents*, pour profiter au mieux de toutes les caractéristiques des documents XML [GRA 03], [TRO 04a], [WEI 04], [SAU 05a]. Elles permettent notamment d'indexer le contenu textuel des documents et de pondérer les termes, ce qui rend ensuite possible un calcul de pertinence des éléments.

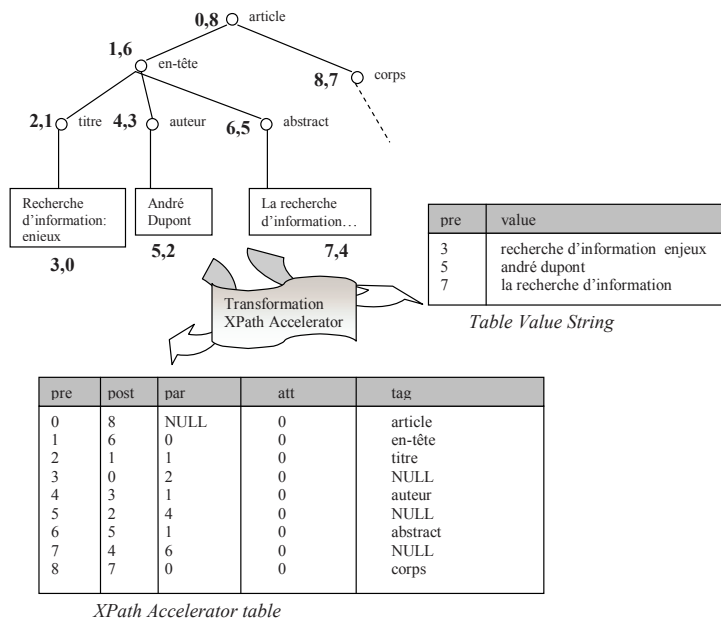


Figure 4.5. Transformation d'un document XML avec l'approche XPath Accelerator

4.3. Langages de requêtes

L'interrogation des corpus de documents XML diffère de l'interrogation habituelle en RI, et ce du fait de l'information structurale contenue dans les documents. D'un point de vue utilisateur, il existe deux façons principales d'interroger les collections de documents XML :

- 1) il peut, s'il n'a pas d'idée précise de ce qu'il recherche, formuler des requêtes comparables à celles utilisées dans les moteurs de recherche traditionnels, c'est-à-dire des requêtes composées de simples mots-clés. On parle de *requêtes orientées contenu* ;
- 2) il peut ajouter des conditions sur l'information structurale des documents, et préciser ainsi son besoin. Ceci présuppose cependant qu'il a une connaissance au moins partielle de la collection qu'il interroge. On parle alors de *requêtes orientées contenu et structure*.

De nombreux langages de requêtes ont été proposés dans la littérature, et nous nous proposons d'en détailler quelques-uns dans cette section. Ces langages, issus de la communauté des bases de données, se concentrent principalement sur l'introduction de la dimension structurale dans les requêtes, et traitent souvent le contenu des éléments de façon booléenne (présent/absent).

D'après [GAR 02], un langage de requêtes XML doit intégrer les fonctionnalités des langages de requêtes pour les systèmes documentaires et pour les bases de données. Les différents langages de requêtes doivent donc supporter les fonctions suivantes :

- sélection des arbres sur critères multiples,
- possibilité d'effectuer toutes les opérations des types de base,
- quantification universelle et existentielle des variables,
- combinaison des données depuis des documents,
- tri des résultats,
- imbrication de requêtes,
- possibilité d'utilisation des agrégats et fonctions associées.

De plus, l'intégration de fonctions des systèmes documentaires nécessite la prise en compte de requêtes par liste de mots-clés du type : *CONTAINS* (<élément>, collection de mots-clés).

Au-delà des requêtes exactes, il serait aussi souhaitable de supporter des requêtes approchées sur mots-clés du type : *SIMILAR* (<élément>, collection de mots clés).

Enfin, de nouvelles fonctionnalités traitant les structures sont nécessaires :

- respect de la hiérarchie et des séquences,
- agrégation de données depuis des documents,
- préservation de structures,
- construction de structures nouvelles.

4.3.1. Langages de requêtes : premières approches

Parmi les premiers langages spécifiés pour l'interrogation des documents semi-structurés, on peut citer UnQL [BUN 96], LOREL [ABI 97], StruQL [FER 97], XML-QL [LEV 98] ou encore XQL [ROB 98]. XQL a été proposé par Microsoft pour interroger des collections de documents XML. Au lieu d'utiliser des canevas XML, XQL propose d'étendre les URL pour interroger des collections de documents XML avec des expressions Xpath. Le langage QUILT [CHA 00] a été développé dans le but d'être un langage flexible, combinant des caractéristiques pour interroger des documents et des bases de données. XML-GL [CER 99] est un langage de requêtes graphique s'utilisant sur des graphes XML. Il présente le principal avantage d'être très ergonomique.

Xquery [FER 03] est le langage de requêtes pour XML proposé par le W3C. Xquery tire très fortement ses constructions et caractéristiques de Quilt, lui-même dérivé de Xpath, XQL, XML-QL, et LOREL.

XQuery peut être perçu comme un surensemble de SQL. Les fonctionnalités de SQL sur les tables (collections de tuples) sont étendues pour supporter des opérations similaires sur les forêts (collections d'arbres). Ces extensions ont conduit à intégrer les fonctions suivantes : projection d'arbres sur des sous-arbres, sélection d'arbres et de sous-arbres en utilisant des prédicats sur les valeurs des feuilles, utilisation de variables dans les requêtes pour mémoriser un arbre ou pour itérer sur des collections d'arbres, combinaison des arbres extraits de collection en utilisant des jointures d'arbres, réordonnement des arbres, imbrication des requêtes, calculs d'agrégats, et utilisation possible de fonctions utilisateur.

XQuery supporte des fonctions orientées RI : en particulier, un prédicat *contains* est intégré pour la recherche par mots-clés. Pour faciliter la recherche dans des structures mal connues, XQuery permet enfin d'exprimer des chemins indéterminés ou partiellement connus, tout comme XPath. On trouvera dans le tableau 4.2 un exemple de requête XQuery.

```

For $R in collection (" Guide ")/Restaurant,
  $H in collection (" Répertoire)/Hôtel
where $H/Rue=$R and $H/Nom= "Le Bon Repos "
return
  <RestauTel>
    <Nom> $R/Nom/text() </Nom>
    <Tel> $R/Téléphone/text() </Tel>
  </RestauTel>

```

Tableau 4.2. Exemple de requête XQuery : lister le nom des restaurants avec leur numéro de téléphone dans la rue de l'hôtel Le Bon Repos

4.3.2. Evolution des langages de requêtes : prise en compte de la composante recherche d'information

Les langages que nous avons présentés jusqu'ici proposent un prédicat de type *contains* pour exprimer des conditions sur le contenu des éléments. Cependant, ce type de prédicat ne permet pas de mesurer la similarité entre les unités d'informations et les conditions de contenu. Les approches que nous décrivons ci-dessous sont issues de la communauté de la RI et cherchent à intégrer ce dernier aspect.

Les langages de requêtes que nous avons décrits précédemment ne peuvent pas traiter des requêtes du genre « trouver les livres et les CD avec des titres similaires ». Le langage ELIXIR (*an Expressive and Efficient Language for XML Information Retrieval*) [CHI 02] étend le langage XML-QL avec un opérateur de similarité textuelle.

En partant d'un constat sur les lacunes du langage XQL (pas de pondération des résultats, pas de prédicats vagues pour mesurer la similarité et pas de correspondance

sémantique entre les différents *tags* XML), Grossjohann [GRO 00] propose le langage XIRQL. XIRQL est une extension du langage XQL et possède des opérateurs permettant l'utilisation de prédicats vagues et l'abstraction des types de données et des balises, et permet une recherche orientée sur la pertinence (c'est-à-dire avec une pondération des résultats). Le langage possède cependant une syntaxe complexe, difficilement utilisable sans une interface appropriée. XIRQL a été implémenté dans le système HyreX et testé dans la campagne d'évaluation INEX 2002 [GOV 02].

Cohen *et al.* [COH 02] proposent le langage EquiX, permettant de combiner la recherche de motifs, la quantification et les expressions logiques pour interroger à la fois les données et les métadonnées des documents XML. Les requêtes peuvent être formulées avec une syntaxe abstraite basée sur des graphes ou une syntaxe formelle concrète. L'algorithme d'évaluation a un coût polynomial et la DTD décrivant les documents résultats est dérivée automatiquement de la requête.

Dans leur système XISS, Li et Moon [LI 01] proposent un modèle pour répondre à des requêtes demandant de retrouver des éléments ou attributs communs dans des documents XML ne suivant pas la même DTD.

Dans [COL 02], le langage Tequyla-TX est présenté. Tequyla-TX est un langage de requête typé permettant d'effectuer aussi bien des requêtes orientées bases de données que des requêtes basées sur des mots-clés. Afin de répondre aux besoins fondamentaux des applications de recherche de texte, il autorise la recherche basée sur des mots ou des caractères, ce qui est particulièrement utile pour des applications littéraires (comme l'analyse de l'utilisation des prépositions dans les textes latins).

Le langage XML Fragment [CAR 03] se propose d'interroger les documents XML sous forme XML, ce qui permet à l'utilisateur d'exprimer des besoins imprécis.

Le langage NEXI a été défini dans [TRO 04b, TRO 04c] pour répondre aux besoins de la campagne d'évaluation INEX. Les requêtes étaient en effet précédemment exprimées en XML (pour 2002) ou XPath (pour 2003), mais dans le premier cas, le langage n'était pas assez puissant, et il était trop complexe dans le second cas (63% des requêtes exprimées par les participants (experts en RI) contenaient des erreurs de syntaxe !). NEXI a alors été conçu comme un sous-ensemble extensible d'XPath interprétable de manière vague (il s'agit d'un langage de requête orienté RI et non base de données). NEXI est amené à évoluer au fil des années pour s'adapter aux différentes tâches proposées aux participants d'INEX (notamment la tâche hétérogène ou la tâche en langage naturel).

On notera enfin que le W3C a récemment proposé un Working Draft [W3C 03], qui a pour but d'étendre les caractéristiques de recherche de XQuery à la recherche plein texte. Le langage TexQuery [AME 04] en est une application.

4.3.3. Conclusion sur les langages de requêtes XML

Comme on peut le voir, de très nombreux langages de requêtes ont été proposés dans la littérature. La plupart sont très puissants, mais leur syntaxe, souvent dérivée de SQL, est difficilement accessible pour les novices. Des interfaces adaptées peuvent être associées, mais leur feraient perdre de leur puissance.

La plupart des langages que nous venons de présenter sont basés sur une approche orientée base de données. Les conditions sur le contenu textuel des documents sont énoncées à l'aide de prédicats de type *contains*, qui ne permettent pas l'évaluation de la similarité entre la requête et les unités d'informations.

Des langages orientés RI font peu à peu leur apparition, et les propositions rencontrés dans la littérature proposent d'ajouter des fonctionnalités concernant la recherche sur le contenu (utilisation d'un prédicat *about* pour remplacer le prédicat *contains*, ou bien encore d'opérateurs booléens dans les conditions de contenu).

Quelle que soit l'approche utilisée (orientée *données* ou orientée *documents*), une connaissance parfaite de la structure des documents est aussi souvent nécessaire à l'utilisateur pour pouvoir formuler des requêtes. Il doit de plus spécifier l'élément qu'il désire voir retourné par le SRI, alors qu'il n'a pas forcément d'idée précise de ce qu'il recherche exactement. L'utilisation de ces syntaxes certes complètes mais aussi complexes nécessite l'utilisation d'interfaces pour les utilisateurs finaux, ce qui engendre l'apparition de langages de transition [GEV 06]. Ceci nous amène à nous interroger sur l'utilité de tels langages : dans [TAN 05], les auteurs montrent en effet que l'utilisation du langage naturel pour exprimer les requêtes peut donner des résultats comparables à ceux obtenus lorsque les requêtes sont exprimées en suivant la grammaire du langage NEXI.

Enfin, nombreuses sont les spécifications de langages, mais rares sont les implémentations concrètes...

4.4. Appariement requêtes/granules documentaires

Les modèles de recherche que nous présentons dans cette partie sont spécifiques aux approches orientées RI, puisqu'ils cherchent à attribuer des scores de pertinences aux nœuds des documents XML. Les approches orientées BD ne se posent pas ce problème, le contenu des documents étant traité de façon booléenne (présent/absent).

Dans les approches présentées dans la littérature, les modèles de RI classiques ont été adaptés pour tenir compte de l'information structurelle contenue dans les documents XML et des tailles variées des éléments (c'est-à-dire des granularités variées de l'information). Ces modèles cherchent à répondre à des requêtes orientées contenu

et /ou à des requêtes orientées contenu et structure. Dans le premier cas, les systèmes doivent décider de la granularité idéale de l'information à renvoyer à l'utilisateur, alors que dans le second, les conditions de structure des requêtes donnent des indications sur le type d'élément à renvoyer.

Dans cette section, nous nous proposons de détailler les différentes méthodes proposées en les classant selon deux approches principales :

- les approches propageant les termes dans l'arbre des documents : pour ce faire, ces approches indexent des sous-arbres imbriqués (paragraphe 4.2.1.1) ;
- et les approches propageant les scores des éléments dans l'arbre des documents : ces approches quant à elles indexent des unités disjointes.

Nous nous attardons ensuite sur les méthodes proposant une approche spécifique pour le traitement des conditions de structure.

4.4.1. Approches par propagation des termes des documents

4.4.1.1. Modèle vectoriel étendu

Dans les approches issues du modèle vectoriel, une mesure de similarité de *chaque* élément à la requête est calculée, et ce, à l'aide de mesures de distance dans un espace vectoriel. Les éléments sont représentés par des vecteurs de termes pondérés. Les éléments sont renvoyés à l'utilisateur par ordre décroissant de pertinence.

Schlieder et Meuss [SCH 02] intègrent la structure des documents dans la mesure de similarité du modèle vectoriel. Leur modèle de requête est basé sur l'inclusion d'arbres : cela permet de formuler des requêtes sans connaître la structure exacte des données.

Les auteurs proposent la notion de *terme structurel*, définie comme un arbre étiqueté. *book[author]*, *book[Bradley, title[XML]]*, *author[Bradley]*, etc., sont des exemples de termes structurels.

Les notions de *tf* et *idf* sont adaptées au processus de recherche dans des documents structurés. Soit E un élément de type t . Le poids $w_{T,E}^t$ d'un terme structurel T dans E est défini par :

$$w_{T,E}^t = tf_{T,E} \cdot idf_T^t = \frac{freq_T(E)}{maxfreq(E)} \cdot (\log\left(\frac{|E^t|}{n_T}\right) + 1) \quad [4.1]$$

avec $freq_T(E)$ le nombre d'occurrences de T dans E , $maxfreq(E)$ le nombre maximal d'éléments de la collection possédant la même étiquette que E , $|E^t|$ le nombre d'éléments de type t et n_T le nombre d'éléments contenant T .

Les auteurs combinent ainsi le modèle vectoriel et le *tree matching* afin de répondre à des requêtes orientées contenu et structure. Dans le modèle proposé, seuls les éléments (c'est-à-dire les sous-arbres) qui ont une structure qui peut être réduite à celle de la requête (c'est-à-dire qu'en supprimant certains éléments du sous-arbre, on peut arriver à la requête) ont un score de pertinence non nul.

Le modèle JuruXML [MAS 02, MAS 03] propose d'indexer les éléments selon leur type (un index par type d'élément) et d'appliquer ensuite le modèle vectoriel pour la pondération des éléments.

Les requêtes orientées contenu sont évaluées sur chacun des index et les résultats, qui ont été normalisés, sont ensuite fusionnés afin de fournir à l'utilisateur une liste unique de résultats.

Une requête structurée est quant à elle évaluée en trois phases. Tout d'abord, la requête originale est décomposée en ensemble de conditions de la forme (*chemin, terme*). Ensuite, une correspondance vague entre les chemins est calculée. Soit c_i^q la condition de chemin pour le terme t_i et c_i^e le XPath du terme t_i dans l'élément e . La fonction de similarité entre les deux chemins est exprimée selon l'équation [4.2] :

$$cr(c_i^q, c_i^e) = \begin{cases} \frac{1+|c_i^q|}{1+|c_i^e|} \text{ si } c_i^q \text{ est une sous - sequence de } c_i^e \\ 0 \text{ sinon} \end{cases} \quad [4.2]$$

Par exemple, $cr(\text{article/bibl}, \text{article/bm/bib/bibl/bb}) = 3/6 = 0.5$. On a enfin :

$$RSV(e, q) = \frac{\sum_{(t, c_i^q) \in q} \sum_{(t, c_i^e) \in e} w_q(t) * w_e(t) * cr(c_i^q, c_i^e)}{|q| * |e|} \quad [4.3]$$

où $w_q(t)$ et $w_e(t)$ sont les poids du terme t dans q et e , et $|q|$ et $|e|$ sont les nombres de termes dans q et e .

Cette dernière approche, évaluée dans le cadre de la campagne INEX 2004 permet d'obtenir de bons résultats par rapport à l'ensemble des participants.

On trouvera une autre approche implémentant le modèle vectoriel dans [HAS 05].

4.4.1.2. *Modèles probabilistes*

Dans [KAM 04], les auteurs proposent une approche basée sur les *modèles de langage* pour traiter les requêtes orientées contenu. Les auteurs considèrent que comme n'importe quel élément XML peut potentiellement être renvoyé à l'utilisateur, chaque élément doit être traité comme une unité d'indexation à part entière. Par conséquent, pour chaque élément, le texte qu'il contient ainsi que le texte contenu dans ses descendants est indexé. Un modèle de langage est ensuite estimé pour chaque élément de la collection. Pour une requête donnée, les éléments sont triés par rapport à la probabilité que le modèle de langage de l'élément génère la requête. Ceci revient à estimer la probabilité $P(e,q)$, où e est un élément et q une requête :

$$P(e, q) = P(e).P(q|e) \quad [4.4]$$

Deux probabilités doivent donc être estimées : la probabilité *a priori* de l'élément $P(e)$ et la probabilité qu'il génère la requête $P(q|e)$. Pour la seconde probabilité, les auteurs considèrent que les termes de la requête sont indépendants, et utilisent une interpolation linéaire du modèle d'élément et du modèle de collection pour estimer la probabilité d'un terme de la requête. La probabilité d'une requête t_1, t_2, \dots, t_n est ainsi calculée de la façon suivante :

$$P(t_1, \dots, t_n | e) = \prod_{i=1}^n (\lambda.P(t_i|e) + (1 - \lambda).P(t_i)) \quad [4.5]$$

où $P(t_i)$ est la probabilité d'observer le terme t_i dans l'élément e , $P(t_i)$ est la probabilité d'observer le terme dans la collection et λ est un paramètre de lissage.

Le calcul des probabilités peut être réduit à la formule de calcul des scores ci-dessous, pour un élément e et une requête t_1, \dots, t_n :

$$s(e, t_1, t_2, \dots, t_n) = \beta \cdot \log\left(\sum_t tf(t, e)\right) + \sum_{i=1}^n \log\left(1 + \frac{\lambda \cdot tf(t_i, e) \cdot (\sum_t df(t))}{(1 - \lambda)df(t_i) \cdot (\sum_t tf(t, e))}\right) \quad [4.6]$$

où $tf(t, e)$ est la fréquence du terme t dans l'élément e , $df(t)$ est le nombre d'éléments contenant t , λ est le poids donné au modèle de langage de l'élément en lissant avec le modèle de la collection, et β est un paramètre servant à combler le fossé entre la taille de l'élément moyen et la taille de l'élément pertinent.

Dans [WOL 00], l'utilisation de la fréquence inverse d'élément *ief* est proposée pour faciliter les pondérations par élément : un nouveau poids probabiliste pour les termes est alors formulé, utilisant *ief* et la fréquence du terme dans chaque élément. Les poids des termes de la requête peuvent être étendus avec des conditions sur l'appartenance du terme à un certain élément ou chemin.

On trouvera d'autres approches basées sur les modèles de langages dans [ABO 04], et [SIG 03].

Dans [PIW 02, PIW 03a], on trouve un exemple d'utilisation des *réseaux bayésiens* à la recherche d'information structurée. La structure de réseau bayésien utilisée reflète directement la hiérarchie des documents, c'est-à-dire que les auteurs considèrent que chaque élément de la hiérarchie possède une variable aléatoire associée. La variable aléatoire associée à un élément structurel peut prendre trois valeurs différentes dans l'ensemble $V = \{N, G, E\}$, avec N indiquant que l'élément n'est pas pertinent, G que l'élément est peu spécifique et E que l'élément possède une forte spécificité.

Pour chaque élément e et pour une requête donnée q , la probabilité $P(e = E|q)$ donne le score de pertinence *final* de l'élément, qui permet ensuite de classer les éléments selon leur degré de pertinence.

Deux autres types de variables aléatoires sont considérés. Le premier est la requête, qui est représentée par un vecteur de fréquences de termes. Le second est associé aux modèles de pertinence utilisés pour évaluer la similarité locale de l'élément à la requête et peut prendre deux valeurs : *pertinent* ou *non pertinent*.

Pour une requête donnée, un score local de pertinence est calculé pour chaque élément. Ce score dépend uniquement de la requête et du contenu de l'élément. Pour calculer ce score local, plusieurs modèles peuvent être utilisés. La fréquence des termes de la requête dans la requête, dans l'élément, dans le parent de l'élément et la longueur de l'élément peuvent par exemple être utilisés comme paramètres par les modèles.

La probabilité qu'un élément soit dans l'état N , G ou E dépend ensuite de l'état de l'élément parent, et du jugement par le(s) modèle(s) de pondération utilisé(s) que l'élément est pertinent ou non pertinent, comme le montre la figure 4.6.

On a alors (si on considère deux modèles de base M_1 et M_2 pour le calcul du score local de l'élément) :

$$P(e = v|q) = \sum_{v_p \in V, r_1, r_2 \in \{R, \neg R\}} \theta \cdot P(e \text{ parent} = v_p) \cdot P(M_1 = r_1|q) \cdot P(M_2 = r_2|q) \quad [4.7]$$

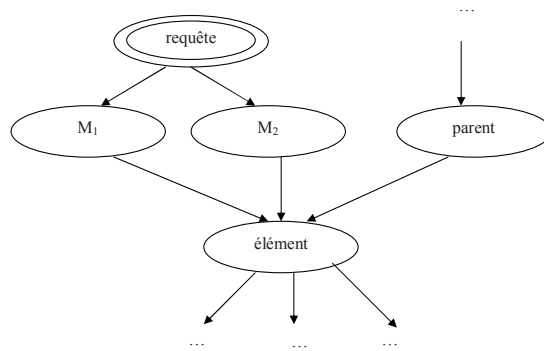


Figure 4.6. *Modèle de réseau bayésien. L'état de l'élément dépend de l'état du parent et de la pertinence de l'élément pour les modèles M_1 et M_2*

où $v \in V$, q est une requête composée de simples termes, et θ est un paramètre obtenu par apprentissage. Il dépend des différents états des quatre variables aléatoires (état de l'élément, état du parent, pertinence des modèles de base M_1 et M_2), et de la catégorie $c(e)$ de l'élément.

Les scores de pertinence sont calculés récursivement dans le réseau bayésien en commençant par la racine des documents. Le modèle est étendu au traitement des requêtes orientées contenu et structure dans [VIT 04].

4.4.2. Approches par propagation des scores des éléments

4.4.2.1. Le modèle d'inférence probabiliste

Pour étendre le modèle probabiliste inférentiel aux documents XML, les probabilités doivent tenir compte de l'information structurelle. Une approche est d'utiliser des probabilités conditionnelles de jointure, avec par exemple $P(d|t)$ devenant $P(d|p \text{ contains } t)$, où d représente un document ou une partie de document, t est un terme et p est un chemin dans l'arbre structurel de d .

Une méthode d'augmentation basée sur le modèle probabiliste est proposée par Fuhr *et al.* dans [FUH 03a], [GOV 02]. Cette méthode est basée sur le langage de requêtes XIRQL, et a été implémentée au sein du moteur de recherche HyRex.

Dans cette approche, les nœuds sont considérés comme des unités disjointes (paragraphe 4.2.1.1). Tous les nœuds feuilles ne sont cependant pas indexés (car d'une granularité trop fine). Dans ce cas-là les termes sont propagés jusqu'au nœud indexable le plus proche. Afin de préserver des unités disjointes, on ne peut associer à un nœud que des termes non reliés à ses nœuds descendants.

Le poids de pertinence des nœuds dans le cas de requêtes orientées contenu est calculé grâce à la *propagation* des poids des termes les plus spécifiques dans l'arbre du document. Les poids sont cependant diminués par multiplication par un facteur, nommé facteur *d'augmentation*.

Par exemple, considérons la structure de document de la figure 4.7, contenant un certain nombre de termes pondérés (par leur probabilité d'apparition dans l'élément), et la requête « XML ».

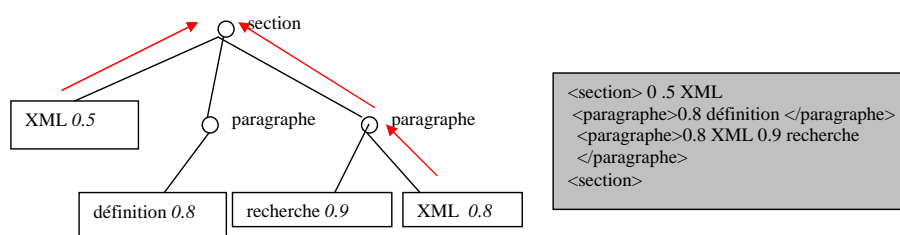


Figure 4.7. Modèle d'augmentation [FUH 03a]

Le poids de pertinence de l'élément *section* est calculé comme suit, en utilisant un facteur d'augmentation égal à 0.7 :

$$\begin{aligned}
 & P([\text{section}, \text{XML}]) + P([\text{paragraphe}[2]]) \cdot P([\text{paragraphe}[2], \text{XML}]) \\
 & - P([\text{section}, \text{XML}]) \cdot P([\text{paragraphe}[2]]) \cdot P([\text{paragraphe}[2], \text{XML}]) \\
 & = 0.5 + 0.7 * 0.8 - 0.5 * 0.7 * 0.8 = 0.68
 \end{aligned}$$

Le nœud *paragraphe* (ayant une pertinence de 0.8 à la requête) sera donc mieux classé que le nœud *section*.

Pour les requêtes orientées contenu et structure, des probabilités d'apparition de chaque terme de la condition de contenu dans les éléments répondant aux conditions de structure sont calculées, et des sommes pondérées de ces probabilités sont ensuite effectuées.

4.4.2.2. Le modèle XFIRM

Le modèle XFIRM [SAU 05a] est basé sur un modèle de données générique permettant l'implémentation de nombreux modèles de RI et le traitement de collections hétérogènes (c'est-à-dire contenant des documents ne suivant pas la même DTD).

Le traitement des requêtes orientées contenu est effectué comme présenté ci-dessous : une première étape consiste à évaluer la similarité des nœuds feuilles de l'index à la requête (on parle alors de calcul des poids des nœuds feuilles) et une seconde étape consiste à rechercher les sous-arbres pertinents. La pertinence des sous-arbres

est évaluée en effectuant la propagation des poids des feuilles dans l'arbre du document.

4.4.2.2.1. Calcul du score des nœuds feuilles

Les scores des nœuds feuilles identifiés dans l'arbre du document sont calculés grâce à la fonction de similarité $RSV(q, nf)$ (*Retrieval Status Value*).

Si la requête est composée de termes et des poids associés, on a :

$$RSV(q, nf) = \sum_{i=1}^T w_i^q \cdot w_i^{nf}, \text{ avec } w_i^q = tf_i^q \text{ et } w_i^{nf} = tf_i^{nf} \cdot ief_i \cdot idf_i [4.8]$$

où w_i^q et w_i^{nf} sont respectivement le poids du terme i dans la requête q et le nœud feuille nf , avec tf_i^q et tf_i^{nf} sont respectivement la fréquence du terme i dans la requête q et dans le nœud feuille nf , $idf_i = \log(|D|/|d_i|)$ permet d'évaluer l'importance du terme i dans la collection de documents, où $|D|$ est le nombre total de documents de la collection et $|d_i|$ est le nombre de documents contenant i , et $ief_i = \log(|F_c|/|nf_i|)$ permet d'évaluer l'importance du terme i dans la collection de nœuds feuilles, où $|F_c|$ est le nombre total de nœuds feuilles de la collection, et $|nf_i|$ est le nombre de nœuds feuilles contenant i .

4.4.2.2.2. Propagation de la pertinence des nœuds feuilles

Une valeur de pertinence est ensuite calculée pour chaque nœud de l'arbre de document, en utilisant les poids des nœuds feuilles qu'il contient. Les termes apparaissant près de la racine d'un sous-arbre paraissent plus porteurs d'information pour le nœud associé que ceux situés plus bas dans le sous-arbre. Il semble ainsi intuitif que plus grande est la distance entre un nœud et son ancêtre, moins il contribue à sa pertinence. Cette intuition est modélisée par l'utilisation dans la fonction de propagation du paramètre $dist(n, nf_k)$, qui représente la distance entre le nœud n et un de ses nœuds feuille nf_k dans l'arbre du document, c'est-à-dire le nombre d'arcs séparant les deux nœuds. Il paraît aussi intuitif que plus un nœud possède de nœuds feuilles pertinents, plus il est pertinent. Le paramètre $|F_n^p|$, qui est le nombre de nœuds feuilles descendants de n ayant un score non nul est alors introduit dans la fonction de propagation. Une première évaluation de la pertinence p_n d'un nœud peut être calculée selon la formule [4.9] :

$$p_n = |F_n^p| \cdot \sum_{nf_k \in F_n} \alpha^{dist(n, nf_k)-1} * (RSV_m(q, nf_k)) \quad [4.9]$$

où F_n est l'ensemble des nœuds feuilles nf_k descendants de n , et $\alpha \in]0..1]$ est un paramètre permettant de quantifier l'importance de la distance séparant les nœuds dans la formule de propagation.

Deux propositions sont également faites pour évaluer de manière plus juste l'informativité des nœuds résultats :

– l'utilisation du paramètre β pour augmenter l'importance des nœuds de petite taille durant la propagation (nœuds supposés être utilisés pour faire ressortir des informations importantes, comme par exemple les nœuds *titres* qui permettent de situer avec précision le sujet de leur nœud ancêtre) :

$$\beta(nf_k) = \begin{cases} l_k/\Delta l \text{ si } dist(n, nf_k) = 1 \text{ et } l_k < \Delta l \\ \log(\Delta l/l_k) \text{ si } dist(n, nf_k) > 1 \text{ et } l_k < \Delta l \\ 1 \text{ sinon} \end{cases} \quad [4.10]$$

avec Δl la taille moyenne d'un nœud feuille de la collection ;

– l'utilisation du contexte des éléments (c'est-à-dire de la pertinence du documents qui les contient) : un nœud appartenant à un document fortement pertinent doit être mieux classé qu'un nœud se trouvant dans un document de pertinence moindre.

La pertinence p_n d'un nœud n est alors définie de la façon suivante :

$$p_n = \rho * |F_n^p| \cdot \sum_{nf_k \in F_n} \alpha^{dist(n, nf_k)-1} * \beta(nf_k) * RSV(q, nf_k) + (1 - \rho) * p_{racine} \quad [4.11]$$

avec F_n et F respectivement l'ensemble des nœuds feuilles nf_k descendants de n et l'ensemble des nœuds feuilles nf_k du document, $|F_n^p|$ et $|F^p|$ respectivement le nombre de nœuds feuilles descendant de n ou du document et ayant un score non nul, $RSV(q, nf_k)$ calculé d'après [4.8], $\beta(nf_k)$ calculé d'après 4.10 et $\rho \in [0..1]$ est un paramètre servant de pivot et permettant d'ajuster l'importance de la pertinence du nœud racine lors de la rétropropagation.

Les nœuds sont ensuite renvoyés à l'utilisateur par ordre décroissant de pertinence à la requête.

Les expérimentations menées sur le modèle au sein de la campagne d'évaluation INEX montrent ses bonnes performances par rapport à l'ensemble des participants [SAU 05b], [SAU 06b], [PIN 06].

On trouvera d'autres méthodes utilisant la propagation des scores des éléments dans [GRA 02], [ANH 02], [KAK 05], [KAZ 02], [GEV 05], [ROE 02], [OGI 03], [OGI 05], [HUB 05], appliquées aux modèles vectoriels ou probabilistes.

4.4.3. Remarques concernant le traitement de la structure

La plupart des modèles proposés dans la littérature ne proposent pas d'approches réellement orientées RI pour le traitement des conditions de structure des requêtes. Certaines approches proposent par exemple d'effectuer un simple filtre sur les résultats des conditions de contenu [SIG 03], [THE 02]. D'autres approches, indépendantes des modèles de pondération de termes utilisés, existent cependant, et cherchent à évaluer la pertinence des conditions structurelles. Nous nous proposons de les décrire dans le paragraphe suivant.

Dans [BRA 02], les auteurs proposent le langage FXpath (*Fuzzy XPath*), possédant les caractéristiques suivantes :

- une *correspondance d'arbres floue*, ce qui permet de renvoyer à l'utilisateur une liste triée d'éléments et non un ensemble non-ordonné comme le fait XPath ;
- des *prédicats flous*, permettant à l'utilisateur de spécifier des conditions de sélection imprécises et approximatives (introduction d'un prédicat *NEAR* et d'un prédicat *CLOSE*) ;
- une *quantification floue*, permettant la spécification d'opérateurs linguistiques comme opérateurs d'agrégation (par exemple *tout*, *au moins un*, *la plupart*, etc.).

D'autres approches cherchent elles aussi à effectuer la correspondance entre l'arbre du document et l'arbre de la requête [SCH 02], [MAS 03], [YOO 02].

Dans [YOO 02], l'auteur définit la notion de proximité à l'aide de distances. Dans des documents structurés, la distance peut être définie en terme de nombres de mots entre des termes de nœuds feuilles ou en termes de nœuds entre les nœuds. La distance des nœuds peut être quantifiée grâce à la distance horizontale (nombre de nœuds du même niveau entre les nœuds) et à la distance verticale (nombre d'unités logiques qui peuvent être groupées pour aller d'un nœud à un autre).

Dans [SAU 06a], le modèle XFIRM propose d'effectuer une correspondance vague entre l'arbre de la requête et l'arbre des documents. L'évaluation des requêtes est effectuée de la façon suivante :

- 1) les requêtes sont décomposées en sous-requêtes élémentaires du type $tg[q]$, où tg est une contrainte structurelle (un nom de balise) et q est une requête composée de simples mots-clés ;
- 2) une valeur de pertinence est calculée entre les nœuds feuilles et les conditions de contenu q des sous-requêtes élémentaires, et ce en utilisant l'équation [4.8] ;
- 3) les valeurs de pertinences sont propagées dans l'arbre du document pour répondre aux conditions de structures (équation [4.9]) ;
- 4) les requêtes originales sont évaluées grâce à une propagation vers le haut et vers le bas des poids de pertinence des nœuds répondants aux sous-requêtes élémentaires.

L'évaluation du système dans le cadre d'INEX a montré le grand intérêt de la proposition [SAU 05c]. Le modèle a, par exemple, été classé premier pour la tâche SVCAS de la campagne d'évaluation d'INEX 2005.

4.5. Evaluation : la campagne d'évaluation INEX

INEX (*Initiative for the Evaluation of XML Retrieval*) est à ce jour la seule campagne d'évaluation des différents SRI pour la recherche d'information sur des documents XML. Elle a lieu en 2006 pour la cinquième année consécutive. L'année passée, près de 70 équipes du monde entier ont participé à cette campagne d'évaluation.

Le but principal d'INEX est de promouvoir l'évaluation de la recherche sur des documents XML en fournissant une collection de test, des procédures d'évaluation et un forum pour permettre aux différentes organisations participantes de comparer leurs résultats.

La collection de test consiste en un ensemble de documents XML, requêtes et jugements de pertinence. Les requêtes et les jugements de pertinence associés sont obtenus grâce à la collaboration des participants.

4.5.1. Collection

Jusqu'en 2005, la collection INEX était composée d'articles scientifiques provenant de la IEEE Computer Society, balisés au format XML. La collection, d'environ 700 Mo, a atteint plus de 17 000 articles, publiés de 1995 à 2004, et provenant de 18 magazines ou revues différents.

En 2006, la collection s'est renouvelée : d'environ 4,5 Go, elle est composée de plus de 650 000 articles de l'encyclopédie Wikipedia [DEN 06].

4.5.2. Requêtes

Les requêtes (ou *topics*) sont créées par les différents participants et doivent être représentatives des demandes de l'utilisateur moyen sur la collection. Longtemps séparés en requêtes orientées contenu et requêtes orientées structure, les deux formes de requêtes sont regroupées depuis 2006 et les *topics* comportent aujourd'hui deux formulations différentes :

- le titre (champ <*title*>) du *topic* exprime le besoin en information de l'utilisateur sous forme de simples mots-clés ;
- le titre structuré (champ <*castitle*>) du *topic* exprime le même besoin en information de l'utilisateur mais en y ajoutant des précisions sur la structure des documents et/ou des unités d'information recherchées.

Enfin, les champs *Description* et *Narrative*, explicités en langage naturel, indiquent les intentions de l'auteur [SIG 04].

4.5.3. Tâches

4.5.3.1. Tâche ad hoc

La tâche principale d'INEX est la tâche de recherche *ad hoc*. Comme en recherche d'information traditionnelle, la recherche ad hoc est considérée dans INEX comme une simulation de l'utilisation d'une bibliothèque, où un ensemble statique de documents est interrogé avec des besoins utilisateurs, c'est-à-dire des requêtes. Les requêtes peuvent contenir à la fois des conditions structurelles ou de contenu, et en réponse à une requête, des éléments (et non forcément des documents) peuvent être retrouvés à partir de la bibliothèque. La tâche *ad hoc* se divise en 2006 en quatre sous-tâches :

- la tâche de recherche exhaustive (*thorough task*), qui consiste à retourner tous les éléments (éventuellement imbriqués les uns dans les autres) répondant au sujet de la requête et triés par degré de pertinence ;
- la tâche de recherche focalisée (*focused task*), qui consiste à retourner les éléments les plus précis possibles satisfaisant le besoin en information de l'utilisateur (éléments imbriqués interdits) ;
- la tâche de recherche de pertinence en contexte (*relevant in context task*), qui consiste à retourner des éléments dans le contexte d'articles entiers (les unités d'informations sont triées par document) ;
- la tâche de recherche du meilleur en contexte (*best in context task*), qui consiste à renvoyer un seul élément par article, à savoir le meilleur point d'entrée dans l'article.

Pour chacune de ces stratégies de recherche, les champs *<title>* ou *<castitle>* des *topics* peuvent être utilisés.

4.5.3.2. Autres tâches

Depuis 2004, de nouvelles tâches sont proposées aux participants, parmi lesquelles on peut citer :

- la tâche de *relevance feedback*, qui a pour but d'expérimenter l'utilisation du contenu ET de la structure comme informations de base pour la formulation d'une nouvelle requête ;
- la tâche de *langage naturel*, dans laquelle les utilisateurs formulent leurs requêtes en langage naturel, et donc sans avoir besoin d'apprendre un langage complexe ;
- la tâche *interactive* qui a pour but d'étudier le comportement des utilisateurs face à des corpus XML et donc de cerner au mieux leurs besoin ;

- la tâche *hétérogène*, qui propose aux participants de nouvelles collections, afin de développer des approches indépendantes des DTD ;
- la tâche *multimedia*, dédiée pour le moment à la recherche de contenus images au sein de documents XML.

4.5.4. Jugements de pertinence

L'évaluation de la pertinence des SRI passe par une première phase de validation des documents renvoyés par les SRI. Chaque élément/document est jugé à la main (par les participants) pour chaque requête, en utilisant le système de jugement en ligne [PIW 06]. Les participants surlignent dans les documents les parties qui ne contiennent que de l'information pertinente (indépendamment de la structure des documents). Les niveaux d'exhaustivité et de spécificité de chaque élément sont ensuite déduits par le système.

4.5.5. Evaluation

L'évaluation de la performance des différents systèmes proposés par les participants a tout d'abord utilisé des méthodes basées sur les mesures de rappel et précision, en cherchant à prendre en compte la structure des documents XML [FUH 02], [FUH 03b], [FUH 04]. Ces mesures présentent cependant un inconvénient majeur : elles ne prennent pas en compte l'imbrication (*overlap*) des éléments et évaluent le retour d'un élément pertinent sans prendre en compte le fait qu'il ait été déjà peut-être vu entièrement ou en partie par l'utilisateur. Par exemple, un système A renvoyant une section pertinente et aussi un de ses paragraphes pertinent obtient les mêmes performances qu'un système B renvoyant deux éléments pertinents non imbriqués.

En 2003, de nouvelles mesures ont été fournies pour essayer de résoudre ce problème. On peut par exemple citer les mesures proposées dans [GöV 03], qui incorporent la taille des éléments et le concept d'imbrication dans les mesures de rappel et précision, ou encore la mesure ERR (*Expected Ratio of Relevant Units*) [PIW 03c], qui est basée sur le comportement hypothétique d'un utilisateur (consultation du contexte structurel d'un élément retourné (parents, enfants, frères)).

Cependant, les mesures décrites ci-dessus ne prennent pas en compte un problème essentiel de l'évaluation : la *surpopulation de la base de rappel* [KAZ 04]. Cette surpopulation est due aux règles d'inférence utilisées lors de l'élaboration des jugements de pertinence [PIW 03b] : si un nœud est jugé pertinent, ses ancêtres doivent aussi être jugés pertinents, même si leur degré de pertinence est moindre (et ce notamment à cause de la propagation de l'exhaustivité dans l'arbre du document). Par conséquent, un taux de rappel idéal ne peut être obtenu que par les systèmes référençant tous les composants de la base de rappel, y compris les éléments imbriqués.

Afin de résoudre ce problème, Gabriella Kazai *et al.* établissent dans [KAZ 04] la définition d'une base de rappel idéale, qui supporterait la procédure d'évaluation suivante : les éléments de la base de rappel idéale *doivent* être retournés par les systèmes, les éléments proches de ceux contenus dans la base de rappel idéale *peuvent* être vus comme des succès partiels, mais les autres systèmes *ne doivent pas* être pénalisés s'ils ne les renvoient pas. Les mesures XCG sont proposées pour répondre à ces besoins. Les mesures XCG (*XML Cumulated Gain*) sont des extensions du « gain cumulatif » proposé par Järvelin et Kekäläinen dans [JÄR 02]. Les mesures de gain cumulatif ont été développées pour évaluer les systèmes selon le degré de pertinence des documents retournés, et sont aujourd'hui utilisées comme mesures officielles [KAZ 05].

Les mesures xCG sont cependant loin d'être adoptées de manière définitive, une réflexion continue ayant lieu sur les mesures utilisées [PIW 05], [KAZ 06], [PEH 06b] et la notion de pertinence [PEH 06a].

Les résultats et réflexions issus des premières campagnes d'évaluation INEX et des *workshops* qui ont suivi sont disponibles dans [FUH 02], [FUH 03b], [FUH 04], [FUH 05], [FUH 06], et permettent d'établir un aperçu complet de la recherche d'information structurée actuelle.

4.6. Conclusion

La recherche d'information dans les documents semi-structurés a, comme nous venons de le montrer, réactualisé de nombreuses problématiques de la recherche d'information : indexation, interrogation, modèle d'appariement granules documentaires/requêtes, etc. De nombreuses approches ont été proposées dans la littérature et un certain nombre de solutions ont été trouvées, même si de nombreuses voies de recherche sont encore ouvertes.

Parmi elles, on peut citer :

- *le problème des corpus hétérogènes* : l'interrogation de corpus hétérogènes (c'est-à-dire composés de documents suivant des DTD différentes) reste un problème ouvert. Les conditions de structures exprimées par les utilisateurs dans la requête ne correspondent pas forcément exactement aux DTD des documents présents dans le corpus, mais ces derniers pourraient pourtant être pertinents pour l'utilisateur. Alors que les approches que nous avons présentées jusqu'ici cherchent à vérifier des *correspondances syntaxiques* entre les arbres de la requête et des documents, les approches pour les corpus hétérogènes cherchent, quant à elles, à vérifier des *correspondances sémantiques*.

Une première solution est d'utiliser un lexique, un *thesaurus* ou une ontologie pour faire correspondre les conditions de structures exprimées dans la requête avec les types d'éléments effectivement présents dans la collection [THE 02].

D'autres approches, comme celle proposée par Denoyer *et al.* dans [DEN 04] ou Abiteboul *et al.* dans [ABI 04] visent à proposer un format médian dans lequel tous les documents du corpus (et éventuellement les requêtes) peuvent être transformés pour ensuite appliquer des techniques traditionnelles de traitement des requêtes structurées.

Depuis 2004, une tâche visant à proposer des solutions pour l'interrogation de corpus hétérogènes a été introduite dans INEX [SZA 04], et permettra d'évaluer ces différentes approches ;

– *l'interaction avec l'utilisateur* : la RI structurée se heurte à des problèmes d'interaction avec les utilisateurs, tant au niveau de l'interrogation que de la visualisation des résultats. Les interfaces d'interrogation devraient guider l'utilisateur dans la formulation de la requête, si possible de façon dynamique, en lui présentant par exemple les éléments de structure sur lesquels il peut interroger le système. La présentation des résultats soulève un grand nombre de questions : les résultats doivent-ils être présentés dans leur contexte (c'est-à-dire au sein du document) ou bien doivent-ils, puisqu'ils sont censés être informatifs, apparaître indépendamment ? Doit-on regrouper les résultats par document ou bien présenter une simple liste triée de résultats ? Ce dernier point nous amène aussi à réfléchir au regroupement des unités d'informations [PIW 03a] : la réponse à un besoin utilisateur peut être amenée par plusieurs éléments indépendants, chacun apportant une information supplémentaire à l'utilisateur. Pour répondre au mieux au besoin de l'utilisateur, ces éléments pourraient être regroupés, et les résultats seraient alors présentés à l'utilisateur sous forme d'une liste de groupes d'éléments ;

– *l'introduction de données multimedia* : la naissance du standard XML et la croissance des documents multimedias (texte, image, vidéos, etc.) ont amené la recherche d'information vers de nouvelles problématiques, qui ont jusqu'à très récemment été traitées séparément. Peu d'approches se sont jusqu'à présent intéressées à la prise en compte de l'information structurelle des documents pour aider à identifier les contenus multimedia pertinents, alors que la structure en tant que source d'évidence a déjà prouvé son efficacité dans le cadre de la recherche textuelle. Un nombre croissant de travaux s'intéressent cependant à la question, comme le montre par exemple l'apparition de la tâche multimedia au sein de la campagne d'évaluation INEX.

4.7. Bibliographie

- [ABI 97] ABITEBOUL S., QUASS D., MCHUGH J., WIDOM J., WIENER J.-L., « The Lorrel query language for semi-structured data », *International Journal on Digital Libraries*, vol. 1, n° 1, p. 68-88, 1997.
- [ABI 04] ABITEBOUL S., MANOLESCU I., NGUYEN B., PRADA N., « A test platform for the INEX heterogeneous track », *Pre-proceedings of INEX 2004*, Dagstuhl, Allemagne, p. 177-182, 2004.

- [ABO 04] ABOLHASSANI M., FUHR N., « Applying the divergence from Randomness approach for content-only search in XML documents », *Proceedings of ECIR 2004*, Sunderland, Grande-Bretagne p. 409-419, 2004.
- [AME 04] AMER-YAHIA S., BOTEV C., SHANMUGASUNDARAM J., « TeXQuery : A Full-Text Search Extension to Xquery », *Proceedings of WWW 2004*, 2004.
- [ANH 02] ANH V. N., MOFFAT A., « Compression and an IR approach to XML Retrieval », *Proceedings of INEX 2002 Workshop*, Dagstuhl, Allemagne, 2002.
- [BRA 01] BRADLEY N., *The XML Companion*, Addison-Wesley Professional Publisher, 2001.
- [BRA 02] BRAGA D., CAMPI A., DAMIANI E., LANZI P., PASI G., « FXpath : Flexible querying of XML documents », *Proceedings of Eurofuse 2002*, 2002.
- [BUN 96] BUNEMAN P., DAVIDSON S., HILLEBRAND G., SUCIU D., « A query language and optimization techniques for unstructured data », *Proceedings of ACM-SIGMOD International Conference on Management of Data*, Montréal, Canada p. 505-516, 1996.
- [CAR 03] CARMEL D., MAAREK Y., MANDELBROT M., SOFFER A., « Searching XML documents via XML fragments », *Proceedings of SIGIR 2003*, p. 151-158, 2003.
- [CER 99] CERI S., COMAI S., DAMIANI E., FRATERNALI P., PARABOSCHI S., TANCA L., « XML-GL : A graphical language for querying and restructuring WWW Data », *Proceedings of the 8th Int. WWW Conference, WWW8*, Toronto, Canada, mai 1999.
- [CHA 00] CHAMBERLIN D., ROBIE J., FLORESCU D., « Quilt : An XML query language for heterogeneous data sources », *Proceedings of the 3rd International Workshop on World Wide Web and databases*, Dallas, Etats-Unis, p. 1-25, 2000.
- [CHI 96] CHIARAMELLA Y., MULHEM P., FOUREL F., A Model for Multimedia Information Retrieval, Rapport, Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, 1996.
- [CHI 02] CHINENYANGA T. T., KUSHMERICK N., « An Expressive and Efficient Language for XML Information Retrieval », *Journal of the American Society for Information Science and Technology (JASIST)*, vol. 53, n° 6, p. 538-543, 2002.
- [CLA 99] CLARK J., DEROSE S., XML Path Language (XPath) , Version 1.0, Rapport, World Wide Web Consortium (W3C), W3C Recommendation, novembre 1999.
- [COH 02] COHEN S., KANZA Y., KOGAN Y. A., SAGIV Y., NUTT W., SEREBRENİK A., « EquiX - A search and query language for XML », *Journal of the American Society for Information Science and Technology*, vol. 53, n° 6, p. 454-466, 2002.
- [COL 02] COLAZZO D., SARTIANI C., ALBANO A., MANGHI P., GHELLI G., LINI L., PAOLI M., « A typed text retrieval query language for XML Documents », *JASIST*, vol. 53, n° 6, p. 647-488, 2002.
- [DEN 04] DENOYER L., WISNIEWSKI G., GALLINARI P., « Document Structure matching for heterogeneous corpora », *Proceedings of XML and IR workshop, SIGIR 2004*, Sheffield, Grande-Bretagne, 2004.

- [DEN 06] DENOYER L., GALLINARI P., « The Wikipedia XML Corpus », *Pre-proceedings of INEX 2006*, Dagstuhl, Allemagne, p. 379-384, 2006.
- [FAL 01] FALLSIDE D.C., XML Schema, Rapport, World Wide Web Consortium (W3C), W3C Recommendation, 2001.
- [FER 97] FERNANDEZ M., FLORESCU D., LEVY A., SUCIAU D., « A query language for a Web site management system », *SIGMOD Record*, vol. 26, n° 3, p. p. 4-11, septembre 1997.
- [FER 03] FERNANDEZ M., MALHOTRA A., MARSH J., NAGY M., WALSH N., XQuery 1.0 and XPath 2.0 Data Model, Rapport, World Wide Web Consortium (W3C), W3C Working Draft, mai 2003.
- [FLO 99] FLORESCU D., KOSSMANN D., « Storing and Querying XML Data using an RDMBS », *IEEE Data Engineering Bulletin*, vol. 22, n° 3, p. 27-34, 1999.
- [FUH 02] FUHR N., GOVERT N., KAZAI G., LALMAS M., *Proceedings of the first workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2002)*, Dagstuhl, Allemagne, 2002.
- [FUH 03a] FUHR N., GROSSJOHANN K., « XIRQL : a query language for Information retrieval in XML documents », *Proceedings of SIGIR 2001, Toronto, Canada*, 2003.
- [FUH 03b] FUHR N., LALMAS M., MALIK S., *INEX 2003 Workshop proceedings*, Dagstuhl, Allemagne, 2003.
- [FUH 04] FUHR N., LALMAS M., MALIK S., SZLAVIK Z., *2004 Workshop Proceedings*, Dagstuhl, Allemagne, Springer, 2004.
- [FUH 05] FUHR N., LALMAS M., MALIK S., KAZAI G., *INEX 2005 Workshop Proceedings*, Dagstuhl, Allemagne, 2005.
- [FUH 06] FUHR N., LALMAS M., TROTMAN A., *INEX 2006 Workshop Pre-Proceedings*, Dagstuhl, Germany, 2006.
- [GAR 02] GARDARIN G., *XML : Des bases de données aux services Web*, Dunod, Paris, 2002.
- [GEV 05] GEVA S., « GPX - Gardens Point XML IR at INEX 2005 », p. 240-253, novembre 2005.
- [GEV 06] GEVA S., HASSLER M., TANNIER X., « XOR - XML Oriented Retrieval Language », *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, Seattle, Etats-Unis, août 2006.
- [GOL 90] GOLDFARB C., *The SGML Handbook*, Oxford University Press, Oxford, 1990.
- [GOV 02] GOVERT N., ABOLHASSANI M., FUHR N., GROSSJOHANN K., « Content-oriented XML retrieval with HyReX », *Proceedings INEX 2002*, Dagstuhl, Allemagne, 2002.
- [GRA 02] GRABS T., SCHECK H.-J., « Flexible information retrieval from XML with PowerDB XML », *Proceedings of INEX 2002*, Dagstuhl, Allemagne, p. 26-32, décembre 2002.
- [GRA 03] GRABS T., Storage and Retrieval of XML Documents within a Cluster of Database Systems, PhD thesis, Ecole Polytechnique Fédérale de Zürich, 2003.

- [GRO 00] GROSSJOHANN K., « Query Formulation and Result Visualization for XML Retrieval », *Proceedings of the SIGIR 2000 Workshop on XML and Information Retrieval, Athènes, Grèce, 2000.*
- [GRU 02] GRUST T., « Accelerating XPath Location Steps », *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, Etats-Unis, M. J. Franklin, B. Moon, A. Ailamaki (dir.), ACM Press, 2002.*
- [GöV 03] GÖVERT N., KAZAI G., FUHR N., LALMAS M., Evaluating the effectiveness of content-oriented XML retrieval, *Technischer Bericht, University of Dortmund, Computer Science 6, 2003.*
- [HAS 05] MARCUS HASSLER, BOUCHACHIA A., « Searching XML documents - preliminary work », p. 30-42, novembre 2005.
- [HEA 97] HEARST M., « TextTiling : A Quantitative Approach to Discourse Segmentation », *Computational Linguistics*, vol. 23, n° 1, p. 33-64, mars 1997.
- [HUB 05] HUBERT G., « XML retrieval based on direct contribution of query components », Dans [FUH 05], p. 172-186, novembre 2005.
- [HUC 00] HUCK G., MACHERIUS I., FANKHAUSER P., « PDOM : Lightweight Persistency Support for the Document Object Model », *textitSucceeding with Object Databases, John Wiley, 2000.*
- [JAN 99] JANG H., KIM Y., SHIN D., « An effective mechanism for index update in structured documents », *Proceedings ACML CIKM, Kansas City, Etats-Unis p. 383-390, 1999.*
- [Jär 02] JÄRVELIN K., KEKÄLÄINEN J., « Cumulated gain-based evaluation of IR techniques », *ACM Transactions on Information Systems*, vol. 20, n° 4, p. 422-446, 2002.
- [KAK 05] KAKADE V., RAGHAVAN P., « Encoding XML in vector spaces », *Proceedings of ECIR 2005, Saint Jacques de Compostelle, Espagne, 2005.*
- [KAM 04] KAMPS J., DE RIJKE M., SIGURBJORNSSON B., « Length normalization in XML retrieval », *Proceedings of SIGIR 2004, Sheffield, Grande-Bretagne, p. 80-87, 2004.*
- [KAN 00] KANNE C.-C., MOERKOTTE G., « Efficient Storage of XML Data », *Proceedings of the 16th International Conference on Data Engineering, San Diego, Californie, Etats-Unis, p. 198, 2000.*
- [KAZ 02] KAZAI G., LALMAS M., ROELLEKE T., « Focused document retrieval », *9th International Symposium on string processing and information retrieval, Lisbonne, Portugal, septembre 2002.*
- [KAZ 04] KAZAI G., LALMAS M., DE VRIES A. P., « The overlap problem in Content-oriented XML retrieval evaluation », *Proceedings of SIGIR 2004, Sheffield, Grande-Bretagne, p. 72-79, juillet 2004.*
- [KAZ 05] KAZAI G., LALMAS M., « INEX 2005 Evaluation Metrics », Dans [FUH 05], novembre 2005.
- [KAZ 06] KAZAI G., « Choice of Parameter Values for the INEX Evaluation Metrics : Sensitivity Analysis », Dans [FUH 06], 2006.

- [LAL 97] LALMAS M., « Dempster-Shafer's theory of evidence applied to structured documents : modeling uncertainty », *Proceedings of SIGIR'97*, Philadelphie, Etats-Unis, p. 110-118, 1997.
- [LAM 94] LAMBOLEZ P.-Y., Recherche d'Informations pour la maintenance logicielle, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1994.
- [LAM 95] LAMBOLLEZ P.-Y., QUEILLE J.-P., VOIDROT J.-F., CHRISMENT C., « EXREP : un outil générique de réécriture pour l'extraction d'informations textuelles », *Revue ISI*, vol. 3, n° 4, p. p. 471-487, 1995.
- [LEE 96] LEE Y., YOO S., YOON K., « Index Structures for structured documents », *Proc. ACM Workshop on XML and IR*, Bethesda, p. 91-99, 1996.
- [LEV 98] LEVY A., FERNANDEZ M., SUCIU D., FLORESCU D., DEUTSCH A., XML-QL : A query language for XML, Rapport, World Wide Web Consortium technical report, Number NOTE- xml-ql-19980819, 1998.
- [LI 01] LI Q., MOON B., « Indexing and querying XML data for regular path expressions », *Proceedings of the 27th VLDB Conference*, Rome, Italie, 2001.
- [LUK 02] LUK R. W., LEONG H., DILLON T. S., SHAN A. T., CROFT W. B., ALLAN J., « A survey in indexing and searching XML documents », *Journal of the American Society for Information Science and Technology*, vol. 53, n° 3, p. 415-435, 2002.
- [MAS 02] MASS Y., MANDELBROD M., AMITAY E., CARMEL D., MAAREK Y., SOFFER A., « JuruXML- an XML retrieval system at INEX'02 », *Proceedings of INEX 2002*, Dagstuhl, Allemagne, p. 73-80, 2002.
- [MAS 03] MASS Y., MANDELBROD M., « Retrieving the most relevant XML components », *Proceedings of INEX 2003*, Dagstuhl, Allemagne, 2003.
- [MIG 03] MIGNET L., BARBOSA D., VELTRI P., « The XML Web : A first study », *Proceedings of WWW2003*, Budapest, Hongrie, 2003.
- [MOF 93] MOFFAT A., SACKS-DAVIS R., WILKINSON R., ZOBEL J., « Retrieval of Partial Documents », *Proceedings of TREC-2*, 1993.
- [OGI 03] OGILOVIE P., CALLAN J., « Using Language Models for Flat Text Queries in XML Retrieval », *Proceedings of INEX 2003 Workshop*, Dagstuhl, Allemagne, p. 12-18, décembre 2003.
- [OGI 05] OGILOVIE P., CALLAN J., « Parameter estimation for a simple hierarchical generative model from XML retrieval », p. 211-224, novembre 2005.
- [PEH 06a] PEHCEVSKI J., « Relevance in XML retrieval : the user perspective », *Proceedings of the SIGIR 2006 Workshop on XML Element Retrieval Methodology*, Seattle, Etats-Unis, août 2006.
- [PEH 06b] PEHCEVSKI J., THOM J. A., VERCOUSTRE A.-M., « XML Retrieval Evaluation Revisited : Comparison of Metrics », Dans [FUH 06], 2006.
- [PIN 06] PINEL-SAUVAGNAT K., BOUGHANEM M., « Propositions pour la pondération des termes et l'évaluation de la pertinence des éléments en recherche d'information structurée », *Information - Interaction - Intelligence*, vol. 6, n° 2, Cepaduès Editions, 2006.

- [PIW 02] PIWOWARSKI B., FAURE G.-E., GALLINARI P., « Bayesian networks and INEX », *Proceedings in the First INEX Workshop*, décembre 2002.
- [PIW 03a] PIWOWARSKI B., Techniques d'apprentissage pour le traitement d'information structurées : application à la recherche d'information, Thèse de doctorat : Université Paris 6, 2003.
- [PIW 03b] PIWOWARSKI B., « Working group report : the assessment tool », *Proceedings of INEX 2003*, Dagstuhl, Allemagne, p. 181-183, décembre 2003.
- [PIW 03c] PIWOWARSKI B., GALLINARI P., « Expected Ratio of Relevant Units : a measure for structured information retrieval », *Proceedings of INEX 2003*, Dagstuhl, Allemagne, p. 158-166, décembre 2003.
- [PIW 05] PIWOWARSKI B., « EPRUM metrics and INEX 2005 », Dans [FUH 05], p. 30-42, novembre 2005.
- [PIW 06] PIWOWARSKI B., « INEX 2006 Relevance Assessment Guide », *Pre-proceedings of INEX 2006*, Dagstuhl, Allemagne, p. 401-408, 2006.
- [ROB 98] ROBIE J., LAPP J., SCHACH D., « XML Query Language (XQL) », *Proceedings of W3C QL'98 (Query Languages 98)*, Massachusetts, 1998.
- [ROE 02] ROELLEKE T., LALMAS M., KAZAI G., RUTHVEN J., QUICKER S., « The accessibility dimension for structured document retrieval », *Proceedings of ECIR 2002*, 2002.
- [SAL 93] SALTON G., ALLAN J., BUCKLEY C., « Approaches to passage retrieval in full text information systems », *Proc. of SIGIR'93*, Pittsburgh, Etats-Unis, 1993.
- [SAU 05a] SAUVAGNAT K., Modèle flexible pour la recherche d'information dans des corpus de documents semi-structurés, Thèse de doctorat, Toulouse, Université Paul Sabatier, 2005.
- [SAU 05b] SAUVAGNAT K., BOUGHANEM M., « A la recherche de nœuds informatifs dans des corpus de documents XML - Ou pourquoi on a toujours besoin de plus petit que soi... », *Actes de CORIA 05*, Grenoble, France, 2005.
- [SAU 05c] SAUVAGNAT K., HLAOUA L., BOUGHANEM M., « XFIRM at INEX 2005 : adhoc and relevance feedback tracks. », Dans [FUH 05], 2005.
- [SAU 06a] SAUVAGNAT K., BOUGHANEM M., CHRISMENT C., « Answering content-and-structure-based queries on XML documents using relevance propagation », *Information Systems, Special Issue SPIRE 2004*, vol. 31, p. 621-635, Elsevier, 2006.
- [SAU 06b] SAUVAGNAT K., HLAOUA L., BOUGHANEM M., « XML retrieval : what about using contextual relevance ? », *ACM Symposium on Applied Computing (SAC) - IAR (Information Access and Retrieval)*, Dijon, 23-27 avril 2006.
- [SCH 02] SCHILEDER T., MEUSS H., « Querying and ranking XML documents », *Journal of the American Society for Information Science and Technology*, vol. 53, n° 6, p. p. 489-503, 2002.
- [SIG 03] SIGURBJÖRNSSON B., KAMPS J., DE RIJKE M., « An element-based approach to XML retrieval », *Proceedings of INEX 2003 workshop*, Dagstuhl, Allemagne, décembre 2003.

- [SIG 04] SIGURBJÖRNSSON B., LARSEN B., LALMAS M., MAALIK S., « INEX04 Guidelines for topic development », *Pre-proceedings of INEX 2005*, Dagstuhl, Allemagne, p. 212-218, 2004.
- [SZA 04] SZALIK Z., ROELLEKE T., « Building and experimenting with a heterogeneous collection », *Pre-proceedings of INEX 2004*, Dagstuhl, Allemagne, p. 24-32, 2004.
- [TAN 05] TANNIER X., GIRARDOT J.-J., MATTHIEU M., « Utilisation de la langue naturelle pour l'interrogation de documents structurés », *Actes de CORIA 05*, Grenoble, France, 2005.
- [THE 02] THEOBALD A., WEIKUM G., « The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking », *EDBT 2002, 8th International Conference on Extending Database Technology*, Prague, République tchèque, p. 477-495, 2002.
- [TRO 04a] TROTMAN A., « Searching structured documents », *Information Processing and Management*, vol. 40, p. 619-632, 2004.
- [TRO 04b] TROTMAN A., SIGURBJÖRNSSON B., « Narrowed Extended XPath I (NEXI) », *INEX 2003 proceedings*, Dagstuhl, Allemagne, p. 219-237, décembre 2004.
- [TRO 04c] TROTMAN A., SIGURBJÖRNSSON B., « NEXI, now and next », *INEX 2003 proceedings*, Dagstuhl, Allemagne, p. 10-15, décembre 2004.
- [TRO 05] TROTMAN A., « Choosing document structure weights », *Information Processing and Management*, vol. 41, n° 2, p. 243-264, mars 2005.
- [VIT 04] VITTAUT J.-N., PIWOWARSKI B., GALLINARI P., « An algebra for Structured Queries in Bayesian Networks », *INEX 2004 Pre-proceedings*, Dagstuhl, Allemagne, p. 58-65, 2004.
- [W3C 98] W3C, EXtensible Markup Language (XML) 1.0, Rapport, World Wide Web Consortium (W3C), Technical report, février 1998.
- [W3C 03] W3C, XQuery and XPath Full-Text Use Cases, Rapport, World Wide Web Consortium (W3C), W3C working draft, février 2003.
- [WEI 04] WEIGEL F., MEUSS H., BRY F., SCHULZ K. U., « Content-Aware DataGuides : Interleaving IR and DB Indexing Techniques for Efficient Retrieval of Textual XML Data », *Proceedings of ECIR 2004*, Sunderland, Grande-Bretagne, p. 378-393, 2004.
- [WOL 00] WOLFF J., FLÖRKE H., CREMERS A., « Searching and browsing collections of structural information », *Proceedings of IEEE advances in digital libraries*, Washington, 2000, p. 141-150, 2000.
- [YOO 02] YOO S., « An XML Retrieval Model based On structural proximities », *INEX 2002 Workshop Proceedings*, Dagstuhl, Allemagne, p. 60-64, 2002.
- [ZAR 04] ZARGAYOUNA H., « Contexte et sémantique pour une indexation de documents semi-structurés », *Actes de CORIA 04*, Toulouse, France, p. 161-178, 2004.
- [ZIP 49] ZIPF G., *Human Behaviour and the Principle of Least Effort*, Addison-Wesley, 1949.