République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed Bougara-Boumerdes
Faculté des Sciences
Ecole Doctorale D'informatique

MEMOIRE

En vue de l'obtention du diplôme de Magister en Informatique
Spécialité : Spécification de Logiciels et Traitement de l'Information

Par
Ourdia Bouidghaghen

Prise en compte de l'hétérogénéité structurelle en recherche d'information semi-structurée

Soutenu le 11/04/2007 devant le jury :

Mr Mohamed Mezghiche	Professeur à l'université de Boumerdes (**Président**)
Mr Rachid Ahmed Ouamer	Maître de conférences à l'Université de Tizi-Ouzou (**Examinateur**)
Mr Abdelkarim Harzallah	Docteur à l'université de Boumerdes (**Examinateur)**
Mr Mohand Boughanem	Professeur à l'université Paul Sabatier de Toulouse (**Rapporteur**)

Année universitaire 2006/2007

Remerciements

Je tiens à remercier mon Rapporteur Mr Mohand Boughanem d'abords pour m'avoir offert la possibilité de réaliser ce travail et m'avoir encadré durant cette année, mais aussi pour ses relectures et commentaires.

Je remercie également les membres du jury :
Mr Mohamed Mezghiche, professeur à l'université de Boumerdes de m'avoir fait l'honneur d'être président de ce jury ;
Mr Rachid Ahmed Ouamar, maître de conférences à l'Université de Tizi-ouzou, et Mr Abdelkarim Harzallah, docteur à l'université de Boumerdes, d'avoir accepter d'évaluer mon travail.

Je tiens aussi à remercier madame Issad, la responsable PG pour son écoute et pour toutes ses qualités humaines.

Pour finir, je remercie mes parents, mes frères et sœurs ainsi que tous mes amis pour leur amour et leur soutien.

Résumé

Les travaux présentés dans ce mémoire se situent dans le contexte général de gestion automatisée de corpus de documents XML de structures hétérogènes. Leur objectif est de proposer des solutions pour l'interrogation de ce type de documents sans se soucier de cette hétérogénéité.

L'émergence d'XML comme langage de représentation a créé une grande quantité de documents qui bien que se rapportant au même domaine sont structurés différemment. Cela est une conséquence directe de la liberté qu'offre XML aux concepteurs pour représenter leurs données. En effet, deux concepteurs différents peuvent employer différents noms de balises pour désigner un même concept. De même, le nombre des balises et leur agencement, peuvent varier à travers des sources disparates de documents. L'hétérogénéité des structures des documents est de ce fait inévitable.

L'accès aux documents semi structurés suivant des structures hétérogènes, dans le cadre de la recherche d'information soulève un réel problème. En effet, comme ces documents peuvent être interrogés à la fois à travers des requêtes comportant que des mots clés ou des requêtes combinant mots clés et contraintes structurelles (balises), la connaissance de toutes les structures dans le second cas par un utilisateur est impossible. Il appartient alors au système de recherche d'information de fournir des moyens adéquats pour l'interrogation de tels corpus. Il est nécessaire alors de répondre aux questions suivantes : quelle méthode peut être utilisée pour établir les correspondances entre les différentes structures? Les correspondances doivent-elles se focaliser uniquement sur la différence des noms de balises, ou bien faut-il considérer aussi la différence de structuration de ces balises?

Nous nous sommes intéressés dans ce mémoire à proposer des solutions pour répondre à de telles problématiques. Dans ce cadre, nous avons présenté principalement trois contributions. Dans la première, pour remédier au problème de la variation linguistique, nous proposons de concevoir un dictionnaire des balises synonymes de la collection en utilisant une ontologie (WordNet). Dans la seconde, nous tentons de répondre aux deux problèmes de la différence des noms de balises et leur structuration dans les différents schémas des documents. Pour cela, nous proposons d'utiliser une ontologie pour concevoir une structure générique unifiant tous les schémas des documents de la collection. Dans la dernière, nous proposons de convertir les documents XML de structures hétérogènes vers un schéma de médiation. Cette conversion se fait de manière automatique à partir de règles de transformation applicables pour toute la collection.

Mots clefs

Recherche d'information, documents XML, structures hétérogènes, variation linguistique, variation de la hiérarchisation, ontologie, interrogation générique, schéma médian, apprentissage automatique, conversion de documents.

Table des matières

Chapitre 2 : La recherche d'information structurée

Parie II : Interrogation de corpus hétérogènes

Chapitre 3 : Interrogation de corpus hétérogènes : Un état de l'art.

Chapitre 4 : Contribution à l'interrogation de corpus hétérogènes

.

Conclusion générale

Bibliographie

Listes des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Chapitre 4 :

Liste des tableaux

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Chapitre 4 :

Introduction générale

Les corpus de documents ont considérablement évolué ces dernières années. Si auparavant on distinguait principalement les documents non structurés comme les documents textuels plats et les images, les documents très structurés comme les données relationnelles en Base de données (BD). Nous assistons aujourd'hui à l'émergence des documents dits semi-structurés, mixant structure et contenu. Ces documents sont souvent décrits dans le langage XML dont l'utilisation a connu un énorme succès aussi bien en BD qu'en Recherche d'Information (RI).

Contexte de travail

Les travaux présentés dans ce mémoire s'inscrivent principalement dans le domaine de la Recherche d'Information. Ils abordent précisément un problème d'actualité, en l'occurrence la Recherche d'Information dans des documents semi-Structurés de type XML (RIS).

La problématique engendrée par ce type de document est liée à la nature de leur contenu.
En effet, comme ces documents comportent de l'information (du texte) et des contraintes structurelles (des balises), ils ne peuvent pas être efficacement exploités par les techniques classiques de RI, qui considèrent le document comme un granule d'information indivisible.

Or, dans un document XML toute partie du document peut être considérée comme réponse potentielle à la requête de l'utilisateur. La partie concernée peut être spécifiée directement dans la requête de l'utilisateur ou calculée automatiquement par le système de RIS. Les requêtes dans les systèmes de RIS peuvent en effet avoir deux formes : une forme « contenu seulement », la requête est dans ce cas composée que de mots clés et une forme combinant la structure et le contenu.

Dans le cas où les documents ont des structures hétérogènes l'écriture d'une requête de type contenu et structure devient très difficile, car d'une part, l'utilisateur ne connaît pas forcément toutes les structures des documents et d'autre part il n'est pas possible d'exprimer la notion de synonymie structurelle dans aucun langage existant aujourd'hui.

C'est dans la perspective de s'affranchir de ces structures hétérogènes que se situent nos travaux. Notre objectif est de construire un moyen permettant de manipuler les structures « similaires » de manière transparente.

Problématique

Comme il n'existe aucun standard universel pour la représentation des données arbitraires sous XML, l'hétérogénéité des structures des documents est inévitable. Ainsi les structures (noms des balises et leurs agencements) utilisées varient à travers les sources de données. La connaissance de toutes ses structures ne peut être attendue de la part de l'utilisateur (à moins de le supposer omniscient), diverses questions se présentent alors lors de la recherche d'information dans de telles collections. Or, cette hétérogénéité de structures peut être seulement liée à des informations sémantiquement similaires mais codées dans des structures XML très variées :

- Variation linguistique, c'est-à-dire utilisation de différents noms de balises pour désigner un même concept dans les diverses sources d'information.
- Variation de la structuration (ou hiérarchisation) des balises, c'est-à-dire différence de leur agencement et leur nombre dans les diverses sources d'information.

Il est alors nécessaire de répondre aux questions suivantes :

- Quelle méthode utiliser pour établir les correspondances (matchings) entre les différentes structures ?
- Ces "matchings" doivent-ils se focaliser uniquement sur les noms des balises, ou bien faut-il considérer aussi la différence de structuration ou de hiérarchisation des balises ?

Ce sont là des questions ouvertes.

Les réponses à ces questions ont été abordées dans deux principaux domaines, à savoir :

- Le domaine des BDs à travers la tâche du « schema matching » pour l'intégration de différentes sources de données où diverses méthodes automatiques ont été élaborées.
Bien que le contexte des données issues des BDs soit différent de celui de la recherche d'information, il peut être intéressant de voir s'il est possible d'en tirer profit.

- Le domaine de la RI, notamment à travers la tâche hétérogène nouvellement introduite (à partir de 2004) dans la campagne INEX qui permet aux participants de présenter leurs travaux par rapport à cette problématique.

En général, les solutions proposées par les deux communautés peuvent être classées en deux groupes:

- Les approches se basant sur l'utilisation d'un lexique, un thésaurus ou une ontologie pour faire correspondre les conditions de structure exprimées dans la requête avec les types d'éléments effectivement présents dans la collection.
- D'autres approches visent à proposer un schéma de médiation dans lequel sera formulée la requête de utilisateur. Ainsi, soit c'est la requête qui sera transformée afin qu'elle s'adapte aux formats des documents interrogés, soit c'est les documents qui seront transformés dans le format médian pour leur appliquer enfin la requête.

Notre problématique se situe alors dans le cadre de la RI. Dans ce contexte, et pour réponde aux problématiques inhérentes aux questions précédentes, nous nous intéressons à la proposition d'une méthode permettant une gestion automatique de corpus de documents XML hétérogènes.

Contribution

Notre contribution dans le cadre de l'interrogation de corpus hétérogène comporte trois propositions qui couvrent les deux types d'approches proposées en RI pour cette problématique.

Dans le cadre l'utilisation d'une ressource sémantique (une ontologie), nous proposons deux approches :
1- La première s'intéresse à l'utilisation de l'ontologie pour permettre le matching de balises morphologiquement différentes mais qui désignent le même concept. La solution que nous proposons se base sur la construction et l'utilisation lors de l'interrogation d'un dictionnaire regroupant les balises sémantiquement équivalentes. L'approche comprend trois étapes principales : dans la première, une projection de chaque DTD du corpus sur l'ontologie permet l'extraction des concepts candidats pour chaque balise. Puis, un traitement de désambiguïsation permettra de choisir les concepts adéquats aux sens des balises telles qu'elles sont utilisées dans les documents de la collection. Une dernière étape consiste en la construction du dictionnaire des synonymes. Cela se fait en définissant une entrée pour chaque concept retenu dans l'étape précédente et où sera sauvegardée de plus, la liste des balises qui lui correspond dans la collection. Ainsi, un utilisateur pourra poser sa requête dans les termes d'une quelconque DTD de la collection, le système se chargera de réécrire la requête dans les termes des autres DTDs en consultant le dictionnaire pour d'éventuels synonymies.

2- La seconde approche d'utilisation d'ontologie est plus générale dans la mesure où l'on s'intéresse aux deux problèmes : de la variation linguistique et de la variation de la structuration des balises. Dans cette approche, nous faisons recours à l'ontologie pour concevoir une structure générique unifiant les différentes structures des documents de la

collection. L'approche comprend trois étapes : en premier lieu, on procède à la projection des DTDs sur l'ontologie pour construire les arbres conceptuels correspondants. Comme dans la première approche, un traitement de désambiguïsation est effectué pour sélectionner pour chaque balise le concept qui correspond le mieux à son sens dans son contexte. Puis, un éventuel groupement (ou classification) des arbres selon les domaines qu'ils décrivent est opéré en effectuant une comparaison des arbres conceptuels deux à deux. L'idée est que plus deux arbres présentent des concepts communs, plus ils se rapportent à un même domaine. Enfin, les éléments de chaque groupe construit précédemment seront représentés par rapport à un même référentiel. Il correspond au sous arbre minimal de l'ontologie qui contient tous ces arbres et qui servira d'interface pour l'interrogation de la classe de documents qu'il représente dans la collection.

3- Dans le cadre d'utilisation des techniques d'apprentissage automatique dans le domaine de la recherche d'information, et plus précisément la recherche d'information structurée, nous proposons une approche par classification probabiliste et arbre de dérivation pour la conversion de documents XML. Dans cette approche, on se propose de convertir les documents de structures hétérogènes dans un schéma désigné pour la médiation. Nous tentons de résoudre le problème général de conversion sans faire des restrictions sur les structures sources ou cibles de la conversion. L'approche proposée comprend trois phases : la première phase dite classification probabiliste, consiste à parcourir la séquence des feuilles du document et d'estimer une séquence d'étiquettes associée à ces feuilles parmi les éléments terminaux du schéma de médiation. Ceci est basé sur un modèle d'apprentissage combinant différents classifieurs probabilistes permettant ainsi de prendre en charge les caractéristiques des documents XML. Ces classifieurs seront entraînés sur un ensemble de documents exprimés à la fois dans le schéma d'origine et dans le schéma de médiation. Dans la seconde étape, on se propose d'estimer un arbre de dérivation pour la séquence d'étiquettes terminales obtenue dans la première étape. Nous partons du fait qu'une DTD XML peut être facilement réécrite sous forme de grammaire hors contexte probabiliste, et nous exploitons les propriétés de cette grammaire pour estimer cet arbre. La dernière phase consiste à combiner les résultats des phases précédentes pour réussir la conversion automatique des documents en respectant à la fois la sémantique et la structure des documents. La combinaison s'effectue en maximisant la probabilité jointe d'estimer une séquence d'étiquettes terminales et de dériver un arbre pour cette séquence.

Plan du mémoire

Ce mémoire est organisé en deux parties : la première présente le contexte général dans lequel se situent notre travail, à savoir la recherche d'information et plus précisément la recherche d'information dans des documents semi-structurés ; la seconde partie comprend

un état de l'art sur l'interrogation de corpus de documents comportant des structures hétérogènes.

L'objectif de la première partie, ***Recherche d'information et structure***, est de porter la lumière sur le domaine de la recherche d'information dans des documents plats, puis son extension pour embrasser la recherche dans des documents semi-structurés.

Le chapitre 1, ***La recherche d'information,*** présente les concepts de base de la recherche d'information. Nous commençons par retracer les notions clés liées à ce domaine, puis nous décrivons le processus fondamental de la RI. Nous exposons ensuite les principaux modèles qui sont à la base de la majorité des systèmes de RI. Nous présentons enfin les mesures et les collections de test développées pour permettre l'évaluation des systèmes de recherche.

Le chapitre 2, ***La recherche d'information structurée***, est consacré aux enjeux de la recherche d'information dans des documents semi-structurés principalement de type XML. Nous commençons par l'introduction de la notion de document structuré, nous nous intéressons plus particulièrement au format XML et les différentes technologies qui s'y rapportent. Par la suite, nous présentons les nouvelles problématiques issues de la recherche d'information dans ce type de documents et les solutions qui ont été proposées dans la littérature pour y faire face. Les principales adaptations, faites aux modèles de RI classiques pour pouvoir exploiter l'information supplémentaire de structure portée par les documents semi-structurés et permettre des requêtes portant sur le contenu et/ou la structure des documents, sont présentées. Enfin, de nouvelles mesures d'évaluation de la performance des systèmes de recherche d'information structurée sont exposées au sein de la présentation de la campagne d'évaluation qui en est la source, à savoir INEX.

La seconde partie du mémoire intitulée : ***Interrogation de corpus hétérogène***, présente un état de l'art des travaux de la littérature qui essayent de résoudre le problème de l'interrogation de corpus de documents XML de structures hétérogènes, puis le détail de notre contribution.

Le chapitre 3, ***interrogation de corpus hétérogène : un état de l'art***, est consacré à l'état de l'art sur l'interrogation de corpus hétérogène. Il introduit la problématique liée à l'interrogation dans ces corpus, puis passe en revue les différentes approches proposées dans la littérature pour permettre sa résolution. Il commence d'abord par les travaux issus principalement de la communauté BD, puis, ceux de la communauté de la RI dans le cadre de la tâche hétérogène d'INEX 2004.

Le dernier chapitre, ***contribution à l'interrogation de corpus hétérogène***, englobe nos propositions sur la gestion automatisée d'un corpus hétérogène pour faciliter son

interrogation aux utilisateurs. Nous commençons par deux approches exploitant toutes deux une ontologie pour réconcilier les différents schémas des documents en exploitant la sémantique portée par les balises XML. Tandis que la première approche construit un dictionnaire des balises synonymes et l'utilise pour étendre les éléments structurels d'une requête en ajoutant leurs synonymes dans les autres DTDs. La seconde approche, préconise plutôt d'utiliser une ontologie pour générer une structure générique englobant toutes les autres structures du corpus, et qui servira comme interface générique pour l'interrogation. La dernière approche proposée, approche par classification probabiliste et arbre de dérivation pour la conversion de documents XML, entre également dans le cadre d'une interrogation générique, mais elle essaye par contre, de convertir les documents de la collection vers un schéma désigné pour l'interrogation. Elle se base en cela sur des techniques issues de l'apprentissage automatique.

En **conclusion**, nous faisons une synthèse de nos contributions dans le cadre général de gestion automatisée d'un corpus hétérogène pour une interrogation générique dans le domaine de la recherche d'information structurée. Nous terminons par les perspectives d'évolution envisageables pour nos travaux.

Chapitre 1 :

La recherche d'information

1.1 Introduction

La croissance du volume de données textuelles, comme les livres et les articles dans les bibliothèques, a imposé de définir des mécanismes efficaces pour les localiser. Les premières techniques, comme l'indexation et l'utilisation des catégories de classification ont marqué la naissance de la "Recherche d'Information" comme discipline de recherche.

Avec l'apparition des premiers ordinateurs dans les années cinquante, naquit l'idée d'automatiser la recherche d'information dans les bibliothèques. D'énormes efforts ont été déployés depuis, pour développer des approches et des techniques permettant de retrouver l'information voulue effectivement et efficacement à partir de vastes collections de données textuelles.

Depuis les années 1990, notamment avec l'avènement d'Internet, la recherche d'information est devenue plus d'actualité et plus exigeante que jamais. En effet, le web représente incontestablement la plus grande source d'information disponible jusqu'à présent et qui ne cesse de croître. Un moteur de recherche populaire rapporte plus de huit (8) milliards de pages dans son index en juillet 2005 alors qu'elles étaient seulement 320 millions en 1997 et 3.3 milliards en septembre 2002. Le nombre d'utilisateurs quant à lui, est estimé aujourd'hui à plusieurs centaines de millions. Ces facteurs ont conduit à l'émergence de nouveaux défis pour les tâches de collecte et de gestion, du stockage et de la recherche efficace de l'information.

Dans ce chapitre, nous présentons les concepts de base de la recherche d'information, puis nous passons en revue les principaux modèles de RI. Pour ce dernier point, il faut noter qu'un large éventail d'initiatives de recherches en RI durant la deuxième moitié du siècle passé ont donné naissance à un très grand nombre de publications traitant de la recherche d'information. Même si chacune de ces publications apporte quelque chose de

nouveau, il est pratiquement impossible de les passer toutes en revue. Nous restreignons cet état de l'art donc aux modèles piliers de la RI.

1.2 Concepts de base de la recherche d'information

La Recherche d'Information (RI) ([Rijsbergen, 79], [Salton et al, 83], [Grossman et al, 98], [Baeza-Yates et al, 99]) est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la distribution de l'information. En bref, un système de recherche d'information (SRI) permet de sélectionner à partir d'une collection de documents, des informations pertinentes répondant à des besoins utilisateurs, exprimés sous forme de requêtes.
Plusieurs concepts clés s'articulent autour de cette définition :

Collection de documents : la collection de documents (ou fond documentaire, corpus) constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents. Nous utiliserons dans la suite les termes : corpus, collection ou fonds documentaires de manière indifférente.

Document : le document constitue l'information élémentaire d'une collection de documents. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document. Nous utiliserons dans la suite du document les termes information ou document pour désigner un granule documentaire.

Besoin en information : cette notion est souvent assimilée au besoin de l'utilisateur. Trois types de besoin utilisateur ont été définis par [Ingwersen, 92] :

- *Besoin vérificatif :* l'utilisateur cherche à vérifier le texte avec les données connues qu'il possède déjà. Il recherche donc une donnée particulière, et sait même souvent comment y accéder. La recherche d'un article sur Internet à partir d'une adresse connue serait un exemple d'un tel besoin. Un autre exemple serait de chercher la date de publication d'un ouvrage dont la référence est connue. Un besoin de type vérificatif est dit stable, c'est-à-dire qu'il ne change pas au cours de la recherche.
- *Besoin thématique connu :* l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et domaine connus. Un besoin de ce type peut être stable ou variable ; il est très possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche. Le besoin peut aussi s'exprimer de façon incomplète, c'est-à-dire que l'utilisateur n'énonce pas nécessairement tout ce qu'il sait dans sa requête mais seulement un sous-ensemble. C'est ce qu'on appelle dans la littérature le label.
- *Besoin thématique inconnu* : cette fois, l'utilisateur cherche de nouveaux concepts ou de nouvelles relations hors des sujets ou domaines qui lui sont familiers. Le besoin est intrinsèquement variable et est toujours exprimé de façon incomplète.

Requête : la requête constitue l'expression du besoin en information de l'utilisateur. Elle représente l'interface entre le SRI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.

Pertinence : Une information est dite pertinente si elle satisfait les besoins de l'utilisateur. C'est sur cette base que le système doit juger si un document doit être retourné à l'utilisateur comme réponse. Comme la notion de pertinence d'un document face à une requête ne dépend que de l'utilisateur elle reste difficile à automatiser.

1.3 Approche générale de la Recherche d'Information

Pour établir la mise en correspondance des besoins en information des utilisateurs d'une part et des informations contenues dans les fonds documentaires d'autre part, les systèmes de RI adoptent une démarche simple, couramment appelée : Processus en U de recherche d'information, ce processus est composé de trois fonctions principales (comme l'illustre la figure 1.1) qui sont :

- ***Processus de représentation ou d'indexation* :** Comme il n'est pas pratique de chercher l'information désirée par un parcours complet de tous les textes des collections, les documents et les requêtes se voient représentés par un ensemble de termes caractérisant leur contenus. C'est une opération fondamentale en RI, à l'issu de laquelle un descripteur du document ou de la requête, est construit. Un descripteur est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids pour différencier leur degré de représentativité.

- ***Processus de recherche ou d'appariement requête-document* :** Il représente le processus noyau d'un SRI. Il mesure un degré de correspondance (pertinence) d'un document vis-à-vis d'une requête Il est étroitement lié au modèle de représentation des documents et des requêtes. La problématique majeure des SRI est de retrouver les quelques dizaines ou milliers de documents pertinents parmi des millions de documents. Cet écart de cardinalité rend cette tâche encore plus difficile.

- ***Processus de reformulation de requêtes*** : Ce processus est optionnel. Quelques systèmes de RI tentent d'améliorer les performances des recherches, en offrant en plus des fonctions précédentes la possibilité de reformuler (reconstruire) la requête en fonction de ressources externes (thésaurus, ontologie) ou à partir de documents restitués par le système et jugés par l'utilisateur. Une des méthodes utilisées pour la reformulation de requêtes est la réinjection de pertinence, qui ajoute des termes issus des premiers documents restitués.

Dans ce qui suit seront décrites chacune de ces fonctions en détail.

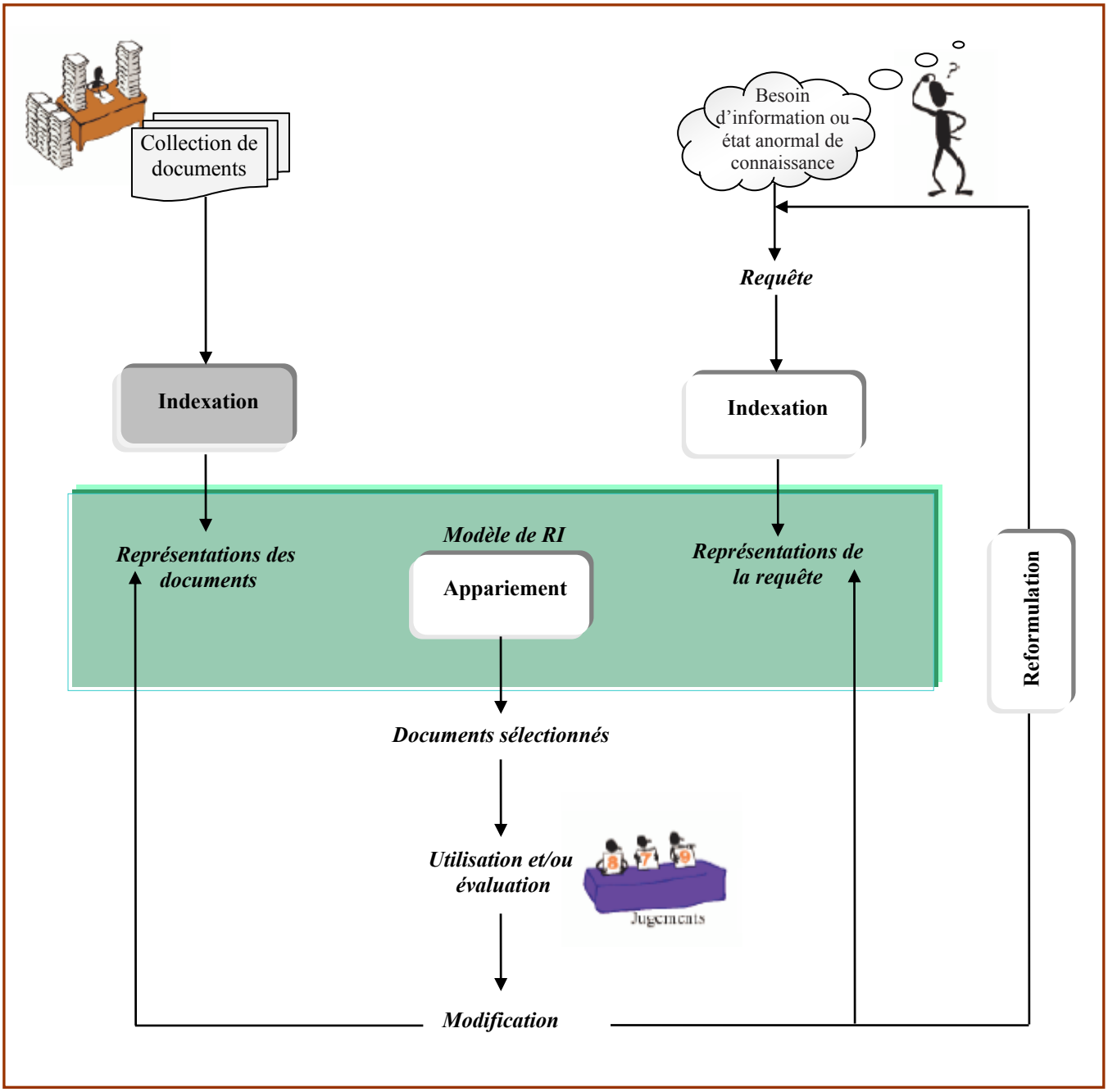

Figure -1.1- Approche générale de RI : Processus en U.

1.3.1 Le processus d'indexation

L'objectif principal de cette étape est de trouver une représentation des documents et des requêtes facile à exploiter par la machine dans la phase de recherche, étant donné que les documents (sous forme de texte libre) sont difficiles à exploiter tels quels lors des recherches. Cette représentation est souvent une liste de mots clés significatifs, que l'on nomme : descripteur du document (ou de la requête). L'indexation peut être :

- **Manuelle :** chaque document est analysé par un spécialiste du domaine ou un documentaliste, qui effectue le choix des mots en utilisant un vocabulaire contrôlé (liste hiérarchique, thésaurus, lexique,…). L'indexation manuelle fournit une terminologie standard pour indexer et rechercher des documents, assurant ainsi de meilleure pertinence dans les réponses apportées par le SRI [Nie et al, 99]. Cependant, elle présente plusieurs inconvénients, à savoir l'effort et le prix qu'elle exige (en temps et en nombres de personnes). De plus, un degré de subjectivité lié au facteur humain fait que pour un même document, des termes différents peuvent être sélectionnés par des indexeurs différents. Il peut même arriver qu'une personne, à des moments différents, indexe diversement le même document.

- **Semi-automatique :** les termes du document sont extraits par un processus automatique, puis l'indexeur intervient pour effectuer le choix final des termes significatifs et établir les relations entre les mots clés [Jacquemin et al, 02].

- **Automatique :** le processus d'indexation est complètement automatisé [Maron et al, 60]. L'indexation automatique peut se faire selon une méthode linguistique ou statistique. L'approche courante est plutôt statistique. Elle comprend un ensemble de traitements automatisés sur les documents, qui sont : l'extraction automatique des descripteurs, l'élimination des mots vides, la normalisation et la pondération des mots. Ces traitements, seront détaillés ci-dessous.

1.3.1.1 Extraction des mots simples

Cette opération consiste à extraire du document un ensemble de termes ou de mots simples par une analyse lexicale permettant d'identifier les termes en reconnaissant les espaces de séparation des mots, des caractères spéciaux, des chiffres, les ponctuations, etc.

1.3.1.2 Elimination des mots vides

La liste des mots simples extraite précédemment peut contenir des mots vides non significatifs tels que : les pronoms personnels, les prépositions, …, ou même des mots athématiques qui peuvent se retrouver dans n'importe quel document (par exemple des mots comme *contenir*, *appartenir*, etc). L'élimination de ces mots peut se faire en utilisant une liste dressée de mots vides (également appelée *anti-dictionnaire),* ou en écartant les mots dépassant un certain nombre d'occurrences dans la collection. Bien que ce traitement présente l'avantage de diminuer le nombre de termes d'indexation, il peut cependant réduire le taux de rappel (par exemple, en éliminant le mot a de vitamine a).

1.3.1.3 La normalisation (lemmatisation ou radicalisation)

C'est un processus morphologique permettant de regrouper les variantes d'un mot. En effet, on peut trouver dans un texte différentes formes d'un mot désignant le même sens. Il seront représentés par un seul mot désignant le concept véhiculé (ex : écologie, écologiste, écologique → écologie). On distingue quatre principaux types de lemmatisation : par analyse grammaticale en utilisant un dictionnaire (ex : Tree-tagger [tree-tagger]), par utilisation de règles de transformation de type condition action surtout pour l'anglais (ex : l'algorithme de Porter [Porter, 80]), par troncature des suffixes à X caractères (ex : la troncature à 7 caractères), ou encore par la méthode des n-grammes utilisée pour le chinois et très intéressante pour la radicalisation. Il reste cependant à mentionner que ces traitements peuvent induire certains inconvénients tels que la production de normalisation agressive, par exemple les mots university/universe, organization/organ, policy/police sont normalisés par l'algorithme de Porter. Ou l'oubli de quelques normalisations intéressantes, par exemple : matrices/matrix, Europe/European, machine/machinery ne sont pas normalisés. Il existe des techniques (analyse de corpus) pour réduire ces effets négatifs.

1.3.1.4 La pondération des termes

La pondération est une fonction fondamentale puisqu'elle traduit l'importance des termes dans les documents. Cette importance est souvent calculée sur la base d'aspects statistiques, si l'on ne veut pas citer certaines méthodes qui proposent d'introduire des éléments linguistiques dans l'indexation des documents. Les approches statistiques tirent leur origine de la loi de Zipf [Zipf, 49] et de la conjecture de Luhn [Luhn, 58].

➢ ***Loi de Zipf :*** selon Zipf [Zipf, 49] les mots dans les documents ne s'organisent pas de manière aléatoire mais suivent une loi inversement proportionnelle à leur rang. Le rang d'un mot est sa position dans la liste décroissante des fréquences des mots du corpus. Ainsi, la fréquence du second mot le plus fréquent dans le corpus est la moitié de celle du premier, la fréquence du troisième mot le plus fréquent, son tiers, etc. Formellement, cette loi s'exprime de la manière suivante :

$$Fréquence * Rang = Constante.$$

Zipf qui ne put expliquer cette curieuse loi, invoqua le principe du moindre effort : il considère qu'il est plus facile pour un auteur d'un document de répéter certains termes que d'en utiliser de nouveaux. C'est la conjecture de Luhn [Luhn, 58] qui a permis de définir comment utiliser la loi de Zipf pour sélectionner les termes représentatifs d'un document.

➢ ***La conjecture de Luhn:*** La loi de Zipf est à la base de la conjecture de Luhn. Cette dernière émet une hypothèse sur l'information contenue dans les termes (informativité)

d'un document schématisée par les deux courbes de fréquence et d'informativité dans la Figure 1.2. Elle considère que les mots qui reviennent souvent de rangs faibles (très fréquents), sont non significatifs car ils n'ont pas de pouvoir discriminant. Elle considère aussi ceux de rangs élevés (très rares), comme peu significatif. En effet ces mots sont rares et donc peu utilisés. Les termes significatifs sont ceux de rangs intermédiaires. Cette conjecture permet de diminuer la taille des index des documents. Deux seuils de fréquence (*seuil max* et *seuil min* dans la Figure 1.2), sont fixés pour éliminer les termes dont le contenu informatif est jugé faible. Seuls les termes entre ces deux seuils sont alors significatifs pour représenter les documents.

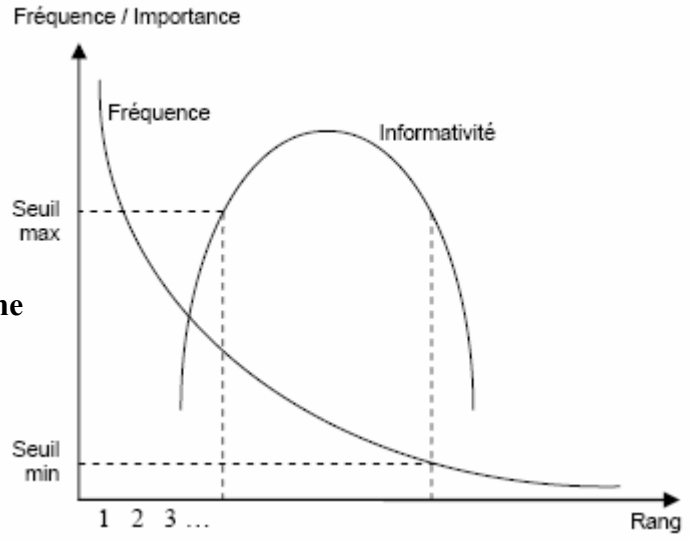

Figure -1.2-Informativité d'un terme en fonction de sa fréquence d'apparition dans un document.

A partir de ces constatations, différentes techniques de pondération ont vu le jour. Nous citons parmi d'autres : les approches se basant sur les mesures de *Tf-IDF* et les approches se basant sur le pouvoir discriminatoire d'un terme [Salton, 75].

- Pondération des mots TF-IDF : La plupart des méthodes de pondération est fondée sur la combinaison de deux facteurs. Un facteur *tf* de pondération locale, quantifiant la représentativité locale d'un terme dans le document, et un second facteur *idf*, de pondération globale, mesurant la représentativité globale du terme vis-à-vis de la collection des documents.

♦ *TF (Term Frequency)* : Cette mesure a été introduite pour tenir compte de la fréquence d'un terme dans un document. L'idée sous-jacente est que plus un terme est fréquent dans un document plus il est important dans sa description. Elle représente une "pondération locale" d'un terme dans un document. On trouve plusieurs variantes de cette mesure :

-fonction binaire : elle vaut 1 si la fréquence d'occurrence du terme dans le document est supérieure ou égale à 1, et 0 sinon;

- fonction logarithmique : combine tf_{ij} avec un logarithme : $\alpha + log\ (tf_{ij})$, *où* α est une constante, elle a pour but d'atténuer les effets de larges différences entre les fréquences d'occurrences des termes dans le document.

- fonction normalisée : permet de réduire les différences entre les valeurs associées aux termes du document. Elle est donnée par la formule suivante :

$$0.5 + 0.5 \frac{tf_{ij}}{\max_{t_i \in D_j}(tf_{ij})}$$

Où : $\max_{t_i \in D_j}(tf_{ij})$ est la plus grande valeur de tf_{ij} des termes du document D_j .

♦ *IDF (Inverse Document Frequency) :* ce facteur mesure la fréquence d'un terme dans toute la collection, c'est la "pondération globale". En effet, un terme fréquent dans la collection, a moins d'importance qu'un terme moins fréquent. Il est exprimé comme : $log\ (N / n_i)$, avec N est la taille (nombre de documents) de la collection et n_i le nombre de documents contenant le terme t_i.

La mesure *TF*IDF* est une bonne approximation de l'importance d'un terme dans un document, notamment dans des corpus de documents de tailles homogènes. Cette mesure a eu en revanche un succès très limité dans les corpus de tailles très variables, puisque elle ne tient pas compte d'un facteur très important du document : sa longueur. Le problème posé est que les termes appartenant aux documents longs apparaissent très fréquemment et l'emportent en poids sur les termes appartenant à des documents moins longs. Les documents longs auront alors plus de chance d'être sélectionnés. Pour pallier cet inconvénient, [Singhal et al., 95] et [Robertson et al., 97] proposent d'intégrer la taille des documents à la formule de pondération, on parle de facteur de normalisation. Robertson et Sparck-Jones proposent de normaliser la fonction de pondération de la façon suivante :

$$wd_{ij} \frac{tf_{ij}.(K_1+1)}{K_1.((1-b)+b.\frac{dl_i}{\Delta l})+tf_{ij}}$$

Où wd_{ij} est le poids du terme t_i dans le document D_j ; K_1 contrôle l'influence de la fréquence du terme t_i dans le document D_j, sa valeur optimale dépend de la longueur et de l'hétérogénéité des documents dans la collection; b est une constante appartenant à l'intervalle [0, 1] et contrôle l'effet de la longueur du document ; dl_j est la longueur du document D_j, et Δl est la longueur moyenne des documents dans la collection entière.

1.3.2 Le processus d'appariement document-requête

Le processus d'appariement document-requête permet de mesurer la pertinence d'un document vis-à-vis d'une requête, en calculant un score de correspondance entre la représentation de chaque document et celle de la requête. Ce score traduit un degré de pertinence système, que l'on essaye de rapprocher le plus possible du jugement de pertinence de l'utilisateur vis-à-vis du document. Cette valeur est calculée à partir d'une

probabilité ou une similarité appelée : ***RSV(Q, d)*** (*Retrieval Status Value*), où *Q* est une requête et *D* un document . Cette mesure tient compte des poids des termes.
Le processus d'appariement est étroitement lié au processus d'indexation et de pondération des termes des requêtes et des documents du corpus. Notons également que d'une façon générale, le modèle de représentation des documents et des requêtes ainsi que l'appariement document-requête, permettent de caractériser et d'identifier un modèle de recherche d'information. Les principaux travaux effectués dans ce domaine ont fait l'objet de modèles basés sur différentes approches. Nous les présentons dans la section 1.4.

1.3.3 Le processus de reformulation de requête

La requête est souvent une représentation approximative et imprécise du besoin de l'utilisateur. Par conséquent, parmi les documents qui lui sont retournés par le SRI, certains l'intéressent moins que d'autres. Afin de rapprocher au mieux la pertinence système de la pertinence utilisateur, quelques systèmes incorporent une étape supplémentaire de reformulation de la requête. Elle consiste à construire une nouvelle requête par ajout, suppression et re-pondération des termes pour mieux représenter les besoins de l'utilisateur. On distingue deux types de reformulation :

- **Reformulation indirecte :** également dite reformulation automatique de la requête. Dans ce cas, l'utilisateur n'intervient pas, l'extension de la requête est effectuée à partir d'une ressource externe qui peut être un thesaurus. Un thesaurus est une ressource comportant des termes reliés entre eux par différents types de relations (équivalence, association, hiérarchie), ou bien à partir de matrices d'association construites par analyse globale ou locale.

- **Reformulation directe :** ou reformulation manuelle de la requête. Il s'agit de la stratégie de reformulation de la requête la plus populaire. On la nomme communément *réinjection de la pertinence* ou *relevance feedback*. Dans un cycle de réinjection de pertinence, on présente à l'utilisateur une liste de documents sélectionnés par le système comme réponse à la requête initiale. Après les avoir examinés, l'utilisateur indique ceux qu'il considère pertinents. L'idée principale de la réinjection de pertinence est de sélectionner les termes importants appartenant aux documents jugés pertinents par l'utilisateur, et de renforcer l'importance de ces termes dans la nouvelle formulation de la requête. Cette méthode a pour double avantage une simplicité d'exécution pour l'utilisateur qui ne s'occupe pas des détails de la reformulation, et un meilleur contrôle du processus de recherche en augmentant le poids des termes importants et en diminuant celui des termes non pertinents. Il existe une variante très intéressante de ce processus, dite : la réinjection aveugle ou pseudo feedback, elle tente d'automatiser la partie manuelle du processus de

réinjection en supposant que les X (X=5 à 10) premiers documents sélectionnés par le système pertinents sans demander le jugement de l'utilisateur.

1.4 Les modèles de RI

L'objectif des modèles de RI est de fournir une formalisation du processus de RI qui soit la meilleur approximation possible de ce processus. Un modèle de RI se définit principalement, par sa modélisation de la mesure de la pertinence document-requête, mais aussi par sa représentation des documents et sa représentation des requêtes.
La littérature nous offre une panoplie de modèles, qui peuvent être regroupés en trois classes, à savoir :

- *les modèles booléens* : ces modèles trouvent leurs fondements théoriques dans la théorie des ensembles. On distingue le modèle booléen pur (boolean model), le modèle booléen étendu (extended boolean model) et le modèle basé sur les ensembles flous (fuzzy set model).
- *les modèles vectoriels* : basés sur l'algébrique, plus précisément le calcul vectoriel. Ils englobent le modèle vectoriel (vector model), le modèle vectoriel généralisé (generalized vector model), Latent Semantic Indexing (LSI) et le modèle connexionniste.
- *les modèles probabilistes* : se basent sur les probabilités. Ils comprennent le modèle probabiliste général, le modèle de réseau de document ou d'inférence (Document Network) et le modèle de langage.

Nous présentons dans la suite les principaux modèles issus de chacune de ces trois classes.

1.4.1 Les modèles booléens

Le modèle booléen pur

Modèle pionnier de la RI. Il est basé sur la théorie des ensembles. Un document est représenté par l'ensemble des termes qu'il contient. Une requête est représentée comme une formule logique où chaque atome correspond à un terme. Par exemple, la requête ((ciel OU terre) ET (NON air) renverra les documents contenant le mot "ciel" ou le mot "terre" et qui ne contiennent pas le mot "air".
La correspondance $RSV(D_l, Q)$, entre une requête Q et un document D_l est déterminée comme suit :

$RSV(D_l, q_i) = 1$ si $q_i \in D_l$; 0 sinon,
$RSV(D_l, q_i \square q_j) = 1$ si $RSV(D_l, q_i) = 1$ et $RSV(D_l, q_j)=1$; 0 sinon,
$RSV(D_l, q_i \square q_j) = 1$ si $RSV(D_l, q_i)= 1$ ou $RSV(D_l, q_j)=1$; 0 sinon,

RSV (D_l, $\neg q_i$) = *1 si RSV* (D_l, q_i) = *0 ; 0 sinon,*

Où : q_i est un terme de la requête Q_k.

Ainsi, le modèle booléen affirme que chaque document est soit pertinent soit non pertinent. Ceci est un inconvénient majeur de ce modèle. En effet, des documents pertinents, dont les représentations ne correspondent qu'approximativement à la requête, ne seront pas sélectionnés. D'autres inconvénients de ce modèle sont dus au fait que premièrement, dans ce modèle tous les termes ont la même importance, deuxièmement, ce modèle est incapable de trier les documents sélectionnés étant donnée que la pertinence ne peut avoir que deux valeurs (1 ou 0). Enfin, la formulation des requêtes n'est pas toujours évidente pour beaucoup d'utilisateurs.

Le modèle booléen étendu

Pour remédier aux inconvénients du modèle précédent, le modèle booléen étendu [Salton et al, 83] a été proposé. Il tient compte de l'importance des termes dans la représentation des documents et dans la requête. L'extension du modèle booléen se base sur une interprétation des conjonctions et des disjonctions. Ainsi, si l'on considère un document *d* et une requête *q* à deux termes, les scores de similarité document-requête seront calculés comme suit :

$$RSV(d, q_{ou}) = \sqrt{\frac{x^2 + y^2}{2}}$$
$$RSV(d, q_{et}) = 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}}$$

Où : *x* et *y* sont les poids des termes du document *d* appartenant à la requête *q*.

Le modèle *p-norme* généralise cette notion de distance euclidienne à *p-distances*, avec $1 \leq p \leq \infty$. Si *m* est le nombre de termes dans la requête, les fonctions de similarité seront alors :

$$RSV(d, q_{ou}) = \left(\frac{x_1^p + x_2^p + \ldots + x_m^p}{m}\right)^{\frac{1}{p}}$$
$$RSV(d, q_{et}) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + \ldots + (1-x_m)^p}{m}\right)^{\frac{1}{p}}$$

Si $p = 1$, on se ramène au modèle booléen, si $p = 2$, on retrouve les formules ci-dessus, enfin si $p = \infty$, on peut vérifier que l'on se ramène aux opérateurs flous :

$$RSV(d, q_{ou}) = max(x_i)$$
$$RSV(d, q_{et}) = min(x_i)$$

Ce modèle est puissant : il présente une combinaison des propriétés des modèles booléen et vectoriel. Cependant, il présente l'inconvénient de la complexité de ses calculs, ce qui a fait qu'il ne soit pas beaucoup utilisé.

Le modèle basé sur les ensembles flous

Une autre extension du modèle booléen est basée sur la théorie des ensembles flous proposée dans [Zadeh, 65]. Dans cette théorie un élément x appartient avec un certain degré à un ensemble A. Ce degré est généralement notée $\mu_A(x)$ et prend ses valeurs entre 0 et 1.Cette théorie a été appliquée dans le domaine de la RI pour modéliser les notions de vague et d'imprécision qui existent à différents niveaux du processus de la RI.

Dans ce modèle, un document est représenté comme un ensemble de termes pondérés comme suit :

$$D = \{(t_1,a_1)..., (t_i,a_i), ...\}$$

où : a_i est le degré d'appartenance du terme t_i au document D.
La correspondance RSV entre une requête Q et un document D est déterminée comme suit :

$$RSV (D_l, q_i \wedge q_j) = min(RSV (D_l, q_i),RSV (D_l, q_j)),$$

$$RSV (D_l, q_i \vee q_j) = max(RSV (D_l, q_i),RSV(D_l, q_j)),$$

$$RSV (D_l, \neg q_i) = 1- RSV (D_l, q_i),$$

Ces opérateurs ne conviennent pas parfaitement à un processus de recherche d'information pour la raison suivante : prenons la requête *a* ***et*** *b*, un document D appartenant à l'ensemble flou relatif à *a* avec $RSV(D,a)=0.9$ et à l'ensemble flou relatif à *b* avec $RSV(D,b)=0$ sera considéré de la même manière qu'un document D appartenant à l'ensemble flou relatif à *a* avec $RSV(D,a)=0.9$ et à l'ensemble flou relatif à *b* avec $RSV(D,b)=0.9$. Cela fait qu'ils ne sont pas adaptés au classement (ranking) des documents pertinents. Toutefois, les modèles de recherche d'information intègrent les ensembles flous pour réduire l'imperfection et traiter l'imprécision qui caractérise le processus d'indexation, pour contrôler l'imprécision de l'utilisateur dans sa requête et aussi pour traiter les réponses reflétant la pertinence partielle des documents par rapport aux requêtes.

1.4.2 Les modèles vectoriels

Le modèle vectoriel

Proposé par Salton dans le système SMART [Salton, 70], ce modèle représente les documents et les requêtes sous forme de vecteurs dans l'espace vectoriel engendré par tous les termes de la collection.
Chaque document est représenté par un vecteur (voir schéma de la Figure 1.3) :
$d_j= (w_{1j}, w_{2j}, ..., w_{Mj})$.

Chaque requête est représentée par un vecteur :
$q = (w_{1q}, w_{2q}, \ldots, w_{Mq})$.
avec : w poids du terme dans le document ou dans la requête. La pondération des composantes de la requête est soit la même que celle utilisée pour les documents, soit donnée par l'utilisateur lors de sa formulation.

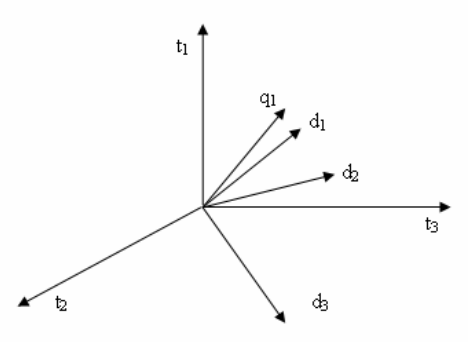

Figure -1.3- Le modèle vectoriel.

La pertinence dans le modèle vectoriel est interprétée comme une mesure de similarité vectorielle. Le mécanisme de recherche consiste donc à retrouver les vecteurs documents qui s'approchent le plus du vecteur requête. Dans la Figure 1.3, le vecteur documents d_1 est plus proche du vecteur requête q_1. Les principales mesures de similarité utilisées sont :

Produit scalaire :

$$RSV\ (Q, D_j) = \sum_{i=1}^{N} q_i . d_{ij}$$

Mesure de Jaccard :

$$RSV\ (Q, D_j) = \frac{\sum_{i=1}^{N} q_i . d_{ij}}{\sum_{i=1}^{N} {q_i}^2 + \sum_{i=1}^{N} {d_{ij}}^2 - \sum_{i=1}^{N} q_i . d_{ij}}$$

La mesure cosinus :

$$RSV\ (Q, D_j) = \frac{\sum_{i=1}^{N} q_i . d_{ij}}{\left(\sum {q_i}^2\right)^{1/2} . \left(\sum {d_{ij}}^2\right)^{1/2}}$$

L'aspect le plus intéressant de ces mesures est l'influence d'un terme isolé sur le score de recherche. Si un terme est présent à la fois dans la requête et le document, il contribue au score. S'il est présent uniquement dans l'un des deux, il diminue le score parce que la requête et le document se correspondent moins.

Les avantages d'un tel modèle sont nombreux : la pondération des termes améliore les résultats des recherches et la fonction d'appariement permet de trier les documents selon leur pertinence vis-à-vis de la requête. Théoriquement, le modèle vectoriel présente l'inconvénient de supposer l'indépendance entre les termes de l'index. Néanmoins, la pratique nous montre que la prise en compte globale de la dépendance des termes peut faire baisser les performances d'un système (puisque les dépendances sont généralement

locales). C'est pour toutes ces raisons qu'aujourd'hui le modèle vectoriel est le plus populaire en recherche d'information et le mieux accepté.

Le modèle connexionniste

C'est un autre type de modèle algébrique qui se base sur l'utilisation de réseaux neuronaux [Kwok, 89], [Boughanem et al, 92]. Un réseau est construit à partir des représentations initiales des documents et des informations associées (termes, mots clés, auteurs, …). Le fonctionnement du réseau se fait par propagation de signaux de la couche d'entrée vers la couche de sortie. Brièvement, l'activité d'un neurone est représentée par un nombre réel, plus celui-ci est grand, plus le neurone est actif. Certains neurones sont liés entre eux, plus cette liaison est forte, plus l'activité du neurone source augmente l'activité du neurone cible. La force de cette liaison sera notée : $w(n_1, n_2)$ où n_1 et n_2 sont deux neurones.

Cette représentation sous forme de réseau permet de mettre en évidence l'importance des différentes relations et associations qui peuvent exister entre les éléments d'un système documentaire. Des relations qui peuvent être :

- Relations entre les termes : synonymie, voisinage,…
- Relations entre les documents : similitude, référence,…
- Relations entre les termes et les documents : fréquences, poids,…

La figure 1.4, illustre un modèle de réseau neuronal pour la recherche d'information. Il est composé de trois couches :

- La couche d'entrée qui est formée de neurones représentant les termes de la requête.

- Puis, la couche interne qui est formée de neurones représentant les termes du vocabulaire *T* de la collection. Les neurones de la couche d'entrée sont liés aux termes correspondant dans cette couche.

- En fin, la couche de sortie, formée de neurones représentant l'ensemble des documents. Seuls les neurones liant les termes apparaissant dans un document *d* sont reliés au neurone représentant le document *d.*

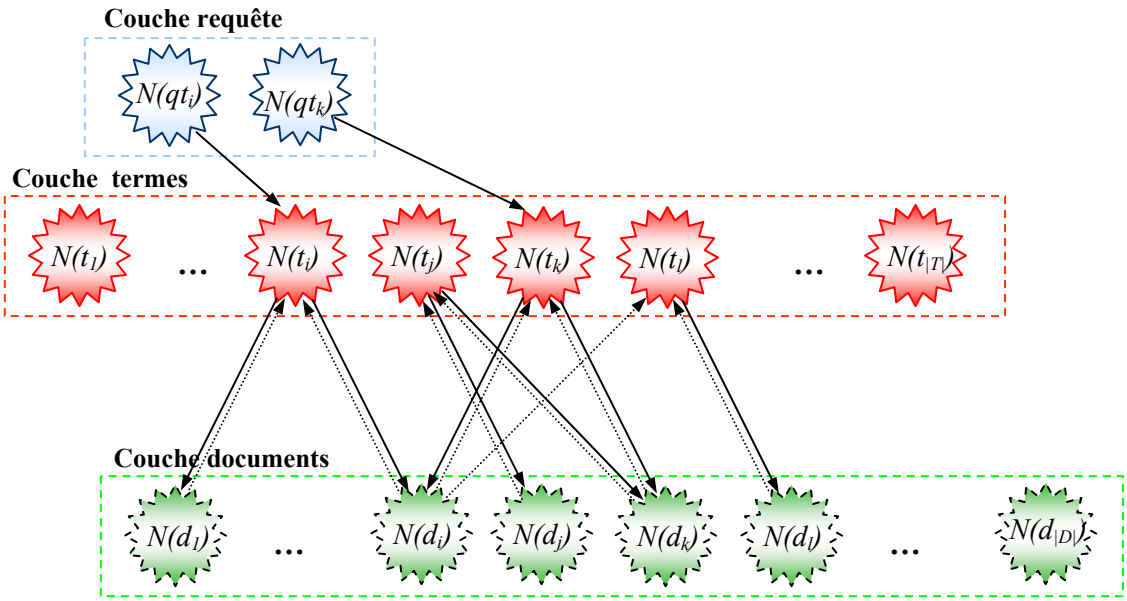

Figure -1.4- Réseau de neurones pour la recherche d'information.

Dans la suite, nous noterons *N(x)* le neurone associé à *x* où *x* peut être un terme de la question, du vocabulaire ou d'un document. Nous noterons $A_{N(x)}$ le degré d'activation du neurone *N(x)*. Afin de répondre à une requête *q*, les neurones de la couche d'entrée sont activés :

$$A_{N(qt)} = 1, t \in termes(q),$$

cette activation se propage dans la seconde couche (vocabulaire) :

$$A_{N(t)} = w(N(q_t), N(t))\, A_{N(qt)},$$

puis dans la troisième (documents) :

$$A_{N(d)} = \sum w(N(q_t), N(d))\, A_{N(qt)},$$

Le score d'un document aura pour valeur le degré d'activation du neurone correspondant. Il est possible de retrouver n'importe quel modèle vectoriel en fixant de manière appropriée les différents poids.

L'approche permet d'apporter plusieurs fonctionnalités souhaitées dans la RI. Elle va d'une simple comparaison de requêtes et de documents à des techniques associatives basées sur des associations de documents pour l'expansion de la réponse. Pour cela, il suffit de lier les documents aux termes qu'ils contiennent (liens en pointillés dans la figure 1.4). Deux nouvelles propagations vont avoir lieu : des documents vers le vocabulaire puis du vocabulaire vers les documents, ainsi des documents qui ne

contiennent pas des termes de la requête pourront être retrouvés. Sans oublier également son utilisation pour répondre à des problèmes de classification et de filtrage d'information.

1.4.3 Les modèles probabilistes

Le modèle probabiliste général

Un des premiers modèles de RI qui remonte aux années 1960 avec [Maron et al, 60]. Ce modèle tente d'estimer la probabilité qu'un document *d* soit pertinent pour une requête *q*, notée : *P(pert /d,q)*. Cette approche est justifiée dans le *Probability Ranking Principle* (PRP) formulé par Robertson [Robertson, 77].

On distingue deux classes de documents pour une requête : les pertinents (Pert) et les non pertinents (Npert), ainsi deux mesures de probabilité seront calculées :

- $P(Pert_q/ d)$: probabilité que *d* soit dans Pert.
- $P(Npert_q/ d)$: probabilité que *d* soit dans Npert.

Un document sera sélectionné si *P(pert /d) > P(Npert /d),* ce qui est équivalent à ordonner les documents par rapport à un degré de vraisemblance :

$$RSV(q, d) = P(Pert /d) / P(Npert /d). \qquad (1)$$

En appliquant la règle de Bayes sur la formule *(1),* cela donnera :

$$RSV(q, d) = P(d /Pert) / P(d /Npert). \qquad (2)$$

Pour estimer les probabilités *P(d /Pert)* et *P(d /Npert)* un document sera décomposé en un ensemble d'événements. Chaque événement dénotera la présence ou l'absence d'un terme dans un document, c'est le modèle de recherche indépendant "Binary Independence Retrievel" (BIR) qui suppose l'indépendance des termes des documents. La formule *(2)* devient :

$$RSV(q,d) = \sum_{i=1}^{t} log \frac{P(t_i / Pert)}{P(t_i / Npert)}$$

Avec :

- $P(t_i /Pert) = r /R$
- $P(t_i /NonPert) = n_i - r_i /N - R$

Où:

r_i : est le nombre de documents pertinents dans lesquels le terme t_i apparaît.

R : est le nombre de documents pertinents pour la requête.

$n_i – r_i$: est le nombre de documents non pertinents dans lesquels le terme t_i apparaît.

N : est le nombre total de documents dans la collection.

Parmi les applications du modèle probabiliste, citons le modèle 2-poisson développé par Robertson et Walker [Robertson et al, 94a] ou bien encore le moteur de recherche Okapi l'un des modèles les plus connu en RI avec la formule BM25 donnant les meilleurs performances en terme de rappel et de précision [Robertson et al, 94b], [Walker et al, 97].

La complexité apparente de la formule (par rapport à la modélisation vectorielle) et sa grande efficacité (ce modèle est l'un des plus performants actuellement) montre bien les avantages de la modélisation probabiliste. A partir d'une modélisation de la RI et d'un ensemble d'hypothèses, il est possible d'arriver à des modèles performants. Les hypothèses peuvent être révisées et donnent lieu à de nouvelles mesures de similarité. Un exemple flagrant est l'apparition ces dernières années des modèles de langage qui ont eux aussi des bases théoriques fortes.

Les réseaux bayésiens

Une modélisation du processus de RI peut être obtenue en utilisant le formalisme des réseaux bayésiens. Un réseau bayésien [Pearl, 88] est un graphe de dépendances acyclique et orienté dans lequel les nœuds représentent des variables aléatoires et les arcs traduisent des relations de causalité entre ces variables. En associant des probabilités initiales pour les racines du graphe, on calcule de proche en proche le degré de croyance associé à chacun des noeuds restants. Deux principaux modèles ont été développés : le modèle inférentiel bayésien par Turtle dans [Turtle, 91] puis dans [Callan et al, 92] et le réseau de croyance de Ribeiro [Ribeiro et al, 96].
Les réseaux inférentiels bayésiens [Turtle, 91] considèrent le problème de la recherche d'information d'un point de vue épistémologique. Ils associent des variables aléatoires avec les termes de l'index, les documents et les requêtes de l'utilisateur. Les termes de l'index et les documents sont représentés comme des noeuds. Une variable aléatoire associée avec un document d_j représente l'événement d'observer ce document. Les arcs sont dirigés du noeud document vers ses noeuds termes : ainsi, l'observation d'un document est la cause d'une augmentation de la valeur des variables associées avec ses termes d'index. La variable aléatoire associée à la requête de l'utilisateur modélise l'événement que la requête a été vérifiée. La valeur de ce noeud requête est une fonction des valeurs des noeuds associés aux termes de la requête. Ainsi, les arcs sont orientés des noeuds des termes de l'index vers le noeud de la requête.

La figure 1.5, issue de [Turtle et al, 90], illustre un réseau inférentiel bayésien simple de pertinence d'un document vis à vis d'une requête composée de trois termes. L'événement "la requête est accomplie" (Q=1) est réalisé si le sujet lié à un terme est vrai (T1=1, T2=1 ou T3=1), ou une combinaison de ces événements.

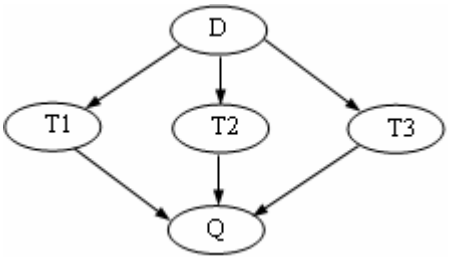

Figure -1.5- Un réseau inférentiel bayésien simple.

Les trois sujets sont inférés par l'événement "le document est pertinent" (D=1). Par l'enchaînement de règles de probabilités, la probabilité jointe des autres noeuds du graphe est :

$P(D, T1, T2, T3, Q) = P(D)\, P(T1|D)\, P(T2|D, T1)\, P(T3|D, T1, T2)\, P(Q|D, T1, T2, T3)$

La direction des arcs indiquant les relations de dépendance entre les variables aléatoires, l'équation devient :

$$P(D, T1, T2, T3, Q) = P(D)P(T1|D)P(T2|D)(T3|D)P(Q|T1, T2, T3)$$

La probabilité de réalisation de la requête $P(Q = 1|D = 1)$ peut être utilisée comme score d'ordonnancement des documents :

$$P(Q = 1|D = 1) = \frac{P(Q = 1, D = 1)}{P(D = 1)}$$

$$= \frac{\sum P(D = 1, T1 = t_1, T2 = t_2, T3 = t_3, Q = 1)}{P(D = 1)}$$

Le modèle nécessite la connaissance de $P(D = [0|1])$, $P(Ti = [0|1]|D = [0|1])$ et $P(Q = [0|1]|\ (T1, T2, \ldots, Tn) \in \{0, 1\}^n)$, cette dernière étant la plus difficile à trouver car le nombre de probabilités à spécifier augmente exponentiellement avec le nombre de termes de la requête. Pour résoudre ce problème, Turtle [Turtle, 91] a identifié quatre formes canoniques de $P(Q|T1, T2, \ldots Tn)$: *and, or , sum* et *wsum*.

Le modèle inférentiel bayésien a été mis en oeuvre dans le système Inquery [Allan et al, 98]. Le cadre probabiliste dans lequel se situe Inquery peut être utilisé pour formuler des requêtes simples basées sur des mots clés, des requêtes booléennes, des requêtes basées sur des phrases ou bien une combinaison des trois types. Pour ce faire, Inquery propose des opérateurs de moyenne et de moyenne pondérée, des opérateurs booléens probabilistes ou stricts (on conserve alors les probabilités), des opérateurs de proximité et de synonymie. Une procédure d'analyse de la requête permet de générer une forme inférentielle prête à être évaluée. Inquery propose également une expansion de requête. La figure 1.6 représente un réseau bayésien utilisé par Inquery.

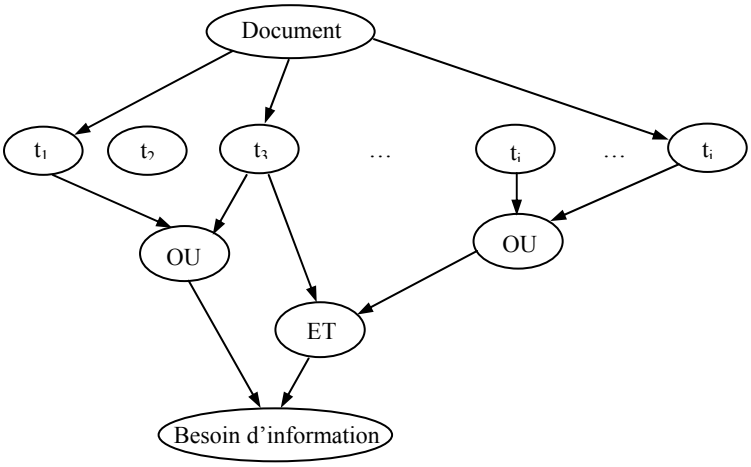

Figure -1.6- Un réseau bayésien utilisé par INQUERY.

Basés sur les réseaux inférentiels bayésiens, les "belief networks" ont été introduits en 1996 par Ribeiro-Neto et Muntz [Ribeiro et al, 96]. Ils sont aussi basés sur une interprétation épistémologique des probabilités, mais travaillent dans un espace différent. En conséquence, on obtient une topologie de réseau différente, qui permet la séparation entre l'espace des documents et l'espace des requêtes. La figure 1.7 représente un réseau bayésien proposé par Ribeiro.

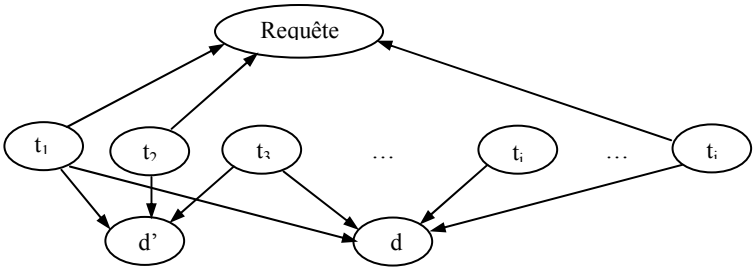

Figure -1.7- Un réseau bayésien proposé par Ribeiro.

On peut ainsi combiner des sources distinctes d'évidence (requêtes passées, cycles de feedback précédents, formulations distinctes de requêtes), ce qui permet d'augmenter les performances du système (c'est à dire augmenter la qualité de la liste ordonnée de documents renvoyée par le système).

L'inconvénient principal des réseaux bayésiens reste le calcul des probabilités, qui demande un temps exponentiel au nombre de termes dans la requête même si l'introduction des quatre formes canoniques dans [Turtle, 91] résout partiellement le problème.

Les modèles de langages

Les modèles de langues (ou de langages) tentent de modéliser l'agencement de mots dans une langue en estimant la probabilité de distribution d'une séquence de mots.
Ponte et Croft [Ponte et al, 98] ont été les premiers à proposer leur utilisation en RI. Pour cela, un document sera vu comme une suite de mots, généré par son propre modèle de langue. L'idée est alors, de mesurer la probabilité qu'une requête Q ait été générée par le modèle de langage M_d d'un document d, qui sera considérée comme le score de pertinence du document vis-à-vis de la requête, soit : RSV$(Q, d) = P(Q/M_d)$.
Suivant la complexité du modèle, l'estimation de cette probabilité sera plus ou moins complexe. En règle générale, l'indépendance des termes de la requête est supposée (modèle unigramme), le modèle de langage du document est estimé par la technique de l'estimateur du maximum de vraisemblance (MLE) :

$$P(Q / M_d) = \prod_{t \in Q} P(t / M_d)$$

$$= \prod_{t \in Q} \frac{tf(t,d)}{dl_d}$$

où : - $tf\ (t, d)$ est la fréquence du terme t dans le document d;
- dl_d est le nombre total de termes dans le document d;

Cette formule présente cependant un problème : lorsque un document ne contient pas un ou plusieurs termes de la requête donnant une probabilité nulle : $p\ (t\ /M_d) = 0$, il y lieu alors d'assigner des probabilités différentes de zéro à de tels mots.

Le modèle mixte (mixture modèle) [Song et al, 99] apporte la solution en combinant le modèle de langage du document et le modèle de langage de la collection. Ce dernier sera utilisé comme un modèle de référence pour les mots non observés dans le document :

$$P(Q\ /d) = \prod_{t \in Q} ((1-\lambda)P(t\ /M_c) + \lambda P(t\ /M_d)).$$

Où la valeur optimale du paramètre λ est déterminée empiriquement
Avec :

$$P(t / M_c) = \frac{total\_tf_t}{total\_tf\_col}$$

et où : - $total\_tf_t$ est la fréquence du terme t dans la collection ;
- $total\_tf\_col$ est le nombre total de termes dans la collection.

1.4.4 Conclusion

De nombreux modèles de recherche d'information existent à l'heure actuelle, et les supports théoriques sont mieux définis et compris aujourd'hui. Un des moteurs de l'évolution des différentes théories et modèles est sans conteste le cadre de campagnes d'évaluation des systèmes de RI. Ces campagnes sont basées sur des mesures de performance relativement simples issues des recherches sur l'évaluation des systèmes de RI que nous présentons dans la section suivante.

1.5 Evaluation de la performance des systèmes de RI

L'évaluation de la performance des systèmes de RI est vite apparue comme un problème essentiel en RI. La démarche d'évaluation adoptée est expérimentale plutôt qu'analytique. En effet, plusieurs facteurs : pertinence, distribution des termes, etc., sont difficiles à formaliser mathématiquement. L'évaluation des systèmes de RI ne se fait pas d'une façon absolue. Elle se fait plutôt par comparaison des systèmes de RI entre eux sur de mêmes bases documentaires assurées par de nombreuses campagnes d'évaluation. Une première évaluation expérimentale est lancée dès les années 1960 par Cleverdon dans le cadre du projet Cranfiel [Cleverdon, 67] qui a eu pour objectif de construire des collections de tests et d'évaluer les systèmes sur cette base.

Les suggestions de mesures et de techniques d'évaluation des systèmes de RI se sont multipliées des lors, parmi elles on peut citer :

- Le temps de réponse acceptable : un SRI doit pouvoir fournir à l'utilisateur les documents correspondants à sa demande dans des temps très courts.

- La présentation claire des résultats avec facilité d'utilisation : capacité du système à comprendre les besoins de l'utilisateur et à mettre en valeur les documents correspondants à ceux-ci. Ceci est lié à l'interface avec l'utilisateur.

- Le rang du premier document pertinent : cette mesure a été proposée pour prendre en compte la satisfaction de l'utilisateur qui chercherait un seul document pertinent (comme c'est éventuellement le cas pour les moteurs de recherche sur Internet).

- La longueur de recherche : elle est égale au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre *n* de documents pertinents.

- La capacité du système à sélectionner tous les documents pertinents et à rejeter tous les documents non pertinents : qui est considérée comme la plus importante du fait qu'elle traduit l'objectif premier des systèmes de RI.

La performance des systèmes de RI est évaluée à partir de la pertinence des documents renvoyés. Cette notion de pertinence est ambiguë. En effet, on peut parler de pertinence objective, c'est à dire une pertinence calculée à partir des résultats du SRI, mais aussi de pertinence subjective : un document peut être jugé pertinent à une requête par un utilisateur et pas par un autre. De même, la pertinence d'un document dépend des connaissances de l'utilisateur sur le sujet, et peut affecter la pertinence des documents examinés par la suite. C'est pour ces raisons, qu'un certain nombre d'hypothèses sur la façon dont s'effectue la tâche de RI ont été introduites afin de permettre de formuler directement les mesures :

- *Présentation* : Les documents sont ordonnés par score décroissante ; toutefois, on peut supposer dans certaines mesures que plusieurs documents peuvent avoir un même score (et donc associer une probabilité sur le rang des documents).

- *Ordre du parcours* : L'utilisateur parcourt la liste des documents en partant du premier jusqu'au dernier présenté. Il ne se décourage jamais, et ne procède jamais de façon aléatoire.

- *Jugement absolu* : Un document reste pertinent même s'il contient exactement la même information qu'un autre document déjà présenté à l'utilisateur.

- *Non additivité* : Deux documents non pertinents ne pourront jamais former une unité d'information pertinente (même s'ils se complètent).

- *Pertinence binaire* : Un jugement de pertinence doit pouvoir se ramener *au mieux* à un nombre réel borné.

Nous présentons dans la suite les mesures les plus courantes dont les plus classiques "Rappel" et "Précision", ainsi qu'un exemple de campagne d'évaluation.

1.5.1 Les mesures de Rappel/Précision

Elles permettent de calculer la capacité du système à retourner des documents pertinents et de rejeter les non pertinents. Pour ce faire, on considère |A| le nombre de documents sélectionnés par un système de RI pour une requête donnée. On considère de plus, le nombre |R| des documents pertinents dans la collection pour cette requête et |Ra| le nombre des documents pertinents renvoyés par le système (figure 1.8).

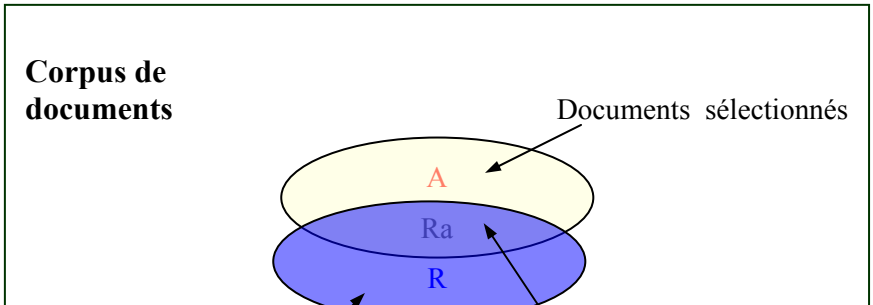

Figure -1.8- Précision et Rappel.

On peut alors définir les mesures comme suit :

- *La précision :* mesure la capacité du système à rejeter tous les documents non pertinents pour une requête. Elle est donnée par le rapport entre les documents sélectionnés pertinents et l'ensemble des documents sélectionnés :

$$Précision = \frac{|Ra|}{|A|}$$

- *Le rappel :* mesure la capacité du système à renvoyer tous les documents pertinents pour une requête. Il est donné par le rapport entre les documents pertinents sélectionnés et l'ensemble des documents pertinents pour la requête :

$$Rappel = \frac{|Ra|}{|R|}$$

1.5.2 Courbe de Rappel /Précision

La précision mesurée indépendamment du rappel (et inversement) est peu significative. En pratique, les valeurs du rappel et de précision sont conjointement calculées à chaque document restitué pour les *i* premiers documents dans la liste des réponses du système. Le tableau 1.1 donne un exemple de calcul de précision et de rappel pour les 10 premiers documents renvoyés pour deux requêtes R1 et R2 pour lesquelles la collection contient respectivement 6 et 3 documents pertinents. La figure 1.9 représente la courbe de rappel / précision qui en résulte.

	R1			R2		
Rang du document	Pertinent	Rappel	Précision	Pertinent	Rappel	Précision

renvoyé						
1	x	0.17	1	x	0.25	1
2	x	0.33	1		0.25	0.50
3		0.33	0.67	x	0.50	0.67
4	x	0.5	0.75		0.50	0.50
5		0.5	0.60		0.50	0.40
6	x	0.67	0.67		0.50	0.33
7		0.67	0.57		0.50	0.29
8		0.67	0.50		0.50	0.25
9	x	0.83	0.56		0.50	0.22
10	x	1	0.6	x	0.75	0.30

Tableau -1.1- Calcul de rappel et de précision pour deux requêtes R1 et R2.

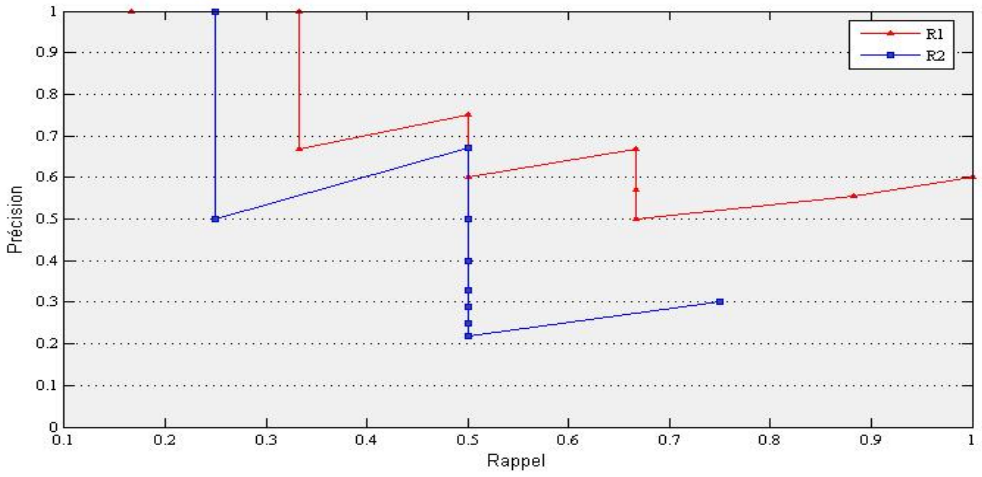

Figure-1.9- Courbes de Rappel/ Précision des requêtes R1 et R2.

On voit facilement que pour un même niveau de rappel on a plusieurs précision possibles cela est du au fait que le niveau de rappel reste constant alors que la précision baisse jusqu'au prochain document pertinent retrouvé. Afin d'obtenir des courbes plus aisées à lire, on ne présente généralement que la précision calculée à chaque document pertinent de la liste, la valeur de précision la plus haute sera choisie, comme l'illustre la figure 1.10.
On peut remarquer de plus, que la courbe rouge n'est pas décroissante. Il arrive souvent qu'on applique l'interpolation sur la courbe de chaque requête. L'interpolation vise à créer une courbe descendante. Le principe est le suivant :

Soit i et j deux points de rappel avec i<j,
Si la précision au point i < à la précision au point j,
Alors on affecte la précision du point j à celle du point i.

Concrètement, cela signifie qu'on remplit un creux de la courbe par une ligne horizontale, comme l'illustre la Figure 1.10. L'idée derrière l'interpolation est que les creux de la courbe ne représentent pas vraiment la performance du système. S'il existe un point à un rappel et

une précision plus élevée, on peut toujours donner plus de documents dans la réponse pour augmenter la performance. Le creux est donc surmontable.

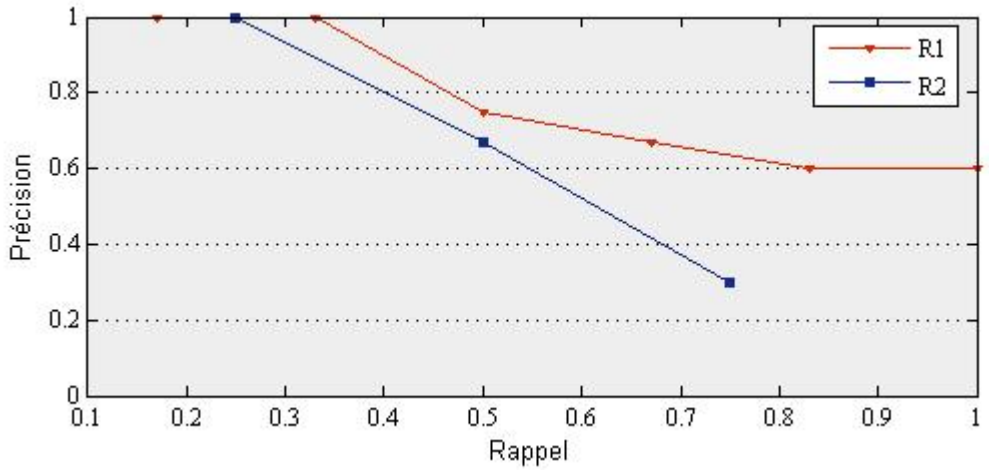

Figure -1.10- Courbes de Rappel/ Précision simplifiée des requêtes R1 et R2.

Pour avoir une évaluation de la performance du système sur toutes les requêtes et pouvoir effectuer une comparaison avec d'autres systèmes, on calcule la précision moyenne. Cette précision représente la moyenne des valeurs de précision à chaque niveau de rappel. Pour ce faire, il faudra unifier les niveaux de rappel pour l'ensemble des requêtes. Onze (11) points standard de rappel sont généralement retenus de 0 à 1 à pas de 0.1. Les valeurs de précisions non obtenues à partir du rappel sont alors calculées par interpolation linéaire. La figure 1.11 représente les courbes de rappel/précision interpolées des requêtes R1 et R2 et la courbe de la précision moyenne des ces requêtes.

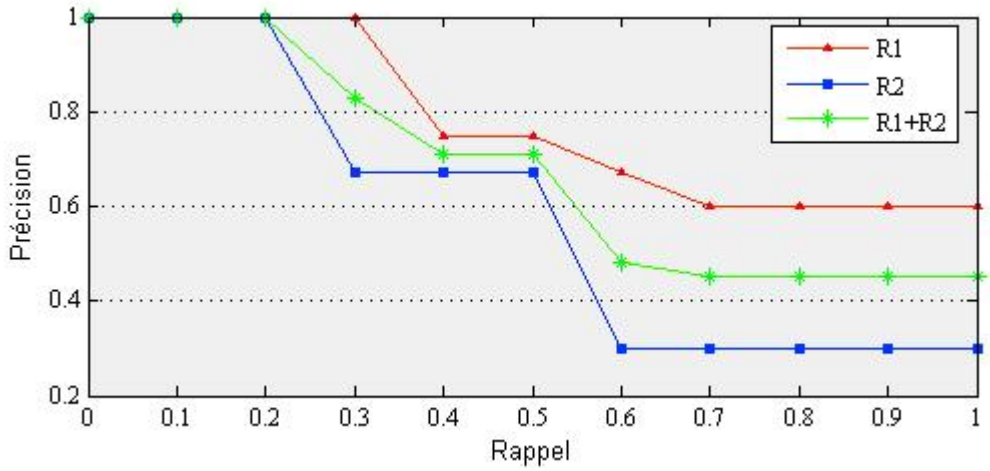

Figure -1.11- Courbes de Rappel/ Précision interpolées des requêtes R1 et R2 et de leur précision moyenne.

Une autre mesure communément utilisée en RI est la précision à x documents. Elle est souvent reliée à ce que l'on appelle la précision exacte ou la R- Précision. Elle représente la précision obtenue à l'endroit où elle vaut le rappel : Si la requête admet x documents

pertinents, la précision exacte est celle calculée pour les x premiers documents de la liste ordonnée des documents renvoyés par le système.

Dans le cas d'un système idéal, les valeurs de précision et de rappel seront égales à 1, c'est-à-dire que tous les documents pertinents et rien que ceux-ci, sont sélectionnés, on aurait alors une droite. En pratique, les valeurs de rappel et de précision évoluent inversement ce qui fait que la courbe Précision/rappel est décroissante. Ainsi, plus la courbe d'un système est élevée, plus il est performant.

1.5.3 Les mesures combinées

L'inconvénient principal des mesures de rappel et de précision est qu'elles représentent des aspects différents de l'ensemble des documents sélectionnés. Des mesures agrégatives ont donc été proposées pour synthétiser l'information contenue dans ces deux mesures. Nous présentons les plus importantes :

- **Moyenne harmonique**

Une moyenne harmonique combine le rappel et la précision en un nombre compris entre 0 et 1, elle est définit dans [Shaw et al, 97] :

$$\text{harmonique} = \frac{2}{\frac{1}{\text{rappel}} + \frac{1}{\text{précision}}}$$

Elle vaut 0 lorsque aucun document pertinent n'est sélectionné et 1 lorsque tous les documents sélectionnés sont pertinents. La valeur de la moyenne harmonique est élevée lorsque les valeurs de précision et de rappel sont élevées, garantissant ainsi un compromis entre ces deus mesures.

- **La E-mesure**

C'est une extension de la mesure harmonique, proposée par [Rijsbergen, 79]. Elle permet à l'utilisateur de spécifier s'il est plus intéressé par le rappel ou par la précision :

$$\text{E-mesure}(b) = 1 - \frac{1 + b^2}{\frac{b^2}{\text{rappel}} + \frac{1}{\text{précision}}}$$

Le paramètre *b* désigne l'importance relative de la précision et du rappel. Si $b>1$, on privilégie la précision et si $b<1$ on privilégie le rappel.

Malgré leur caractère hautement subjectif, les mesures de rappel et de précision ont permis aux modèles de RI de se développer depuis les années 1960, les courbes de

précision/rappel et la précision moyenne se sont imposées et sont utilisées dans les conférences d'évaluation.

D'autres mesures moins conventionnelles peuvent aussi servir à évaluer la performance d'un système de RI. Mizzaro dans [Mizzaro, 97], fait une étude complète des différentes mesures d'évaluation utilisées en RI.

1.5.4 Collections de test- Un exemple : TREC

Les évaluations des systèmes de RI se font sur des collections des tests qui offrent la possibilité de les comparer dans un cadre unifié. Une collection de test est composée d'un ensemble de documents, d'un ensemble de requêtes-tests, avec pour chaque requête la liste des documents jugés pertinents pour cette requête ou communément appelés la "réponse idéale" associée à cette requête.

De nombreux projets basés sur les corpus de tests se multiplient depuis les années 1970, on peut citer la collection CACM, la collection CISI ou encore la campagne CLEF. La campagne la plus connue est sans doute TREC (Text REtrieval Conference) [TREC] organisée annuellement depuis 1992 par la NISI[1] et la DARPA[2]. Elle a pour but de fournir un environnement de compétition /collaboration entre les groupes de recherche en RI à travers le monde, tout en fournissant des procédures uniformes et appropriées pour l'évaluation.

De TREC 1 à TREC 6, les recherches étaient centrées sur deux tâches principales : la tâche de routage et la tâche adhoc. La tâche ad hoc est constituée d'un ensemble de nouvelles requêtes qui sont lancées sur une collection de documents fixés, et la tâche de routage est composée d'un ensemble de requêtes fixes lancées sur une collection de documents en évolution perpétuelle.

Autour de ces tâches, bon nombre de pistes ont été explorées, et de nouvelles tâches sont apparues. On peut par exemple citer des évaluations de recherche de documents écrits dans une autre langue que l'anglais (exemple : espagnol, français, ou encore chinois, arabe) (à partir de 1994), des évaluations de recherche à travers des langages multiples, des évaluations sur de très grands corpus (tâche Terabyte en 2004), ou des évaluations portant sur des aspects plus diversifiés (la tâche interactive depuis 1994, la tâche QA (question-réponse) en 1999. . .), ou encore des évaluations de recherche sur des vidéos (depuis 2001) ou des documents Web (depuis 1999).

[1] National Institue of Standards and Technology
[2] Defense Advanced Research Project Agency

1.6 Conclusion : Vers la Recherche d'information Structurée

Ce chapitre a donné un aperçu sur les approches et modèles fondamentaux utilisés en recherche d'information classique.

Le défi des chercheurs en RI était de résoudre les différents problèmes inhérents à ce domaine, à savoir la création d'une base documentaire, son indexation et le choix du modèle utilisé pour la recherche d'une part, le choix d'un langage de requête facilitant l'expression du besoin utilisateur et le retour éventuel de pertinence d'autre part. Plusieurs techniques en été proposées. Les évaluations des performances de ces techniques ont aidé à leurs évolutions en permettant de justifier bon nombre de propositions qu'elles soient théoriques ou empiriques.

Néanmoins, les systèmes de RI que nous avons décrits jusqu'ici considèrent les documents de manière plate. En effet, même si les documents comportent des structures seul le contenu sémantique des documents, c'est à dire leur texte, est exploité.

Aujourd'hui, Internet et plus particulièrement le format XML gagne vraiment du terrain en favorisant la prolifération des documents structurés. Les systèmes de RI s'intéressent actuellement à cette évolution pour bien des raisons. Le chapitre suivant portera sur les motivations, les influences de la communauté bases de données, et les approches issues de la RI pour permettre une utilisation effective de cette information supplémentaire qu'est la structure.

Chapitre 2 :

La Recherche d'Information Structurée

2.1 Introduction

Le format de documents structurés XML est devenu un standard populaire, adopté par de nombreuses applications dans le cadre de la représentation, du stockage et de l'échange de documents et de données, en particulier sur le Web. XML étant un méta-langage, les applications qui l'adoptent définissent et utilisent en réalité des langages fondés sur XML. Ces langages adoptent la syntaxe, la structure et les mécanismes définis par XML, apportant de leur côté un vocabulaire (fonction du domaine auquel est associé le langage) qui permettra de décrire le contenu des documents de manière exploitable par les applications.

Ce n'est que très récemment que l'utilisation de l'information apportée par la structure a commencé dans le domaine du traitement automatique des données textuelles. Du point de vue des systèmes de Recherche d'Information, l'accès à ce type de documents soulève de nouvelles problématiques liées à la co-existence de l'information structurelle et de l'information de contenu. La prise en compte de la dimension structurelle devrait permettre de mieux répondre aux différents besoins des utilisateurs.

La Recherche d'Information Structurée (RIS) est un domaine complexe. Dans la suite, nous définissons tout d'abord ce qu'est un document structuré (en particulier un document XML), et les technologies qui s'y rapportent. Puis nous décrivons les dernières évolutions et idées que l'apparition du XML a apportées à la notion d'unité d'information et les différentes problématiques qu'elle a soulevées. Pour représenter le besoin d'information d'un utilisateur, de nombreux langages de requêtes ont été proposés. Nous donnons un aperçu de ces langages et des caractéristiques que doit avoir un langage d'interrogation XML. Une fois décrits les différents aspects de la RIS, nous passons en revue les différents modèles permettant d'effectuer de telles recherches. Enfin, comme pour la RI, nous terminons par la présentation des mesures d'évaluation proposées dans le cadre d'INEX, afin de pouvoir estimer la performance des différents systèmes et de stimuler la recherche dans ce domaine.

2.2 Une évolution des corpus

Jusqu'au milieu des années 1980, la notion d'unité d'information était étroitement liée à celle de document. Ainsi un système de RI s'occupait uniquement du traitement des documents dans leur globalité. Mais à présent, le Web a complètement révolutionné la notion de document en proposant d'une part, d'intégrer dans un même document des informations de natures différentes (image, texte, son etc…) sous diverses formes (texte en gras, en couleur, tableaux etc…), d'autre part en proposant de relier les documents par l'intermédiaire d'hyperliens. Puis les formats semi-structurés comme le XML ont commencé à se répandre. Ils sont aujourd'hui adoptés par la majorité des éditeurs et sont devenus un standard de facto.

Cette évolution des documents a entraîné la nécessité d'adapter les modèles de recherche d'information pour qu'ils prennent en compte ces nouveaux formats de documents.

Dans la suite seront brièvement présentés les principaux concepts liés aux documents XML.

2.2.1 Les documents semi-structurés et le XML

Parallèlement aux documents de type HTML, de nouveaux documents appelés documents semi-structurés sont apparus conjointement dans les domaines de la RI ou du document numérique et dans le domaine des Bases de Données (BD). Ce type de document représente un compromis entre les données fortement structurées issues de la communauté BD (données relationnelles par exemple) et les données faiblement structurées issues des communautés document numérique et RI (documents plats, images etc…).

Les modèles de documents semi-structurés représentent le contenu sous forme d'éléments typés organisés en une structure hiérarchique qui est principalement un arbre même si dans certain cas elle décrit aussi un graphe avec les attributs ID/IDREF. La structure du document est donc un ensemble de nœuds connectés par des relations hiérarchiques père-fils. Il est possible d'ajouter des informations au niveau de chaque élément sous forme d'attributs. Par exemple, le document de type article est présenté dans la figure 2.1 sous forme d'arbre. Un tel arbre est composé d'une racine *article,* de noeuds internes représentant les éléments ou les attributs, et de noeuds feuilles contenant les valeurs d'éléments ou d'attributs. Dans la suite de ce mémoire, nous représentons les documents XML sous cette forme, et utilisons indifféremment les termes éléments ou noeuds pour désigner des sous-arbres de documents XML.

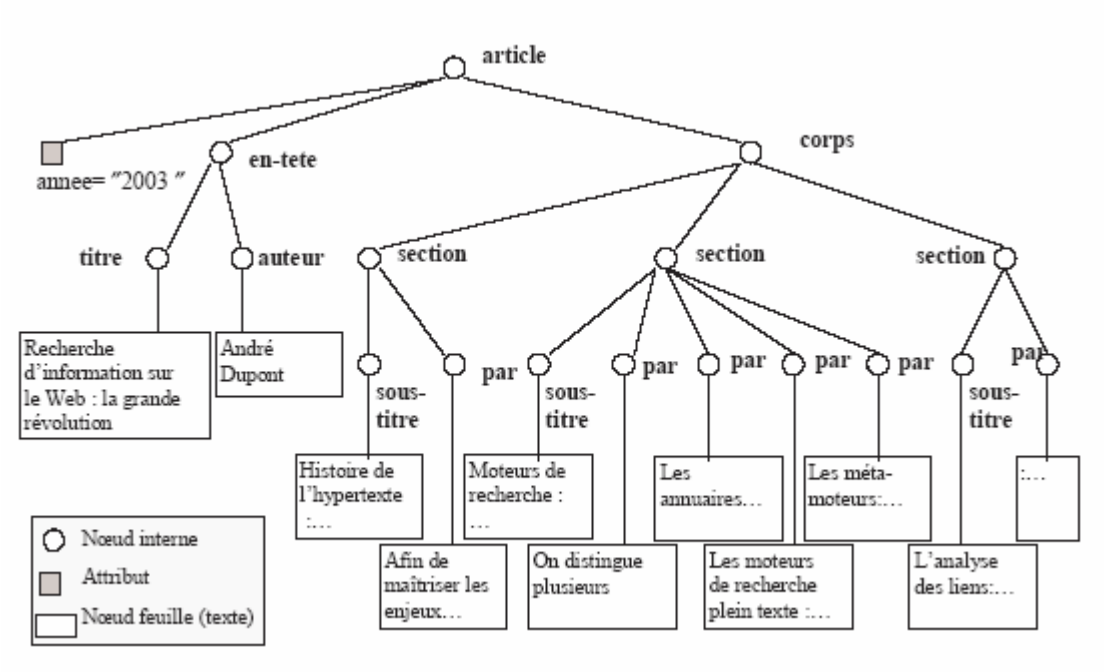

Figure -2.1- Représentation en arbre du document "*article*".

Il n'existe pas de règles absolues quant à l'utilisation des attributs ou des éléments. La ligne de conduite préconisée étant cependant, de modéliser les données en tant que contenu au moyen d'élément/fils et les méta données en tant qu'attributs attachés à l'élément qu'elles décrivent (les méta données sont des données à propos des données).

Les formats associés aux modèles des documents structurés nécessitent l'emploi de balises (tags ou labels) qui représentent la structure logique. Les langages utilisant ces constructions sont appelés "langages de balisage". Les deux formats standards les plus important sont : le SGML (Standard Generalized Markup Language) [Goldfarb, 90] une norme de l'ISO, et le XML (eXtensible Markup Language) développé par le W3C [W3C, 98a] et présenté comme un sous ensemble de SGML simplifié et optimisé pour le Web. XML est aujourd'hui le format de représentation le plus utilisé. C'est pour cela, dans la suite nous nous focalisons sur les documents semi-structurés XML et les technologies qui s'y rapportent.

2.2.2 La notion de structure

Comme nous l'avons vu dans le paragraphe précédent, la structure des documents est définie par des balises encadrant les portions d'informations. Nous présentons ici les notions de base liées à la structure, s'appliquant aussi bien à des documents SGML que des documents XML :

- Une *balise* (ou *tag* ou *label)* est une suite de caractères encadrés par "<" et ">", comme par exemple : <*tagname*>.

- Un *élément* est une unité sémantique identifiée, délimitée par des balises de début < *b* > et de fin < */b* >, comme par exemple : <*unebalise*> *c'est un exemple* </*unebalise*>. Les éléments peuvent être imbriqués formant ainsi une structure hiérarchique (un arbre) comme dans l'exemple de la figure 2.2 suivante :

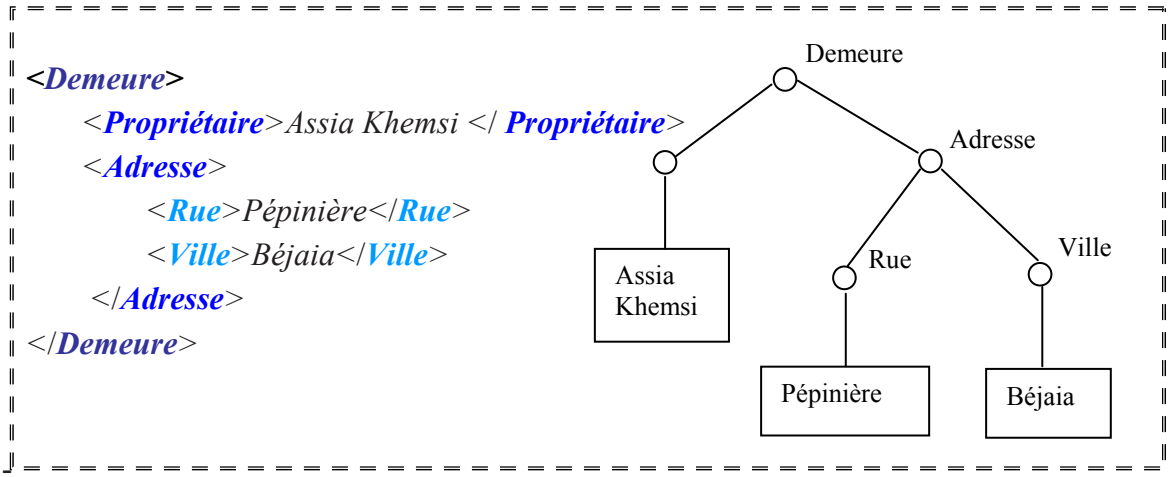

Figure -2.2- Exemple d'éléments imbriqués avec la représentation en arbre.

Les *attributs* des balises sont spécifiés au début de l'élément et après le nom de la balise, en utilisant la syntaxe : *nomattribut = valeur*. Par exemple :
<*unebalise sonattribut = unevaleur*> *c'est un exemple* </*unebalise*>.

Nombreuses sont les technologies reliées au format de données XML, nous présentons dans la suite, les plus importantes.

2.2.3 Schémas pour les documents XML

Les documents se conformant aux règles syntaxiques définies par les recommandations XML du W3C, sont dits "bien formé". Cette propriété garantit qu'un analyseur XML pourra analyser le document XML avec succès. La correction syntaxique des documents XML n'est cependant pas suffisante, il est intéressent de pouvoir exprimer des contraintes sur leur structure en fonction du vocabulaire utilisé. Ces contraintes sont en effet, exprimées par des langages de schéma qui permettent de décrire des classes de documents. D'où vient la notion de document XML valide, c'est-à-dire se conformant à un schéma. Divers langages de schéma sont proposés pour la validation de documents XML, nous présentons les deux plus populaires, à savoir les DTD (Document Type Definition) et les XML Schema. On dira alors qu'une classe de documents possède une structure *générique* définie par la DTD (ou le schéma XML), alors qu'un document instance de cette classe possède une structure *spécifique*.

2.2.3.1 Les DTD XML

La DTD (*Document Type Definition*) associée au document contient l'ensemble des balises qu'il est possible d'inclure, ainsi que des relations de composition entre ces balises. Ce sont un sous ensemble des DTD SGML et dont elles héritent la popularité. Leur expressivité est limitée comparée aux autres langages et elles sont les seules à ne pas employer la syntaxe XML. Elles sont nécessaires pour la définition d'entités générales utilisées dans les instances des documents, mais il n'est pas obligatoire d'associer une DTD à un document XML. Elles fournissent un mécanisme d'inclusion, mais ne gèrent pas les espaces de noms. Le système de type est pauvre (le contenu des nœuds textes -#PCDATA- ne peut pas être typé), mais il est toutefois possible d'assigner des valeurs par défauts aux attributs et de définir les valeurs possibles par une énumération. Les contraintes sur les modèles de contenu sont : la séquence ",", le choix "|" et les opérateurs de cardinalité : ? (1|0), + (au moins un), * (0 ou plus). La figure 2.3 suivante illustre un exemple de DTD.

```
<!ELEMENT company (address, cname, personnel)>
<!ATTLIST comapny id ID #REQUIRED>
<!ELEMENT address (street, city, state, zip)>
<!ELEMENT personnel (person)+>
<!ELEMENT person (name, email?, url?, fax+)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT name (family|given|middle?)*>
<!ELEMENT cname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
```

Figure -2.3- Exemple d'une DTD XML représentant une société.

2.2.3.2 Les XML Schema

C'est une recommandation du W3C depuis Mai 2001[W3C, 01a] pour remplacer les DTD jugées trop peu expressives. Elle est divisée en plusieurs sous-recommandations : *XML Schema Part 0* qui décrit l'utilisation d'XML Schema, *XML Schema Part 1* qui décrit les structures et *XML Schema Part 2* qui décrit les types de données. Elles adoptent la syntaxe XML, et proposent un système de type très riche permettant à l'utilisateur de définir ses propres types de données ainsi que des mécanismes d'héritage et de sous

typage. L'occurrence des éléments peut être contrôlée d'une façon très fine, les espaces de noms sont gérés et des mécanismes d'importation et d'inclusion sont proposés.
Un exemple de schéma XML est donné dans la figure 2.4.

```
1 : <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2 :     <xs:element name="PO">
3 :         <xs:complexType>
4 :             <xs:sequence>
5 :                 <xs:element ref="POHeader"/>
6 :                 <xs:element ref="ShipTo"/>
7 :                 <xs:element ref="BillTo"/>
8 :                 <xs:element ref="Line"/>
9 :             </xs:sequence>
10:         </xs:complexType>
11:     </xs:element>
12:     <xs:element name="POHeader">
13:         <xs:complexType>
14:             <xs:sequence>
15:                 <xs:element name="Number" type="xs:string"/>
16:                 <xs:element name="Date" type="xs:string"/>
17:             </xs:sequence>
18:         </xs:complexType>
19:     </xs:element>
20:     <xs:element name="ShipTo">
21:         <xs:complexType>
22:             <xs:sequence>
23:                 <xs:element name="PostalCode" type="xs:string"/>
24:                 <xs:element name="Country" type="xs:string"/>
25:                 <xs:element name="City" type="xs:string"/>
26:                 <xs:element name="Street" type="xs:string"/>
27:             </xs:sequence>
28:         </xs:complexType>
29:     </xs:element>
30:     <xs:element name="BillTo">
31:         <xs:complexType>
32:             <xs:sequence>
33:                 <xs:element name="PostalCode" type="xs:string"/>
34:                 <xs:element name="Country" type="xs:string"/>
35:                 <xs:element name="City" type="xs:string"/>
36:                 <xs:element name="Street" type="xs:string"/>
37:             </xs:sequence>
38:         </xs:complexType>
39:     </xs:element>
40:     <xs:element name="Line">
41:         <xs:complexType>
42:             <xs:sequence>
43:                 <xs:element ref="Item"/>
44:             </xs:sequence>
45:         </xs:complexType>
46:     </xs:element>
47:     <xs:element name="Item">
48:         <xs:complexType>
49:             <xs:sequence>
50:                 <xs:element name="UOM" type="xs:string"/>
51:                 <xs:element name="UnitPrice" type="xs:string"/>
52:                 <xs:element name="QTY" type="xs:string"/>
53:             </xs:sequence>
54:         </xs:complexType>
55:     </xs:element>
56: </xs:schema>
```

Figure -2.4- Exemple d'un XML Schema représentant une facture.

2.2.4 DOM (Document Object Model)

C'est une spécification du W3C [W3C, 98b] qui fournit un moyen pour gérer des documents XML dans un programme. Par exemple, il offre la possibilité de créer des documents et des fragments de documents, de naviguer dans les documents, etc. Le DOM s'intercale généralement entre le parseur XML et l'application ayant besoin des informations du document. Ce qui signifie que le parseur lit les données du document pour les diriger vers le DOM qui est alors utilisé par une application de niveau supérieur. L'API DOM est basée sur une structure d'objets pour représenter un document balisé. L'analyseur génère un arbre d'objets reliés entre eux, chaque objet représentant un atome du document XML.

2.2.5 XPath

C'est un langage qui permet d'adresser les sections d'un document XML afin de permettre d'en extraire avec précision les informations voulues, à l'aide 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 [Clark et al, 99]. Il permet la sélection de sous-arbres d'un document XML. Il possède une syntaxe simple et non ambiguë et implémente les types usuels (chaînes, nombres, booléens, variables, fonctions).
La seconde norme XPath 2 a servi de base à la définition du langage de requêtes du groupe de travail INEX 2003 et bien d'autres projets dont HYREX [Govert et al, 02].

2.2.6 Espaces de noms

Les espaces de nom (*namespaces*) permettent de disposer, dans un document XML, de balises provenant de différents catalogues : par exemple des balises HTML, MathML, etc. Il se peut que deux catalogues fournissent des balises de même nom, mais de significations différentes, les espaces de nom résolvent alors ce genre de problème : ils nomment de manière unique un élément ou un attribut en associant un domaine à un ensemble de noms. En pratique, on préfixe l'objet de l'espace de nom correspondant. Les espaces de nom sont identifiés par des URIs (*Uniform Resource Identifiers*), mais l'on précise pour chacun d'eux un label qui servira de préfixe aux balises concernées.

2.2.7 XSL (eXtensible Style sheet Language)

XSL est un langage de feuilles de styles. Il est composé de deux parties principales :

– XSLT (*XSL Transformation*) : Conçu au départ comme le composant de transformation du langage de formatage XSL du W3C, XSLT est devenu un langage de transformation de documents XML indépendant. Il permet la transformation de documents XML en documents XSL-FO (*XSL Formatting Objects*) mais aussi vers n'importe quel autre vocabulaire XML (XHTML, SVG, etc . . .) ainsi que vers d'autres modèles comme le texte pur ou LaTeX. XSLT est principalement axé sur la transformation de la structure des documents XML, même s'il peut aussi, dans une moindre mesure, manipuler leur contenu.

– XSL/FO : langage qui permet de formater l'affichage et/ou l'impression d'un document XML (boîtes, positionnement, ordonnancement et propriétés d'affichage). Il s'agit d'une extension de CSS, associée aux documents HTML. XSLT 1.0 et XSL/FO 1.0 sont des recommandations du W3C [W3C, 01b].

2.2.8 RDF (Ressource Definition Framework)

C'est un cadre de description et d'échange des méta-données. Quelque soit le format utilisé, RDF permet de rendre plus efficace le traitement automatisé des informations du Web, en fédérant les vocabulaires et syntaxes de description des méta-données existantes dans un cadre commun. RDF est piloté par le W3C [Lassila et al, 99] et est largement influencé par le Dublin Core. RDF permet de rendre plus "intelligente" l'information nécessaire aux moteurs de recherche et, plus généralement, nécessaire à tout outil informatique analysant de façon automatisée des pages Web dans un cadre commun. Cela ne veut pas dire qu'il s'agit de définir le modèle de méta-données, mais plutôt de permettre à chaque modèle de s'insérer harmonieusement dans les méta-données décrivant une ressource particulière. Il est utile pour la recherche d'information (pour donner aux outils de recherche de plus grandes possibilités), pour le catalogage (puisqu'il décrit le contenu d'un document et les rapports qu'il a avec les divers contenus d'un site Web), et pour le partage et l'échange de connaissances, via des agents logiciels intelligents. La force de RDF est de ne pas se prononcer sur le sujet et de laisser aux personnes définissant leurs méta-données le choix du vocabulaire utilisé.

2.3 La recherche d'information structurée

Le domaine de la Recherche d'Information Structurée (RIS) est l'extension du domaine de la RI classique aux documents semi-structurés de type XML. C'est un domaine émergeant qui s'est principalement développé ces dernières années notamment avec l'apparition de l'initiative INEX[3] [INEX] et de plusieurs Workshop dans différentes conférences de RI (SIGIR[4] [SIGIR] par exemple). Il s'intéresse à l'adaptation des modèles de RI pour une prise en compte simultanée de la structure et du contenu ainsi qu'à la définition et l'évaluation de nouvelles problématiques spécifiques aux données semi-structurées.

Dans la suite, nous allons tout d'abord montrer que les méthodes de représentation habituelles ne sont pas adaptées à la représentation des données XML. Puis, nous parlons de la nécessité d'adapter les modèles classiques de RI et nous expliquons comment de nouvelles problématiques sont apparues simultanément à l'émergence de ce type de données. Parmi celles-ci, nous mettons en avant tout particulièrement les problématiques issues du phénomène de l'hétérogénéité structurelle qui nous intéresse dans notre travail.

[3] INEX : Initiative for the Evaluation of XML Retrieval
[4] SIGIR : Special Interest Group of Information retrieval

2.3.1 Recherche d'Information Structurée : problèmes et enjeux

2.3.1.1 L'unité d'information recherchée : la redéfinition de la notion de document

Si les systèmes de RI classiques considèrent les documents dans leur globalité bien qu'il s'avère souvent que ces derniers peuvent renfermer des contenus hétérogènes, les systèmes de RI structurée cherchent plutôt à identifier des parties de documents les plus pertinentes à une requête, aidés en cela par la structure des documents définie par les balises.

L'introduction d'une information structurelle dans le codage des documents bouleverse complètement la notion d'unité d'information. Cette unité était jusqu'à présent le document, ce n'est pas le cas aujourd'hui. Prenons l'exemple des sites Web ; faut-il considérer qu'un document est une page HTML ou alors qu'un document est un site Web en entier ? Ce changement quant à la notion d'unité d'information est à l'origine de l'apparition de nouvelles problématiques spécifiques aux documents structurés. Certaines de ces nouvelles problématiques sont aujourd'hui particulièrement étudiées dans le cadre de la recherche documentaire sur les documents structurés notamment au travers de l'initiative INEX.

Ainsi, le concept de "granule ou d'unité d'information" renvoyée à l'utilisateur n'est plus un document en complet. Il correspond plutôt à un nœud de l'arbre du document dont la pertinence vis-à-vis d'une requête est calculée selon deux nouvelles notions : "exhaustivité" et "spécificité". On parle alors d'unité d'information exhaustive à une requête, si elle contient toutes les informations requises par la requête, et d'unité d'information spécifique à une requête, si tout son contenu ne concerne que la requête. Par conséquent, le principe de RI dans les documents structurés est défini comme suit : "Un système de recherche d'information devrait toujours renvoyer l'unité d'information la plus exhaustive et spécifique répondant à une requête", ce qui revient à trouver les sous arbres de taille minimale pertinents à la requête.

De part cette structure, on distingue deux types de requêtes des utilisateurs selon leur connaissance du corpus :

- Les requêtes portant sur le contenu, composées de simples mots clés. Dans ce cas, c'est le système qui se charge de trouver l'unité d'information la plus pertinente et qui décide du granule d'information à retourner à l'utilisateur.

- Les requêtes portant sur le contenu et la structure, où l'utilisateur peut spécifier l'élément qu'il désire voir retourner.

Afin de répondre à ces nouvelles notions, il y a lieu d'adapter les anciens systèmes pour pouvoir répondre aux problématiques liées à la modification de l'unité d'information recherchée que cela soit au niveau de l'indexation, l'interrogation ou même la recherche et le tri des unités d'information renvoyées.

2.3.1.2 Problèmes de représentation

Le problème de la représentation et de l'indexation des documents structurés pour la RI est encore aujourd'hui un problème ouvert. Les problématiques au niveau d'indexation peuvent se résumer en les questions suivantes : Comment relier la structure au contenu même du document ? En fonction de qu'elle dimension doit-on pondérer les termes d'indexation ? Que doit-on indexer de la structure des documents ?

Différentes approches proposées dans la littérature tentent de répondre à ces questions. Nous décrivons dans la suite les solutions apportées pour les deux dimensions d'indexation : information de contenu et information de structure.

2.3.1.2.1 Indexation de l'information de contenu

A la problématique traditionnelle de l'indexation qui consiste en l'identification et l'extraction des termes importants des documents, s'ajoute pour les documents structurés, le problème de la portée des termes d'indexation et celui du niveau de la pondération de ces termes. Ces deux problèmes seront détaillés dans la suite.

1) Portée des termes d'indexation

Cela consiste à répondre à la question comment rattacher les termes d'indexation à l'information structurelle ? Deux solutions sont proposées :

- **Indexation par sous arbres imbriqués :** Les partisans de cette approche [Abolhassani et al, 04], [Kamps et al, 04] et [Sigurbjornsson et al, 03], rattachent les termes d'un nœud feuille à tous ses nœuds ancêtres dans le document. Cependant, comme les documents XML possèdent une structure hiérarchique, les noeuds de l'index sont imbriqués les uns dans les autres et l'index contient de nombreuses informations redondantes.

- **Indexation par unités disjointes :** Pour les partisans de cette approche [Anh et al, 02], [Govert et al, 02], [Kazai et al, 02], [Roelleke et al, 02], [Fuhr et al, 03a] et [Ogilvie et al, 03], les termes des nœuds feuilles sont uniquement reliés au nœud parent qui les contient.

L'approche utilisée pour indexer le contenu du document influe directement sur la méthode utilisée pour la recherche.

2) Pondération des termes d'indexation

Si en RI traditionnelle, la pondération d'un terme tenait compte de son importance locale au sein du document et de manière globale au sein de la collection, s'ajoute en RIS l'importance du terme au niveau de l'élément qui le contient.
Les occurrences des termes ne suivent plus forcément une loi de Zipf [Grabs, 03]. En effet, 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.
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.

On trouvera des exemples d'adaptation des formules de pondération traditionnellement utilisées en RI à la RIS dans [Trotman, 05]. L'utilisation du facteur tf-ief (Terme Frequency - Inverse Element Frequency) a été proposée par de nombreux auteurs comme [Wolff et al, 00], [Grabs et al, 02] et [Sauvagnat, 04].

2.3.1.2.2 Indexation de l'information de structure

Ce n'est pas forcément toute l'information structurelle qui est utilisée dans le processus d'indexation. On distingue trois types d'approches proposées dans la littérature, pour l'indexation de l'information structurelle :

1) Indexation basée sur des champs

Dans cette approche, un document est représenté comme un ensemble de champs (dans l'exemple de la figure 2.5, on peut citer : auteur, intitulé, année, 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 d'un champ avec les termes du contenu, comme l'illustre la figure 2.5. Les champs peuvent être :

- des méta-données codées en RDF dans les fichiers XML.
- Provenir du document dans son format original s'il est transformé en XML.
- Retrouvés à l'aide de différentes techniques d'extraction [Gutierrez et al, 00].
- Extraits de la DTD ou du schéma XML associé.

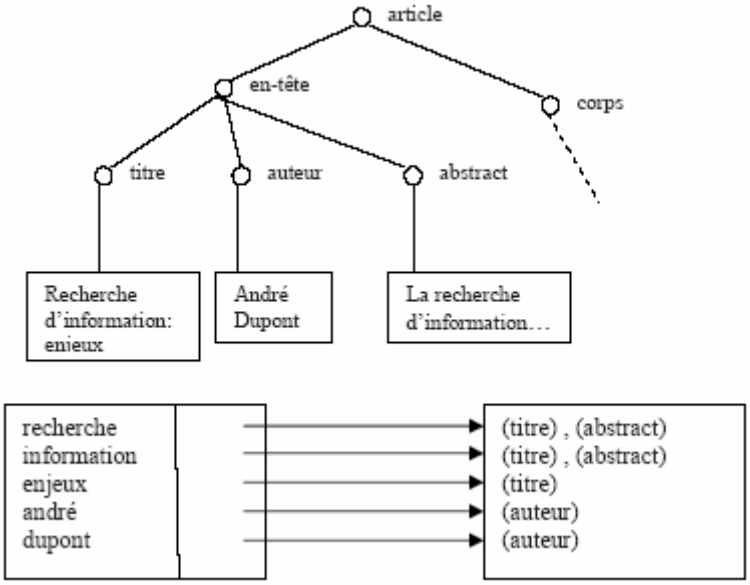

Figure -2.5- Exemple d'indexation basée sur des champs.

2) Indexation basée sur des chemins

Dans cette approche, l'indexation se base sur des index de chemins qui donnent pour chaque valeur répertoriée d'un chemin de balises, la liste des documents contenant un élément atteignable par ce chemin et ayant cette valeur, comme l'illustre la figure 2.6.

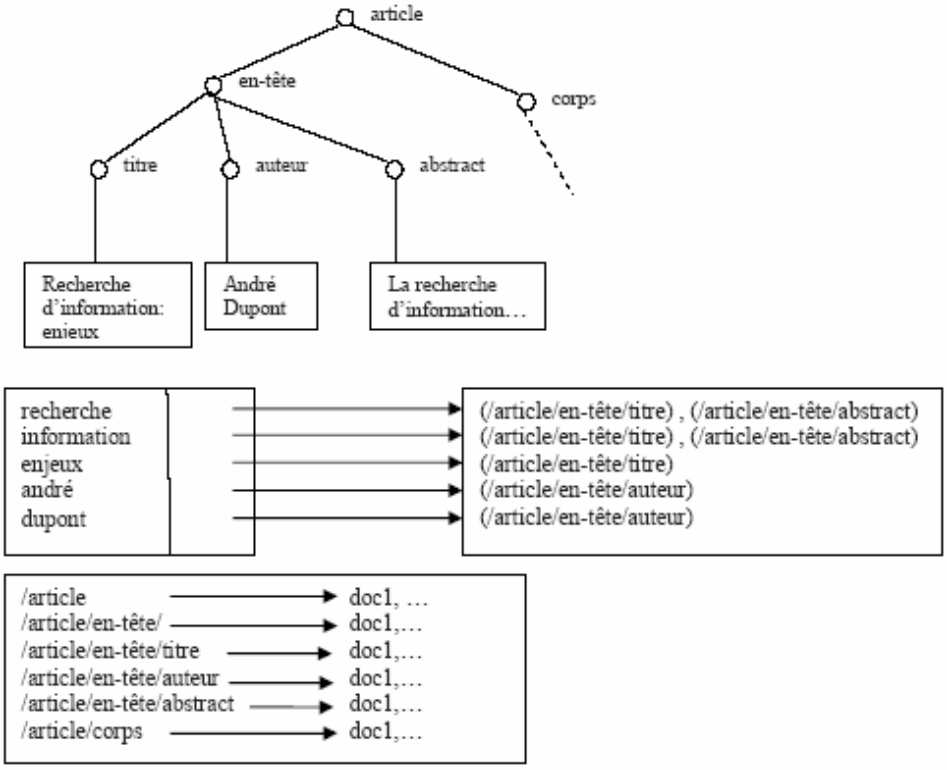

Figure -2.6- Exemple d'indexation basée dur des chemins.

Cette façon de faire, permet de retrouver rapidement des documents ayant des valeurs connues pour certains éléments ou attributs et facilite la navigation de façon à résoudre efficacement des expressions XPath.

3) Indexation basée sur des arbres

Dans cette approche, les nœuds de l'arbre du document sont numérotés dans l'index de sorte à pouvoir reconstruire la structure arborescente du document, pour la navigation et le traitement d'expression XPath. Une liste inverse de tous les termes apparaissant dans les feuilles et les valeurs des attributs est crée. Un meilleur exemple de cette manière d'indexation est l'approche utilisée dans XPath Acclerator [Grust, 02]. Elle permet d'assigner dans une traversée de l'arbre du document deux valeurs de pré et de post ordre à ses nœuds, comme le montre la figure 2.7.

En stockant de plus, la dimension du prédécesseur du nœud parent, un champ indiquant la présence d'attributs et le nom de balise de chaque nœud, la navigation devient efficace et il est possible de répondre à des expressions XPath qui n'ont pas pour origine la racine du document.

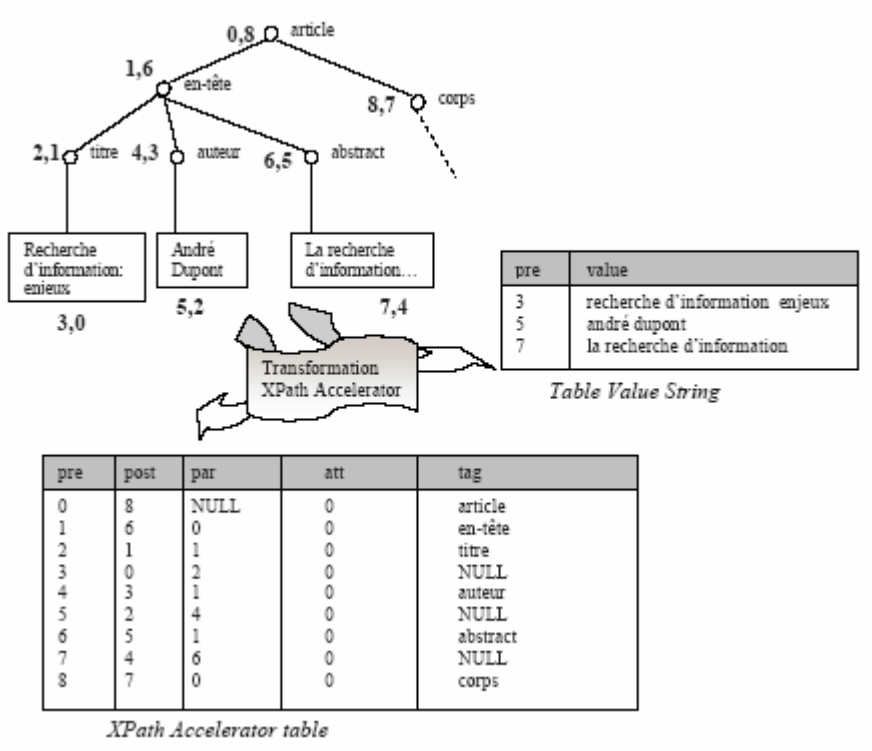

Figure -2.7- Exemple d'indexation basée sur des arbres.

2.3.1.3 Langages de requêtes

Différents langages de requêtes sont proposés dans la littérature pour l'interrogation de documents XML. Ils offrent à l'utilisateur la possibilité d'exprimer des besoins diversifiés (concernant le contenu et/ou la structure) et de la manière la plus simple. La profusion des différents langages peut être décryptée comme illustré dans la figure 2.8.

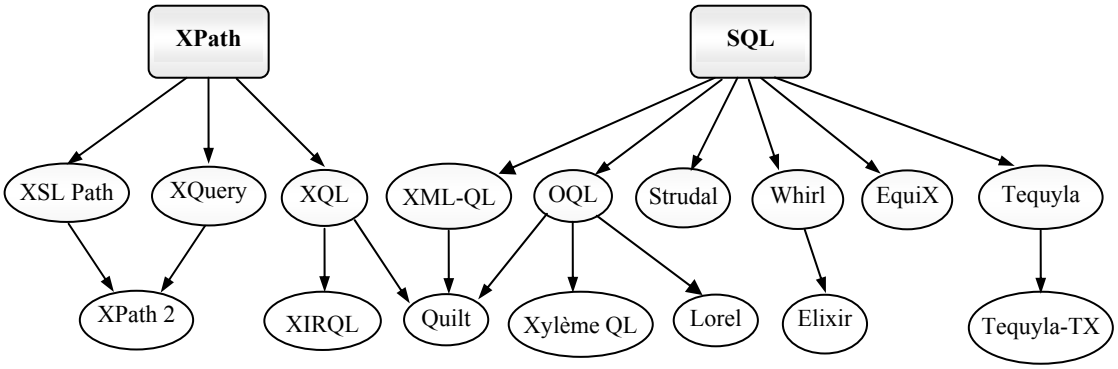

Figure -2.8- Historique des langages d'interrogation XML : Les liens représentent des relations d'inclusion (quelquefois partielle) des différents langages de requête.

Dans [Gardarin, 02] sont définies les fonctionnalités que doit supporter un langage de requête XML pour être à la fois un langage de requête pour les systèmes documentaires et pour les bases de données. Ces fonctionnalités sont les 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,
- Respect de la hiérarchie et des séquences,
- Agrégation de données depuis des documents,
- Préservation de structures,
- Construction de structures nouvelles,
- 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).

Une comparaison des langages de requêtes les plus importants, sur la base des principales caractéristiques est faite dans [Sauvagnat, 04], elle est donnée dans le tableau 2.1.

	LOREL	**XML-QL**	**XQL**	**XML-GL**	**Xquery**
Navigation	Oui	Oui	Oui	Oui	Oui
Support Xpath	Limité	Limité	Oui	/	Oui
Sélection	Oui	Oui	Oui	Oui	Oui
Jointure	Oui	Oui	Non	Oui	Oui
Tri	Oui	Oui	Non	Oui	Oui
Construction	Non	Oui	Non	Oui	Oui
Mot-clé	Non	Non	Non	Non	Oui
Fonction	Oui	Oui	Non	/	Oui
Imbrication	Oui	Oui	Oui	Oui	Oui
Agrégation	Oui	proposée	Non		Oui

Tableau -2.1- Comparaison de différents langages de requêtes pour XML.

La plupart des langages de requêtes proposés dans la littérature est basée sur une approche BD, qui se concentre sur l'introduction de la dimension structurelle dans les requêtes, et traite le contenu des éléments de façon booléenne (présent, absent). Des langages orientés RI commencent à apparaître (exemple : langage NEXI [Trotman et al, 04a] et [Trotman et al, 04b], langage XFIRM [Sauvagnat, 04], TexQuery [Amer-Yahia, 04], etc.). Ils proposent d'ajouter des fonctions de recherche sur le contenu (notamment l'utilisation du prédicat "about" au lieu de "contains").

2.3.2 Modèles de recherche d'information structurée

Nous présentons dans la suite les principales extensions faites sur les modèles de RI classiques pour les adapter à la recherche dans les documents structurés.

2.3.2.1 Extension des modèles booléens

Hatano et al. dans [Hatano et al, 02] se sont basés sur des approches portant sur l'extension des modèles booléens aux documents structurés grâce aux p-normes. Leur travail se basant essentiellement sur la définition de l'unité d'information renvoyée à l'utilisateur, ne considère que les requêtes orientées contenu. L'unité d'information est définie comme étant tout élément contenant au moins un élément feuille. Pour calculer le

score d'un élément vis-à-vis d'une requête, les auteurs propagent des scores, depuis les feuilles de l'arbre documentaire jusqu'à l'élément considéré. Pour ce faire, chaque élément feuille e_j *(j=1, ..., N)* est représenté par un vecteur :

$$F(e_j) = (w_{t_1}^{e_j}, w_{t_2}^{e_j}, ..., w_{t_n}^{e_j})$$

Où *n* représente le nombre de termes indexés dans la collection et $w_{t_i}^{e_j}$ représente le poids d'un terme t_i dans un élément feuille qui est donné par :

$$w_{t_i}^{e_j} = \frac{tf(t_i, e_j)}{\sum_{j=1}^{N}\sum_{k=1}^{n} tf(t_k, e_j)} . \log \frac{N_d}{df(t_i)}$$

Où : $tf(t_k, e_j)$ est la fréquence du terme t_k dans e_j, N_d est le nombre de documents dans la collection et $df(t_i)$ est le nombre de documents contenant t_i.
La similarité entre un élément feuille e_j (représenté par le vecteur $F(e_j)$) et une requête *q* (représentée par un vecteur dont les composantes sont 1 si un terme de la collection apparaît dans la requête, 0 sinon) est calculée grâce au cosinus :

$$sim(F(e_j), q) = \frac{F(e_j).q}{|F(e_j)|.|q|}$$

Ils utilisent ensuite la p-norme pour calculer le score final d'un élément. Ce score est calculé de manière récursive car il prend en compte le score des sous-éléments :

$$RSV(q, e) = 1 - \sqrt[\frac{1}{p}]{\frac{1}{|enf(e)|} \sum_{é \in enf(e)} (1 - RSV(q, é))^p}$$

Où : *p* est généralement choisi avec une valeur égale à 2. *enf(e)* représente les sous éléments de *e*. Lorsqu'un élément *e* est une feuille :

$$RSV(q, e) = sim(F(e_j), q).$$

2.3.2.2 Extension des modèles vectoriels

Ce modèle a connu diverses adaptations [Anh et al, 02], [Crouch et al, 02], [Fuller et al, 93], [Marx et al, 02], [Mass et al, 02], [Theobald et al, 02], [Carmel et al, 03], [Weigel et al, 04] et [Kakade et al, 05]. Toutes les approches issues de ce modèle calculent une mesure de similarité entre un élément et une requête. Pour ce faire, chaque élément est représenté sous forme de vecteur (obtenu par indexation des sous arbres imbriqués) de termes pondérés. Nous présentons les plus importantes.

Fuller dans [Fuller et al, 93] présente une des premières adaptations du modèle vectoriel. La mesure de similarité d'un nœud *n* à une requête $q = \{t_1, t_2, ..., t_k\}$ est donnée par l'équation suivante :

$$sim(q, n) = \alpha(T) \cdot cosm(q, n) + \sum_{k=1}^{s} \frac{cosm(q, n_k)}{\beta^{k-1}}$$

Où $\alpha(T)$ est un facteur qui prend en compte le type du noeud, s est le nombre de noeuds enfants n_k de n, et β est un paramètre permettant d'assurer que le nombre d'enfants n'introduit pas un biais dans la formule. La fonction *cosm* est définie comme suit :

$$cosm(q,n) = \sum_{i=1}^{T} \frac{w_i^q * w_i^n}{|n|}$$

Avec : $w^q{}_i$ et $w^n{}_i$ respectivement le poids du terme t_i dans la requête q et dans le noeud n, et $|n|$ le nombre de termes dans le noeud n. La pertinence d'un noeud peut ainsi être calculée à part, puis combinée avec la pertinence des noeuds descendants.
Le modèle peut être généralisé en permettant le traitement des requêtes orientées contenu et structure. L'idée de base est là encore d'appliquer le modèle récursivement à chaque sous arbre de la hiérarchie pour ensuite effectuer un agrégat des scores.

Les auteurs dans [Schlieder et al, 02] proposent d'intégrer la structure du document dans la mesure de similarité du modèle vectoriel en calculant une distance entre deux arbres. Cette distance est généralement calculée en fonction du coût associé à l'ajout et/ou au retrait de nœuds et de liens dans le graphe d'un élément pour obtenir celui de la requête.
La simplification de ce modèle tient essentiellement dans le fait qu'une première sélection est effectuée : seuls les éléments qui ont une structure qui peut être réduite à celle de la requête sont considérés. Ensuite, une représentation vectorielle simple du poids est utilisée pour le calcul du score, les notions de *tf* et *idf* sont adaptées comme suit :
Soit un élément E de type t, le poids $w^t{}_{T,E}$ d'un terme structurel T dans E est défini par :

$$w_{T,E}^t = tf_{T,E}.idf_T^t = \frac{freq_T(E)}{maxfreq(E)}.(log(\frac{|E^t|}{n_T}) + 1)$$

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.

Dans [Grabs et al, 02], les auteurs proposent d'évaluer l'importance d'un terme dans un élément donné en fonction de l'importance du terme dans les éléments du même type. Lorsque la requête est composée d'une condition sur le type d'un élément (on nommera *cat* ce type) ainsi que d'une condition sur le contenu de cet élément, la similarité d'un élément e de type *cat* à la requête q est calculée selon l'équation suivante :

$$RSV(e,q) = \sum_{t \in terms(q)} tf(t,e).ief_{cat}(t)^2.tf(t,q)$$

Où : $tf(t, e)$ est la fréquence du terme t dans l'élément e et $ief_{cat} = log\ N_{cat} / ef_{cat}(t)$, avec N_{cat} le nombre d'éléments du type *cat* et $ef_{cat}(t)$ la fréquence du terme t dans les éléments du type *cat*.

Les requêtes orientées contenu sont quant à elles traitées de la façon suivante :
Soit *SE*(*e*) l'ensemble des descendants de *e* incluant *e*. $\forall se \in SE(e)$, $l \in path(e, se)$ est une étiquette appartenant au chemin reliant *e* à *se*, c'est-à-dire un type d'élément. Soit enfin $aw_l \in [0, 1]$ un facteur modélisant l'importance de l'étiquette *l*. La similarité d'un élément *e* à une requête *q* composée de simples mots-clés est définie de la façon suivante :

$$RSV(e,q) = \sum_{se \in SE(e)} \sum_{t \in terms(q)} tf(t,se)\Big(\prod_{l \in path(e,se)} aw_l\Big).ief_{cat(se)}(t)^2.tf(t,q)$$

Le modèle JuruXML [Mass et al, 02] et [Mass et al, 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 un ensemble de conditions de la forme (*chemin, terme*). Ensuite, une correspondance vague entre les chemins est calculée par la fonction suivante :

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

Où : $c^q{}_i$ est la condition de chemin pour un terme t_i de la requête *q* et $c^e{}_i$ le XPath de ce terme dans l'élément *e*.
Enfin, on a :

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

La fonction de similarité entre deux chemins *cr* permet de favoriser les termes qui apparaissent dans des endroits proches (au sens de la structure), par exemple :

cr (/doc/title, /doc/title) = 1,
et cr (/doc/title, /doc/sections/section1/title) = 0,5

2.3.2.3 Extension des Modèles probabilistes

Nous présentons deux principales extensions de ce modèle, à savoir :

1) Le modèle FERMI

Le modèle de données multimédia FERMI [Chiaramella et al, 96] est l'un des premiers modèles à considérer la vue logique des documents et à permettre le renvoi à l'utilisateur

non seulement de documents entiers, mais aussi de sous-structures de la structure logique des documents (c'est à dire des *noeuds*). Pour ce faire, chaque document est représenté par un arbre composé d'objets structurels typés (par exemple un livre, un chapitre, une section, un paragraphe, une image,...), et dans lequel les feuilles contiennent des données mono-média. Deux sortes d'attributs sont alors assignés aux nœuds :

- attribut standard (comme l'auteur ou la date de création).
- Attribut de description du contenu du noeud pour l'index.

Ce dernier type d'attributs est initialement assigné aux seuls noeuds feuilles, et leur contenu dépend du type de média. Par exemple, pour le texte, ils servent à décrire le contenu sémantique, alors que pour les images, un contenu spatial et perceptif est ajouté.
On procède à une propagation des valeurs des attributs vers le haut ou vers le bas de la hiérarchie, selon la classe de l'attribut auquel elles sont rattachées, par exemple : les auteurs des différents noeuds sont propagés vers le haut (en utilisant des techniques de fusion), alors que la date de publication d'un document complet est propagée vers le bas.
La recherche suit une approche logique, c'est à dire qu'elle consiste à chercher des noeuds *n* qui impliquent la requête *q*. La formulation originale du modèle est basée sur la logique des prédicats, et la stratégie de recherche est composée de deux phases, "fetch and browse" : dans la première phase, on cherche les noeuds exhaustifs, et dans la seconde phase, on navigue à partir de ces noeuds dans l'arborescence du document pour retrouver (éventuellement) des noeuds plus spécifiques.

Le modèle proposé dans [Lalmas, 97], affine le modèle FERMI en utilisant la théorie de l'évidence de Dempster-Shafer [Shafer, 76], dont l'intérêt tient à l'existence d'un opérateur de combinaison de croyance qui combine la croyance provenant de plusieurs éléments permettant ainsi, d'effectuer une agrégation du score de pertinence des éléments en respectant la théorie de l'incertain.

2) Le modèle d'inférence probabiliste

Une approche d'extension du modèle probabiliste inférentiel aux documents structurés se résume en l'utilisation de probabilités conditionnelles de jointure, avec par exemple : *P(d/t)* devenant *P(d/p contains t)* où *d* représente un document ou une partie du document, *t* est un terme et *p* un chemin dans l'arbre structurel de *d*.

Une méthode d'augmentation basée sur le modèle probabiliste est proposée dans [Fuhr et al, 03a] et [Govert et al, 02]. Elle est basée sur le langage de requêtes XIRQL, et a été implémentée au sein du moteur de recherche HyRex. L'approche adopte une indexation par unités disjointes (sauf pour les nœuds feuilles d'une granularité trop fine, dont les termes sont propagés au nœud indexable le plus proche dans la hiérarchie). De plus pour préserver des unités disjointes, on ne peut associer à un nœud que les termes non reliés à ses descendants. Les calculs de pertinence se font comme suit :

- Dans le cas des requêtes orientées contenu, le poids de pertinence d'un nœud est calculé par propagation des poids des termes les plus spécifiques dans l'arbre du document. Les poids sont de plus diminués par multiplication par un facteur d'augmentation.
- Dans le cas des requêtes orientées contenu et structure, on calcule d'abord les probabilités d'apparition de chaque terme de la condition de contenu dans les éléments répondants aux conditions de structure. Puis, des sommes pondérées de ces probabilités sont effectuées.

2.3.2.4 Autres approches

Des travaux comme [Wolff et al, 00], [List et al, 03], [Ogilvie et al, 03], [Sigurbjornsson et al, 03], [Abolhassani et al, 04] et [Kamps et al, 04] proposent d'appliquer les modèles de langage à la recherche d'information structurée.

Dans l'approche de [Kamps et al, 04] par exemple, on estime un modèle de langage pour chaque élément de la collection. Pour ce faire, tous les éléments d'un document sont indexés en suivant ainsi une approche d'indexation par sous arbres imbriqués. Puis, pour chaque requête (orientée contenu) les éléments sont triés par rapport à la probabilité que le modèle de langage de l'élément génère la requête :

$$P(e, q) = P(e) \,.\, P(q/e)$$

Où : - $P(e)$: est la probabilité à priori de l'élément e,
- $P(q/e)$:est la probabilité que l'élément e génère la requête q.

La probabilité $P(q/e)$ est calculée en supposant l'indépendance entre les terme de la requête $q = (t_1, \ldots, t_n)$ et en utilisant une interpolation linéaire du modèle de l'élément e et celui de la collection, comme suit :

$$P(t_1, \ldots t_n|e) = \prod_{i=1}^{n} (\lambda . P(t_i|e) + (1 - \lambda) . P(t_i))$$

Où : - $P(t_1, \ldots, t_n/e)$ est la probabilité d'observer le terme t_i dans l'élément e,
- $P(t_i)$ est la probabilité d'observer le terme t_i dans la collection.
- λest un paramètre de lissage entre le modèle de l'élément e et celui de la collection.

Le calcul de ces probabilités est donné par la formule ci-dessous :

$$\begin{aligned} s(e, t_1, t_2, \ldots t_n) &= \beta . log(\sum_t tf(t, e)) \\ &\quad + \sum_{i=1}^{n} log(1 + \frac{\lambda . tf(t_i, e) . (\sum_t df(t))}{(1 - \lambda) df(t_i) . (\sum tf(t, e))}) \end{aligned}$$

Où : - $tf(t, e)$ est la fréquence du terme t dans l'élément e,
- $df(t)$ est le nombre d'éléments contenant le terme t,
- λest un paramètre de lissage entre le modèle de l'élément e et celui de la collection.

- et β est un paramètre servant à combler le fossé entre la taille d'un élément et la taille de l'élément moyen pertinent.

Wolff et al. dans [Wolff et al, 00] proposent l'utilisation de la fréquence inverse d'élément *ief* 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.

Les travaux de [Piwowarski et al, 02], [Piwowarski, 03a] et [Vittaut et al, 04], proposent compte à eux, l'application des réseaux bayésiens. Pour ce faire, Les auteurs associent à chaque élément de l'arbre du document une variable aléatoire qui peut prendre trois valeurs dans l'ensemble V= {N, G, E}, où :

- N : signifie qu'un élément est non pertinent,
- G : signifie qu'un élément est peu spécifique,
- E : signifie qu'un élément est très spécifique.

Les éléments seront classés selon leur degré de pertinence à une requête, qui est donné par la probabilité : *P(e=E/q)* où *e* est un élément et *q* une requête.
Deux autres types de variables aléatoires sont utilisés :
- Une variable aléatoire est associée à chaque requête, représentée sous forme de vecteur de fréquence de termes.
- Une variable est associée à chaque modèle de pertinence utilisé pour évaluer la similarité locale d'un élément à une requête. Elle prend deux valeurs : pertinent ou non pertinent.
Un score de pertinence locale d'un élément à une requête, qui ne dépend que de l'élément et de la requête, peut être calculé en utilisant plusieurs modèles : la fréquence des termes de la requête dans la requête, dans l'élément, la longueur de l'élément,…
Ensuite, la probabilité qu'un élément soit dans l'état N, G ou E, qui dépend de l'état de l'élément parent et des jugements des modèles de pondération locale, est calculé comme suit (si on considère deux modèles M_1 et M_2 pour le calcul du score local d'un élément *e*):

$$P(e = v|q) = \sum_{v_p \in V, r_1, r_2 \in \{R, \neg R\}} \theta_{c(e), v, v_p, r_1, r_p} * P(e\ parent = v_p) * P(M_1 = r_1|q) * P(M_2 = r_2|q)$$

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 4 variables aléatoires (état de l'élément, état du parent, pertinence des modèles 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 a été étendu par Vittaut et al. dans [Vittaut et al, 04], pour prendre en compte des requêtes orientées contenu et structure.

2.3.2.5 Approches orientées RI pour le traitement de la structure

La plupart des modèles proposés jusqu'ici ne proposent pas d'approches réellement orientées RI pour le traitement des conditions de structure des requêtes. En effet, ces approches cherchent des correspondances exactes entre les conditions de structure des requêtes et les éléments retournés par le système. D'autres approches, notamment [Braga et al, 02], [Schileder et al, 02], [Yoo, 02], [Mass et al, 03] et [Sauvagnat, 04], qui sont indépendantes des modèles de pondération des termes utilisés, existent cependant. Elles cherchent à évaluer la pertinence des conditions structurelles en effectuant des correspondances entre l'arbre du document et l'arbre de la requête. La correspondance entre l'arbre de la requête et l'arbre du document est effectuée de manière vague. Par conséquent, des documents possédant une structure différente de celle de la requête peuvent être renvoyés à l'utilisateur, même si leur score de pertinence est plus faible que celui des documents pour lesquels toutes les conditions de structure sont respectées. Par exemple, un document possédant la structure /a/b/c sera sélectionné pour une requête /a/d/c, mais aussi pour une requête /a/c/b.

Par exemple, dans [Braga et al, 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*, ...).

Un autre exemple, est celui de Yoo dans [Yoo, 02] qui 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 les termes des noeuds feuilles ou en terme de noeuds entre les noeuds. La distance des noeuds peut être quantifiée grâce à la distance horizontale (nombre de noeuds du même niveau entre les noeuds) et à la distance verticale (nombre d'unités logiques qui peuvent être groupées pour aller d'un noeud à un autre).

Sauvagnat dans [Sauvagnat, 04] propose un modèle de propagation de la pertinence qui repose sur une représentation en arbre des documents XML, dans laquelle une indexation des unités disjointes est adoptée. Pour les requêtes orientées contenu, un premier score de pertinence est calculé pour les noeuds feuilles des documents XML, prenant en compte les pondérations locales et globales des termes *(Tf-Ief)*. Ensuite, ce score est propagé dans

l'arbre du document pour calculer le score des noeuds internes, après l'avoir diminuer en prenant en compte la distance entre un noeud interne et ses noeuds feuilles. Le modèle tient compte également du contexte d'un nœud en définissant deux notions : l'informativité d'un nœud qui est calculée en utilisant la taille des éléments comme indication sur leur importance durant la propagation (plus les éléments sont petits, plus le concepteur du document a cherché à faire ressortir leur contenu), et le contexte d'appartenance des éléments, en prenant en compte la pertinence du document dans son entier dans le calcul du poids de pertinence d'un noeud interne.
Pour les requêtes contenant des conditions de structure, la correspondance entre l'arbre de la requête et l'arbre du document, est effectuée de manière vague, en effectuant diverses propagations dans l'arbre du document. Par conséquent, des documents possédant une structure différente de celle la requête, peuvent être renvoyés à l'utilisateur même si leur score de pertinence est plus faible que celui des documents pour lesquels toutes les conditions de structure sont respectées.
Ce modèle permet également de déterminer la granularité de l'information à renvoyer dans le cas de requêtes possédant des conditions de structure et pour lesquelles aucune indication sur le type d'élément à renvoyer n'est donnée.

2.3.3 Evaluation de la performance des systèmes de RIS

Depuis la fin de l'année 2002, une campagne d'évaluation des systèmes de RI pour la recherche d'information sur des documents XML, nommée INEX (*Initiative for the Evaluation of XML Retrieval*) est lancée. Elle a lieu en 2005 pour la quatrième année consécutive. Elle fournit un corpus de documents XML, un ensemble de requêtes et leurs jugements de pertinence, dans le but de promouvoir l'évaluation de la recherche sur des documents XML.

Dans la suite, nous décrivons brièvement le corpus INEX, les requêtes, les tâches, les jugements de pertinence et les mesures d'évaluation.

2.3.3.1 Corpus INEX

La collection INEX est composée d'articles scientifiques provenant de la IEEE Computer Society, balisés au format XML. La collection, d'environ 500 Mo, contient plus de 12000 articles, publiés de 1995 à 2002, et provenant de 18 magazines ou revues. Un article moyen est composé d'environ 1500 éléments, et la profondeur moyenne des documents est de 6.9. Au total, la collection contient 8 millions de noeuds et 192 balises différentes.

2.3.3.2 Requêtes

Les requêtes (ou Topics) sont crées par les différents participants et doivent être représentatives des demandes de l'utilisateur moyen sur la collection. On distingue deux principaux types de requêtes :

– Les CO (*Content Only*) : ces requêtes sont exprimées en langage naturel. Les mots-clés de la requête peuvent être groupés sous forme d'expressions et précédés par les opérateurs '+' (signifiant que le terme est obligatoire) ou '-' (signifiant que le terme ne doit pas apparaître dans les éléments renvoyés à l'utilisateur).
– Les CAS (*Content And Structure*) : ces requêtes contiennent de plus des contraintes sur la structure des documents.

La figure 2.9 donne un exemple de la composition d'une requête.

```
<inex-topic  topic-id = "numéro-requête"  query type = "type-requête">
<title> "donne la définition formelle de la requête" </title>
<description>  "indiquent brièvement les intentions de l'auteur en langage
   naturel" </description>
 <narrative>"indiquent d'une façon narrative les intentions de l'auteur en
   langage  naturel " </narrative>
 <keywords> "contient un ensemble de mots-clés qui ont permis
   l'exploration du corpus avant la formulation définitive de la requête "
</keywords>
 </inex-topic>
```

Figure -2.9- Anatomie d'une requête INEX.

2.3.3.3 Tâches

La tâche principale d'INEX est la tâche de recherche *ad-hoc*, elle est considérée 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 trois sous-tâches :

- **Tâche CO :** La tâche CO (*Content Only Task*) a pour but de répondre avec des éléments/documents XML à des requêtes utilisateur orientées contenu (topic CO), c'est à dire des requêtes contenant des mots-clés simples. Aucune indication de structure dans la requête ne peut aider les SRI à déterminer la granularité de l'information à renvoyer.

- **Tâche SCAS :** La tâche SCAS (*Strict Content And Structure Task*) consiste à répondre avec des éléments/documents XML aux topics CAS de manière stricte, c'est à dire en respectant toutes les conditions sur la structure et le contenu énoncées dans les requêtes. Le champ *Title* des requêtes de la tâche SCAS est basé sur une syntaxe XPath.

- **Tâche VCAS :** La tâche VCAS (*Vague Content And Structure Task*), utilise elle aussi des requêtes CAS, mais pour lesquelles les participants peuvent répondre de manière vague, c'est à dire avec des éléments/documents qui satisfont globalement les requêtes. Le champ "*Title*"des requêtes de la tâche SCAS est basé sur le langage de requêtes NEXI [Trotman et al, 04a] et [Trotman et al, 04b]**,** l'extension de XPath utilisée en 2003 pour les requêtes CAS étant considérée comme trop complexe : 63% des requêtes exprimées par les participants (experts en RI) contenaient des erreurs de syntaxe !

Depuis 2004, quatre nouvelles tâches ont été proposées aux participants :

- **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 besoins;

- **La tâche "*hétérogène* "**, qui propose aux participants de nouvelles collections, afin de développer des approches indépendantes des DTDs.

2.3.3.4 Jugements de pertinence

Pour juger de la pertinence des documents vis-à-vis des requêtes, une échelle de pertinence à deux dimensions est utilisée. Ces dimensions sont :

1) Une dimension d'exhaustivité

La dimension d'exhaustivité décrit jusqu'à quel point un élément discute du sujet de la requête. Pour cela, 4 niveaux sont définis sur cette échelle :

- *Pas exhaustif* : l'élément ne traite pas du tout du sujet de la requête ;

- *Marginalement exhaustif* : l'élément traite peu d'aspects du sujet de la requête ;
- *Assez exhaustif* : l'élément traite de nombreux aspects du sujet de la requête ;
- *Très exhaustif* : l'élément traite la plupart ou tous les aspects du sujet de la requête.

2) Une dimension de spécificité

La dimension de spécificité décrit jusqu'à quel point un élément se focalise sur le sujet de la requête. Pour cela, 4 niveaux sont également définis sur cette échelle :

- *Pas spécifique* : le sujet de la requête n'est pas un thème de l'élément ;
- *Marginalement spécifique* : le sujet de la requête est un thème mineur de l'élément ;
- *Assez spécifique* : le sujet de la requête est un thème majeur de l'élément ;
- *Très spécifique* : le sujet de la requête est le seul thème de l'élément.

Ces deux dimensions de la pertinence reflètent la pertinence d'un élément par rapport à ses descendants. En effet, un élément peut être plus exhaustif que chacun de ses descendants pris séparément. De même, des éléments peuvent être plus spécifiques que leurs parents.

Le degré de pertinence d'un élément/document renvoyé par les systèmes de RI entrés en campagne pour chaque requête, est jugé (à la main) par les participants en utilisant un système de jugement en ligne [Piwowarski, 03b] et [Piwowarski et al, 04]. Le degré de pertinence est donné par la paire $(e, s)_{\in E.S}$, avec E.S = {(0, 0), (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)} est l'ensemble des 10 valeurs possibles sur l'échelle combinant les deux dimensions d'exhaustivité et de spécificité (les autres combinaisons ne sont pas possibles, par exemple : un élément qui n'est pat exhaustif, ne peut être spécifique, et inversement).

2.3.3.5 Mesures d'évaluation

Une fois les jugements de pertinence sont assignés aux résultats d'un système de RI, les scores de pertinence obtenus avec un ensemble de mesures sont utilisés pour évaluer la pertinence du système. Cet ensemble de mesures qui se basent sur les mesures traditionnelles de rappel et de précision, est défini dans INEX, il cherche à prendre en considération la structure des documents XML.

Dans la suite, nous décrivons quelques une des mesures les plus connues proposées dans INEX.

1) La mesure INEX 2002 (dite inex-eval metric)

Elle est utilisée depuis le début d'INEX, et est restée la mesure d'évaluation officielle utilisée jusqu'à INEX 2004. Elle se base sur la mesure de precall définie dans [Yu et al, 76] et [Raghavan et al, 89]. Elle permet de calculer la probabilité qu'un résultat vu par un utilisateur soit pertinent :

$$P(pert|retr)(x) = \frac{x.n}{x.n + esl_{x.n}}$$

Où : $esl_{x.n}$ (*Expected Search Length* [Cooper, 68]) : désigne la longueur supposée de recherche, c'est-à-dire le nombre attendu d'éléments non pertinents retrouvés jusqu'à ce qu'un point de rappel x soit atteint, et n est le nombre total de résultats pertinents dans la collection pour une requête donnée.

Pour appliquer cette mesure, les deux dimensions de pertinence (exhaustivité et spécificité) doivent être agrégées en une seule valeur de pertinence. Cela se fait par application d'une fonction d'agrégation : f_{quant} (e, s) : ES → [0, 1]. INEX utilise couramment plusieurs fonctions d'agrégation, chacune représente une préférence utilisateur différente.
Les résultats des fonctions d'agrégation peuvent être des valeurs purement binaire comme la fonction d'agrégation "stricte", ou des scores basés sur leur degré de pertinence comme les fonctions d'agrégation généralisées et les fonctions d'agrégation orientées spécificité ou orientées exhaustivité. On trouve des descriptions détaillées de ces fonctions dans [Kazai, 04].

L'inconvénient majeur de la métrique INEX 2002 est qu'elle ignore l'imbrication (*Overlap*) inhérente aux éléments XML et évalue 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 premier système renvoyant une section pertinente et un de ses paragraphes pertinents obtient les mêmes performances qu'un second système renvoyant deux éléments pertinents non imbriqués.

2) La mesure INEX 2003 (dite inex-ng)

Dans INEX 2003, une nouvelle mesure a été proposée pour essayer de résoudre ce problème [Fuhr et al, O3b] et [Govert et al, 03]. Cette mesure incorpore la taille des éléments et le concept d'imbrication dans les mesures de rappel et de précision comme donné dans les équations ci-dessous :

$$rappel_o = \frac{\sum_{i=1}^{k} e(c_i).\frac{|c'_i|}{|c_i|}}{\sum_{i=1}^{N} e(c_i)}$$

$$precision_o = \frac{\sum_{i=1}^{k} s(c_i).|c'_i|}{\sum_{i=1}^{k} |c'_i|}$$

Où les éléments $c_1, ..., c_k$ dans les équations forment une liste triée de résultats, N est le nombre total d'éléments dans la collection, $e(ci)$ et $s(ci)$ sont les valeurs d'exhaustivité et de spécificité de l'élément ci, $|ci|$ est la taille de l'élément et $|c'i|$ est la taille d'un élément c' qui n'a pas été précédemment vu par l'utilisateur.
Au lieu de mesurer le rappel et la précision après qu'un certain nombre d'éléments ait été retrouvé, la taille totale de l'élément retrouvé est utilisée comme paramètre de base, alors que l'imbrication est traitée en ne considérant que les parties de l'élément qui n'aient pas déjà été vues (on considère alors que l'information pertinente est répartie uniformément au sein d'un élément).

Comme cette mesure traite les deux dimensions de pertinence séparément, de nouvelles fonctions ont été définies pour fournir une normalisation séparée de l'exhaustivité et de la spécificité [Govert et al, 03].

Cependant, cette mesure ne prend pas en compte un problème essentiel de l'évaluation : *la surpopulation de la base de rappel* [Kazai et al, 04a], qui est due aux règles d'inférence utilisées lors de l'élaboration des jugements de pertinence [Piwowarski, 03b] : si un noeud est jugé pertinent, ses ancêtres doivent aussi être jugés pertinents, même si leur degré de pertinence est moindre. 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.

3) La mesure XCG (XML Cumulated Gain)

Pour remédier à ce problème, Kazai dans [Kazai et al, 04a] établit 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 [Jarvelin et al, 02].
Les mesures de gain cumulatif ont été développées pour évaluer les systèmes selon le degré de pertinence des documents retournés. La motivation derrière XCG est d'étendre les mesures de gain cumulatif au problème des éléments imbriqués. Les premiers tests de fiabilité de la mesure sont encourageants [Kazai et al, 04b].

Dans INEX 2005, elle a été considérée comme la mesure d'évaluation officielle. La mesure XCG inclue des mesures normalisées du gain cumulé "*nXCG*" qui sont orientées utilisateurs (*user-oriented metric*), et des mesures orientées *effort-précision/gain-rappel (ep/gr)*. Plus de précision sur cette mesure se trouve dans [Kazai et al, 05].

4) La mesure PRUM (Precision-Recall with User Model) [Piwowarski et al, 07]

Cette mesure étend la mesure de Rappel-Précision de [Raghavant et al, 89] pour inclure le comportement de l'utilisateur lors de sa navigation dans les résultats. PRUM définit un comportement stochastique de l'utilisateur en estimant la probabilité qu'un utilisateur a vu un élément particulier, par exemple : si la probabilité qu'un utilisateur a vu un élément parent est 1, alors la probabilité qu'il a vu le premier élément fils de cet élément est 0,95.
La mesure PRUM élimine l'indépendance entre des éléments de même chemin, tout en autorisant à l'utilisateur de consulter le contexte (ascendants, descendants et frères) des éléments renvoyés comme résultats.
De plus, la mesure suppose un ensemble idéal de résultats qui est défini comme la probabilité qu'un utilisateur voit un nouvel élément pertinent en consultant le contexte de l'élément retrouvé, sachant que l'utilisateur veut voir un certain nombre d'éléments pertinents.

Comme nous venons de le voir, les problèmes soulevés par l'évaluation des systèmes de RIS sont nombreux et loin d'être résolus. Ceci s'explique par la "jeunesse" des recherches dans le domaine, l'évaluation de la RIS étant née avec la campagne d'évaluation INEX. Beaucoup de propositions de mesures d'évaluation adéquate sont d'actualité.

2.4 Conclusion

L'apparition puis l'émergence des documents semi-structurés de type XML a considérablement modifié le cadre habituel de la recherche d'information.

Les modèles de documents semi-structurés ont cela de particulier qu'ils représentent le contenu sous forme d'éléments typés organisés en une structure hiérarchique. Cette structuration ajoutée au contenu textuel des documents, a en effet, remis en cause la notion même d'unité d'information recherchée. Ce changement quant à la notion d'unité d'information est à l'origine de l'apparition de nouvelles problématiques spécifiques aux documents structurés.

Les modèles traditionnels de recherche d'information ne prennent pas en compte l'information de structure qui est cependant très importante. De nouveaux modèles sont donc proposés pour l'indexation, l'interrogation et la recherche. Dans ce chapitre nous avons présenté les différentes approches proposées dans la littérature pour répondre à ces problématiques.

L'utilisation de la structure dans les différents modèles présentés en est à ses débuts. Les résultats montrent que l'utilisation de l'information de structure permet d'améliorer les résultats de recherche.

Parallèlement aux problèmes liés à la redéfinition de l'unité d'information recherchée, se pose le problème de l'interrogation de corpus hétérogènes. En effet, l'utilisateur se trouve confronté à la diversité des structures dans les corpus hétérogènes, ne pouvant connaître toutes les structures possibles. Il appartient donc au système de recherche d'information de trouver les moyens adéquats à l'interrogation de tels corpus.

Nous nous sommes intéressés en particulier à cette problématique, dans la suite du mémoire. Nous proposons de voir de plus près dans le chapitre suivant les motivations, les influences de la communauté BD, et les approches issues de la RI pour permettre une interrogation générique des corpus hétérogènes.

Chapitre 3 :

Interrogation de corpus hétérogène :

Un état de l'art.

3.1 Introduction

XML (eXtensible Markup Langage) est extensible dans le sens où il n'est pas limité à un langage prédéfini. Cela offre à l'auteur du document XML une grande liberté pour la création et l'organisation des données selon ses besoins. Cela est un grand bénéfice pour les concepteurs de systèmes spécifiques qui échangent des données pour des buts spécifiques. Mais, cela devient un problème lors de l'intégration, le stockage ou l'interrogation dans des systèmes conçu séparément. La liberté que XML offre pour la conception de documents peut conduire à la création de structures totalement différentes et aussi à des structures qui semblent différentes mais qui représentent des informations similaires. En effet, dans plusieurs cas le concepteur du document prend des décisions arbitraires sur le nombre d'éléments, leurs noms et leurs structures hiérarchiques. Par exemple, la figure qui suit illustre comment deux structures différentes peuvent décrire des collections de CDs musicaux.

Structure 1
<CDCOLLECTION>
<CD>
<NOMARTISTE>Beatles</NOMARTISTE>
<TITRE>Yellow Submarine</TITRE>
<CHANSON> Yellow Submarine </CHANSON>
<CHANSON>Help</SCHANSON>
</CD>
…
</CDCOLLECTION>

Structure 2
<ARTISTE>
<NOM>Beatles</NOM>
<CDCOLLECTION>
<CD>
<TITRE>Yellow Submarine</TITRE>
<CHANSON> Yellow Submarine </CHANCON>
<CHANSON>Help</CHANSON>
</CD>
…
</CDCOLLECTION>
<ARTISTE>

Tableau -3.1- Deux représentations structurelles différentes d'une même information.

Aujourd'hui, XML est largement accepté comme format standard pour le partage et l'échange de données dans divers domaines comme les BDs, le Web, les intranets, … En conséquence, le nombre de documents XML augmente et les intérêts sur la recherche

de données dans de larges collections de documents hétérogènes augmentent aussi. Cette hétérogénéité pose plusieurs problèmes lors de l'interrogation et la fouille de corpus de documents hétérogènes.

Jusqu'aujourd'hui, le problème de l'hétérogénéité des structures a été principalement abordé par la communauté BD. Le problème commence à peine à être abordé en RI à l'occasion de la tâche hétérogène introduite dans la campagne INEX.

Dans la suite du chapitre, nous éclairons ce problème et nous passons en revue les différentes approches de la littérature qui tentent de répondre à cette problématique.

3.2 Problème des corpus hétérogènes

L'accès aux documents semi-structurés pose un autre problème lorsqu'il s'agit de corpus hétérogène. Par corpus hétérogène on entend des corpus où les documents présentent des structures hétérogènes, c'est-à-dire:
- Des différences entre les noms des balises : on utilise souvent un vocabulaire différent pour désigner un même concept (c'est la variation linguistique).
- Des différences dans la structuration des balises (leur agencement ou leur nombre).

Pour comprendre ce problème, considérons un corpus de documents structurés suivant les structures représentées dans le tableau 3.1. Considérons de plus un utilisateur qui veut interroger ce corpus. Il va utiliser une requête contenant des conditions de contenu et de structure, par exemple : « Je cherche un CD dont le NOMARTISTE est "*Beatles*" le TITRE est " *Yellow Submarine*" ». Trouver la réponse à la requête est trivial pour un document qui possède la structure1 car l'information recherchée est parfaitement localisée. Cependant, pour un autre document qui possède plutôt la structure2, ceci est plus difficile car il faut localiser l'information recherchée dans des parties au contenu peu structuré. La différence entre les deux structures du tableau 3.1, donne un bon aperçu des problèmes posés par l'hétérogénéité.

La problématique considérée ici, est donc comment surpasser ces différences ? Par quel procédé est-il possible de réconcilier d'une façon automatique ces documents, pour permettre une interrogation simplifiée et une recherche efficace aboutissant à des résultats couvrant toute la collection.

C'est une problématique émergeante et particulièrement vaste car elle concerne des domaines aussi variés que les bases de données à travers la tâche de « *Schema matching* », le domaine de gestion de documents numériques avec la problématique de conversion des formats et plus récemment le domaine de la RIS avec la tâche d'interrogation de corpus de documents XML hétérogènes.

Plusieurs solutions ont été proposées dans la littérature. Nous présentons d'abord les principaux travaux effectués dans la communauté BD et le domaine de la gestion de documents numériques, puis nous nous focalisons sur les travaux issus de la communauté RI.

Les solutions proposées dans le domaine des BD peuvent être distinguées selon trois approches : approches de spécification de transformation, approches de Schema Matching, et approches sémantiques. Nous les décrivons dans la suite :

- **Approches de spécification de transformation** : Elles se fondent sur un ensemble de spécifications exprimant les transformations que doivent subir les éléments du document source pour les rendre compatibles avec la structure cible. Divers langages simples et hautement déclaratifs de transformation [krishnamurthi et al, 00] et [Tang et al, 01], ont été introduits comme solutions pour éviter la programmation. Des outils graphiques spéciaux sont aussi proposés pour assister à la spécification des transformations [Pietriga et al, 01] et [XSLWIZ, 01]. [Vernet, 02] offre plus d'exemples sur ces langages et ces outils, qui sont très efficaces pour la spécification de transformations.
Cependant, elles nécessitent que les développeurs indiquent manuellement les mappings pour chaque source et cible. Cette tâche qui est aujourd'hui laborieuse et inacceptable pour des volumes de données qui sont grands et qui changent comme ceux des grandes bases de données ou à l'échelle du WEB.

- **Approches de Schema Matching** : Une stratégie alternative utilisée pour réconcilier des documents XML est basée sur des techniques de Schema Matching pour automatiser la tâche du mapping. Elle consiste à trouver des correspondances entre des schémas hétérogènes de données. C'est une tâche ancienne et largement étudiée dans les bases de données pour l'intégration et l'interrogation de sources hétérogènes. L'article de [Rahm et al, 01] présente une vue d'ensemble des approches d'automatisation de cette tâche.
Bien que beaucoup de travail est fait dans ce domaine, diverses issues restent non résolues lorsque cette approche est utilisée dans le cadre de transformation de documents XML, en particulier les problèmes liés à l'évolution des structures et la découverte de relations sémantiques.

- **Approches sémantiques :** Elles offrent un nouveau moyen pour l'interrogation générique de données de structures hétérogènes fondé sur le partage d'une ontologie commune aux différentes sources. Un schéma global est supposé constituer lui-même une ontologie complète du domaine (ontologie formelle), qui apparaît l'approche la plus prometteuse pour une possible automatisation. En effet, plusieurs projets (PICSEL[5], Xyleme[6], C-web[7] et autres), permettent la médiation entre sources de données hétérogènes,

[5] PICSEL project, http://www.iri.fr/~picsel/
[6] Xyleme, http://www.xyleme.com.
[7] C-web project, http://www.cweb.inria.fr

en construisant un médiateur fondé sur la définition d'un schéma globale considéré comme l'ontologie du domaine.

Dans la suite, nous présentons des exemples représentatifs de ces approches. On ne s'intéressera qu'aux approches automatiques ou semi-automatiques. Les approches de spécification de transformations ne pouvant être considérées comme une bonne solution dans le cadre de volume important de données.

3.2.1 Un algorithme efficace de Schema Matching pour une transformation automatique de documents XML

Bagy dans [Bagy] propose une approche automatisée pour la transformation de documents XML en utilisant un algorithme de schema matching fondé sur des mappings one-to-one. L'approche est composée de deux étapes : schema matching et génération automatique de script XSLT basée sur les relations déduites de la première étape. Il utilise pour la modélisation des documents un arbre ordonné, dont chaque nœud possède un label et une valeur. Les éléments et les attributs de la DTD du document deviennent des nœuds de l'arbre. Un nœud prend le nom d'un élément ou d'un attribut comme label. La valeur du nœud sera le type d'un élément ou celui d'un attribut. Uniquement les nœuds feuilles auront des valeurs. L'algorithme de schema matching se déroule comme suit :

3.2.1.1 L'algorithme de schema matching proposé

L'algorithme est composé de deux étapes (comme l'illustre la figure ci-dessous) :

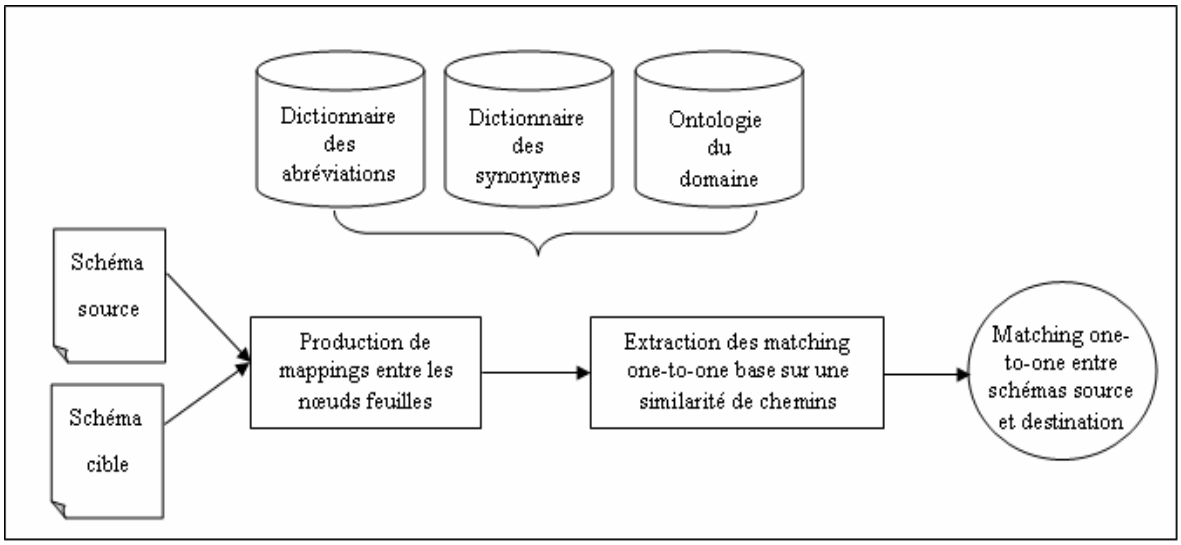

Figure -3.1 - Le processus de schéma matching proposé.

1) Production des mapping entre les nœuds feuilles : Des mappings préliminaires sont trouvés entre les nœuds feuilles dont la similarité est supérieur à un seuil "*ThLeaf*". La similarité ici est calculée par :

$$LeafNodeSimilarity(N_s, N_t) = w_l \cdot LexicalSimilarity(N_s, N_t) + w_t \cdot DataTypeSimilarity(N_s, N_t)$$

Où : - w_l et w_t sont des poids proposés pour *LexicalSimilarity* et *DataTypeSimilarity* respectivement ;
- *LexicalSimilarity* : calcule le degré de similarité entre les labels de deux nœuds N_s et N_t. Elle est donnée par la relation :

$$LexicalSimilariy(N_s, N_t) = \sum_{1 \le i \le n, 1 \le j \le m} \frac{Similarity(T_{si}, T_{ti})}{n+m}$$

où : T_{si} est un token du label source, T_{tj} est un token du label cible. Pour cela, les labels sont analysés pour former des tokens. Si les tokens appartiennent au dictionnaire des abréviations, ils sont remplacés par leurs noms complets. Puis, leur similarité est calculée par l'utilisation d'un dictionnaire de synonymes et d'une ontologie du domaine. Si deux tokens sont identiques, leur similarité = 1.0, s'ils sont synonymes, leur valeur par défaut est 0.8 ;
- *DataTypeSimilarity* : définit le degré d'information préservé lorsqu'un nœud source est transformé en un nœud cible. Ce facteur est classé en quatre groupes selon le degré de perte d'information lors d'une transformation : égale, sans perte, avec quelque perte et impossible. Elles sont assignées des valeurs de similarité de 3 à 0, la plus grande valeur attachée à la transformation égale, et la plus petite à la transformation impossible.

2) Extraction des mapping one-to-one : Les résultats de la première étape sont des matching many-to-many. Dans cette phase seront extraits les matching one-to-one en se basant sur une similarité de chemins : pour deux nœuds N_s et N_t qui sont reliés dans la première étape, la valeur de la similarité de chemins est calculée ainsi :

$$PathSimilarity\ (N_s, N_t) = \frac{Le\ nombre\ de\ nœuds\ liés\ entre\ Ps\ et\ Pt}{|Ps| + |Pt|}$$

où : *Ps* (respec. *Pt*) est le chemin de la racine au nœud feuille N_s (respec. N_t).
Avant d'effectuer ce calcul, le mapping doit être fait pour les nœuds appartenant aux chemins *Ps* et *Pt*. Si la similarité entre deux nœuds N_s et N_t internes est supérieure à un seuil "*ThInternal*", ils sont reliés. Elle est calculée par :

$$InternalNodeSimilrity\ (N_s, N_t) = w_l .LexicalSimilarity\ (N_s, N_t) + w_s .StructuralSimilarity\ (N_s, N_t).$$

Où : - w_l et w_s sont des poids proposés pour *LexicalSimilarity* et *StructuralSimilarity* respectivement ;
- Le facteur *StructuralSimilarity* représente le taux des nœuds feuilles communs inclus entre les sous arbres dont les racines sont N_s et N_t, comme suit :

$$StructuralSimilarity\ (N_s, N_t) = \frac{Le\ nombre\ de\ nœuds\ liés\ entre\ LNs\ et\ LNt}{|LNs| + |LNt|}$$

Où : *LNs* (respec. *LNt*) est l'ensemble des nœuds feuilles du sous arbre de racine N_s (respec. N_t).
Après le calcul de similarité de chemins pour toutes les paires de nœuds feuilles liées, si la valeur de cette similarité est inférieure à un seuil "*ThPath*", le matching est enlevé.

3) Résoudre les mapping one-to-many et many-to-one : Si un nœud source possède un mapping one-to-many, le matching avec une grande similarité de chemins est choisi. S'il existe plus d'un mapping avec la même similarité de chemin, la similarité des nœuds feuilles est considérée. Si un nœud cible possède un mapping one-to-many (ie : mapping many-to-one), le mapping avec une grande similarité de chemin est choisi.

3.2.1.2 Discussion

Les expériences effectuées sur différentes collections de documents XML, montrent une bonne performance, avec une précision moyenne de 97% et un rappel moyen de 81%. Des expériences sont aussi effectuées pour déterminer les bonnes valeurs des paramètres de la méthode. Ainsi, la valeur de "*ThLeaf*" est ajustée à 0.6, "*ThPath*" est ajusté 0.3, les valeurs de 0.8 et 0.2 sont déterminées comme les poids w_l et w_t la plus grande valeur d'entre elles étant attribuée à la similarité lexicale du fait que les documents XML utilisent peu de types de données comme le string.
Cependant, la performance de la méthode dépend de l'utilisation d'une ontologie du domaine et du dictionnaire des synonymes. De plus pour des transformations sophistiquées incluant des opérations de "split" et de "merge", il est important de considérer les mapping complexes ; exemple : pour résoudre le mapping de : Name avec FirstName et LastName.

3.2.2 Approche basée sur un modèle conceptuel pour l'automatisation de la transformation de documents XML

Boukottaya et al. dans [Boukottaya et al, 04] considèrent que le vrai obstacle à la transformation directe entre deux documents XML est qu'il y a lieu de réaliser des mapping manuellement. De plus, l'utilisation des algorithmes courants pour l'automatisation de la tâche du Schema Matching souffre de deux problèmes majeurs à savoir l'évolution possible à cause de la diversité des constructeurs des structures et du besoin de réaliser des matching sémantiques. Pour cela, ils proposent un modèle en couche pour l'interopérabilité entre les schémas XML ou LIMXS (Layered Interoperability Model for XML Schemas). LIMXS offrent une vue sémantique à travers la spécification de concepts et de relations sémantiques entre eux (c'est la couche sémantique). Il offre de plus

une vue logique qui permet d'établir les correspondances entre les schémas au niveau logique. Le modèle sera détaillé ci-dessous.

3.2.2.1 Modèle en couche pour l'interopérabilité de schémas XML

Il comprend deux couches, à savoir :

1) La couche sémantique : Décrit les concepts et les relations sémantiques des schémas sources et cible indépendamment des considérations d'implémentation. Ce modèle conceptuel comprend : - un ensemble de concepts (chaque concept possède un nom et éventuellement un ensemble de propriétés) représentant les objets du monde réel partageant une même structure et une même sémantique, - un ensemble de relations sémantiques entre objets : Généralisation/Spécialisation qui sont inférées grâce au mécanisme d'extension/restriction des types des schémas XML, association qui désigne que deux concepts sont au même niveau et elle est inférée grâce au mécanisme de substitution et la définition de l'intégrité des références (Key/Keyref) des schémas XML, et agrégation qui est une relation de composition entre concepts et elle est inférée des structures hiérarchiques exprimées par des moyens comme les constructeurs de "sequence", "choice" et "all". – et un ensemble de contraintes qui définit les multiplicités au niveau des concepts et des relations.
Un ensemble de règles basées sur l'interprétation par défaut des constructeurs des schémas XML, est proposé pour définir les correspondances entre ces constructeurs et les constructions du modèle conceptuel. Deux ensembles de règles sont distingués : les règles de découverte de concepts et celles de découverte de relations.

2) La couche schématique : Pour dépasser les limites des DTD dans la modélisation de documents XML, plusieurs langages de définition de schémas sont introduits pour décrire la structure logique des documents. Pour prendre en considération la différence entre ces langages, la couche schématique est composée de deux sous couches :

- Une couche logique qui donne une vue orientée objet de l'information indépendamment du langage de schéma XML utilisé. Son but est de décrire les constructions des différents langages de schéma sans considération des détails syntaxiques. Ainsi, pour chaque type simple de schéma, correspond un objet de base ayant un contenu de base et un ensemble de contraintes associées. Un contenu de base comprend une valeur atomique de type de base (string, integer et date) ou une valeur construite qui peut être une liste ou une union de valeurs. Les contraintes peuvent être : unicité, intégrité de référence ou des contraintes sur le domaine.
- La couche sérialisation qui décrit une implémentation spécifique des structures de données décrites dans le modèle logique, et dépend d'un produit ou d'une version particulière. Dans ce contexte de travail, il est utilisé le standard définit par le WWW XML Recommandation.

3.2.2.2 Les opérations de transformation

Deux groupes d'opérations de transformation sont distingués basés sur la cause de l'hétérogénéité :

- **Opérations primitives conceptuelles** : définies sur les concepts et les relations. Elles incluent les opérations : Add (ajoute d'un concept, une relation ou une propriété au schéma cible), Delete (effectue l'opération inverse de Add), Merge (mélange deux entités différentes en une seule entité), Split (effectue l'opération inverse de Merge), Rename (change le nom d'un concept ou d'une propriété) et Connect (connecte deux entités équivalentes sans aucune transformation).

- Opérations logiques : définies sur les types des contenus textuels. Après une analyse des schémas XML, un ensemble de fonctions communes est donné dans une librairie qui peut être facilement étendue ou modifiée. Des exemples de ces fonctions sont : ronder un réel à un entier, tranquer un string à une taille donnée, traitement de différences entre les contraintes de schéma comme les cardinalités des valeurs par défaut et autres.

3.2.2.3 Le processus de matching

Un processus de matching dynamique est proposé. Il comprend : un matching sémantique (SM) qui a pour but de découvrir les similarités exprimées en termes de relations sémantiques entre les modèles sémantiques de la source et de la cible. Après que ces matchings soient générés et validés par une intervention humaine, le résultat passe à la couche logique (LM) qui réalise les mappings au niveau logique, cela consiste à la découverte et la résolution de conflits logiques comme des différences de représentation (liste, choix) et les conflits de types et de contraintes. Finalement, un script de transformation peut être généré automatiquement par la couche de transformation de données (DT). Ce processus est illustré dans la figure ci-dessous.

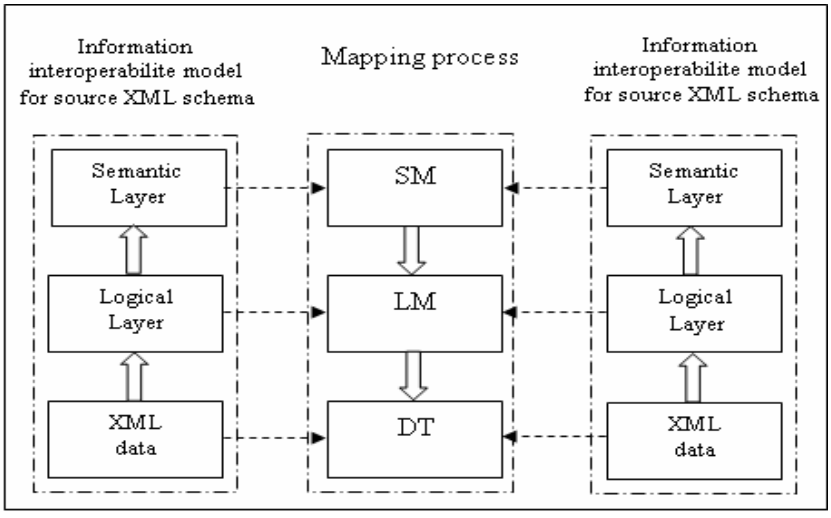

Figure -3.2 - Approche de Matching en couches.

Le matching sémantique consiste en deux étapes, une analyse linguistique des significations des concepts en utilisant WordNet [Miller, 95], où sont considérés cinq relations sémantiques : équivalent, général, spécifique, compatible et nom compatible. Puis, une analyse structurelle dont le but est d'étendre des résultats de l'étape précédente en examinant l'organisation des données, décrite par les relations d'agrégation, d'association et de généralisation/spécialisation. Le résultat du matching sémantique est un ensemble d'éléments sémantiquement liés. La clé caractéristique de l'approche étant de fournir une méthode de modélisation en couche qui inclut la modélisation des mappings et des transformations, le résultat exploite à la fois l'information sur l'interopérabilité des modèles, l'opération primitive de transformation à appliquer et la librairie de fonctions de transformation.

3.2.2.4 Discussion

L'approche est très intéressante, cependant elle se base sur l'utilisation des schémas XML et ne peut donc être appliquée que sur des documents ayant des schémas XML, alors qu'une grande partie de ces documents sont décrits plutôt par des DTD ou n'ont pas de schéma.

3.2.3 Automatisation de la transformation de documents XML

Su et al. dans [Su et al, 01] ont étudié le problème de la réconciliation automatique de différentes structures entre des documents de même type, ie : appartenant au même domaine, et cela en transformant un document de sa DTD origine à une DTD destination. D'abord, chaque DTD source ou cible est modélisée sous forme d'arbre en définissant un modèle de données pour les DTD en arbre. Le problème est ainsi ramené à celui de la comparaison et de la transformation de deux arbres de DTD. Un ensemble d'opérations de transformation, qui établit les relations sémantiques entre deux schémas, a été défini. De plus, pour choisir une meilleure séquence d'opérations à appliquer parmi toutes les séquences possibles, un modèle de coût était proposé.

3.2.3.1 Le modèle de données des DTD

Une analyse de la construction des DTD et la façon dont sont définis les types d'éléments à mener à la modélisation des déclarations des types d'éléments sous forme d'arbres, noté par $T = (N, P, L)$, où : N est l'ensemble des nœuds, P est la relation de parenté entre deux nœuds et L est la fonction de "labling" qui représente les propriétés d'un nœud. Un nœud n de N est caractérisé donc par son label $L(n)$ comme suit :

- Nœud tag : chaque nœud élément n est associé à un type élément T, $L(n)$ correspond au nom de son T et chaque nœud attribut n est associé à un type attribut T, $L(n)$ sera un

quadruple de la forme *[nom, type, def, val]* où : nom est le nom de *T*, type est le type des données de *T* (exemple CDATA), def est la propriété par défaut de *T* (exemple #required) et la valeur default ou fixed de *T*.
- Nœud contrainte : un nœud liste *n* est créé pour indiquer comment les nœuds fils sont composés, par séquence, *L(n) = [",]*, ou par choix, *L(n) = ["/"]*. Un nœud quantifié *n* est créé pour dénoter si ses nœuds fils occurrent, une ou plusieurs fois, *L(n) = ["+"]*, zéro ou plus, *L(n) = ["*"]*, zéro ou une seule fois, *L(n) = ["?"]*.

3.2.3.2 Les opérations de transformation

Une étude des différences entre les DTD a menée à la proposition des opérations suivantes : *Add (T,n)* (ajoute le sous arbre *T* sous le nœud n), *Insert (n, p, C*) (insère un nouveau nœud *n* sous le nœud *p*, avec *n* un nœud quantifié ou liste et déplacer un sous ensemble *C* des fils de *p* au-dessus de *n*), *Delete (T)* (inverse de Add), *Remove (n)* (inverse de Insert), *Rlabel (n, l , l')* (change le label du nœud de *l* à *l'*, deux sortes de changements peuvent être opérés soit dans le même type ou entre des types différents), *Unfold (T, <T_1, T_2, ..., Ti>)* (remplace le sous arbre *T* avec la séquence T_1, T_2, ...,T_i, *T* doit prendre comme racine un nœud quantifié et la séquence T_1, T_2, ...,T_i sont des frères adjacents et leurs sous arbres non quantifiés par "?"), *Fold (<T_1, T_2, ...,T_i>, T)* (inverse de Unfold), *Split (m, <n_1, n_2>)* (un nœud séquence *m* est divisé en un nœud n_1 quantifié "*", et un nœud choix n_2, cette opération correspond à la conversion d'une séquence ordonnée en une autre non ordonnée), et *Merge (<n_1, n_2>, m)* (opération inverse de split). Les opérations add, insert, delete, remove, et relabel effectuent des mappings one-to-one, celles de fold, unfold, merge et split effectuent des mappings one-to-many. Des contraintes sur ces transformations sont définies basées sur des observations sur les DTD. En effet, les déclarations des types d'éléments sont d'une profondeur ne dépassant pas trois, cela est dû au fait que les expressions régulières complexes ne sont pas utilisées à cause de leur difficulté de compréhension. Cela donne l'allusion que les DTD sont conçues de façons similaires. Donc, ils ne seront découverts que les scripts de transformation qui ne violent pas la contrainte suivante : un nœud peut être opéré par une seule opération différente de *"Relabel*" optionnellement suivie ou précédée par une opération "*Relabel*".

3.2.3.3 Un modèle de coût pour les opérations de transformation

Les opérations peuvent être combinées en une variété d'équivalents scripts de transformation. Pour faciliter la sélection entre eux, le modèle de coût est proposé qui évalue le coût des opérations en terme de leur impact sur la capacité d'information des documents. Le coût d'une opération augmente avec l'augmentation de la perte d'information q'elle cause lors d'une transformation. La capacité d'information d'un document XML est considérée comme son contenu PCDATA et les valeurs de ses attributs. Le modèle de coût d'une opération est donné par :

$$Cost\ (op) = (DC\ (op) + PDC\ (op)\ ) * Fac\ (op)$$

Où : - DC (op) (pour Data Capacity gap) est un facteur qui dénote le coût attaché à la différence de la capacité d'information causée par l'opération. Sa valeur est comprise entre 0 et 1, et on distingue 4 classes de valeurs qui sont données dans l'ordre croissant de leurs coûts : *DC-Preserve* (opérations qui conservent les données), *DC-Increase* (opérations causant une augmentation de données, comme exemple l'opération Add d'un sous arbre), *DC-Ambiguous* (affectée aux cas où il est difficile de déterminer l'influence de l'opération qui dépend de l'instance du document, comme exemple : la suppression d'un nœud "*" change l'occurrence du contenu de "non requis" à celle de "requis" qui peut causer l'augmentation de la capacité d'information, mais elle change aussi l'occurrence du contenu de "répétable" à celle de "non répétable" qui a l'effet inverse que le précèdent), et *DC-Reduce* (opération de transformation qui induit une perte de données, comme exemple l'opération Delete d'un sous arbre).

- PDC (op) (pour Potentiel Data Capacity gap) : Dénote le coût d'une différence potentielle de la capacité d'information. En effet, une opération comme celle d'insérer un nœud quantifié "+" bien qu'elle est de la classe DC-Preserve, elle change l'occurrence d'un contenu de "contable" à "non contable" permettant ainsi au document XML d'accommoder plus de données. Uniquement les opérations Insert, Remove et Relabel peuvent avoir un PDC non nul. Il est donné par la relation :

$$PDC\ (op) = w_{req} * required\text{-}change\ (op) + w_{count} * countable\text{-}change\ (op)$$

où : "*required-change*" *et* "*countable-change*" sont deux fonctions booléennes déterminant si les propriétés "*required*" ou "*countable*" d'un élément ont été changées, et les poids w_{req} et w_{count} indiquent l'importance du changement opéré avec $w_{req} + w_{count} = 1$.

- Fac (op) : (pour Relative Factors of Operands) : Dénote l'effet que peut avoir le nombre, la taille ou les propriétés des opérandes mêlés à une opération de transformation sur les facteurs DC et PDC.

3.2.3.4 Génération d'arbres de DTD égaux

1) Arbres d'éléments simplifiés : Comme première mesure heuristique possible de similarité entre les tags de deux nœuds, sera utilisée la similarité des noms. Ainsi, est introduit le concept d'arbre d'éléments simplifié conçu pour capturer les relations entre noms de deux arbres basé sur les informations dictées par une ontologie du domaine. Un nœud tag d'une DTD source sera dit "non renommable" s'il existe un tag dans la DTD cible dont le nom est le même ou reconnu pour être synonyme. Un arbre d'éléments simplifié de type E, noté ST, est un sous arbre de la déclaration de son arbre de type T, qui

a pour racine celle de T et dont les branches se terminent au premier nœud non renommable rencontré dans l'arbre.

2) Algorithme de matching : L'entrée de l'algorithme nommé *DMatch* consiste en deux arbres simplifiés d'éléments T_1 comme source et T_2 comme cible. *DMatch* (n_1, n_2), avec n_1 et n_2 des nœuds de T_1 et T_2 respectivement, découvre la séquence d'opérations qui transforme le sous arbre de racine n_1 à celui de racine n_2. Les nœuds enlevés sont dits reliés à des nœuds spécifiques Φ_1, et ceux supprimés sont dits reliés à des nœuds spécifiques Φ_2.
L'algorithme est composé de deux phases :
- La première est une phase de prétraitement, dans laquelle toutes les séquences de contenu non répétables sont converties à des contenus répétables en effectuant des opérations de fold tout en marquant les nœuds concernés et cela pour ne pas perdre l'information nécessaire pour le calcul du coût total. Des opérations merge sont aussi effectuées pour imposer un ordre sur des représentations permettant des combinaisons arbitraires d'éléments.
- La seconde phase consiste à trouver les mappings one-to-one. Pour dériver la transformation du sous-arbre de racine n_1 à celui de racine n_2, pour chaque fils m_1 de n_1 l'algorithme tente de trouver son correspondant m_2 dans n_2. Cette phase se déroule en deux passes. Dans la première, les nœuds fils de n_1 sont visités séquentiellement et sont comparés avec certains nœuds de n_2 qui forment l'ensemble des nœuds candidats aux mappings de la 1^{ière} passe, noté S (comme illustré dans le tableau ci-dessous). Selon la contrainte qu'un nœud ne peut être opéré qu'une seule fois, le partenaire m_2 de m_1 peut être de même niveau (aucune opération n'est faite sur m_1), d'un seul niveau inférieur (une opération insert est opérée sur m_1) ou un nœud spécifique Φ_1 ou Φ_2 (une opération delete ou remove faite sur m_1).
L'algorithme est appliqué récursivement sur m_1 et tout élément de S, jusqu'à trouver l'élément s_k de S avec le coût c le moins élevé. Pour décider d'accepter le mapping de m_1 à s_k, la stratégie adoptée consiste à comparer c au coût de l'opération de delete m_1. Si c> coût de delete, le mapping est retardé à la deuxième passe.

SOURCE	MATCHING CANDIDATE SET
Element	Element node on the same level.
Attribute	Attribute node on the same level.
Choice	Choice node on the same level.
Sequence	Sequence node on same level or one level deeper; Φ_1.
quantifier	Quantifier node on same level or one level deeper; Φ_1.

Tableau -3.2- Ensemble des nœuds candidats au matching de la 1^{ière} passe.

La seconde passe commence lorsque tous les nœuds fils de n_1 sont visités. Tous les nœuds fils de n_1 qui n'ont pas de correspondant dans n_2, sont comparés avec des nœuds de n_2 qui

forment l'ensemble des nœuds candidats aux mappings de la 2 ième passe, noté *S'* (comme décrit dans le tableau ci-dessous). Un nœud m_1 sera lié à un nœud s_k de *S'* si le coût de ce mapping est inférieur au coût de delete m_1 et add s_k.

SOURCE	MATCHING CANDIDATE SET
Element	Element node on same level or one level deeper; Sequence node on same level; Attribute node on same level.
Attribute	Element node on the same level.
Choice	Choice node on same level or one level deeper.
Sequence	Sequence node on same level or one level deeper; Φ_1; Quantifier node on same level.
quantifier	Quantifier node on same level or one level deeper; Φ_1; Sequence node on same level.

Tableau -3.3- Ensemble des nœuds candidats au matching de la 2ième passe.

Après que les relations entre les deux DTD source et destination soient trouvées par l'algorithme DMatch, un générateur automatique XSLT générera un script de transformation XSLT des documents de la DTD source à celle cible.

3.2.3.5 Discussion

Des expériences pour vérifier l'efficacité de la méthode sont faites sur un prototype de système XTRA et sur des collections de documents XML réels et synthétiques. Les résultats montrent que l'algorithme découvre d'une façon satisfaisante des transformations acceptables. Cependant, la méthode ne reflète pas bien les informations hiérarchiques de structures des schémas des documents XML, de plus elle est dépendante des DTD.

3.2.4 Approche statistique pour la correspondance de schémas

He et al. dans [He et al, 03] proposent une approche holistique (voir figure 3.3) pour établir les correspondances entre les schémas de différentes sources de données relatives au même domaine sur le web. Leur challenge était de trouver, typiquement sans même comprendre la sémantique, les attributs synonymes entre les différentes sources.
Partant des observations que parallèlement à la prolifération des sources des documents décrivant le même domaine, les vocabulaires des schémas correspondants tendent à converger. Ils supposent l'existence d'un modèle (hypothèse) de schéma génératif qui unifie tous les schémas, et proposent un cadre statistique général pour la découverte d'un

tel modèle. Ils développent un algorithme spécifique pour la découverte des attributs synonymes entre les différents schémas. L'algorithme comprend trois étapes principales : modélisation de l'hypothèse, génération de l'hypothèse et sélection de l'hypothèse. Dans la suite, nous détaillons cet algorithme.

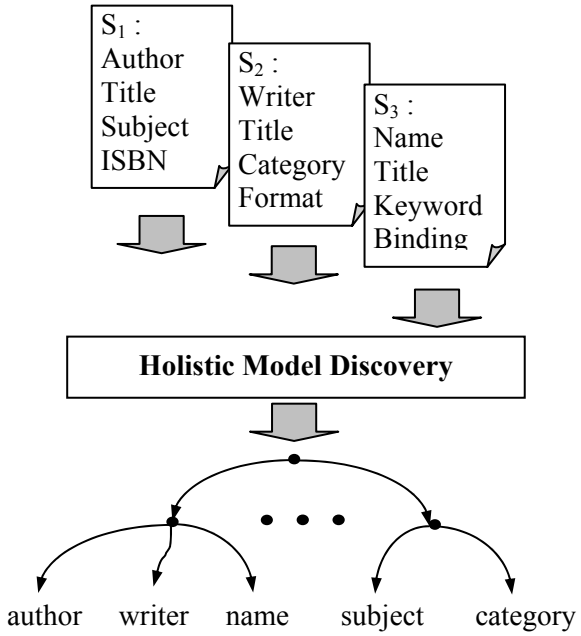

Figure -3.3- Approche holistique pour la découverte de model.

3.2.4.1 Modélisation de l'hypothèse

Pour définir la structure du modèle qui sous-tend un ensemble de schémas de sources de données, chaque schéma observé est d'abord représenté par la suite des attributs qu'ils contient. De plus, pour avoir un modèle simple, un ensemble de suppositions est fait sur la façon dans sont générés les schémas. D'abord, les différents concepts d'un schéma sont supposés être sélectionnés indépendamment (supposition 1). Puis, un même schéma ne peut contenir des attributs synonymes (supposition 2). La dernière supposition, est la non existence de concepts qui se recouvrent (supposition 3). La définition de la structure du modèle se fait en deux étapes :

3.2.4.1.1 La structure du modèle

Un modèle décrit comment sont générés les schémas à partir d'un vocabulaire d'attributs. Pour exprimer la synonymie, le modèle partitionne l'ensemble des attributs en des concepts. Puis décide pour chaque concept s'il sera inclus. Si c'est le cas, il sélectionne un attribut qui le représente dans le schéma généré. Ce processus génère un schéma comme un ensemble d'attributs. Formellement, un modèle de schémas M est un 4-tuple (V, C, P_c, P_a), où V désigne le vocabulaire de l'ensemble des attributs $\{A_1, ..., A_n\}$. C est un ensemble de concepts $\{C_1, ..., C_m\}$ qui partitionne V ($V=UC_i$ *avec* $C_i \cap C_k = \phi$). P_c est la fonction probabilité des concepts, elle détermine la probabilité α_i pour inclure un concept C_i dans

une génération de concepts. P_a est la fonction probabilité des attributs, elle détermine la probabilité β_j pour sélectionner un attribut A_j sachant son concept inclus. Pour chaque concept C_i on a: $\sum_{A_j \in C_i} \beta_j = 1$. Le modèle sera noté comme dans l'exemple de la figure 3.4.

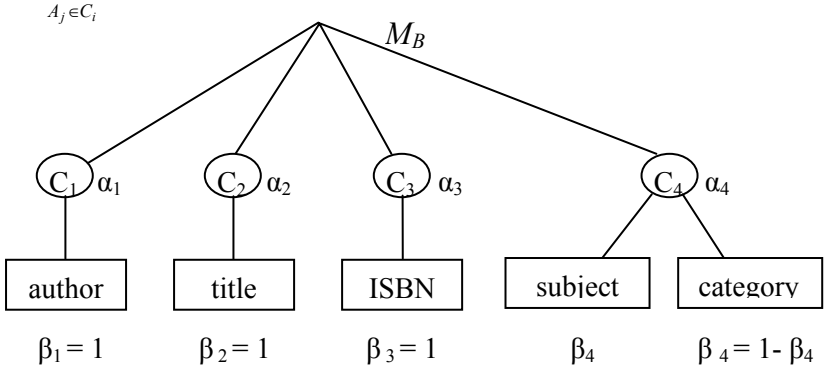

Figure -3.4- Un exemple de modèle de schémas M_B (pour les ressources décrivant les livres "Books").

3.2.4.1.2 Génération de schémas et observations

Le modèle *M* représente un processus de distribution générative donnant une probabilité pour tout schéma qui peut être généré. Pour générer un schéma, le modèle sélectionne les concepts à inclure. Un concept C_i apparaîtra avec la probabilité :

$$P(C_i / M) = \alpha_i, \quad \textbf{(1)} \text{ autrement } P(\neg Ci / M) = 1 - \alpha_i.$$

Pour choisir un attribut A_j sa probabilité sera :

$$P(A_j / M) = \begin{cases} \alpha_i \times \beta_j, & \exists i : A_j \in C_i \\ 0, & \text{sinon.} \end{cases} \quad \textbf{(2)}$$

Pour sélectionner un ensemble d'attributs $A_1, A_2, \ldots, A_m$ à partir de *M*, la probabilité sera :

$$P(A_1, A_2, \ldots, A_m / M) = \begin{cases} 0, & \exists j \neq k, \exists i : A_j \in C_i \wedge A_k \in C_i \text{ (représente la } \textit{supposition } 2) \\ \prod P(A_j / M), & \text{sinon} \qquad \text{(avec la } \textit{supposition } 1) \end{cases} \quad \textbf{(3)}$$

On dérive ainsi la probabilité qu'un modèle $M = (V, C, P_c, P_a)$ génère un schéma $I = \{A_1, A_2, \ldots, A_m\}$, notée $P(I/M)$:

$$P(I / M) = P(A_1, A_2, \ldots, A_m / M) \times \prod_{\forall A_j, A_j \notin C_i} P(\neg C_i / M) \quad \textbf{(4)}$$

Un schéma *I* peut être alors instancié d'un modèle *M* si $P(I/M)>0$.
L'approche cherche à découvrir le modèle caché à partir d'un ensemble de schémas observés en entrée, noté : *I*. Dans cet ensemble, un schéma (I_i) peut être observé plusieurs fois (B_i). L'ensemble *I* est alors un ensemble de 2-tuple $\langle I_i, B_i \rangle$. Pour découvrir le modèle caché, il est nécessaire de répondre à la question : Etant donné un modèle *M*, comment

est-il probable que *M* génère les schémas dans *I*. Suivant l'équation (4), cette probabilité est : $P(I/M) = \prod P(I_i/M)^{Bi}$. Si $P(I/M) = 0$, il est impossible d'observer *I* au-dessus de *M*. On dit alors qu'un modèle *M* est consistant pour un ensemble d'observation *I* si $P(I/M)>0$.

3.2.4.2 Génération de l'hypothèse

Ici, on tente de construire des modèles $M = (V, C, P_c, P_a)$ candidats qui sont consistants avec les observations en entrée *I*. Pour commencer, on définit $V = UI_i$, où $I_i \in I$. Ainsi, *V* sera composé de l'ensemble des attributs apparaissant au moins dans un schéma de *I*.
Puis, la partition de concepts *C* sera construite. Il existe un nombre important de partitions qui peuvent être obtenues à partir de *V*. Ici, on ne s'intéresse qu'aux partitions qui génèrent des modèles qui ne contredisent pas les schémas observés dans *I*. Dans de tels modèles, aucun des concepts ne contiendra deux attributs A_i et A_k qui sont utilisés dans un même schéma dans *I*. Formellement, étant donné un ensemble d'observations *I* et un vocabulaire *V*. Soit $C = \{C_1, C_2, ..., C_m\}$ une partition en concepts de *V*. Un modèle *M* construit à partir de *C* sera inconsistant avec *I* (P(I/ M)=0), si deux attributs A_j et A_k vérifient les deux conditions suivantes : - 1. ∃ un schéma $I \in I$, tel que $A_j \in I$ et $A_k \in I$.

- 2. ∃ un concept $C_i \in C$, tel que : $A_j \in C_i$ et $A_k \in C_i$.

Le processus de construction d'une partition de concepts se compose de deux étapes :
- *ConsistentConceptConstruction* *:* tente de trouver tous les concepts consistants. Ce problème est ramené à celui de trouver toutes les cliques dans un graphe de co-occurrence d'attributs [Cormen et al, 01]. Un graphe de concepts sera construit à partir de *I*. Dans ce graphe, un nœud représente un attribut. Un arc reliera deux attributs uniquement s'ils ne co-occurrent pas dans un même schéma. Les attributs reliés forment un concept consistant.
- *BuildHypothesisSpace* : construit l'espace des modèles consistants à partir des concepts de l'étape précédente. Cette étape correspond au problème classique de couverture d'ensemble [Cormen et al, 01], avec la contrainte que les sous-ensembles de couverture qui ne se chevauchent pas. En d'autres termes, étant donné des sous-ensembles (des concepts consistants) de *V*, on tente de sélectionner les sous-ensembles qui ne se chevauchent pas pour couvrir *V*.
Enfin, les fonctions de probabilité sont construites. Ici, on tente de déterminer les fonctions P_c et P_a vérifiant toujours P(I /M)>0 et qui la maximise. Les valeurs de P_c et P_a doivent être celles qui rendent le modèle *M* le plus consistant avec *I*. Cela fait un problème d'optimisation à résoudre :

$$\max_{p_c, p_a} \quad P(I / M(V, C, P_c, P_a)) \quad \textbf{(5)}$$

qui correspond au problème du maximum de vraisemblance. Dans l'estimation du maximum de vraisemblance des fonctions P_c et P_a, on estime les paramètres α_i et β_j. Comme les concepts sont supposés être sélectionnés indépendamment, chaque α_i sera estimé indépendamment. Pour β_j, la sélection est dérivée de [Bickel et al, 01] puisque β_j

dans un concept C_i forme une distribution multinomiale. En somme, les solutions pour l'équation (5) sont :

$$\alpha_i^* = \frac{\sum_{A_j \in C_i} O_j}{|I|} \quad , \quad \beta_j^* = \frac{O_j}{\sum_{A_j \in C_i} O_j}$$

Où O_j est la fréquence d'un attribut A_j dans I (nombre de schémas dans I contenant A_j) et $|I|$ est le nombre de schémas dans I.

3.2.4.3 Sélection de l'hypothèse

A ce niveau, une hypothèse est un modèle déterminé : $M = (V, C, P_c, P_a)$. Il est nécessaire cependant de choisir parmi les hypothèses consistantes jusqu'ici obtenues, celles qui présentent une signification statistique suffisante. Pour cela, une technique statistique de test d'hypothèse [Bickel et al, 01], en l'occurrence X^2, a été utilisée pour quantifier le degré de la consistance du modèle avec les observations. Ainsi, supposant que l'on a n observations ($|I| = n$). Et dans chaque observation, précisément 1 parmi r évènements (schéma avec une probabilité non nulle) $I_1, ..., I_r$ doit apparaître avec les probabilités respectives $p_1, ..., p_r$ et $\sum_{j=1}^{r} p_j = 1$. Supposant de plus, que $p_{10}, ..., p_{r0}$ sont les probabilités instanciées des schémas $I_1, ..., I_r$ observés en respectant un modèle M avec $\sum_{j=1}^{r} p_{j0} = 1$. On veut tester alors l'hypothèse H : $p_1=p_{10},..., p_r = p_{r0}$ en considérant la statistique :

$$D^2 = \sum_{j=1}^{r} \frac{(B_j - np_{j0})^2}{np_{j0}}.$$

Il est clair que D^2 a asymptotiquement une distribution X^2 avec $r-1$ degré de liberté. Un test des hypothèses nulles H : $p_1=p_{10},..., p_r = p_{r0}$ au niveau de signification 100a% est obtenu en choisissant un nombre b tel que : $P\{D^2 > b\} = a$, où D^2 a une distribution X^2 à $r-1$ degrés et rejetant les hypothèses si une valeur de D^2 supérieur à b est observée.

3.2.4.4 Discussion

Une adaptation de l'algorithme a été proposée pour traiter les données réelles. Ces dernières présentent des caractéristiques qui peuvent compromettre les résultas de l'approche statistique de l'algorithme. Notamment, la distribution non uniforme des attributs ou des schémas dans les observations. Ainsi, les attributs les plus fréquents sont traités par un processus itératif de projection de consensus. Les attributs les plus rares sont traités par un processus de sélection d'attributs. La présence simultanée des attributs fréquents et les moins fréquents dans un même schéma causent l'existence de schémas rares qui peuvent compromettent les résultats des tests X^2. Ils sont traités par un processus dit Rare Schema Smoothing. Ces adaptations sont détaillées dans [He et al, 03].

Les résultats des expériences effectuées sur 200 sources de données couvrant divers domaines sur le Web ont validé l'efficacité de l'approche et ont montré de très bonne performance. L'approche est très intéressante elle doit être adaptées aux cas des documents XML présentant des schémas plus complexes.

3.2.5 Extraction d'arbres fréquents dans un corpus hétérogène de documents XML

Termier [Termier, 04] dans le cadre de la fouille de données, propose des algorithmes qui permettent de trouver des arbres fréquents parmi une collection de différents arbres. Il propose d'utiliser par la suite, les arbres fréquents trouvés pour l'interrogation de bases de données hétérogènes à partir d'un même schéma.

Les algorithmes conçus trouvent des arbres fréquents dans des arbres étiquetés ordonnés, sans tenir compte de l'ordre des frères, et en autorisant les besoins des variations dans l'imbrication des balises, comme dans [Zaki, 02]. Kimpeläinen a montré dans [Kimpeläinen, 92] que si le problème de tester l'inclusion d'un arbre dans un autre était polynomial dans le cas ordonné, il est NP-complet dans le cas non ordonné. Afin d'être efficace et flexible malgré cette plus grande difficulté, deux approches différentes ont été proposées : la première a donné lieu au développement de l'algorithme TREEFINDER, la seconde a abouti à la création des algorithmes DRYAL et DRYADE. La figure 3.5 illustre l'arbre fréquent trouvé par ces algorithmes sur les arbres A_1 et A_2 de l'exemple.

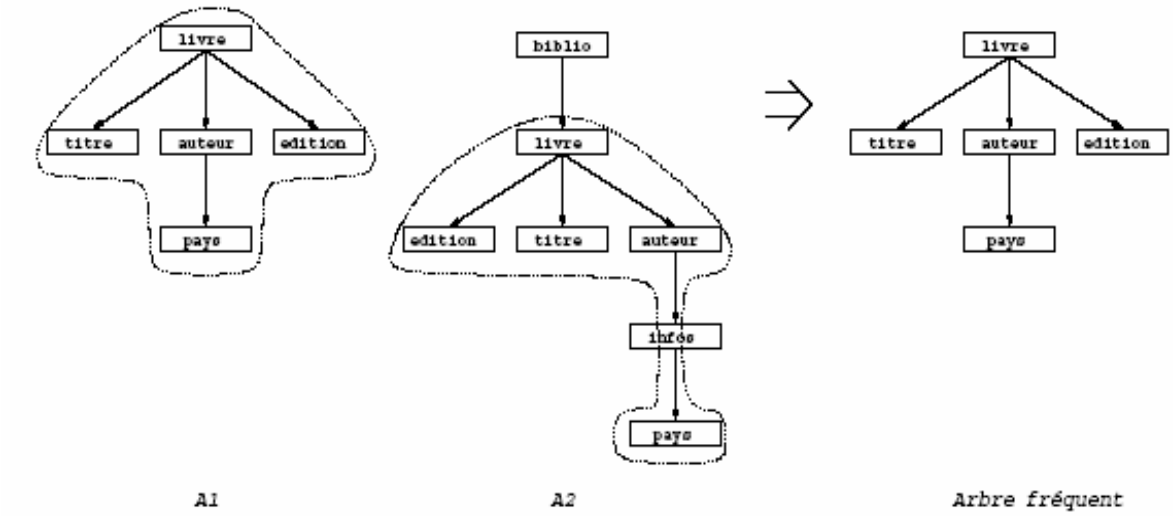

Figure -3.5- Arbre fréquent le plus gros trouvé à partir de A_1 et A_2 par les algorithmes proposés.

Dans la suite, nous décrivons d'abord comment les documents sont représentés sous forme d'arbres, puis nous décrivons ces algorithmes.

3.2.5.1 Présentation des documents par un arbre

Les documents semi-structurés peuvent être représentés par des arbres. Une collection de tels documents sera donc représentée par un ensemble d'arbres $T = \{T_1, T_2, ..., T_m\}$ qui sera regroupé dans une structure appelée : base de données d'arbres, notée : $BA = (D, \Delta, \delta)$, où :
- D est un arbre qui regroupe tous les arbres de la collection, c'est-à-dire l'arbre dont les fils seront les arbres de T. Il est appelé : *Data tree.*
- Δ est une fonction qui associe un identifiant unique à chaque arbre de T.
- δ est une fonction qui associe un identifiant unique à chaque nœud d'un arbre.

Cette structure servira de donnée d'entrée pour tous les algorithmes développés.

3.2.5.2 L'algorithme TREEFINDER : approximation de l'ensemble des arbres fréquents

L'algorithme TREEFINDER prend en entrée un ensemble d'arbres étiquetés $BA = (D, \delta, \sigma)$ dont les sous arbres fils de D sont les arbres de $T = \{T_1, ..., T_n\}$ et un seuil de fréquence ε. Il renvoie un ensemble d'arbres ε-fréquents dans BA, en utilisant la fréquence par identifiants [Termier et al, 02]. Pour découvrir des arbres fréquents l'algorithme TREEFINDER utilise la définition d'inclusion stricte faible [Kimpeläinen, 92] qui ne préserve pas l'ordre des fils et qui préserve la relation d'ancestralité plutôt que la relation de parenté.
Dans TREEFINDER, la recherche d'arbres fréquents est guidée par une première étape de clustering : les arbres fréquents sont en fait *les arbres communs* inclus dans tous les arbres d'un cluster. Pour cette étape de clustering, TREEFINDER est guidé par la recherche d'occurrences fréquentes selon un certain seuil de paires d'étiquettes pour des nœuds en relation d'ancestralité. Les clusters sont les ensembles d'arbres dans lesquels on retrouve les mêmes occurrences de paires d'étiquettes. Cette étape de clustering est effectuée en appliquant un algorithme de recherche d'itemsets fréquents maximaux au codage transactionnel des arbres [Agrawal et al, 94], où chaque arbre est représenté par l'ensemble des paires d'étiquettes correspondant aux nœuds en relation d'ancestralité.
Pour l'extraction d'arbres fréquents, l'algorithme calcule les arbres communs à tous les arbres regroupés dans chaque cluster. Pour cela, TREEFINDER utilise une méthode empruntée à la programmation logique inductive [Plotkin, 70], pour calculer *Least General Generalization (LGG)* sur une représentation adaptée aux arbres de chaque cluster.

L'intérêt de cette approche est de réutiliser des algorithmes classiques (l'algorithme APRIORI [Agrawal et al, 94], le calcul de Least General Generalization) dont l'application était rendue possible par des encodages appropriés des arbres. Termier donne une preuve sur la correction des résultas de l'algorithme. Mais, l'algorithme n'est pas complet dans la mesure où il ne trouve qu'une partie de tous les motifs fréquents présents dans les données.

L'étude expérimentale effectuée a montré qu'en pratique, tant que le taux de recouvrement entre les paires d'étiquettes ancêtre/descendant des arbres fréquents à découvrir n'est pas trop élevé, la complétude des résultats fournis par l'algorithme est acceptable.
Nous présentons dans la suite, les deux autres algorithmes proposés qui sont eux corrects et complets.

3.2.5.3 Les algorithmes DRYAL et DRYADE : une nouvelle approche complète de recherche d'arbres fréquents

Pour affronter un espace de recherche gigantesque, Termier impose une restriction sur les arbres fréquents qui peuvent être découverts par ces deux algorithmes. Ce sont des arbres où un nœud ne peut pas avoir deux fils de même étiquette. Pour les distinguer, de tels arbres fréquents sont appelés des *patterns fréquents*. Il a été montré dans [Delobel et al, 03] que ces patterns correspondent à des patterns de requêtes XML présentant un bon compromis entre efficacité et expressivité. La restriction permet d'éviter une trop grande explosion combinatoire.

L'algorithme DRYAL exploite les spécificités des structures d'arbres pour être efficace. DRYAL accepte des paramétrages lui permettant de prendre en compte différents types d'inclusion d'arbres [Kimpeläinen, 92]. Il est fondé sur les deux principes suivants :

1) Approche constructive : l'approche proposée est une approche par niveaux. Elle commence par créer les arbres fréquents de profondeur 1, puis les combine entre eux pour former les arbres fréquents de profondeur 2, et continue jusqu'à l'obtention de tous les arbres fréquents. Elle s'inspire fortement des techniques verticales récentes ([Zaki et al, 02], [Zaki et al, 03]) qui sont en général plus efficaces que les approches de type APRIORI classiques (dite horizontales) ([Agrawal et al, 94], [Houtsma et al, 95], [Savasere et al, 95], [Toivonen, 96], [Brin et al, 97]). Ce sont des approches qui se révèlent plus adaptées à la découverte de motifs fréquents complexes.

2) Diviser pour reformuler : une technique couramment utilisée en Informatique pour résoudre un problème complexe s'appelle "*diviser pour régner*". On découpe le problème en sous problèmes plus simples que l'on sait résoudre. Et à partir des solutions des sous problèmes, on reconstruit la solution du problème global. Termier s'est inspiré de cette technique pour résoudre son problème de telle sorte que les sous problèmes à résoudre se ramènent, après une *reformulation*, à des recherches d'**itemsets fréquents.** Grâce à ces reformulations, son approche bénéficie de la somme considérable des travaux effectués sur la recherche d'itemsets fréquents. La moindre amélioration sur ces méthodes est directement exploitable pour optimiser son approche.

L'algorithme DRYAL n'exploite pas le fait que si un pattern *A* est fréquent, alors nécessairement tous les patterns inclus dans *A* seront fréquents, pour gagner le temps de calcul consommé pour générer tous ces patterns fréquents inclus dans *A*. Ce défaut est commun à tous les algorithmes de découverte de motifs fréquents.

L'algorithme DRYADE propose la solution à ce problème. Termier présente pour cela la notion d'arbres fréquents fermés, qui n'a jamais été utilisée pour la recherche d'arbres fréquents, et qui permet un gain très important d'efficacité. DRYADE reprend l'approche de DRYAL et s'appuie sur une propriété algébrique (les ensembles fermés) pour trouver des patterns fréquents fermés. Il permet ainsi une résolution plus efficace au-delà des limites de DRYAL.

3.2.5.4 Discussion

Termier donne des preuves de la correction et de la complétude des résultats fournis par ces deux algorithmes. Une étude expérimentale sur un prototype implémentant l'algorithme DRYADE a démontré son efficacité à trouver la structure commune à une collection de 3000 documents XML.

Cependant, dans le cas de documents XML hétérogènes, les balises peuvent non seulement présenter des structurations différentes, plusieurs balises se référant au même concept peuvent être employées. Par exemple, pour désigner une voiture, on peut trouver des balises telles que : véhicule, voiture, automobile, auto (synonymes). Pour que l'algorithme de recherche d'arbre fréquents soit le plus efficace possible sur des documents hétérogènes, idéalement il faudrait que ces problèmes dits de synonymie et de polysémie soient résolus. Dans son travail, Termier suppose qu'il dispose d'un préprocesseur qui unifie les balises se référant au même concept, et donc il n'y a pas de problème de variation linguistique.

3.2.6 Transformation de documents structurés : une combinaison des approches explicites et automatiques

Bonhomme dans [Bonhomme, 98] propose une technique automatique de transformation de documents structurés (par exemple XML), qui repose sur le modèle d'arbre de type, dans le cadre de la conversion électronique de documents pour leur édition. Mais, qu'il est possible d'envisager comme un mécanisme de transformation de structures s'appuyant sur une interface DOM lui permettant de s'intégrer dans de nombreuses applications XML (comme l'intégration de données hétérogènes, la RI, …).

3.2.6.1 Système de type des documents structurés

L'observation de l'existence d'une analogie entre les systèmes de type de langage de programmation définissant des types de données et ceux des documents structurés

définissant des types d'éléments structurés, et la comparaison de ces deux systèmes de types, a permis d'identifier les spécificités et les caractéristiques des systèmes de type d'éléments structurés qui sont :

- Du point de vue de la définition d'un type : les constructeurs de type sont identiques quelque soit la structure générique, les éléments peuvent portés des attributs typés.
- Du point de vue de l'utilisation des données : modularité et inclusion d'éléments appartenant à des structures génériques différentes dans un même document, et traitement des documents incomplets.

Ces différences entre ces systèmes de types, font que les méthodes de conversion de variables entre types de données, qui s'appuient sur la relation d'équivalence calculée par l'algorithme d'unification défini dans [Aho et al, 91], ne peuvent être appliquées telle qu'elles à la transformation de documents structurés. De plus, au cours d'une transformation de documents, des éléments peuvent être créés ou disparaître, il est donc nécessaire de traiter des relations mettant en correspondance des types ayants des structures différentes.

L'auteur propose alors une représentation adaptée, des systèmes de type des documents structurés, qui est donnée comme suit :

- Les types de base : l'ensemble des types de base commun à toutes les DTD est :

B = {texte, symbole, graphique, image, référence}, il est étendu avec des représentations de types d'éléments vides.

- Les constructeurs, qui sont :

1. L'identité : le type est identique au type qui le définit, l'élément n'a qu'un seul fils.
2. Le choix : alternative entre un ensemble d'éléments de type différents.
3. La liste : séquence d'éléments de même type avec les opérateurs "+"et "*".
4. L'agrégation : ensemble d'éléments de type différents.

La structure générique d'un document sera représentée sous la forme d'un arbre de type canonique qui est la structure d'arbre adaptée aux algorithmes de comparaison.

Pour permettre la recherche automatique de similarité de structures syntaxiques entre types dans le but de les transformer, l'auteur définit des relations entre arbres de type pour la transformation de documents structurés, elles sont représentées par les relations antisymétriques de massif et d'absorption définies comme suit :

- ***La relation de massif*** : est une relation injective mettant en correspondance chaque nœud de l'arbre de type source avec un nœud de l'arbre de type cible de même constructeur, en préservant les relations structurales de parenté et d'ordre entre les nœuds et les types de base. Elle permet de répondre au besoin de structuration qui nécessite la création d'éléments de structure intermédiaires.

- ***La relation d'absorption*** : possède les mêmes propriétés que la relation de massif. Elle permet de répondre au besoin de réduction de structure induisant l'élimination d'éléments de la structure de l'arbre de type source n'ayant pas de correspondant dans l'arbre de type

cible. En effet, ces éléments se trouvent définis par le même constructeur que celui de leur nœud père.

Le processus de transformation s'appuiera sur la comparaison des structures génériques et la recherche de relations de massif et d'absorption.

3.2.6.2 Le processus de transformation automatique

Il se compose de deux étapes qui sont la recherche des relations entre les structures génériques (source et cible) et l'exploitation des relations produites pour effectuer les transformations :

1) Recherche des relations : Pour la recherche de relations de massif, un algorithme de comparaison de type a été défini, il présente une extension de l'algorithme de Frilley [Frilley, 89], qui implémente un automate de reconnaissance de massifs entre deux arborescences. L'automate de Frilley ne permettant pas de reconnaître tous les massifs, une modification de l'automate, est opérée pour permettre de rechercher une solution dans la descendance d'un nœud non reconnu en cas d'erreur partiel de l'algorithme. Une autre adaptation de l'algorithme a été aussi réalisée par rapport aux spécificités des arbres de types d'éléments structurés, où l'ensemble des types représentés par les nœuds terminaux (types de base) est différent de celui des nœuds intermédiaires (types construits).
L'algorithme prend en entrée un type d'élément source et un autre cible, pour produire un ensemble de relations entre ces types, appelé : couplage. Pour éviter les complexités élevées et les surcoûts des temps d'exécution, liés aux algorithmes de comparaison d'arbres, les entrées sont représentées linéairement sous forme de chaînes parenthésées appelées "*empreintes de type d'élément*".
Les relations trouvées par cet algorithme sont des relations de massif entre des types non récursifs. Pour prendre en compte les relations d'absorption et trouver des relations entre types récursifs, qui sont fréquents dans les structures considérées, la méthode de comparaison de type est étendue :

- Extensions pour la prise en compte des relations d'absorption : Pour pouvoir procéder à la recherche de relation d'absorption, des formes réduites de l'empreinte source sont définies. Cela consiste à ne pas représenter les nœuds qui ont leur constructeur identique à celui de leur nœud père dans l'empreinte. Mais, les transformations devant préserver le plus possible la structure du document, les relations de massif sont d'abord recherchées par l'algorithme avec l'empreinte générique. En cas d'échec, l'empreinte réduite est utilisée.

- Extensions pour la prise en compte des structures récursives : Les types d'éléments définis de manière récursive, ne sont pas pris en compte par l'algorithme tel qu'il est. Pour résoudre cette limitation, il est proposé de développer les types récursifs lors de la génération des empreintes. Le nombre de récurrences à développer est déterminé pour le

type source en se basant sur l'instanciation des types canoniques. Pour l'empreinte cible, dont la profondeur maximale des développements récursifs ne peut pas être décidée à priori, il est décidé de représenter une occurrence de type récursif comme un type de base associé au caractère @ dans les empreintes. Lorsque l'automate rencontre ce caractère dans l'empreinte cible, des transitions particulières lui sont ajoutées lui permettant ainsi de rechercher un massif dans le développement de la récurrence.

2) Génération des transformations : L'algorithme de génération de l'élément cible se base sur les résultats de l'étape précédente. S'il existe une relation de massif *M* entre l'arbre de type d'un élément source et celui d'un élément cible, il est possible de convertir l'élément dans le nouveau type. La relation de massif étant injective, tous les nœuds source ont une image dans l'arbre cible. De plus, l'ascendance des éléments et les constructeurs étant préservés par cette relation, si le type *t* est fils du type *t'*, alors *M (t)* est un descendant de *M(t')*. Par conséquent, l'algorithme effectue un parcours dans l'ordre préfixe de l'élément à transformer, et insère l'image de chaque élément rencontré comme un descendant de l'image du père de l'élément original. La méthode s'appuie aussi sur le fait que, par construction des arbres de type, il n'existe qu'une branche entre un type canonique et un de ses ancêtres à quelque niveau que ce soit.

3.2.6.3 Application du processus des transformations automatiques et ses limitations

Le processus présenté jusqu'ici, permet de trouver les correspondances entres les éléments d'un type source et ceux d'un type cible se basant sur les structures génériques et ne nécessitant ainsi aucune intervention de l'utilisateur. Son implémentation dans l'éditeur interactive de documents Thot a montré son efficacité.

Cependant, l'application de ce processus, à la conversion et la structuration des documents dans leur intégralité, est plus délicate. Elle présente certaines limitations :

1. Manque de pertinence : En effet, lorsqu'une relation de massif est trouvée, elle n'est pas forcément pertinente par rapport à la sémantique des structures mises en jeu, du fait que la transformation ne s'appuie pas sur des similarités structurales et ignore la sémantique des types d'éléments. De plus, pour les relations d'absorption l'élimination de nœuds conduit à de nombreuses combinaisons de relations de massif et en conséquence, la pertinence des relations trouvées souffre de l'aplatissement de la structure qui introduit de nombreuses combinaisons de couplage possibles.
2. Les attributs ne sont pas inclus dans le modèle de type alors qu'ils pourraient aider à établir des correspondances pertinentes entre type vue la sémantique additionnelle qu'il portent.

3. Manque de souplesse : Si la comparaison d'empreintes échoue, la transformation ne peut tout simplement pas être effectuée. Bien qu'une solution de repli soit proposée, basée sur la comparaison d'empreintes spécifiques, elle ne permet pas de résoudre tous les cas d'échec.
4. Complexité de l'algorithme lors de définition de structures récursives : les expériences ont montré que l'utilisation de la méthode pour la comparaison de structures récursives peut mener à des temps de comparaison élevés lorsqu'elle est appliquée à l'intégralité des documents.

C'est pour toutes ces raisons, et dont le but de pouvoir appliquer la méthode pour d'autres applications, comme la conversion de documents non structurés ou la conversion de documents complets entre structures génériques, qu'il est proposé de combiner cette approche automatique avec des spécifications explicites de transformation.

3.2.6.4 Combinaison de la transformation automatique et des spécifications explicites

Pour remédier aux manques de l'approche automatique, une méthode de spécification de transformation et intégrée au processus automatique de transformation. Les spécifications consistent en des expressions de transformation qui associent un ensemble d'éléments de la structure du document source avec leurs correspondants dans la structure du document cible. Ces associations sont appelées "*pré-couplages*", dont le but est de diriger l'algorithme de transformation automatique sur une relation de massif qui met en relation les nœuds pré-couplés. Les pré-couplages sont définis et mémorisés pour êtres réutilisables. Ils sont décrits par un langage basé sur la syntaxe d'XML proche de celui utilisés pour les spécifications de XSL. Les pré-couplages sont spécifiés de façon interactive si le résultat de la transformation automatique n'est pas satisfaisant.
L'algorithme automatique prend en compte les pré-couplages et cela en effectuant un traitement particulier lorsque l'état courant de son automate contient l'indice d'un nœud pré-couplé. Ce traitement diffère selon la position du nœud cible du pré-couplage dans l'empreinte cible par rapport à l'indice définissant l'état courant et aussi selon la relation entre le constructeur du type source et le constructeur du type cible du pré-couplage.
L'algorithme est aussi étendu pour prendre en compte le cas de la définition de pré-couplages de type récursifs.

3.2.6.5 Discussion

L'algorithme avec transformation explicite a été intégré à l'éditeur de documents Thot. Le processus de conversion de document XML au format de l'éditeur Thot est illustré dans la figure ci-dessous. D'abord, le document est analysé pour produire une empreinte source et une vue XML. L'empreinte source est ensuite comparée avec une empreinte cible d'une

DTD Thot en utilisant l'algorithme de transformation automatique pour produire ainsi une vue Thot. Si l'utilisateur n'est pas satisfait, il peut procéder à des spécifications explicites de transformation en définissant les pré-couplages d'une façon interactive simple avec la possibilité de les modifier ou de les enregistrer pour être utilisées par la transformation automatique lors des transformations futures.

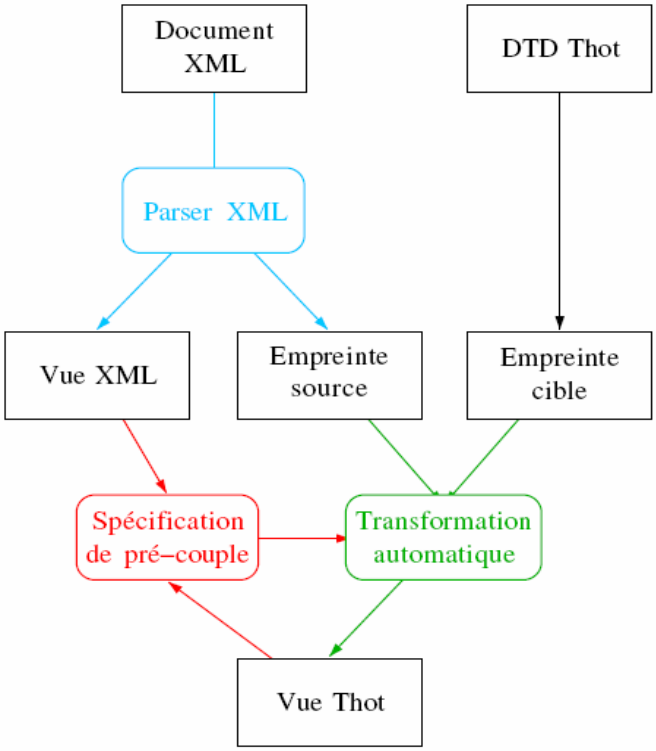

Figure -3.6 - Processus de conversion de documents XML en documents Thot.

L'originalité de la méthode est de donner une part prépondérante à la transformation automatique, les spécifications de relation de pré-couplage viennent compléter et corriger cette transformation. L'auteur estime qu'il est possible d'envisager cette méthode comme un mécanisme de transformation de structures s'appuyant sur une interface DOM lui permettant de s'intégrer dans d'autres applications XML notamment l'intégration de données hétérogènes en BD ou la RI.

3.2.7 Une architecture à base d'ontologie pour une intégration sémantique de documents XML

Pour permettre une intégration et une interrogation générique de documents XML sémantiquement équivalents mais de structures hétérogènes, les auteurs dans [Cruz et al, 04], proposent de générer une ontologie RDF pour chaque document XML préservant sa structure, puis une ontologie globale est construite selon une approche GAV par l'intégration des ontologies locales. L'ontologie globale unifiera l'interrogation en

établissant des connections sémantiques entre les différentes structures des sources des documents. Ainsi, les requêtes posées en termes de l'ontologie globale sont transformées en sous requêtes sur les sources. Et de plus, il est possible de transformer une requête XML en une requête équivalente en termes de l'ontologie globale sur toutes les autres sources.
Le processus de construction des ontologies et leur intégration se fait en deux étapes : construction des ontologies locales et construction de l'ontologie globale.

3.2.7.1 La construction des ontologies locales

Les schémas des documents XML sont transformés en des ontologies RDF locales en transformant chaque élément composé (qui possède d'autres éléments et/ou des attributs) en classes et chaque élément simple ou attribut est transformé en une propriété de la classe de l'élément auquel il appartient. De plus, les relations de structure sont transposées dans l'ontologie locale en définissant deux relations :
- La relation "class-to-literal" avec RDFS entre élément et attribut.
- La relation "element-subelement" entre deux classes, et cela en définissant un nouveau prédicat RDFS nommé "rdfs : contain".

3.2.7.2 La construction de l'ontologie globale

Parmi les différentes techniques de fusion d'ontologies qui existent, c'est l'approche de l'algorithme PROMPT [Noy et al, 00]**,** qui est choisie pour les différentes fonctionnalités qu'elle offre : la fusion de classes équivalentes, la fusion de propriétés équivalentes d'une même classe, la fusion de deux relations équivalentes entre deux classes en une seule relation, la copie d'une classe et/ou propriété si une classe et/ou propriété équivalente n'existe pas dans l'ontologie cible, et la généralisation de classes reliées en une superclasse peut se faire par l'utilisation de connaissances du domaine ou en raisonnant sur un thésaurus.
De part l'ontologie globale ainsi générée, une table de mapping est produite. Elle contient les informations de correspondances entre les ontologies locales et l'ontologie globale utilisée dans le processus de traitement des requêtes.
Dans le processus de traitement des requêtes, deux algorithmes de réécriture de requêtes sont proposés :
- Un algorithme qui transforme une requête RDF (posé sur l'ontologie globale) en des requêtes XML sur toutes les sources.
- Un autre algorithme qui permet de transformer une requête XML (posée en termes d'une source XML) en d'autre requêtes XML posées sur toutes les autres sources.

3.2.7.3 Discussion

Ce travail s'insère dans le cadre des travaux sur le Web sémantique. Les auteurs proposent une architecture de médiation basée sur une ontologie du domaine qui unifiera

des sources de données hétérogènes dans un niveau conceptuel et leur permettra de garder leurs structures propres. L'apport principal de l'approche est qu'elle préserve les relations de structure entre les éléments.
Cependant, la tâche de lier les schémas XML à l'ontologie globale est manuelle.

3.2.8 Vers l'automatisation de la construction d'une ontologie pour un système de médiation

Dans le cadre du Web sémantique et dans le cadre du projet PICSEL, et pour répondre au problème de l'hétérogénéité sémantique dû au fait que les ressources utilisent des vocabulaires propres et différents, Reynaud et al. dans [Reynaud et al, 04] proposent une méthode automatique pour la construction d'une ontologie du domaine au cœur de ce système de médiation. L'ontologie permettra l'expression des besoins des utilisateurs via une seule interface lui donnant ainsi l'impression d'interroger une seule source.

3.2.8.1 La construction de l'ontologie

Elle est basée sur l'utilisation des DTD décrivant les documents. L'ontologie est exprimée dans un langage (CARIN) combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (la logique de description). Elle se fait en deux étapes :
1) La construction d'une ontologie initiale simple : qui décrit la hiérarchie initiale des concepts du domaine. Elle est construite manuellement.

2) L'enrichissement de l'ontologie initiale : qui se fait en deux phases :
- D'abord, une phase d'extraction d'éléments utiles pour compléter l'ontologie initiale, se fait par extraction automatique d'un ensemble de termes-classes, de termes-propriétés et de relations par application d'heuristiques. L'idée de base est : qu'une classe est vue comme une représentation abstraite d'un ensemble d'objets complexes repérés dans les DTD comme étant des éléments décomposables. Les propriétés sont associées à des éléments qui ne sont jamais décomposés dans aucune des DTD. Ce processus est accompagné de traitements consistant à éliminer les doublants, remplacer les abréviations par leurs significations et éliminer les termes jugés non pertinents en exploitant d'une part des fichiers d'explication d'abréviations et de listes de termes jugés non pertinents, et d'autre part, un thésaurus (WordNet) pour l'identification des termes composés retenus en tant que classes du domaine. Ce processus est semi-automatique dans la mesure où une intervention du concepteur est nécessaire pour valider les termes retenus et juger de l'opportunité d'enrichir la liste des termes non pertinents ou des abréviations, et d'exécuter une autre fois ce processus d'extraction en prenant en compte les nouvelles versions de ces fichiers.

- Puis, une phase de structuration, qui consiste à raccrocher les termes extraits des concepts composant la hiérarchie initiale construite manuellement. Pour chacun des concepts, un réseau de relations le liant à d'autres classes du domaine est créé, chacune des classes étant caractérisée par des propriétés.

3.2.8.2 Discussion

Le processus de la construction de l'ontologie a été implémenté comme une partie du système de médiation qui est composé de plus d'un moteur de requêtes générique développé au sein du projet PICSEL, et qui se charge de calculer les réponses aux requêtes posées au médiateur. Les expériences effectuées sur deux séries de données, relatives à l'OTA et la DISA qui représentent des standards de communication basés sur XML pour faciliter l'emploi du e-commerce (l'une sous forme de 15 DTD, l'autre sous forme de 77 schémas XML traduits sous forme de DTD à l'aide du logiciel XML Spy), ont donné des résultats satisfaisants.

3.3 Interrogation et hétérogénéité en RIS

L'interrogation de corpus XML hétérogènes (c'est-à-dire composés de documents suivant des DTD différentes) est un problème ouvert important, les conditions de structure exprimées par les utilisateurs dans les requêtes, ne correspondent pas forcement exactement aux DTD des documents présents dans le corpus, mais ces derniers pourraient pourtant être intéressants pour l'utilisateur.

Les approches courantes de la recherche d'information structurées cherchent à vérifier des *correspondances syntaxiques* entre les arbres de la requête et des documents, les approches pour les corpus hétérogènes quant à elles, devraient vérifier de plus des *correspondances sémantiques*. En effet, les systèmes étant conçus initialement pour des corpus homogènes, il y a lieu de les adapter pour cette nouvelle problématique.

Ainsi, pour interroger un document dont le schéma est nouveau, un système doit pouvoir soit adapter la requête posée au document, soit pouvoir adapter le document pour lui appliquer la requête.

Depuis l'année 2004, une tâche visant à proposer des solutions pour l'interrogation de corpus hétérogènes a été introduite dans INEX, et permettra d'évaluer ces différentes approches. Comme la majorité des approches traitant l'hétérogénéité structurelle a été effectuée dans le cadre d'INEX, nous la décrivons brièvement dans ce qui suit.

3.3.1 La tâche hétérogène d'INEX

Dans cette tâche [Szalik et al, 04], de nouvelles collections ont été proposées aux participants. Ces collections sont décrites dans le tableau 3.4.

Collection	Taille (en Mo)	Nombre de noeuds
IEEE Computer Society	494	8 200 000
Berkeley	33.1	1 194 863
CompuScience	313	7 055 003
bibdb Duisburg	2.08	40 118
DBLP	207	5 114 033
hcibib	30.5	308 554
qmul-dcs-pubdb	1.05	23 436

Tableau -3.4- Collections de la tâche hétérogène.

- **Requêtes CO *(Content Only)***

Elles sont l'équivalent des requêtes CO de la tâche ad-hoc décrite dans le chapitre 2. Le but est de développer des méthodes indépendantes de toute DTD.

- **Requêtes BCAS *(Basic Content And Structure)***

Ces requêtes se focalisent sur la combinaison d'une seule condition de contenu associée à une seule condition de structure (par exemple : *sec[about(.,search engines)]* ou *section[about(.,search engines)]* . Le but est d'être capable de traiter les conditions de structure avec des noms de balises n'appartenant pas nécessairement à toutes les collections, mais pouvant avoir des synonymes dans certaines.

- **Requêtes CCAS *(Complex CAS)***

Elles sont l'équivalent des requêtes CAS définies en langage NEXI pour la tâche ad-hoc. Le but est de permettre des transformations et des correspondances partielles de chemins entre les différentes collections, sans perdre le composant RI de la requête.

- **Requêtes ECCAS *(Extended Complex CAS)***

Ces requêtes supposent que l'utilisateur est capable de donner la probabilité d'existence d'une contrainte structurelle donnée. Par exemple, la requête : *//author(0.8) [about(title(0.5), "Information retrieval")]* signifie que l'utilisateur recherche des auteurs de publications sur la recherche d'information, avec une probabilité de 70% que la balise concernée soit *author* (c'est à dire qu'il y a 30% de probabilité que l'information recherchée soit dans un élément portant un nom différent). Pour

déterminer que la publication parle de la recherche d'information, l'utilisateur pense que dans 50% des cas, le titre de la publication va contenir les termes "*Information retrieval*".

Dans la suite, seront présentées les différentes approches des systèmes de recherche d'information qui ont participé à la tâche hétérogène d'INEX pour résoudre cette problématique.

3.3.2 Les approches des systèmes de RIS pour la tâche hétérogène d'INEX

3.3.2.1 Une plateforme de test pour la tâche hétérogène d'INEX

Lors de leur participation à la tâche hétérogène d'INEX 2004 et dans le but d'apprivoiser la diversité des structures, les auteurs dans [Abiteboul et al, 04], proposent de créer une structure hétérogène unifiée sur les structures des données hétérogènes. L'approche était implémentée comme une application de la plate forme distribuée *KADOP P2P* (*peer-to-peer warehousing)* de documents XML. *KADOP* permet la construction d'entrepôt de ressources distribuées comme des documents XML et des informations sémantiques sur ces documents, des services Web et des collections de ces items. *KADOP P2P,* est construit sur une table de hachage distribuée comme une couche de communication, et utilise *Active XML* comme modèle pour la construction et l'interrogation des ressources dans le réseaux *P2P*. Ce système offre une interface uniforme pour l'interrogation des ressources diverses qu'il renferme. Il se charge de la traduction de la requête dans le format spécifique pour chaque source de données, de l'exécution des requêtes résultantes séparément et d'intégrer les résultats.
Pour cela, un modèle conceptuel des données est crée, il est basé sur la définition de concepts et de relations comme *"Is a"* et *"Part of"* entre eux. Ainsi, un modèle est créé pour chaque DTD, en prenant comme concept tout type de la DTD (incluant les attributs qui sont considérés comme des éléments). Chaque relation parent-fils est représentée par une relation *"Part of"* entre les deux concepts correspondants. Pour extraire le modèle conceptuel global, des groupes de concepts sont identifiés à partir des modèles construits pour les différentes sources, en identifiant les concepts similaires à l'aide d'un outil comme WordNet [Miller, 95] et un seuil de similarité. Puis pour chaque groupe de concepts ou cluster, un concept général est créé. Les relations *"Is a"* sont ajoutées pour relier chaque concept du modèle local d'une source au concept générique unifié du cluster auquel il appartient.

Pour traiter des balises non significatives, il était nécessaire de capturer les commentaires qui les précèdent et à partir de ces descriptions choisir le meilleur cluster ou concept où les mettre.

3.3.2.2 Cheshire II dans INEX 2004

Larson dans [Larson, 04] propose pour la recherche d'information structurée une technique de fusion de modèles probabilistes, en l'occurrence le modèle de régression logistique (LR) avec une version de l'algorithme Okapi BM-25 conjointement avec des opérations booléennes et des opérateurs de fusion .

Son approche pour l'hétérogénéité consistait à traiter les différentes collections comme des bases de données indexées séparément, chacune avec sa propre DTD. Bien que ces bases de données soient traitées comme une seule base virtuelle sur son système. Pour cela, il utilise les facilités offertes par le protocole standard de recherche d'information Z39.50, qui permettent d'effectuer des recherches distribuées parallèles ou séquentielles sur les bases de donnée, à savoir :

- Le service Z39.50 Explain pour l'identification des metadata des bases de données ;
- Le service Z39.50 SCAN pour l'extraction d'index distribués.

La technique de correspondance d'attributs du Z39.50 est utilisée pour relier les éléments d'indexation d'une requête aux éléments appropriés dans une collection de documents. Le système utilise un fichier de configuration qui spécifie les ensembles de labels qui peuvent être utilisés avec le même sens. Ainsi lorsqu'une requête est soumise à la base virtuelle, elle sera envoyée à chacune des bases physiques où elle sera traitée. Les résultats seront retournés par des wrappers XML standards.

3.3.2.3 Le système XFIRM à la tâche hétérogène d'INEX

Dans le modèle de recherche l'informations structurées XFIRM [Sauvagnat et al, 05], qui est par construction même un modèle flexible et générique, l'hétérogénéité est prise par nature lors de la représentation et de l'indexation des documents.
Le modèle de représentation des documents adopté peut être décrit comme sut :

1) Le modèle logique de représentation des données : Un document sd_i est un arbre composé de nœuds n_j, de nœuds feuilles nf_j et d'attributs a_j, ie : $sd_i = (Tree_i) = (n_{ij}, nf_{ij}, a_{ij})$. Pour permettre un parcours facile de l'arbre et une rapidité dans la recherche des relations ancêtres-descendants, le modèle XFIRM utilise une représentation des nœuds et des attributs basée sur l'approche de Xpath Accelerator [Grust, 02]. Ainsi, un nœud est identifié par ses valeurs de *pre-order* et *post-order* (*pre, post*), la valeur *pre-order* de son nœud parent (*parent*), et selon son type - nœud simple ou nœud feuille- par un champ indiquant la présence ou non d'attribut (*attribute*) ou par les termes qu'il contient

($\{t_1, t_2, \ldots, t_n\}$). Un nœud attribut est défini par la valeur pre-order (*pre*) du nœud auquel il est attaché et par sa valeur (*val*). Les valeurs *pre* et *post* sont calculées par un parcours "pre-fixed" et un autre "post-fixed" de l'arbre du document.

2) Le modèle physique de représentation des données : Toutes les données sont stockées dans une base de données relationnelle, avec comme index : - *The Path Index (PI)* permettant la construction de la structure du document (grâce au modèle Xpath Accelerator) ; pour chaque nœud, son type et ses valeurs *pre* et *post* sont stockés.
- *The Term Index (TI)* c'est un fichier inverse traditionnel, ie : pour chaque terme d'index garde les nœuds le contenant et ses positions dans ces derniers.
- *The Element index (EI)* décrit le contenu de chaque nœud feuille, ie : le nombre total de termes et aussi de termes différents qu'il contient.
- *The Attribute Index (AI)* qui donne les valeurs des attributs.

En effet, comme les structures d'index du modèle sont prévues pour traiter des collections de données hétérogènes, le processus d'indexation n'a pas posé de réels problèmes.
Pour les requêtes CO, un modèle identique à celui proposé pour la tâche ad-hoc a été utilisé. Pour les requêtes BCAS et CCAS, un nouvel index de dictionnaire (DICT) a été construit manuellement (en comparant les différentes DTD), c'était nécessaire pour établir des relations entres les balises des différentes DTD qui étaient sémantiquement proche.
Ainsi, une requête contenant des conditions de structure, sera d'abord réécrite dans les termes des balises équivalentes, puis subira les traitements comme dans le cas des requêtes de la tâche ad-hoc.
Les résultats obtenus étaient encourageants, cependant la construction de l'index dictionnaire étant faite manuellement, il fallait penser à proposer une approche plutôt automatique.

3.3.2.4 Un modèle universel pour la recherche d'information XML

Partant de l'observation théorique que les étiquettes des documents XML sont sémantiquement reliées au contenu qu'elles délimitent, l'approche proposée par les auteurs dans [Izabel et al, 04], n'utilise aucun aspect formel d'XML comme les DTD. L'association des étiquettes aux contenus est faite par des mesures statistiques comme celles reliant les fréquences des termes au contenu informationnel du document dans le modèle vectoriel standard.
Le modèle vectoriel était de ce fait, adapté à la recherche d'informations structurées dans des documents XML en établissant les mesures statistiques sur les éléments XML, la formule utilisée est la suivante :

$$P(Q,D) = \sum t_i \in Q \cap D \ \frac{w_q(t_i) * w_d(t_i) * fxml(t_i, e)}{\|Q\| * \|D\|}$$

Le facteur *fxml* a été introduit pour prendre en compte les spécificités des documents XML. Son expression est la suivante :

$$fxml\ (t_i, e) = fnh\ (t_i, e) * fstr\ (t_i, e) * focr\ (t_i, e)\ où:$$

- *fnh* (t_i, e) *:* (pour Nesting Factor), exprime l'importance des termes considérant leur position dans la structure de l'arbre du document, son expression est donnée par :

$$fnh(t_i, e) = \frac{1}{(1 + nl)}$$

Où nl est le nombre de niveaux entre l'élément *e et* son descendant contenant le terme t_i

- *fstr* (t_i, e) *:* (pour Structure Factor) décrit comment les conditions de structure d'une requête (CAS) sont satisfaites dans le contexte d'un élément, son expression est :

$$fstr(ti, e) = \frac{(commun\_markups + 1)}{(nr\_qmarkups + 1)} \qquad \text{où:}$$

- *commun_markups :* désigne le nombre de labels communs entre la requête et dans le cotexte de l'élément *e* contenant le terme t_i.
- *nr_qmarkups* : le nombre de labels dans les contraintes structurelles de la requête.

- *focr* (t_i, e) *:* (pour CO-occurence Factor) exprime le lien sémantique entre un label et son contenu. Pour décrire cette relation, ce facteur est donné par l'équation :

$$focr\ (t_i, e) = cf\ (t_i, e) * idf\ (t_i, e) * N * icf\ (e),\ \text{où:}$$

- *cf* (t_i, e) désigne le nombre de fois le label de l'élément *e* noté *m* délimite un contenu textuel contenant le terme t_i et le label *m* dans la collection.
- *idf* (t_i, e) représente l'inverse du nombre d'éléments *e* qui contient t_i.

Le produit *cf* (t_i, e) * *idf* (t_i, e) représente la raison entre le nombre de fois le terme t_i apparaît avec le label *m* au nombre d'éléments contenant t_i dans la collection.

- *N* représente le nombre total d'éléments dans la collection.
- *icf (e)* désigne l'inverse du nombre d'occurrences du label *m* dans la collection exprimant ainsi la popularité de *m*.

Le facteur *focr,* valorise ainsi la co-occurrence terme-label qui explore la caractéristique du langage XML issue de l'idée de conception de documents en utilisant des labels décrivant le contenu. Une meilleure efficacité de ce modèle est obtenue lorsque une relation sémantique existe entre les termes et les labels des documents.

3.3.2.5 Le système de recherche d'information SphereSearch

Ce système est décrit dans [Schenkel et al, 05]. Ses possibilités de recherche incluent les requêtes orientées contenu et celle orientées structure. Le calcul de la pertinence est basé

sur des modèles statistiques de la RI et sur des relations ontologiques quantifiées statistiquement.

En effet, ce système étant un descendant des travaux précédents sur le système XXL [Theobald et al, 02], il supporte l'opérateur de similarité "~", pour autoriser des requêtes sémantiques portant sur les noms des éléments XML (ou attributs) et aussi leur contenu. Il permet de les étendre avec des termes similaires donnés par une ontologie. Le système utilise un schéma de médiation pour interroger les collections hétérogènes, pour évaluer une requête contenant cet opérateur, le processus de traitement de la requête extrait les termes de l'ontologie qui sont sémantiquement reliés aux termes de la requête et dont la valeur de similarité est supérieur à un seuil prédéfini.
Le service d'ontologie implémenté dans ce système est responsable de fournir des informations quantifiées de similarité entre termes. Il se base sur l'utilisation de thesaurus comme WordNet [Miller, 95] et d'autres sources comme les gazettes géographiques, pour construire un graphe de relations sémantiques entre concepts. Ainsi, chaque terme de l'ontologie est une paire *(w,s)*, où : *w* désigne le mot sur un alphabet et *s* désigne son sens. Un synset *syn(s)* est l'ensemble de tous les mots du même sens. Des relations sémantiques sont définies entre concepts dérivés de sens commun, comme les synonymie, hyperonymie, hyponymie, holonymie et méronymie. Se basant sur ces relations, le graphe de l'ontologie $O = (V,E)$ est une structure de données où V représente les concepts sous forme de nœuds et E représente les relations sémantiques entre deux concepts par des arcs étiquetés par un poids et le type de la relation. Le poids exprime la similarité sémantique entre concepts connectés, pour le calculer, ils se basent sur des statistiques effectuées sur les corpus, en estimant la corrélation statistique, entre les mots caractéristiques des concepts reliés, avec le cœfficient de Dice [Manning et al, 99]:

$$Dice\ (c_1, c_2) = \frac{2|\{documents\ contenants\ c_1\} \cap \{documents\ contenants\ c_2\}|}{|\{documents\ contenants\ c_1\}| + |\{documents\ contenants\ c_2\}|}$$

Un indexe d'ontologie *OI* est construit qui implémente le graphe de l'ontologie. Il est responsable de la recherche des mots sémantiquement reliés à un mot donné de la requête. Il les délivre au processeur de la requête dans un ordre décroissant de similarité. La recherche peut être limitée à certains types d'arcs, ainsi pour trouver des noms similaires de tags, tous les types d'arcs peuvent être utilisés, par contre pour trouver des contenus similaires, la recherche est restreinte aux arcs d'hoponymie pour éviter la dérive de la requête. L'indexe *OI* offre de plus des méthodes de désambiguïsation, c'est-à-dire trouver le sens courant d'un mot entre tous les sens candidats selon le contexte du mot.

3.3.2.6 Restructuration automatique de documents structurés

Denoyer dans [Denoyer , 04] propose de convertir automatiquement les documents dans un schéma de médiation. Il propose pour cela un modèle stochastique de documents structurés qui permet de déduire les correspondances directement du contenu et de la structure du document. Deux hypothèses initiales sont posées par rapport au corpus de documents : d'abord, il suppose que tous les documents considérés appartiennent à des domaines similaires, de sorte qu'il est possible de supposer qu'il existera toujours un élément de la structure de médiation qui correspond à un élément de la structure d'origine. De plus, un ensemble de documents exprimés dans le schéma de médiation est supposé disponible pour effectuer l'apprentissage, ce qui nécessite soit de convertir une partie des documents à la main ou d'utiliser le schéma d'un des corpus que l'on cherche à intégrer comme schéma de médiation.

3.3.2.6.1 Modèle stochastique de documents semi-structurés

Les documents seront représentés sous forme d'arbre, chaque nœud du document sera composé d'une information de contenu et d'une étiquette. Ainsi, pour un document d composé de $|d|$ nœuds, chaque nœud n_i peut s'écrire $n_i = (s_i, t_i)$ où s_i représente l'étiquette du nœud et t_i représente le contenu de celui-ci. Il est alors donné :

$$\begin{aligned} P(d/\Theta) &= P(n_1, \ldots, n_{|d|}/\Theta) \text{ ; où } \Theta \text{ désigne l'ensemble des paramètres du modèle} \\ &= P((s_1, t_1), \ldots, (s_{|d|}, t_{|d|})/\Theta) \\ &= P(s_1, \ldots, s_{|d|}/\Theta).P(t_1, \ldots, t_{|d|} / s_1, \ldots, s_{|d|}, \Theta) \quad \textbf{(1)} \end{aligned}$$

$P(d/\Theta)$ peut donc se décomposé en le produit de deux termes :

- $\boldsymbol{P(s_1, \ldots, s_{|d|}/\Theta)}$: décrit la manière dont le modèle génère les différentes étiquettes du document d, elle est appelée : probabilité de structure. Pour simplifier son calcul, une hypothèse d'indépendance du premier ordre est posée : l'ensemble ordonné des étiquettes fils d'un nœud n_i (nommé $childrentag(n_i)$) vérifie la propriété d'indépendance conditionnelle de toute autre variable d'étiquette étant donné leur parent, on obtient ainsi :

$$P(s_1, \ldots, s_{|d|}/\Theta) = \prod_{i=1}^{|d|} P(childrentag(n_i) / s_i, \Theta)$$

Cette relation de dépendance entre les variables d'étiquettes peut être modélisée par un réseau bayésien comme le montre la figure suivante. Cette probabilité correspond à la probabilité de voir une séquence particulière d'étiquettes au-dessous d'un nœud d'étiquette t_i . Elle sera estimée dans la collection des documents d'apprentissage.

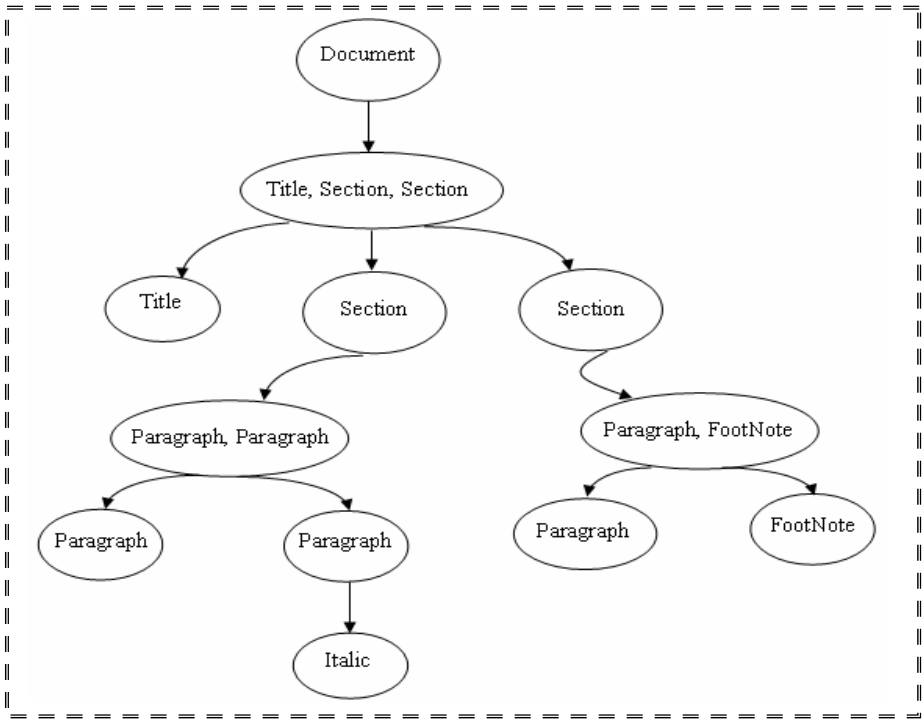

Figure -3.7- Le réseau bayésien correspondant au calcul de la probabilité de structure.

- $P(t_1, \ldots, t_{|d|} / s_1, \ldots, s_{|d|}, \Theta)$: décrit la manière dont le contenu de chaque nœud est généré, elle est dite : probabilité de contenu. L'hypothèse faite ici est que le contenu d'un nœud ne dépend que de l'étiquette de celui-ci :

$$P(t_1, \ldots, t_{|d|} / s_1, \ldots, s_{|d|}, \Theta) = \prod_{i=1}^{|d|} P(t_i / s_i, \Theta)$$

Soit : $ti = (w_i^1, \ldots, w_i^{Ki})$, où les (w_i^j) sont l'ensemble des mots apparaissant dans le i-ème nœud. Pour le calcul de cette probabilité un modèle Naive Bayes est utilisé, comme suit :

$$P(t_1, \ldots, t_{|d|} / s_1, \ldots, s_{|d|}, \theta) = \prod_{i=1}^{|d|} \prod_{j=1}^{Ki} P(w_j / s_i, \Theta)$$

L'équation (1) peut donc se réécrire :

$$P(d / \Theta) = \prod_{i=1}^{|d|} (P(childrentag(n_i) / s_i, \Theta) \prod_{j=1}^{Ki} P(w_j / s_i, \Theta))$$

3.3.2.6.2 Apprentissage des paramètres du modèle

L'apprentissage du modèle nécessite que soit calculé : $P(childrentag(i) / s_i)$ quelque soit $i \in [1, |d|]$ (c'est la probabilité de trouver un ensemble de nœuds ordonnés sous une balise donnée), et $P(w, s)$ quelque soit s et w (c'est la probabilité de trouver un mot sous une étiquette donnée). Pour cela, le logarithme de la vraisemblance du modèle est maximisé sur un corpus d'apprentissage D :

$$L_D = log(\prod_{d \in D} P(d/\Theta))$$

Ceci revient à résoudre l'équation : $\nabla_\Theta L_D = 0$ sous les contraintes assurant que les sommes des différentes probabilités estimées soient égales à 1. Le système proposé admet une solution analytique, les estimateurs robustes dérivés de cette solution, après lissage (pour éviter les valeurs nulles), sont :

$$P(s_1, \ldots, s_m/\Theta) = \frac{N^S_{s_1, \ldots, s_m}}{N^S} + \varepsilon \quad et \quad P(w/s, \Theta) = \frac{N^S_w + 1}{N^S + |V|}$$

Où : - N^S est le nombre de nœuds avec l'étiquette s ;
- $N^S_{s_1, \ldots, s_m}$ est le nombre de nœuds dont l'étiquette est *s* et les enfants sont $s_1, \ldots, s_m$.
- N^S_w est le nombre de fois où le mot *w* apparaît sous le nœud *s*.
- $|V|$ est la taille du vocabulaire.

3.3.2.6.3 Modèle de restructuration de documents

Il est proposé d'adopter une démarche similaire à celle des modèles de langages en RI, au problème de la restructuration. Ainsi, pour chaque document d^{orig} qu'on souhaite transformer, on cherche à évaluer à quel point un document *d* est une restructuration de ce document. Pour cela, il faudra déterminer $P(d/d^{orig}, \Theta)$, la probabilité que le document ait été généré par un modèle de langage du document. La nouvelle structure pourra alors être définie comme étant :

$$D_{optim} = \arg\max_d P(d/d^{orig}, \Theta),$$

Le calcul de cette probabilité est fondé sur le modèle de document structuré présenté précédemment. Résoudre cette équation revient à considérer tous les documents *d* possibles. Comme leur nombre peut devenir rapidement très grand, des hypothèses simplificatrices sont posées.
Le processus génératif considéré alors est le suivant : pour obtenir une restructuration d'un document d'origine, la structure de celui-ci sera tout d'abord transformer afin de l'exprimer dans le schéma de médiation, puis une fois cette structure définie, l'information de contenu sera « distribuée » dans les différents nœuds du document final. Ce processus est illustré dans le réseau bayésien de la figure suivante.

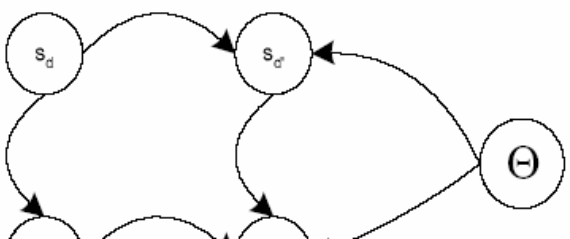

Figure -3.8- Le modèle statistique de restructuration de documents : le document exprimé dans le schéma de médiation dépend à la fois du document original et des paramètres appris sur le corpus d'apprentissage.

L'ensemble de ces hypothèses permet de simplifier l'expression de $P(d/d^{orig}, \Theta)$:

$$P(d/d^{orig}, \Theta) = P((s_1, t_1), \ldots, (s_{|d|}, t_{|d|}) / (s_1^{orig}, t_1^{orig}), \ldots, (s_{|d|}^{orig}, t_{|d|}^{orig}), \Theta)$$
$$= P(s_1, \ldots, s_{|d|} / s_1^{orig}, \ldots, s_{|d|}^{orig}, \Theta). P(t_1, \ldots, t_{|d|} / t_1^{orig}, \ldots, t_{|d|}^{orig}, s_1, \ldots, s_{|d|}, \Theta) \quad \textbf{(2)}$$

L'équation **(2)** correspond à la décomposition en probabilité de contenu/probabilité de structure du modèle de document semi-structuré présenté auparavant. Elles sont calculées comme suit :

- $\boldsymbol{P_{struct} = P(s_1, \ldots, s_{|d|} / s_1^{orig}, \ldots, s_{|d|}^{orig}, \Theta)}$: correspond à la probabilité de restructuration de la structure, avec l'hypothèse que les étiquettes des nœuds enfants d'un nœud n_i dépendent à la fois de l'étiquette de n_i et des étiquettes de ces nœuds dans le document original, la probabilité de restructuration structurelle s'écrit sous la forme :

$$P_{\text{struct}} = \prod_{i=1}^{|d|} (P(childrentag(n_i) / s_i, childrentag(n_i^{orig}), \Theta) ;$$

Comme le corpus d'apprentissage utilisé ne possède pas de données exprimées à la fois dans le schéma de médiation et dans le schéma original, il n'est pas possible d'estimer la probabilité : $P(childrentag(n_i) / s_i, childrentag(n_i^{orig}), \Theta)$, d'où vient la simplification suivante :

$$P(childrentag(n_i) / s_i, childrentag(n_i^{orig}), \Theta) = (P(childrentag(n_i) / s_i, \Theta).$$

- $\boldsymbol{P_{contenu} = P(t_1, \ldots, t_{|d|} / t_1^{orig}, \ldots, t_{|d|}^{orig}, s_1, \ldots, s_{|d|}, \Theta)}$: correspond à la probabilité de restructuration du contenu. Avec la supposition que le contenu d'un nœud ne dépend que de l'étiquette de celui-ci et avec l'hypothèse que le contenu des nœuds n'était pas modifié, il ressort : $P(t_i / t_i', s_i) = 0$ si $t_i \neq t_i'$. L'équation précédente devient :

$$P_{\text{contenu}} = \prod_{i=1}^{|d|} P(t_i / s_i, \Theta))$$

Au final, la structure optimale du document reconstruit s'obtient en résolvant :

$$d_{\text{optim}} = arg\ max\ P(d/d^{orig}, \Theta) = arg\ max(\prod_{i=1}^{|d|} (P(childrentag(n_i) / s_i, \Theta). \prod_{i=1}^{|d|} P(t_i / s_i, \Theta))$$

d d i=1 i = 1

ce qui permet d'utiliser les paramètres du modèle de document proposé auparavant. La manière optimale de résoudre cette équation s'inspire des méthodes de programmation dynamique utilisées lors de l'inférence des modèles de Markov cachés ou des grammaires stochastiques (algorithme de Viterbi [Viterbi, 67] et [Rabiner, 89]). La complexité de cet algorithme est linéaire par rapport au nombre d'étiquettes possibles, au nombre de nœuds dans le document et au nombre de règles de dérivation apprises.

Les résultats obtenus lors des expériences effectuées sur le corpus INEX, sont encourageants dans leur ensemble et ont permis de montrer que l'utilisation simultanée de l'information de structure et de contenu était indispensable pour obtenir de bonnes performances. Ce travail est un travail préliminaire, il a besoin d'enrichissement pour pouvoir surpasser les contraintes posées et traiter le problème général. Le modèle peut par exemple être enrichi par des informations additionnelles comme les noms d'étiquettes et autres.

3.4 Conclusion

L'émergence de XML comme langage de représentation a créé une grande quantité de documents qui bien que se rapportant au même domaine pourraient être structurés différemment et ce à cause de la liberté qu'offre XML aux concepteurs pour représenter leurs données.

Le problème de l'hétérogénéité nouvellement présenté dans le cadre de la tâche hétérogène d'INEX pour la recherche d'information dans des documents XML, est très largement étudié en BD dans le cadre de l'intégration et l'interrogation de sources de données hétérogènes. En effet, on est passé des approches structurelles se basant sur les mapping one-to-one entre les différents schémas des sources de données (dont l'automatisation de la tâche de "schema matching" et l'adaptation à l'échelle du Web sont actuellement l'objet de divers travaux), aux approches sémantiques (surtout dans le cadre du Web sémantique). Les approches sémantiques offrent un nouveau moyen pour l'interrogation générique de données de structures hétérogènes fondé sur le partage d'une ontologie commune aux différentes sources.

Certaines idées proposées dans ce contexte peuvent être utiles pour la recherche d'informations structurées. Cependant, le contexte des BD étant différent de celui de la recherche documentaire, des solutions spécifiques au domaine de la RI, doivent être développées.

Les solutions proposées jusqu'ici, souvent dans le cadre de la tâche hétérogène d'INEX, peuvent être classées selon deux points de vue **:** celles qui proposent d'utiliser un lexique,

un thésaurus ou une ontologie pour faire correspondre les conditions de structure exprimées dans la requête avec les types d'éléments effectivement présents dans la collection. D'autres approches 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.

Les travaux présentés dans le domaine de la RI restent préliminaires. La majorité des solutions proposées est liée à un système de RI particulier ou pose des restrictions sur les structures des documents XML. Il faut cependant développer des solutions qui soient indépendantes du système de RI et des structures des documents.

Dans le chapitre suivant, nous présentons nos propositions dans le cadre d'une gestion automatisée de corpus de documents de structures hétérogènes. Plusieurs pistes de recherches ont été investies. Une première approche simple serait d'établir des méthodes de traduction des requêtes dans chacune des DTD de la collection. Cette traduction se basera sur l'exploitation de la sémantique portée par les balises XML. Ainsi, quelle que soit la DTD utilisée par l'utilisateur au moment de sa requête, la recherche pourra être effectuée dans tout le corpus. Une seconde piste de recherche consiste à élaborer automatiquement une structure générique des documents, qui permettra à l'utilisateur de ne gérer qu'une seule DTD lorsqu'il interroge le corpus. Tous les documents (éventuellement les requêtes) devront cependant être transformés selon cette structure générique, au risque de perdre quelque peu de la sémantique portée par leur structure. Deux approches sont proposées dans se sens : une en utilisant une ontologie et l'autre en se basant sur les techniques d'apprentissage automatiques.

Chapitre 4 :

Contribution à l'interrogation de corpus hétérogène.

4.1 Introduction

Jusqu'à aujourd'hui le problème de l'hétérogénéité des structures a été principalement abordé par la communauté BD à travers la tâche du "Schema Matching" dans le cadre de l'intégration et l'interrogation de BD de schémas hétérogènes. Le problème commence à peine à être abordé dans la communauté de la RI grâce notamment à la tâche hétérogène proposée par la campagne d'évaluation INEX.

Dans le chapitre précédent, nous avons présenté un état de l'art des travaux sur cette problématique. Nous nous sommes particulièrement intéressés aux problèmes concernant l'interrogation de corpus de documents XML de structures hétérogènes.

Les travaux en BD sont intéressants, mais il faudra les adapter à la nature des documents XML qui présentent des schémas plus complexes et plus variés que ceux des BD. Les travaux de RIS restent préliminaires, la majorité des solutions proposées est liée à un système de RI particulier ou pose des restrictions sur les structures des documents XML. Il faut cependant développer des solutions qui soient indépendantes du système de RI et des structures des documents.

Nous nous sommes intéressés à cette problématique et nous avons proposé des solutions génériques et automatiques pour pouvoir traiter efficacement les volumes importants des documents XML. Dans la suite de ce chapitre, nous décrivons trois contributions traduisant différentes pistes de recherche possibles. Dans le cadre d'utilisation d'une ontologie, nous proposons deux approches : dans la première, nous proposons d'utiliser l'ontologie pour construire d'une façon automatique un dictionnaire des balises synonymes qui établit les correspondances entre les balises morphologiquement différentes mais qui désignent le même concept. Dans la seconde, nous utilisons l'ontologie pour construire une structure générique d'interrogation couvrant toutes les différences (de noms de balises ou de leur

hiérarchisation) entre les sources disparates de documents. Dans une troisième contribution, on se propose de convertir les documents dans un schéma désigné pour la médiation en utilisant des méthodes d'apprentissage automatique.

4.2 Utilisation d'une ontologie

Préambule

XML est uniquement un langage de représentation permettant la spécification de la structure du document et sa dimension syntaxique. Bien que la structure des documents présente une certaine sémantique il n'est pas clair comment il est possible de la déployer en dehors de certaines applications spécifiques. Pour permettre une réelle interrogation sémantique, XML doit être complété par un modèle conceptuel qui décrit adéquatement la sémantique des balises. De même, les langages actuels d'interrogation des documents XML, bien qu'ils soient très puissants pour la recherche du contenu des documents se basant sur la structure, ils ne reflètent que cette dernière et ne permettent pas ainsi des requêtes sémantiques.

L'introduction des ontologies dans le processus de RI a connu récemment une large expansion. Elles sont souvent utilisées en RI pour indexer les documents ou reformuler les requêtes [Gonzalo et al, 98], [Guarino et al, 99], [Khan, 00], [Mihalcea et al, 00], [Baziz, 05], etc.

Les résultats de telles pratiques nous laissent très enthousiastes de leur apport à la RI en général, et nous poussent à les considérer également dans le cadre de l'interrogation de corpus de documents XML de structures hétérogènes. L'idée est d'exploiter la sémantique portée par les balises XML et les relations pouvant exister entre ces balises pour faire correspondre les conditions de structure exprimées dans les requêtes avec les éléments présents dans la collection.

Dans la suite, nous présentons tout d'abord très brièvement les ontologies, puis nous décrivons deux approches que nous proposons pour l'utilisation d'une ontologie, en l'occurrence WordNet dans ce cadre.
La première s'intéresse à l'utilisation de l'ontologie pour permettre le matching de balises morphologiquement différentes mais ayant le même sens. La solution que nous proposons se base sur la construction et l'utilisation lors de l'interrogation d'un dictionnaire regroupant les balises sémantiquement équivalentes. Un utilisateur pourra poser sa requête dans les termes d'une quelconque DTD de la collection, le système se chargera de réécrire la requête dans les termes des autres DTD.

La seconde approche s'intéresse également à l'utilisation d'une ontologie cette fois-ci non seulement pour établir les correspondances entre des noms de balises synonymes. Mais, de plus on tente de traiter le problème de la diversité des structures (ie : agencement différents des balises). Nous présenterons l'approche proposée pour construire à partir des différentes DTD et grâce à l'utilisation de l'ontologie, une structure générique unifiant les différentes structures des documents et servant d'interface à l'interrogation de la collection.

4.2.1 Les ontologies

Depuis les années 90, les ontologies sont devenues un des champs de recherche les plus populaires en informatique, investi par différentes communautés dont celle de l'intelligence artificielle (IA) et de la RI. La raison pour laquelle les ontologies sont devenues si importantes est due actuellement au manque de standard pour l'ingénierie des connaissances dans la communication sémantique : on attend des ontologies qu'elles jouent ce rôle de standardisation.

4.2.1.1 Définition des ontologies

La littérature nous offre plusieurs définitions pour l'ontologie [Gruber, 93], [Guarino, 97], [Borst, 97]. Toutes ces définitions insistent sur le fait que les ontologies sont des descriptions formelles et génériques des entités du domaine nécessaires pour la conception d'applications basées sur le partage et l'interopérabilité de connaissances [Uschold, 98]. La définition la plus répondue, est celle de Gruber : "une ontologie est une spécification explicite et partielle rendant compte d'une conceptualisation".
Un large éventail de structures de connaissances est regroupé sous le nom "ontologie", qui en pratique peut référencer des hiérarchies de mots (thesaurus), des terminologies structurées, des bases de connaissances terminologiques, les ontologies formelles actuelles, etc. Ces structures différent selon : - leur échelle (spécifique à un domaine ou générique), - type de contenu (terminologique ou conceptuel), - le type des relations sémantiques représentées et leur degré de formalisation, - les langages de représentation et les méthodes de leur conception.
Généralement, une ontologie comprend les éléments ou composants suivant : - *les concepts* : souvent représentés par des termes, - *les relations* : qui relient ces concepts, telle la relation *partie-de*, - *les fonctions* : qui sont des cas particuliers de relations dans lesquelles le nième élément de la relation est défini de manière unique, à partir des n-1 premiers, - *les axiomes* : qui sont utilisés pour structurer des "phrases" qui sont toujours vraies, - *les instances* : elles sont utilisées pour représenter des éléments.
De façon globale, les concepteurs d'ontologies classent les ontologies existantes selon le degré d'implication de leurs composants. Si une ontologie contient seulement les concepts et les relations entre concepts, on parle d'ontologie moins formelle ou "light-weight".

Si l'ontologie contient en plus de cela les fonctions et les axiomes qui offrent une capacité de raisonnement sur les concepts, on parle alors d'ontologie formelle ou "heavy-weight" [Staab et al, 01], [Fensel et al, 01] et [Ding et al, 01].
En pratique, du moins dans le domaine de la recherche d'information, il existe très peu d'ontologies "heavy-weight" qui regroupent tous les composants listés plus haut, et quand c'est le cas, elles sont trop réduites pour être exploitables à grande échelle. En effet, si le raisonnement que les axiomes peuvent procurer est réalisable sur un ensemble de concepts réduit, il devient vite compliqué (voir impossible) dès que la taille de cet ensemble commence à augmenter sensiblement (grandeur nature). Les ontologies les plus vastes, qui sont utilisées actuellement à grande échelle, simplifient cette représentation. Elles se contentent souvent de la définition des concepts et des relations entre ces concepts [Ding et al, 01]. Parmi ces ontologies "Light weight" les plus répandues nous citons Gene Ontology (GO), UMLS (domaine médical), Mikrokosmos [Beale et al, 95], WordNet [Miller, 95], Sensus [Knight et al., 94], Yahoo Directory (hiérarchie de catégories de Yahoo!) et Cyc [Lenat, 95].
Parmi cette liste d'ontologies et de ressources lexico-sémantiques utilisées en RI, WordNet occupe sans doute avec MeSH et UMLS, les premiers rangs [Smeaton et al., 95], [Gonzalo et al., 98], [Guarino et al., 99], [Moldovan et al., 99], [Chua et al., 04], [Liu et al., 04], [Gao et al., 05], [Theobald et al, 05] et [Schenkel et al, 05].
Concernant WordNet, les raisons de cette large utilisation sont dues au fait que cette base de données lexicale couvre de façon quasi-totale la langue anglaise, ce qui la place souvent en adéquation avec les données traitées en recherche d'information qui sont dans le cas général de type presse (journaux et périodiques). Nous proposons, pour notre part, de l'utiliser dans notre approche. Nous la décrivons succinctement dans la suite.

4.2.1.2 WordNet

WordNet est une base de données lexicales construite par un groupe de psychologues et de linguistes du laboratoire de sciences cognitives de l'université de Princeton, dirigé par le professeur Georges A. Miller. Elle a été initialement conçue dans le cadre d'un projet lancé en 1985 et financé par l'agence de renseignements américaine (CIA), avec l'objectif de tester les déficits lexicaux dans des expériences de psychologie cognitive. A l'origine, ses concepteurs ne prétendaient construire ni une structure conceptuelle, ni une ontologie, mais bien une ressource lexicale rendant compte de l'usage des mots et de leur mise en relation dans la langue. Ce n'est qu'ensuite que le réseau lexical de WordNet a été perçu comme une représentation conceptuelle (Lexical Conceptual Graph ou LGC) [Guarino et al, 99] qui pourrait tenir lieu d'ontologie.
Comme impact, WordNet a inspiré d'autres projets tel EuroWordNet (lancé en 1996), qui construit une représentation conceptuelle indépendante des langues (inter langue), qui sert de noyau pour la majorité des langues européennes. Plusieurs autres propositions et initiatives ont aussi vu le jour ayant pour fin le "nettoyage" de parties de WordNet et leur

attachement à des ontologies de haut niveau pour en faire une ontologie. Parmi ces travaux, on peut citer ceux de Guarino dans le projet ONTOCLEAN [Guarino et al, 00] et [Guarino et al, 02] pour corriger les inadéquations dans les liens taxonomiques de WordNet, le projet ONION [Gangemi et al, 99] où une structure sophistiquée est proposée pour regrouper plusieurs ontologies (ontology merging) ou encore les travaux de Niles et Pease [Niles et al, 03] pour relier (manuellement) une ontologie formelle de haut niveau appelée SUMO (Suggested Upper Merged Ontology) à WordNet.

Concernant le contenu de WordNet, il couvre la majorité des noms, verbes, adjectifs et adverbes de la langue Anglaise. Sa dimension ainsi que le domaine de la langue générale qu'il traite lui permettent souvent de couvrir les sujets traités dans les collections de test conventionnelles de la RI (TREC, CLEF). Ces dernières sont le plus souvent de type presse. WordNet a un réseau de 144 684 termes organisés en 109 377 noeuds (concepts) appelés Synsets. Le Tableau 4.1, donne des statistiques sur le nombre de mots et de concepts dans WordNet.

Catégorie	**Mots**	**Concepts**	***Total Paires Mot-Sens***
Nom	107 930	74 488	132407
Verbe	10 806	12 754	23255
Adjectif	21 365	18 523	31077
Adverbe	4 583	3 612	5721
TOTAL	144 684	109 377	192460

Tableau -4.1- Nombre de mots et de concepts dans WordNet.

Dans WordNet, une entrée est donc un concept qui est représenté par un Synset, c'est-à-dire l'ensemble des termes (mots ou groupes de mots) synonymes qui peuvent désigner ce concept. Les concepts reliés sémantiquement par une relation donnée à un Synset, sont représentés par une classe qui porte le nom de la relation. La relation de base entre les termes dans WordNet est la Synonymie. Les Synsets sont liés par des relations telles que spécifique-générique :hyponyme-hyperonyme (is-a) et la relation de composition meronymie-holonymie (part-of) comme représentées dans le schéma de la Figure 4.1. Elles peuvent être définies comme suit :

- *Synonymie* : les synonymes étant associés à la classe concept**; - *Hyperonymie*** : C'est le terme générique utilisé pour désigner une classe englobant des instances de classes plus spécifiques : *y* est un *hyperonyme* de *x* si *x* est un type de (*kind of*) *y*. **- *Hyponymie*** : C'est le terme spécifique utilisé pour désigner un membre d'une classe (relation inverse de *Hyperonymie*). *x* est un *hyponyme* de *y* si *x* est un type de (*kind of*) *y*. **- *Holonymie*** : Le nom de la classe globale dont les noms *meronymes* font partie. *y* est un *holonyme* de *x* si *x* est une partie de (*is part of*) *y*. **- *Méronymie*** : Le nom d'une partie constituante (*part of*), substance de (*substance of*) ou membre (*member of*) d'une autre classe (relation inverse de *l'holonymie*). *x* est un m*éronyme* de *y* si *x* est une partie de *y*, ex :{*voiture*}a pour *méronymes* {*porte*, *moteur*}.

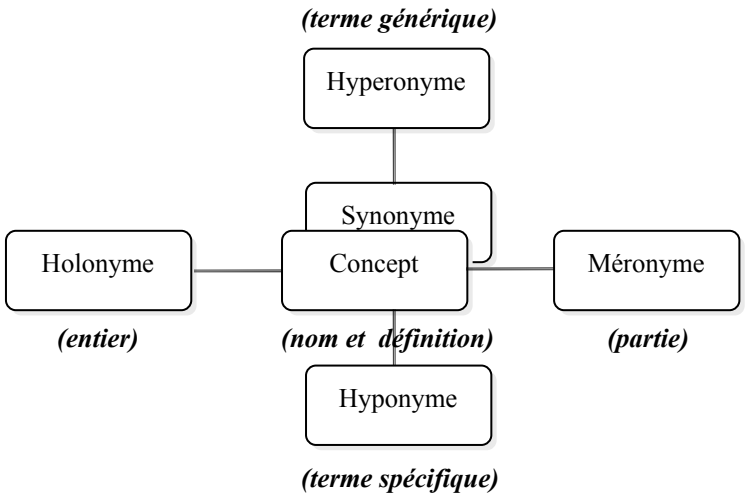

Figure -4.1- Principales relations sémantiques dans WordNet.

En plus de ces relations, on peut citer quelques autres relations qui sont cependant moins utilisées en pratique. C'est le cas notamment de la relation *domain* (rajoutée récemment pour la version WordNet 2.0), de la relation *antonymy* pour exprimer les sens opposés pour les synsets, de la relation *entailment* pour les verbes : Un verbe *x entails* (nécessite) *y* si *x* ne peut être fait à moins que *y* ne le soit ou n'ait été déjà fait. Ou encore de la relation *troponymy* pour les verbes: *x* est un *troponyme* de *y* si to *x* est pareil que to *y* dans une certaine façon. La Figure 4.2, donne un exemple d'une sous hiérarchie extraite de WordNet correspondant au concept "Goal".

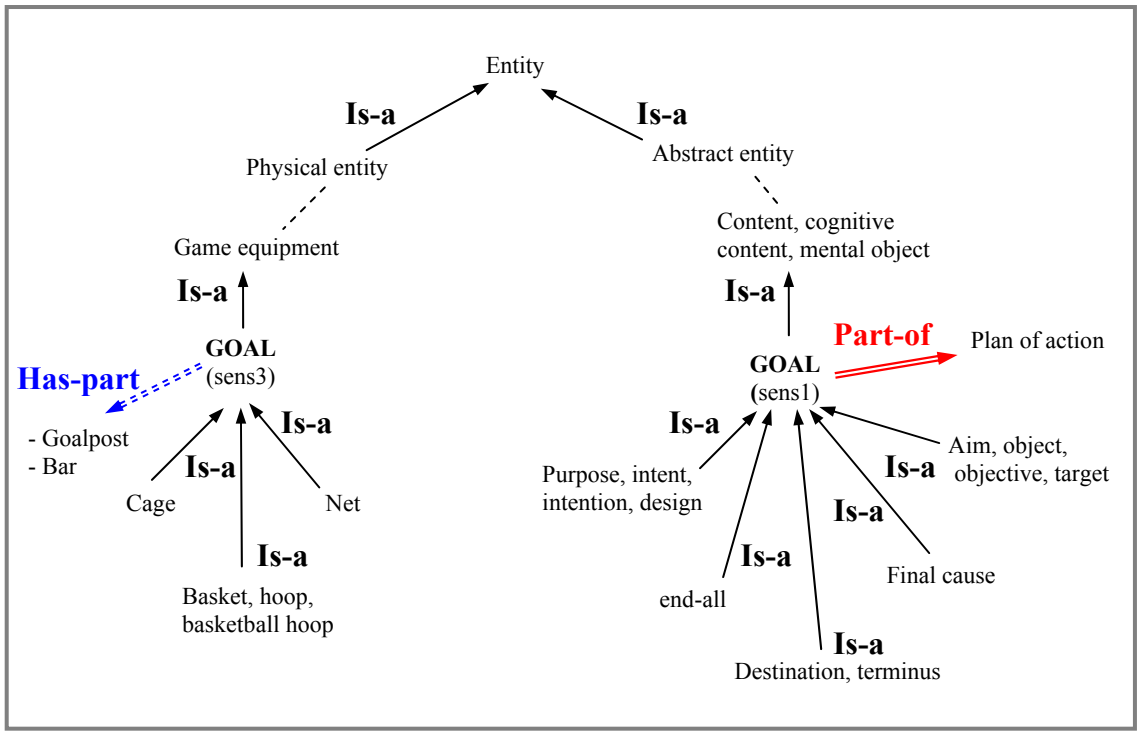

Figure -4.2- Exemple de sous hiérarchie dans WordNet correspondent au concept "Goal".

Notre but n'était pas de détailler ici les ontologies et la manière de les construire à partir de textes, domaine qui est beaucoup plus du ressort de la communauté de l'ingénierie des connaissances [Aussenac-Gilles et al, 00], mais plutôt de donner une petite idée dessus. En effet, l'objectif pour notre part, n'est pas tant de créer des ontologies à partir de textes, mais de proposer un moyen pour leur intégration dans un processus de RI, pour répondre au problème de l'interrogation de corpus hétérogène. C'est l'objet des sections ci-dessous.

4.2.2 Première approche : construction d'un dictionnaire de balises synonymes

4.2.2.1 Vue globale de l'approche

Ici nous cherchons à exploiter la sémantique portée par les noms des balises XML pour remédier au problème de la variation linguistique lors de l'interrogation des documents XML.

A la lumière des travaux de M. Baziz [Baziz, 05] effectués dans le cadre de l'indexation conceptuelle guidée par des ressources sémantiques, nous proposons d'utiliser une ontologie pour établir des relations de synonymie entre les balises XML partageant un même sens.

Pour ce faire, on se basera sur un modèle de représentation de la structure arborescente des documents permettant de déterminer rapidement les relations ancêtres-descendants. Pour chaque balise, nous interrogeons l'ontologie pour avoir son sens. Comme une balise peut avoir plusieurs sens, il est nécessaire de désambiguïser pour avoir le concept adéquat avec son contexte. Cela est réalisé en calculant la similarité entre les différents concepts déterminant le contexte de chaque balise. Pour chaque concept retenu, on lui construit une entrée dans le dictionnaire des balises synonymes tout en gardant la liste des balises qu'il représente dans la collection.

De façon générale, comme l'illustre la figure 4.3, l'approche comprend trois étapes principales, qui sont comme suit :

(1) : La première étape correspond à l'extraction des concepts candidats pour chaque balise d'une DTD par une projection sur l'ontologie.

(2) : Dans cette étape, il s'agit d'un traitement de désambiguïsation qui permettra, sur la base d'un calcul de la proximité sémantique entre concepts, de choisir les concepts adéquats aux sens des balises telles qu'elles sont utilisées dans les documents de la collection.

(3) : La dernière étape, consiste en la construction d'un dictionnaire des synonymes. Cela se fait en définissant une entrée pour chaque concept retenu dans l'étape précédente et où sera sauvegardée de plus, la liste des balises qui lui correspond dans la collection.

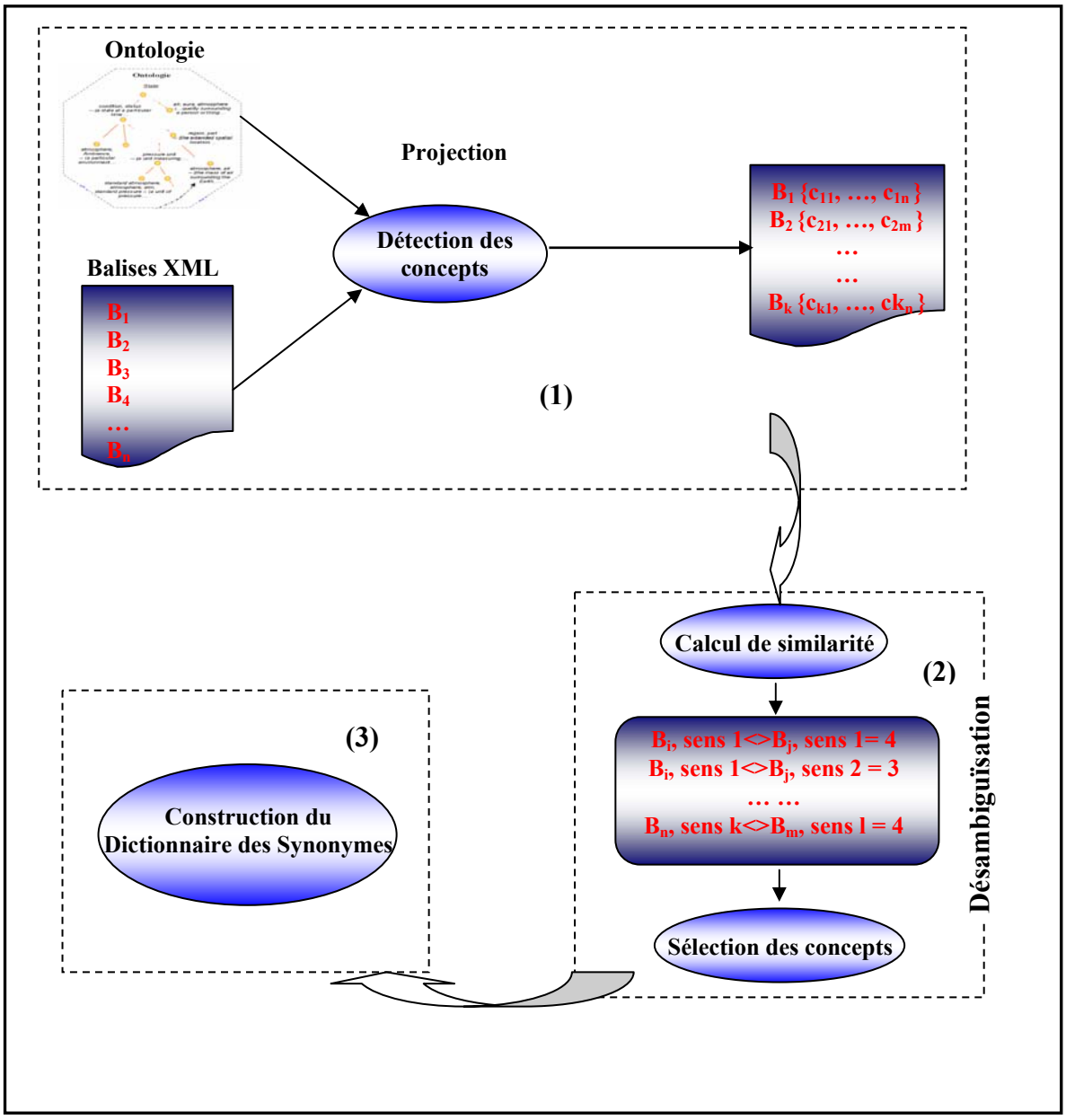

Figure -4.3- Schéma général de la première approche.

Avant de décrire en détail chacun de ces traitements, nous présentons d'abord le modèle de représentation des documents sur lequel se base notre approche.

4.2.2.2 Modèle de représentation de la structure des documents

Les documents XML possèdent des structures arborescentes décrites par des DTD. On exploitera cette structure pour représenter chaque DTD de la collection sous forme d'arbre. Une DTD sera donc représentée par un *arbre (ds)*, défini par les ensembles *N, A* et *L* : *ds* = (*N, A, L*). Avec $N = \{n_1, n_2, ...\}$ l'ensemble des nœuds éléments, $A = \{a_1, a_2, ...\}$ l'ensemble des attributs et *L* est un ensemble d'*arcs orientés*. Un arc orienté est une paire (*u, v*) formée de deux éléments des ensembles *N* ou *A* tels que :

– *u* est parent de *v*

– chaque $n_i \in N$ appartient au moins une fois à *L* en tant que premier composant d'une paire formant un arc

– chaque $n_i \in N$, $a_i \in A$ excepté le noeud racine appartient une et une seule fois à *L* en tant que second composant d'une paire formant un arc.

Les noeuds sont ainsi reliés entre eux par des arcs qui forment les relations parent/enfant. Tous les noeuds excepté le noeud racine ont exactement un nœud parent. De cette façon, il sera facile de déterminer rapidement les relations de hiérarchie entre les balises de la même DTD.

Dans la suite, on désignera par une balise un nom d'un élément (qu'il soit composé ou simple) ou d'un attribut XML. Pour chaque balise on disposera des informations sur sa balise mère et éventuellement ses balises filles et/ou attributs.

4.2.2.3 Projection sur l'ontologie

Toutes les balises XML identifiées dans la collection sont projetées sur l'ontologie pour obtenir les concepts auxquels elles sont associées. Les nominations des balises sont généralement sous forme de noms ou d'abréviation de noms. Pour les balises abrégées, il faudra d'abord interroger un dictionnaire des abréviations pour avoir les noms appropriés.

Cependant, comme chaque nom de balise peut avoir plusieurs sens, et ainsi correspondre à plusieurs synsets (ou concepts) de l'ontologie, des mesures de similarité entre les différents sens des noms, sont calculées en vue de sélectionner, pour chaque balise, le meilleur sens correspondant dans l'ontologie.

La mesure de similarité entre deux noeuds représente une valeur condensée résultant de la comparaison de deux sens possibles pour deux termes (donc deux concepts candidats) en utilisant la distance entre les positions des deux concepts candidats dans l'ontologie ou encore les relations sémantiques de l'ontologie. Cette valeur n'a pas de sens précis mais exprime le degré du lien entre les deux concepts candidats. Nous l'explicitons dans la section suivante.

4.2.2.4 Le traitement de désambiguïsation

Se déroule en deux phases :

1) Calcul de la similarité entre concepts

L'ontologie n'offre pas une quantification des liens sémantiques entre les différents concepts qu'elle définit. Pour cela, diverses mesures permettant de calculer la valeur de la proximité sémantique entre concepts sont proposées dans la littérature. On peut distinguer :
1. Les mesures se basant sur le chemin (Path based measures) entre deux concepts à comparer, telles que définies par exemple dans [Rada et al. ,89] [Leacock et al, 98] [Jiang et al, 97].
2. Les mesures se basant sur la notion de contenu d'information (Information Content IC), telles que définies dans [Resnik, 99].
3. Les mesures se basant sur une combinaison du chemin et du contenu d'information [Lin, 98] ou sur l'algorithme de Lesk adapté à WordNet dans [Patwardhan et al, 03].

Il serait intéressant de combiner plusieurs de ces mesures de sorte à couvrir tous les types cités ci-dessus. De plus, il faudra identifier le contexte de chaque balise qui servira pour le choix du sens qui lui soit le plus approprié. Une première idée, serait de considérer le cotexte formé de toutes les balises de la DTD à laquelle elle appartient, donc on pourra penser à évaluer la proximité sémantique avec chacun des concepts relatifs à ces balises.
Une autre façon est de considérer le contexte local d'une balise en se restreignant à l'ensemble formé de sa balise mère et éventuellement la liste de ses balises filles et attributs. Dans ce cas, il suffira de calculer la proximité sémantique avec les concepts relatifs à ce seul ensemble.

Nous décrirons la mesure de Lesk adaptée à WordNet dans [Patwardhan et al, 03]. Elle représente le nombre de mots communs entre deux concepts. Formellement elle est décrite comme suit : étant donné un ensemble de relations $R = \{R_1, R_2, ..., R_n\}$ et deux mots b_i et b_j auxquels sont affectés deux sens S_α^i et S_β^j. La similarité sémantique entre S_α^i et S_β^j, notée : $Sim\ (S_\alpha^i , S_\beta^j)$ est définit comme suit :

$$Sim\ (S_\alpha^i , S_\beta^j) = \sum_{l,\ k \in \{1,...,n\}} ||R_k\ (S_\alpha^i) \cap R_l\ (S_\beta^j)||$$

Les relations dépendent de l'ontologie utilisée. Dans le cas de WordNet, les relations disponibles sont : les relations de *synonymie,* d*'hypéronymie,* d*'hyponymie,* de *méronymie* et d*'holonymie* qui sont déjà définies dans (la section 4.2.1.2), plus les relations de glossaire " *gloss* " et les relations de domaines et leurs inverses membre de domaine suivantes :

- Gloss : elle n'est pas une relation en elle-même [Patwardhan et al, 03] mais représente la définition d'un concept avec éventuellement des exemples spécifiques du monde réel.

- Les relations de domaine et leurs relations inverses, membre de domaine: elles sont aussi utilisées pour le calcul de la similarité entre concepts. Elles peuvent dénoter la catégorie, l'usage ou la région.

L'utilisation de ce nombre relativement élevé de relations a pour but de couvrir au maximum les différents types de liens que deux concepts peuvent partager.

2) Sélection des concepts

A cette étape, nous connaissons pour chaque balise son sens représenté par l'ensemble des concepts associés (les synsets de WordNet), noté $S_i = \{S_1^i, S_2^i, ..., S_n^i\}$, ainsi que les valeurs de sa proximité sémantique calculées avec les balises du même contexte. Il reste uniquement à choisir pour chaque balise le meilleur concept parmi tous les sens candidats extraits de l'ontologie. Le principe de la désambiguïsation consiste à supposer que, parmi les différents concepts candidats (sens) pour une balise donnée, le plus adéquat (vraisemblable) est celui qui a le plus de liens avec les autres concepts du même contexte qu'elle. En généralisant cette règle à toutes les balises, on se retrouve avec des balises qui se désambiguïsent mutuellement et de manière globale par rapport au contexte de chaque DTD.

Pour formaliser cette idée, on affecte à chaque concept candidat (ou sens d'une balise) un score (*C_score*). Le score d'un concept candidat est égal à la somme des valeurs de similarité qu'il a obtenu avec les autres concepts candidats des balises de son contexte sauf ceux qui sont dans le même ensemble de sens que le sien : pour une balise b_i, le score de son $k^{ième}$ sens est alors calculé comme suit :

$$C\_score(S_k^i) = \sum_{\substack{j\in[1..m], j\neq i \\ l\in[1..n]}} Sim(S_k^i, S_l^j) \ (*)$$

Où m représente le nombre des balises formant le contexte d'une balise et n le nombre de sens qui est propre à chaque balise b_i. Le meilleur concept (synset) S_i retenu est celui qui représente au mieux le sens de la balise b_i. C'est celui qui maximise *C_score* :

$$S_i = Best\_score(b_i) = \underset{k=1..n}{ArgMax} \ \left\| C\_score(S_k^i) \right\|$$

Le concept ainsi sélectionné, représentera une entrée dans le dictionnaire des synonymes qui sera construit dans l'étape suivante.

4.2.2.5 Construction du dictionnaire des concepts

La dernière étape de l'approche concerne la construction du dictionnaire des concepts. Pour chaque concept sélectionné à l'issue de la phase de désambiguïsation, on lui crée une entrée dans le dictionnaire des balises synonymes. On lui associe de plus, la liste des balises le référençant dans la collection. Pour chaque balise, on gardera une référence vers son concept.

Ainsi, pour chaque requête d'un utilisateur contenant des conditions de structure formulées dans les termes d'une quelconque DTD de la collection, il sera possible de chercher pour chaque balise qui y figure, le concept correspondant dans le dictionnaire et d'identifier la liste des balises synonymes pour étendre la requête aux autres documents de la collection qui suivent d'autres DTD et les inclure dans la recherche.

4.2.2.6 Un exemple

Nous illustrons les étapes précédentes en les appliquant sur deux simples DTD (représentées dans la figure ci-dessous sous forme d'arbres) :

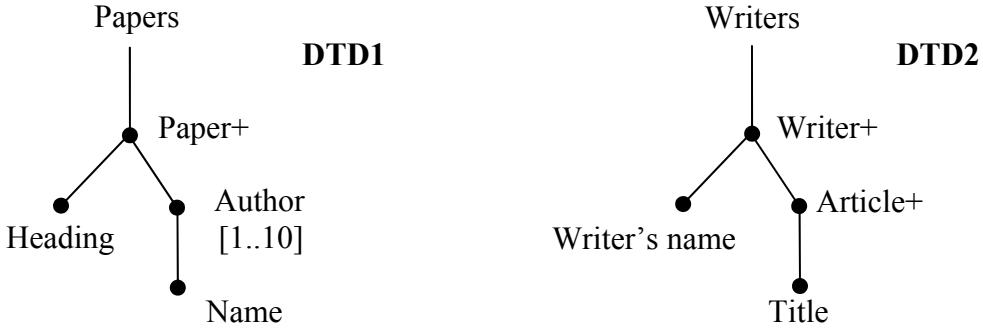

Figure -4.4- Deux DTD différentes décrivant le même domaine.

En premier lieu, toutes les balises contenues dans les documents (représentés ici par les DTD), sont extraites et projetées sur l'ontologie WordNet pour avoir leurs sens. Vu la polysémie des sens, les noms de balises se voient attribuer plusieurs synsets, par exemple la balise "name" possède 6 sens, la balise "paper" 7 sens, etc. (comme le montre la figure 4.5).

The noun name has 6 senses (first 6 from tagged texts)
1. (698) **name** -- (a language unit by which a person or thing is known; "his name really is George Washington"; "those are two names for the same thing")
2. (44) **name** -- (by the sanction or authority of; "halt in the name of the law")
3. (26) **name** -- (a person's reputation; "he wanted to protect his good name")
4. (15) **name**, figure, public figure -- (a well-known or notable person; "they studied all the great names in the history of France"; "she is an important figure in modern music")
5. (6) **name**, gens -- (family based on male descent; "he had no sons and there was no one to carry on his name")
6. (2) **name**, epithet -- (a defamatory or abusive word or phrase)

The noun paper has 7 senses (first 6 from tagged texts)
1. (31) **paper** -- (a material made of cellulose pulp derived mainly from wood or rags or certain grasses)
2. (21) composition, **paper**, report, theme -- (an essay (especially one written as an assignment); "he got an A on his composition")
3. (12) newspaper, **paper** -- (a daily or weekly publication on folded sheets; contains news and articles and advertisements; "he read his newspaper at breakfast")
4. (5) **paper** -- (a scholarly article describing the results of observations or stating hypotheses; "he has written many scientific papers")
5. (4) **paper** -- (medium for written communication; "the notion of an office running without paper is absurd")
6. (2) newspaper, **paper**, newspaper publisher -- (a business firm that publishes newspapers; "Murdoch owns many newspapers")
7. newspaper, **paper** -- (the physical object that is the product of a newspaper publisher; "when it began to rain he covered his head with a newspaper")

Figure-4.5- Les différents sens que peut avoir un mot comme "name" ou "paper".

En second lieu, les similarités sémantiques sont calculées entre tous les concepts candidats, c'est-à-dire : les sens possibles des balises identifiées précédemment, en utilisant les différentes mesures de similarité sémantique. Ici, et en guise d'illustration, nous avons utilisé la mesure de Lesk décrite dans (la section 4.2.2.4) avec les relations de synonymie, et de glossaire.

Par exemple, (comme on le voit dans la figure 4.6), la première ligne de la deuxième colonne veut dire que la similarité entre le sens1 du nom "heading" et le sens4 du nom "paper" est égale à 4.

author#n#1 <> name#n#1=3	paper#n#4<>heading#n#1=4
author#n#1 <> name#n#3=1	paper#n#4<>heading#n#3=0
author#n#2 <> name#n#1=1	paper#n#5<>heading#n#1=1
author#n#2 <> name#n#3=1	paper#n#5<>heading#n#3=0
… …	… …
title#n#1 <> article#n#1=3	writer #n#1<> article #n#1=4
title#n#1<> article#n#4=0	writer #n#1<> article #n#4=0
title#n#6 <> article#n#1=0	writer #n#2<> article #n#1=1
title#n#6 <> article#n#4=0	writer #n#2<> article #n#4=0
… …	… …

Figure-4.6- la similarité calculée entre les concepts.

Puis, vient l'étape de sélection du concept qui représente au mieux le sens d'une balise. Pour chaque concept candidat, son sens ayant le plus grand score cumulé calculé avec la formule (*), est retenu comme le concept approprié. Les résultats pour notre exemple sont illustrés ci-dessous :

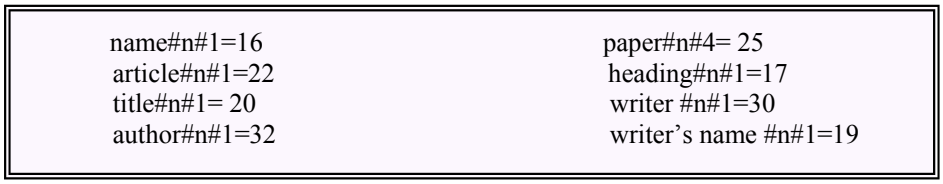

Figure -4.7- Le meilleur score cumulé des concepts retenus.

Par exemple pour la balise "article" possédant 4 sens, le sens1 qui est sélectionné, correspond effectivement au sens approprié dans le contexte de la DTD à laquelle elle appartient. De même pour toutes les autres balises.

Enfin, on crée pour chaque concept retenu une entrée dans le dictionnaire des balises où on gardera une liste de ses balises synonymes. Par exemple la balise "author" est identifiée comme synonyme de la balise "writer", elles correspondent au même concept de l'ontologie, à savoir "writer#n#1", elles seront insérée dans la liste de ses synonymes. On fera de même pour toutes les autres balises, on obtiendra à la fin de cette opération l'ensemble des classes suivantes :

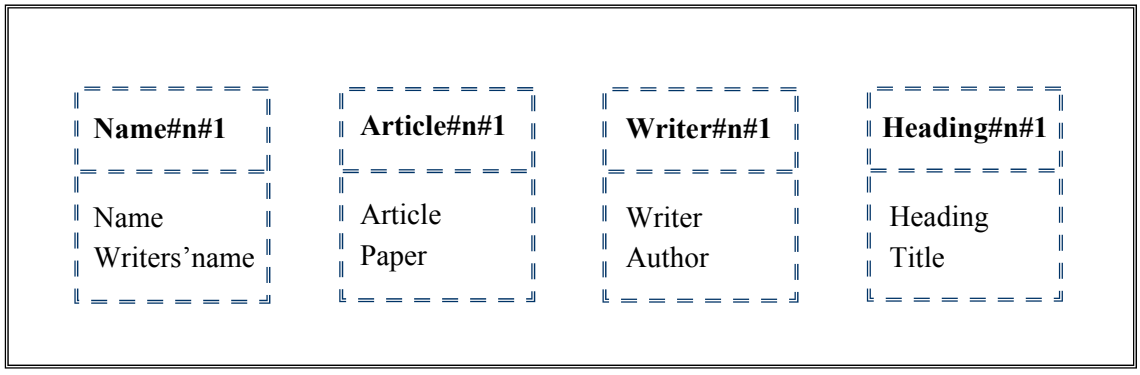

Figure -4.8- Ensemble des concepts insérés dans le dictionnaire des synonymes.

Ainsi, un utilisateur pourra utiliser l'un ou l'autre vocabulaire des DTD pour formuler ses requêtes, il suffira de garder un pointeur pour chaque balise vers son concept dans le dictionnaire, et le système se chargera d'aller chercher ses éventuels synonymes dans le dictionnaire, pour lancer une recherche dans les termes des autres DTD.

4.2.2.7 Conclusion

Nous avons présenté notre première approche pour résoudre le problème de l'interrogation de corpus hétérogène. Dans cette approche, l'utilisateur pourra poser sa requête dans les termes d'une quelconque DTD de la collection, le système se chargera de réécrire la requête dans les termes des autres DTD. La clé de la solution est l'utilisation d'un dictionnaire de balises synonymes construit au préalable à partir de WordNet.

Cependant, cette approche ne s'intéresse qu'à la résolution du problème lié à la variation linguistique, c'est-à-dire établir les correspondances entre les différents noms de balises désignant le même concept, négligeant l'autre problème concernant la structuration de ces balises qui peut être elle aussi différente dans les documents.

Dans la section suivante, nous présentons une seconde approche d'utilisation de l'ontologie. Cette approche couvre ces deux problèmes et tente d'unifier les différentes structures en une seule qui servira de structure générique pour l'interrogation.

4.2.3 Seconde approche : Interrogation par une structure générique

L'idée de construire automatiquement une structure générique (ou schéma de médiation) pour permettre à un utilisateur de ne gérer qu'une seule DTD au moment de l'interrogation d'une collection de documents XML de structures hétérogènes est très répandue ([Termier, 04], [He et al, 03], etc.)

Cette idée, bien qu'exploitée différemment, se base sur deux caractéristiques principales relatives aux documents XML. La première caractéristique est l'existence de plusieurs structures décrivant le même domaine. La seconde, est que parallèlement à la prolifération des sources des documents, les vocabulaires des schémas correspondants tendent à converger.

Dans le même ordre d'idées, on se propose d'utiliser une ontologie pour construire cette structure générique qui unifiera les structures propres aux documents décrivant un même domaine, et fournira aux utilisateurs une interface unique pour l'interrogation.

4.2.3.1 Vue globale de l'approche

Cette approche est plus générale que la précédente. En effet, nous traitons à la fois le problème de la variation linguistique (utilisation de plusieurs noms de balises désignant un même concept) et celui de la différence entre les structurations des balises.

Nous partons de l'observation que les différentes structures des documents XML renferment des relations sémantiques entre les éléments qu'elles comportent. Ces relations peuvent se résumer en les relations de spécification/généralisation (est-un ou is-a, par exemple : "book" is-a "publication"), et de composition (partie-de ou part-of, par exemple : "title" part-of "book"). Les structures des documents XML pourront ainsi être représentées par des hiérarchies formées par les concepts qu'elles contiennent en les projetant sur le sous réseau conceptuel de l'ontologie formé des relations de spécification/généralisation (représentées par les liens d'hypéronymie) et celles de composition (représentées par les liens d'holonomie). Les différents arbres ainsi obtenus, sont comparés pour évaluer le degré de leur ressemblance. Les arbres qui partagent le même thème, sont fusionnés pour former un arbre minimal qui servira pour l'interrogation de tous les documents appartenant au même domaine.

De façon générale, comme l'illustre la figure 4.9, l'approche comprend 3 étapes :

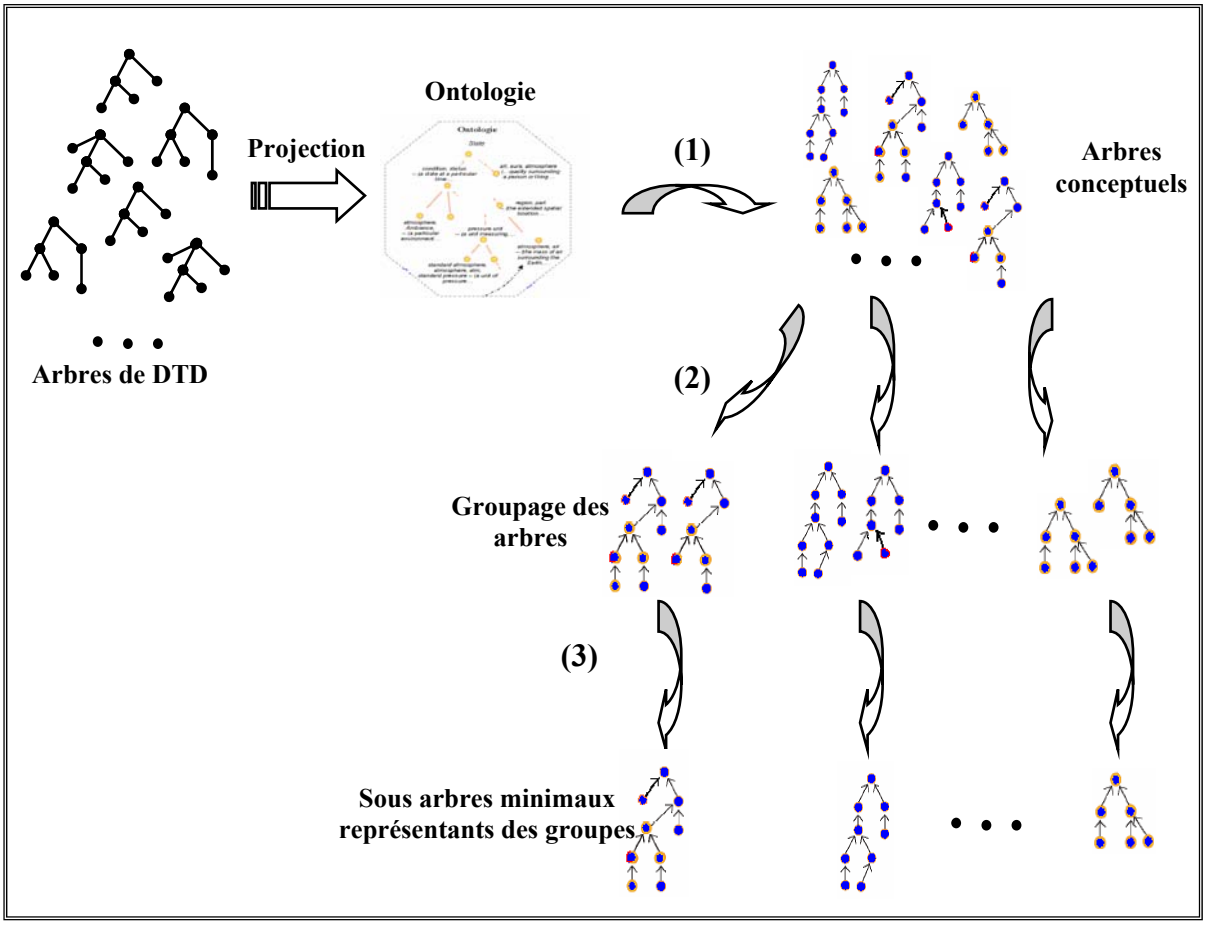

Figure -4.9- Schéma général de la deuxième approche.

(1) : La première étape consiste à projeter les arbres des DTD sur l'ontologie pour construire les arbres conceptuels correspondants. Dans cette étape, une phase de désambiguïsation est nécessaire pour sélectionner pour chaque balise le concept qui correspond le mieux à son sens dans son contexte. Elle se fait comme dans la première approche.

(2) : Dans la seconde étape, nous cherchons à trouver d'éventuels groupements des arbres selon les domaines qu'ils décrivent en effectuant une comparaison des arbres conceptuels deux à deux qui déterminera s'il décrivent un même domaine.

(3) : Dans la dernière étape, les éléments de chaque groupe construit précédemment seront représentés par rapport à un même référentiel, qui consiste en un sous arbre minimal de l'ontologie qui contient tous ces arbres. Il servira comme interface pour l'interrogation de la collection.

Dans les sections suivantes seront décrits en détails chacun de ces traitements.

4.2.3.2 Projection des DTD sur l'ontologie

Pour chaque DTD de la collection, on construit un arbre comportant ses balises en gardant les relations de hiérarchie entre elles.
Soit un arbre de DTD représenté comme dans la première approche : $d_s = (N, A, L)$. Cet arbre est ensuite projeté sur l'ontologie à fin de détecter les concepts correspondants aux noms des balises qu'il comporte. Cela se fera comme dans la première approche. L'arbre d_s sera représenté comme suit : $d_c = (N_c, L)$ où : N_c correspond aux noeuds représentant les concepts correspondant aux noms des balises (éléments ou attributs) de la DTD.
Soit la hiérarchie de concepts H d'une ontologie où tous les arcs sont représentés par les liens est-un (is-a) et partie de (part of). Les noeuds de d_c, une fois projetés sur H, peuvent ne pas former un sous arbre de H. Dans un souci de parachever la représentation, on les complète par des noeuds intermédiaires pour former le *sous arbre minimal* de d_c, nommé H_d..
Nous obtenons à la fin de cette étape l'ensemble des DTD de la collection sous forme de hiérarchies de concepts reliés par les liens d'hypéronymie et d'holonomie.

4.2.3.3 Regroupement des arbres conceptuels obtenus

Une collection de documents XML peut contenir des documents de divers domaines. Pour cela, il peut être nécessaire de comparer les arbres conceptuels obtenus à l'étape précédente pour déterminer des groupes d'arbres se rapportant au même thème.
L'idée est que si deux documents décrivent un même domaine, les deux représentations se rapprocheront, et auront des structures qui se recouvrent en partageant beaucoup de concepts, et l'inverse est correct.

Ainsi, les arbres sont comparés deux à deux pour évaluer le degré d'inclusion de l'un dans l'autre et déterminer jusqu'à quel point ils décrivent les éléments d'un même domaine. Les comparaisons des arbres sont basées sur une simple somme des concepts appartenant aux deux arbres comparés :

$$Compare(H_{di}, H_{dj}) = \sum_{n \in (H_{di} \cup H_{dj})} \mu_{H_{di}}(n) \wedge \mu_{H_{dj}}(n)$$

Où $\mu_{Hd}(n)$ est une fonction binaire telle que : $\mu_{H_d}(n) = \begin{cases} 1, & si\ n \in H_d. \\ 0, & sinon. \end{cases}$

La fonction Compare permet ainsi de calculer une valeur traduisant jusqu'à quel point deux arbres H_{di} et H_{dj} inclus les mêmes concepts. D'autres fonctions peuvent être utilisées, notamment pour tenir en compte de la hiérarchie des balises.

4.2.3.4 Représentation commune des DTD

Dans cette dernière étape, nous cherchons à représenter tous les arbres conceptuels relatifs à un même domaine par rapport à un même référentiel, c'est-à-dire trouver l'arbre conceptuel minimal qui contiendra au même temps tous ces arbres. Il s'agit donc, de trouver le sous-arbre minimal de l'ontologie qui contient au même temps tous les concepts apparaissant dans les arbres individuels.

A la fin de ces traitements, nous obtenons pour chaque domaine couvert par les documents de la collection, un arbre minimal qui servira de schéma d'interrogation pour les utilisateurs.

4.2.3.7 Comparaison des deux approches

Sur le plan démarche de résolution du problème posé, la première approche tente plutôt de traduire la partie structurelle de la requête utilisateur vers des noms de balises des autres DTD de la collection. La seconde tente par contre de proposer une structure générique pour l'interrogation englobant toutes les structures du corpus.

Sur le plan implémentation, les deux approches partagent les parties extraction et désambiguïsation en vue de rattacher les noms des balises aux concepts de l'ontologie. Cependant, la première approche semble plus simple, le calcul des similarités entre les différents arbres conceptuels dans la seconde approche est plus complexe, dans la mesure où il nécessite des fonctions de gestion de chemins.

4.2.3.8 Conclusion

Nous avons présenté une approche pour la construction d'une structure générique d'interrogation de corpus hétérogène en utilisant une ontologie. Cela se fait en trois étapes. En premier lieu, les arbres des DTD sont représentés par des sous arbres de l'ontologie. Les nœuds correspondent aux concepts extraits relatifs aux noms des balises XML et les

arcs les relations *IS-A et PART-OF*. Une phase de désambiguïsation permet de sélectionner pour chaque balise le concept qui correspond le mieux à son sens dans son contexte. En second lieu, une comparaison des arbres conceptuels deux à deux est faite pour déterminer éventuellement des groupements (classes) des arbres selon les domaines qu'ils décrivent. En dernier lieu, les éléments de chaque groupe construit précédemment seront représentés par rapport à un même référentiel, qui consiste en un sous arbre minimal de l'ontologie qui contient tous ces arbres. Il servira comme interface pour l'interrogation de la classe qu'il représente dans la collection.

Dans la section suivante, nous présentons une autre approche qui tente non pas de concevoir une structure générique, mais plutôt de convertir les documents dans un schéma de médiation en utilisant des techniques d'apprentissage automatique. Ce schéma de médiation peut être fourni par les experts du domaine ou choisi parmi les schémas du corpus.

4.3 Utilisation des techniques d'apprentissage

Préambule

Les méthodes d'apprentissage automatiques ont connu un succès important dans le domaine de la RI. Divers travaux dans la littérature tentent actuellement de les adapter aux données structurées, notamment dans les tâches de classification, de clustering, et de recherche. On trouve aussi, des travaux qui proposent d'utiliser les méthodes d'apprentissage pour résoudre le problème de l'hétérogénéité des structures des données. Nous citons d'abord les travaux de [Doan et al, 03] dans le cadre de la correspondance entre schéma XML en BD. Ces travaux proposent l'utilisation de classifieurs probabilistes fondés sur l'exploitation de plusieurs types d'information (comme les noms des éléments, leurs types, …) pour établir des règles d'association entre les éléments de structure des documents et du schéma choisi pour la médiation. Cependant, bien que les résultats sont très encouragent dans le domaine des BD, cette approche transposée au domaine de la recherche documentaire ne s'adapte pas bien à la structure des documents. Une structure qui présente des schémas complexes où il sera nécessaire, en plus de l'attribution d'une étiquette à chaque élément, de positionner correctement les uns par rapport aux autres.
Puis, les travaux de [Denoyer, 04] dans le cadre de la RIS et la tâche hétérogène d'INEX proposent de transformer les documents structurés issus de diverses sources dans un schéma de médiation connu. Ils se basent sur un modèle stochastique de représentation des documents structurés défini dans un cadre statistique général. Cette approche est également intéressante, mais il y a lieu de dépasser certaines restrictions qu'elle pose, à savoir que les

documents ne sont pas tous de mêmes structures et qu'aucune information sur la structure n'est utilisée (comme la sémantique des étiquettes).
D'où, la nécessité de proposer une solution qui prend en considération au même temps la sémantique et la structure des documents. Un travail dans se sens est effectué dans le cadre de la conversion des documents HTML vers des documents XML [Fuselier et al, 05]. Il propose une méthode d'apprentissage pour réaliser cette conversion en utilisant à la fois des informations sur le contenu des documents et les grammaires probabilistes pour générer la structure cible voulue en respectant les contraintes grammaticales décrites par la DTD cible.

Dans le même ordre d'idées, nous proposons d'appliquer l'approche de [Fuselier et al, 05] à la conversion de documents d'une structure XML vers une autre structure XML. Nous utilisons des classifieurs probabilistes sur les données structurés XML pour estimer des étiquettes terminales de la structure cible pour les fragments textuels des documents de la structure source, et les grammaires probabilistes pour générer la structure cible voulue. Nous tentons de surpasser les contraintes posées dans [Denoyer, 04] en considérant le cas général où les documents peuvent être étiquetés et structurés différemment, et nous proposons de traiter les cas de suppression ou de fusion d'éléments qui peuvent en résulter. De plus, nous proposons d'exploiter à la fois les informations sur la structure et le contenu des éléments XML.

Dans la suite, nous donnons un aperçu des divers modèles d'apprentissage et leurs applications aux données structurées. Puis, nous décrivons notre approche pour la conversion de documents XML.

4.3.1 L'apprentissage

L'apprentissage automatique est un processus qui permet de chercher une fonction pour estimer sur la base d'un ensemble d'observations X (ensemble d'apprentissage) les paramètres d'un modèle qui sera applicable sur toutes les données.
On distingue deux types d'apprentissage : l'apprentissage supervisé où à chaque observation de l'ensemble X est associé une réponse observée dans un ensemble Y. La recherche d'une fonction se fera dans un sous ensemble de fonctions de $X \rightarrow Y$. Dans l'apprentissage non supervisé où l'on dispose pas de l'ensemble Y, on cherche plutôt à caractériser l'ensemble des observations.
L'apprentissage automatique est appliqué dans divers domaines. En particulier, dans le domaine du traitement des documents textuels, l'apprentissage automatique propose une gamme de modèles permettant la recherche et l'analyse de l'information textuelle. Ils sont largement utilisés en RI, notamment pour les tâches de classification, clustering et de recherche.

Avec l'avènement des documents structurés, des travaux spécifiques au traitement de ce type de données, ont été proposés ([Wolff et al, 00], [Piwowarski, 03a], [Kamps et al, 04], [Vittaut et al, 04], etc.)

Dans la suite nous décrivons les principaux modèles d'apprentissage de la littérature appliqués pour le traitement des documents textuels et leurs extensions aux documents structurés. Nous distinguons deux principaux types de modèles : génératifs et discriminants.

4.3.1.1 Les modèles génératifs

Un modèle génératif est associé à un processus stochastique qui modélise comment certaines informations présentes dans un document ont été générées. Nous pouvons distinguer trois types de modèles génératifs classiques qui ont été appliqués pour le traitement de données textuelles puis structurées, à savoir :

4.3.1.1.1 Les modèles Naïve Bayes

Le modèle Naïve Bayes est l'un des modèles génératifs classiques qui fut notamment utilisé pour la classification de documents plats [Lewis, 98]. Sa simplicité, sa robustesse et ses bonnes performances pour cette tâche en font aujourd'hui un modèle de référence. Plusieurs versions de ce modèle existent, elles reposent sur des hypothèses statistiques légèrement différentes ou des représentations différentes des documents. Par exemple, le modèle de Poisson et ses variantes (e.g. 2-Poisson) constituent un des modèles de référence proposés pour la recherche d'information [Robertson et al, 94b], le modèle zéro-binomial négatif qui est un modèle alternatif au modèle de poisson où les distributions de mots sont paramétrées par deux paramètres au lieu d'un seul [Katz, 96]. Une revue des différents modèles Naïve Bayes se trouve dans [Eyheramendy et al, 03].

4.3.1.1.2 Les Modèles de Markov Cachés (MMC)

Le modèle de Markov caché est un modèle stochastique classique dédié au traitement des séquences, il a été introduit dans les années 1970 par [Baum et al, 70] puis par [Levinson et al, 83]. Une très bonne présentation des MMCs se trouve dans [Rabiner, 89]. La première application de ces modèles était dans le domaine de la reconnaissance de la parole [Jelinek, 82], puis ils ont été adoptés pour différentes problématiques de la recherche d'information et du traitement du langage naturel. Le modèle de langage proposé dans [Miller et al, 99] pour la recherche d'information, le modèle d'extraction d'information dans des documents plats [Amini, 01] et le modèle génératif pour la classification de documents plats présentant une hétérogénéité thématiques [Denoyer, 04] sont tous basés sur les MMCs.

Actuellement, plusieurs modèles stochastiques basés sur le formalisme des MMCs ont été créés et étudiés pour le traitement des documents structurés. Les plus connus sont les modèles de Markov Cachés Hiérarchiques (HHMM- *Hierarchical Hidden Markov Model*) pour le traitement de séquences [Fine et al, 98], [Murphy et al, 01], [Skounakis et al, 03] et [Xie et al, 03], à coté avec les modèles d'arbres de Markov Cachés (HTMM- *Hidden Tree Markov Model*) qui sont une généralisation des MMCs pour la génération d'arbres, qui ont été utilisés pour la classification de pages HTML [Diligenti et al, 01a] et pour la classification d'images [Diligenti et al, 01b] et [Diligenti et al, 03].

4.3.1.1.3 Les Réseaux Bayésiens

Un réseau bayésien est un formalisme probabiliste de représentation graphique des dépendances conditionnelles entre des variables aléatoires. Les réseaux bayésiens ont historiquement été développés dans le domaine de la prise en compte de l'incertain dans les systèmes de décision et les systèmes experts. Pour de plus amples détails les articles [Charniak, 91], [Jensen, 96] sont de bonnes références.

Pour permettre une prise en compte des dépendances séquentielles entre variables, les réseaux bayésiens dynamiques sont définis comme extension aux réseaux bayésiens classiques [Gharamani, 97]. Les réseaux bayésiens ont été largement utilisés dans le domaine de la recherche d'information surtout pour la tâche de recherche, un exemple est le modèle INQUERY [Callan et al, 92], ils ont même été adoptés pour la recherche dans des documents structurés, un exemple est le modèle dédié à la recherche d'information dans des documents XML [Piwowarski, 03a].

4.3.1.2 Les modèles discriminants

Les modèles discriminants s'inscrivent dans le cadre spécifique de la problématique de classification de textes. Ils sont considérés comme des méthodes permettant de trouver des frontières entres des exemples pour les classer en groupe. Nous distinguons deux grandes classes de modèles discriminants, à savoir :

4.3.1.2.1 Les réseaux de neurones

Les réseaux de neurones sont eux aussi une classe des modèles d'apprentissage, ils ont vu leurs débuts avec le modèle de perceptron [Rosenblatt, 58] et l'Adaline [Widrow et al, 60]. Dans les années 1980, l'essor de ces modèles a été déclanché par la découverte du mécanisme de rétro-propagation [Rumelhart et al, 86]. Les modèles RAAMs (Recursive Auto Associative Memory) et leurs dérivés les LRAAMs ([Pollack, 90], [Melnik et al, 90], [Angeline et al, 93] et [Sperduti, 93]) sont basés sur les réseaux de neurones et permettent

d'encoder une structure en un vecteur puis de décoder un vecteur en la structure correspondante.
Dans le domaine de la recherche d'information, les réseaux de neurones ont été utilisés pour l'extraction d'information, le travail de [Amini, 01] en est un exemple, et pour la classification, un exemple est le travail de [Zaragoza, 99]. Ces travaux ont été étendus pour répondre aux besoins de l'apprentissage à partir de données structurées. La technique du BPTS (Back Propagation Through Structure) proposée dans [Goller et al, 96] permet de faire de l'inférence à partir de graphes acycliques dirigés.

4.3.1.2.2 Les machines à Vecteur de Support (MVS)

Les machines à vecteur de support sont développées aux débuts des années 1990 [Vapnik, 95]. Une MVS est un classifieur binaire qui sépare les données dans un espace vectoriel par un hyperplan séparateur en se basant sur le calcul d'un produit scalaire entre les données projetées sur cette espace. Les MVSs ont été utilisées pour la classification supervisée de documents textuels plats à partir de 1998 par [Joachims, 98]. En tant que machines discriminantes, elles ont montré une supériorité assez nette par rapport aux modèles génératifs classiques et ont dès lors suscité un certain engouement pour la classification de documents.
Cependant, de nombreux exemples de classification ont montré que les données ne sont pas toujours linéairement séparables. Pour surpasser ce problème, les modèles basés sur les fonctions noyaux ont été développés, ils permettent de projeter les données dans un autre espace où ces dernières seront linéairement séparables. Les fonctions noyaux présentent de multiples propriétés qui sont présentées dans [Cristianinni et al, 00].

Les modèles à base de noyaux ont été étendus pour les adapter aux séquences et aux structures, les *strings kernels* définis dans [Watkins, 02] et [Lodhi et al, 02], sont des noyaux qui calculent une similarité entre deux chaînes de caractères. Plusieurs autres noyaux sur le traitement des séquences existent, ils sont détaillés dans [Shawe et al, 04]. Des noyaux d'arbres ont été également développés pour le calcul de similarité entre différents arbres [Collins et al, 02]. Les noyaux de convolution [Haussler, 99] offrent un cadre général de construction de fonctions noyaux récursives, ils ont été appliqués dans divers domaines comme la bioinformatique, la reconnaissance d'écritures manuscrites, etc.

4.3.1.3 Les modèles mixtes

Il est habituel dans le cas de la classification, et notamment dans le cas de la classification de grands corpus textuels d'utiliser de préférence des modèles discriminants plutôt que des modèles génératifs moins performants. En effet, ces derniers montrent généralement, sur les corpus habituels de test, des performances moins bonnes. Une des raisons du choix des modèles discriminants fût avancée par Vapnik dans [Vapnik, 95] par

la phrase suivante : «*Il est préférable de résoudre directement le problème de classification plutôt que de résoudre un problème général intermédiaire* (au sens des *P(d/ c)* des modèles génératifs) ».
Cependant, une comparaison entre les modèles génératifs (présenté par le modèle Naïve Bayes) et les modèles discriminants (présentés par le modèle de régression logistique qui est le modèle discriminant équivalent au modèle Naïve Bayes) effectuée par [Ng et al, 01] modère le propos précédant. Elle a montré que les modèles génératifs sont plus performants lorsque l'on dispose de peu d'exemples d'apprentissage, et l'inverse est vrai. C'est-à-dire que les modèles discriminants sont plus performants lorsque l'on dispose d'un nombre suffisant d'exemples d'apprentissage. C'est pourquoi le choix entre classifieurs discriminants et classifieurs génératifs dépend grandement du nombre d'exemples disponibles.

Plus récemment, plusieurs méthodes ont été développées qui tentent de tirer profit des performances des deux types de modèles : génératifs et discriminants. Cela soit par combinaison des deux modèles, par exemple le modèle de combinaison « Naïve Bayes-Régression logistique » utilisé dans [Raina et al, 03] pour la classification de données textuelles. Soit par transformation d'un modèle génératif en modèle discriminant, par exemple en utilisant les fonctions « p-noyau » [Cristianinni et al, 00] ou la méthode du « noyau de Fisher » [Jaakkola et al, 99] et [Tsuda et al, 04].

4.3.1.4 Autres modèles d'apprentissage

Parallèlement au développement de méthodes générales de traitement des données structurées, comme les méthodes à base de noyaux et les méthodes stochastiques que nous venons de décrire, de nombreux travaux se sont intéressés au traitement spécifique des données HTML ou XML. Deux principales familles de méthodes de classification ont été proposées pour ce type de documents :

- La première famille est spécifique à la classification de pages HTML, elle regroupe un ensemble de modèles qui se basent sur la combinaison de classifieurs et/ou la pondération des différentes parties d'un document HTML, on peut citer notamment les travaux de [Quek, 97], [Chakrabarti, 98], [Cline, 99], [Dumais et al, 00] et [Yang et al, 02].

- La seconde famille de modèles est plus adaptée au traitement des documents structurés en général. Le travail de [Yi et al, 00] propose de représenter un document par un vecteur et développe un modèle probabiliste pour la classification des documents. Ce modèle présente cependant des difficultés à s'adapter à des structures complexes. Le travail de [Diligenti et al, 01a] propose d'utiliser le formalisme des HHMMs pour la modélisation de documents structurés, les performances de classification obtenues présentent une amélioration faible par rapport au modèle Naïve Bayes. Le travail de [Denoyer, 04] propose un modèle générique pour la représentation de documents structurés qu'il utilise

pour la classification de grands corpus de documents XML. Le travail de [Doan et al, 03] propose l'approche dite « IMAP » pour la classification supervisée d'étiquettes d'éléments XML pour répondre au problème du « Schema Matching » en BD, la méthode repose principalement sur la combinaison de classifieurs qui prennent en compte différents types d'information (Naïve Bayes pour le contenu, classifieur XML pour la structure, classifieur de noms d'éléments, etc.) pour caractériser un nœud d'un document XML. Puis un méta classifieur est défini qui permet l'apprentissage de la combinaison des scores des différents classifieurs locaux.

4.3.1.5 Conclusion

Nous avons introduit un certain nombre de techniques d'apprentissage statistique. Nous avons montré comment il était possible d'utiliser (ou d'adapter) des modèles pour traiter des données structurées. Les documents structurés, avec notamment le fort développement des documents XML et le rapprochement des communautés BD et RI ont marqué le début du développement de modèles capables d'appréhender des documents structurés.

Dans la suite, nous allons proposer une approche par classification probabiliste et arbre de dérivation pour la conversion de documents XML.

4.3.2 Approche par classification probabiliste et arbre de dérivation pour la conversion de documents XML

4.3.2.1 Vue globale de l'approche

Le problème que nous cherchons à résoudre est la conversion automatique de documents XML de structures différentes vers une structure générique qui représente la DTD de médiation. Nous supposons que l'on dispose d'une collection de documents destinés à être transformés, et d'un ensemble d'apprentissage pour pouvoir apprendre un schéma de conversion reproductible. Nous supposons de plus que nous possédons également un schéma de médiation qui est le plus adapté pour l'interrogation du corpus considéré. Nous considérons le cas général, où les documents peuvent être étiquetés et structurés différemment. Pour notre proposition, Les documents seront ainsi transformés selon ce format avec un risque de perte d'un peu de leur sémantique étant donné que certains éléments peuvent ne pas se présenter dans le schéma de médiation.

Ainsi, un document sera représenté par une séquence de contenu $T\{t_1, t_2, \ldots, t_n\}$, où chaque t_i est un fragment de contenu qui représente le contenu d'un nœud feuille de l'arbre du document à qu'il faudra choisir la bonne étiquette parmi les étiquettes finales du schéma

cible (schéma de médiation) sachant qu'il est entouré d'information contextuelles telles que sa longueur, le type du contenu et le nom de la balise mère.

L'approche peut être divisée en trois phases :

1. La première phase se base sur une classification probabiliste. Cela consiste à parcourir la séquence des feuilles du document et d'estimer, en utilisant différents classifieurs probabilistes, une séquence d'étiquettes associée à ces feuilles. Les étiquettes possibles sont choisies parmi les éléments terminaux de la DTD cible qui sont de plus augmentées par une étiquette supplémentaire qui sera nommée "Delete" où seront assignés les contenus qui n'ont pas de correspondant dans le schéma cible. La séquence d'étiquette estimée S_{est} sera utilisée comme point d'entrée à la deuxième phase.

2. La deuxième phase consiste à estimer un arbre de dérivation *d* associé à la séquence S_{est} en utilisant les contraintes grammaticales décrites par la DTD cible.

3. La troisième phase consiste à combiner les résultats des phases précédentes pour réussir la conversion automatique des documents en respectant à la fois la sémantique et la structure des documents.

Dans les sections suivantes sont explicitées chacune de ces phases.

4.3.2.2 Classification probabiliste

Dans la première phase de l'approche, nous proposons d'utiliser les techniques issues de l'apprentissage supervisé. Divers classifieurs probabilistes sont définis dans la littérature qui prennent en charge les données semi structurées. Le but est d'estimer à partir d'une séquence de feuilles *t* la séquence d'étiquettes *s* qui est la plus probable. Cette estimation sera basée sur un modèle d'apprentissage entraîné sur un ensemble de documents exprimés à la fois dans le schéma d'origine et dans le schéma de médiation, et où les nœuds feuilles seront couplés avec leurs étiquettes. Le but de la phase d'entraînement est de déterminer des caractéristiques propres à chaque étiquette qui permettront de déterminer à partir d'un contenu t_i la probabilité de chaque étiquette s_i.

Donc, dans cette phase on cherchera à estimer pour chaque étiquette du schéma cible la probabilité $P(s_i / t_i)$ qui correspond à la probabilité que le contenu t_i peut apparaître sous une étiquette s_i.

Pour pouvoir appliquer les méthodes d'apprentissage existantes sur les instances à classer (c'est-à-dire les nœuds feuilles t_i), il faudra faire ressortir les caractéristiques propres à ces étiquettes qui permettront de décrire les spécificités d'un contenu textuel apparaissant sous une étiquette donnée.

Sur les données XML, on peut tirer diverses caractéristiques qui peuvent fournir des informations utiles sur le contexte d'une feuille. On peut classer ces caractéristiques en deux catégories :

- La première catégorie qui offre des propriétés exploitables, concerne les fragments textuels des documents, c'est-à-dire le contenu des nœuds feuilles. Ces propriétés décrivent des caractéristiques spécifiques aux chaînes de caractères contenues dans les feuilles. On peut penser au type des données (caractères numériques ou autres), et à la taille des fragments textuels (nombre de caractères contenus). Il sera ainsi possible de distinguer par exemple une date sachant que la chaîne "2000" est numérique, ou de classer la chaîne " Livre d'apprentissage supervisé" sous l'étiquette nommée "Titre" sachant sa taille réduite.

- Une seconde catégorie concerne les éléments structurels qui entourent une feuille. Il est intéressant de connaître l'étiquette mère de la feuille dans le schéma d'origine et de pouvoir établir des correspondances avec les étiquettes du schéma cible.

En utilisant, cette représentation, une feuille peut être projetée dans ce modèle et le classifieur probabiliste travaille alors sur cette représentation simplifiée.

Différents classifieurs probabilistes sont définis dans la littérature. On peut par exemple utiliser comme dans [Doan et al, 03], des classifieurs pour chaque type d'information exploitée par exemple :

- le classifieur Naive Bayes et le classifieur de contenu qui travaillent tous deux sur les contenus textuels,
- de plus un classifieur de nom et un classifieur XML qui travaillent sur les éléments structurels).

Puis un méta classifieur sera défini qui permet de combiner les résultats des classifieurs en apprenant des poids qui déterminent leur influence sur la probabilité finale.

On peut encore suivre la méthode de Fuselier et al. [Fuselier et al, 05], qui propose un classifieur basé sur le principe du maximum d'entropie (Classifieur MaxEnt). Il est très performant dans le domaine du traitement du langage pour résoudre des problèmes similaires. Ce classifieur cherche à maximiser la probabilité conditionnelle $P(s/t)$, il fait l'hypothèse qu'elle suit une loi exponentielle :

$$P(s/t) = \frac{1}{Z_\alpha(t)} exp\left(\sum_\alpha \lambda_\alpha \cdot f_\alpha(t,s)\right)$$

où $Z_\alpha(t)$ est un facteur de normalisation qui permet d'assurer que la valeur obtenue est une probabilité:

$$Z_\alpha(t) = \sum_s exp\left(\sum_\alpha \lambda_\alpha \cdot f_\alpha(t,s)\right)$$

La variable α permet d'effectuer une somme sur l'ensemble des propriétés choisies pour représenter le contexte d'une feuille t et la fonction $f_\alpha\ (t, s)$ représente la valeur de cet attribut α, pour le couple d'apprentissage (t, s). Les valeurs λ_α représentent une pondération des propriétés et permettent de déterminer un modèle pour lequel la distribution définie soit la plus exacte possible pour les données de l'ensemble d'apprentissage. Chaque choix de $\lambda = (\lambda_{\alpha 1}, \lambda_{\alpha 2}, \ldots, \lambda_{\alpha m})$ possible définit donc un modèle différent, le classifieur MaxEnt va déterminer parmi toutes ces possibilités le modèle optimal.

Une expérimentation de ces deux méthodes sur des collections de test, déterminera la performance de chacune d'elle.

A la fin de cette étape, on obtient les probabilités conditionnelles $P(s_i / t_i)$ pour toutes les étiquettes des éléments terminaux du schéma cible et tous les nœuds feuilles d'un document d'entrée. Deux cas particuliers peuvent se présenter :

- Le premier, concerne les feuilles qui ont une plus forte probabilité pour l'étiquette "Delete" (en effet, si des contenus textuels présentent des probabilités faibles pour toutes les étiquettes terminales, ils seront affectés à l'étiquettes "Delete"). Cela peut correspondre aux contenus qui ne sont pas présentés dans le schéma cible, ils seront présentés pour confirmer leur suppression.

- Le second cas, concerne les feuilles qui présentent une plus grande probabilité pour la même étiquette. Cela peut correspondre au cas de deux nœuds d'un schéma source qui sont fusionnés dans le schéma cible, ils seront également présentés pour confirmer leur fusion.

4.3.2.3 Estimation d'un arbre de dérivation

Pour l'estimation d'un arbre de dérivation pour la séquence des étiquettes terminales obtenue à l'issue de la première phase, on utilise les grammaires probabilistes hors contexte comme proposé dans les travaux de [Fuselier et al, 05] et de [Denoyer, 04]. En effet, les schémas ou DTD XML contiennent des déclarations qui définissent la structure recherchée des arbres des documents. Ces déclarations peuvent être transformées d'une manière équivalente vers le formalisme des grammaires hors-contextes [Papakonstantinou et al, 00].

D'une façon générale, une grammaire hors-contexte probabiliste G est définie par un 5-uplet $<N, T, R, S, P>$ où :

- N est l'ensemble des symboles non terminaux ;
- T est l'ensemble des symboles terminaux ;
- R est l'ensemble des règles r, de la forme : $A \longrightarrow \alpha$, avec $A \in N$ et $\alpha \in (N \cup T)^*$;
- S est l'axiome de départ ;

- P est l'ensemble des probabilités p_i associées aux règles r_i telles que :

$$\sum_{\alpha} P(A \to \alpha) = 1, \ \forall A \in N$$

Dans notre cas, nous cherchons à trouver une séquence s qui soit dérivable à partir de l'axiome de départ S d'une grammaire hort-contextes G qui est aussi la racine de l'arbre du schéma cible de la transformation. La suite des règles de production $S \xrightarrow{*} s$ utilisée pour produire la séquence définit un arbre de dérivation d qui est équivalent à un arbre XML.
Les règles de la grammaire hors-contexte probabiliste G peuvent être écrites manuellement en se référant à la grammaire XML. Elles peuvent également être inférées automatiquement à partir des documents instances du schéma cible de la collection d'apprentissage. Les probabilités de chacune de ces règles peuvent être calculées automatiquement en les dénombrant dans la collection d'apprentissage, en utilisant la formule suivante :

$$P(A \to \alpha) = \frac{nombre(A \to \alpha)}{\sum_{A \to \beta \in R} nombre(A \to \beta)}.$$

L'utilisation des grammaires probabilistes permet de proposer plusieurs arbres de dérivations pour une séquence d'éléments terminaux et donc d'introduire la notion d'arbre le plus probable. La probabilité d'un arbre de dérivation est calculée en effectuant le produit des probabilités des règles de production qui ont été utilisées pour le créer.

4.3.2.4 Combinaison des résultats

Pour réussir à convertir automatiquement des documents semi structurés, nous devons combiner les résultats de la classification probabiliste qui nous informent sur la probabilité de la classification des contenus textuels sous leurs étiquettes avec les résultats de l'estimation de la probabilité de structure décrite par la grammaire probabiliste hors contexte. ¨Pour ce faire, nous proposons d'utiliser la méthode employée dans [Fuselier et al, 05], pour les avantages qu'elle présente.

Cette méthode consiste à trouver le couple (s*, d*) qui soit le plus probable étant donné une grammaire probabiliste G et un document d'origine projeté sous la forme d'une séquence de contenu t. Avec s désigne une séquence d'étiquettes terminales du schéma cible obtenue à l'issu de la phase de la classification probabiliste et d un arbre de dérivation associé à s. Cela peut s'écrire :

$$(s, d)_{\max} = \underset{(s, d)}{argmax}\ P(s, d /t, G) \qquad (1)$$

L'avantage de cette approche est que nous cherchons à maximiser une probabilité jointe entre *s* et *d*. On peut simplifier l'équation (1) en utilisant le théorème de Bayes, on obtient l'équation (2) suivante :

$$(s, d)_{\max} = \underset{(s, d)}{argmax}\, P(d/s, t, G) * P(s/t, G) \quad (2)$$

Avec les hypothèses d'indépendances entre *t* et *d* et entre *s* et *G*, il est possible de reformuler l'équation précédente comme suit :

$$(s, d)_{\max} = \underset{(s, d)}{argmax}\, P(d/s, G) * P(s/t) \quad (3)$$

Où, la première partie correspond à la partie grammaticale et à la recherche de l'arbre de dérivation le plus probable pour une séquence d'étiquettes terminales fixée. Elle est calculée comme le produit des probabilités des règles de *G* formant la suite de dérivation $S \xrightarrow{*} s$ pour une séquence d'éléments terminaux donnée *s*. La deuxième partie correspond à la classification probabiliste d'une séquence de feuilles pour estimer une séquence d'étiquettes terminales. La formule indique qu'il faut calculer la probabilité jointe pour tous les couples (s*, d*) et prendre le couple dont la valeur est maximale. La théorie impose d'effectuer le test pour tous les couples de valeurs possibles afin de trouver un maximum global. Ce n'est malheureusement pas réalisable en pratique. Afin de pallier ce problème, une modification est effectuée sur l'algorithme inside-outside pour les grammaires hors-contextes probabilistes [Lari et al, 90], qui permet de calculer efficacement la probabilité d'un arbre de dérivation à partir d'une séquence donnée, en modifiant la partie inside de l'algorithme en injectant la distribution de probabilité estimée par le classifieur probabiliste. La modification apportée a permis de conserver la complexité de l'algorithme initial en $O(n3)$ avec *n* la longueur de la séquence.

4.3.2.5 Un exemple

Considérons les deux simples DTD suivantes (voir le tableau 4.2 ci-dessous), la première représentant le schéma cible vers lequel tous les documents doivent être transformés. La seconde représentant un schéma d'un ensemble de documents à transformer vers le schéma cible.

< !ELEMENT	MusicStore	(Location, CompactDisc +)>	
< !ELEMENT	CompactDisc	(AlbumTitle, Songlist)>	
< !ELEMENT	Songlist	(Track +)>	
< !ELEMENT	Track	(SongTitle, Singer +)>	
< !ELEMENT	Location	(#PCDATA)>	**DTD1 (Cible)**
< !ELEMENT	AlbumTitle	(#PCDATA)>	
< !ELEMENT	SongTitle	(#PCDATA)>	
< !ELEMENT	Singer	(#PCDATA)>	

< !ELEMENT	CDStore	(Address, CD +)>	
< !ELEMENT	Address	(City, Street, State)>	
< !ELEMENT	CD	(CDTitle, TrackList)>	
< !ELEMENT	TrackList	(Passage +)>	
< !ELEMENT	Passage	(Title, Singer +)>	
< !ELEMENT	City	(#PCDATA)>	**DTD2**
< !ELEMENT	Street	(#PCDATA)>	
< !ELEMENT	State	(#PCDATA)>	
< !ELEMENT	CDTitle	(#PCDATA)>	
< !ELEMENT	Title	(#PCDATA)>	
< !ELEMENT	Singer	(#PCDATA)>	

Tableau -4.2- Schéma source et cible de la transformation.

Avant de commencer la transformation, nous supposons avoir effectué au préalable une phase d'entraînement des classifieurs sur un ensemble de données d'apprentissage, et de plus, nous avons appris les probabilités des règles de la grammaire *G* du schéma cible, cela bien sûr après l'avoir transformé en une grammaire hors contexte.

Nous supposons maintenant, avoir testé les classifieurs probabilistes sur une séquence de 8 feuilles $t = \{t_1, \ldots, t_8\}$ d'un document suivant la DTD 2 du tableau 4.2. Pour estimer la distribution de la probabilité pour chaque classe terminale de la grammaire cible DTD 1 (à savoir : Location, AlbumTitle, SongTitle et Singer) de chaque feuille de *t,* et avoir obtenu les résultats suivants :

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Location	**0,4**	**0,5**	**0,4**	0,1	0,1	0,1	0,1	0,1
AlbumTitle	0,2	0,2	0,3	0,3	**0,4**	0,2	0,2	0,2
SongTitle	0,3	0,1	0,2	**0,4**	0,3	**0,4**	**0,4**	0,2
Singer	0,1	0,2	0,1	0,2	0,2	0,3	0,3	**0,5**

Tableau -4.3- Estimation des probabilités pour chaque classe et chaque feuille.

En suivant la distribution de probabilité estimée, la séquence des classes terminales la plus probable s_{max} *(représentée en rouge gras dans le tableau 4.3),* est composée des classes terminales les plus probables pour tous les fragments textuels des feuilles $t_{i\ ,\ i=1\ldots8}$, il s'agit de la séquence :

s_{max} = "Location Location Location SongTitle AlbulTitle SongTitle SongTitle Singer"

avec la probabilité :

$$P(s_{max}) = P(s_{max}/t) = \Pi\, P(s_{imax})/t_i) = 0{,}4*0{,}5*0{,}4*0{,}4*0{,}4*0{,}4*0{,}4*0{,}5 = 0{,}001024.$$

Mais, comme on le remarque, cette séquence bien qu'elle soit maximale, elle ne possède pas d'arbre de dérivation dans la grammaire du schéma cible. De plus, avant d'aller lui chercher un arbre de dérivation, il est bien de remarquer que les trois premiers fragments textuels présentent tous une plus forte probabilité pour la même classe, ils seront donc présentés d'abord à l'utilisateur pour décider s'il n'y a pas lieu de les fusionner dans la même classe. Et comme dans notre cas il y a raison à les fusionner, on obtient alors la nouvelle séquence :

s'= "Location SongTitle AlbulTitle SongTitle SongTitle Singer"

qui ne possède toujours pas d'arbre de dérivation dans notre grammaire G de la DTD1 cible.

Néanmoins, il existe deux arbres de dérivation valides pour cette séquence, qu'on notera : (s_1, d_1) et (s_2, d_2) comme le montre la figure 4.10.

En calculant la probabilité jointe $P(s_1,d_1/t,G)$ telle que:

$$P(s_1,d_1/t,G) = P(d_1/s_1, t, G) * P(s_1/t, G)$$ (par application de la règle de Bayes).
$$= P(d_1/s_1, G) * P(s_1/t)$$ (en supposant l'indépendance entre d_1 et t et entre s_1 et G).

où : la probabilité $P(d_1/s_1, G)$ est calculée en utilisant les probabilités des règles de la grammaire G composant la dérivation $S \xrightarrow{*} s_1$, et la probabilité $P(s_1/t)$ est calculée en utilisant les résultats des estimations dans le tableau 4.3.

En calculant de la même façon la probabilité jointe $P(s_2,d_2/t,G)$:

$$P(s_2,d_2/t,G) = P(d_2/s_2, t, G) * P(s_2/t, G)$$ (par application de la règle de Bayes).
$$= P(d_2/s_2, G) * P(s_2/t)$$ (en supposant l'indépendance entre d_2 et t et entre s_2 et G).

où : la probabilité $P(d_2/s_2, G)$ est calculée en utilisant les probabilités des règles de la grammaire G composant la dérivation $S \xrightarrow{*} s_2$, et la probabilité $P(s_2/t)$ est calculée en utilisant les résultats des estimations dans le tableau 4.3, on trouve que la dérivation d_1 est celle qui maximise la probabilité jointe et produit donc le résultat optimal de la conversion donnant la séquence des terminaux :

s_1= "Location AlbumTitle SongTitle Singer SongTitle Singer".

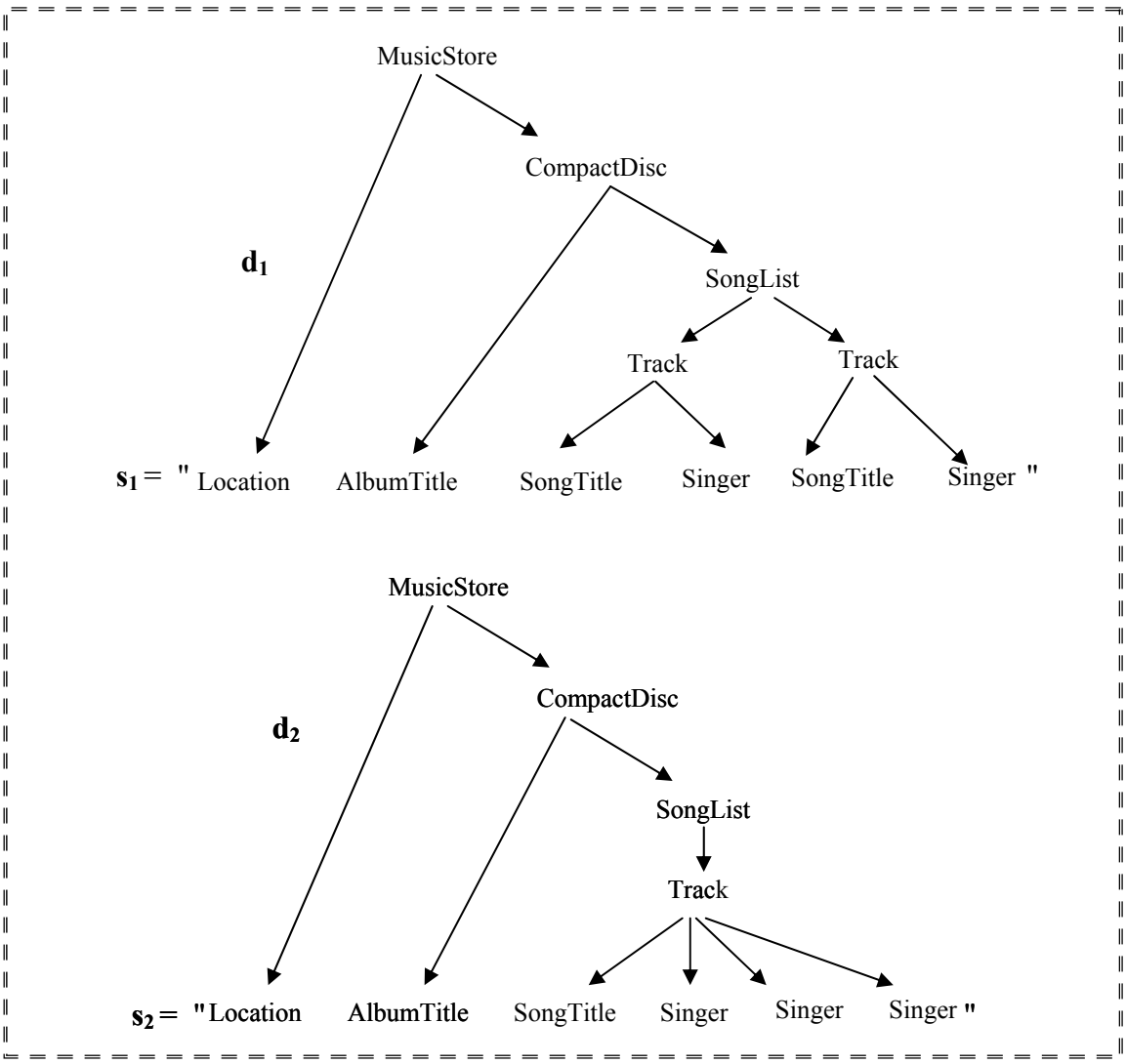

Figure -4.10- Arbres de dérivation possibles pour la séquence de l'exemple.

4.3.2.6 Conclusion

Nous avons présenté une approche de conversion automatique de documents XML vers une structure de médiation qui servira pour l'interrogation de la collection. De façon générale, l'approche présentée s'inscrit dans le cadre de l'application des techniques de l'apprentissage automatique dans le domaine de la RI. Elle consiste en trois étapes. Dans la première, nous cherchons à estimer pour une séquence de feuilles d'une structure source, une séquence d'étiquettes terminales de la structure cible. Cela sera basé sur un modèle d'apprentissage entraîné sur un ensemble de documents exprimés à la fois dans le schéma d'origine et dans le schéma de médiation, et où les nœuds feuilles seront couplés avec leurs étiquettes terminales du schéma cible. Nous proposons de combiner des classifieurs probabilistes permettant de prendre en charge les caractéristiques des documents XML. Dans la seconde étape, on se propose d'estimer un arbre de dérivation pour la séquence

d'étiquettes terminales obtenue dans la première étape en se basant sur les grammaires hors contexte probabiliste. Dans la dernière étape, nous combinons les résultats de la classification probabiliste (qui nous informe sur la probabilité de la classification des contenus textuels sous leurs étiquettes) avec les résultats de l'estimation de la probabilité de structure (décrite par la grammaire probabiliste hors contexte). La combinaison s'effectue en maximisant la probabilité jointe d'estimer une séquence d'étiquettes terminales et de dériver un arbre pour cette séquence.

Nous proposons un traitement semi automatique permettant de prendre en charge les cas de suppression et de fusion d'éléments entre les structures sources et cible.

Nous estimons que cette approche est prometteuse. Une implémentation et des tests avec un système de recherche d'information sur des documents structurés nous permettront de mieux juger de sa performance et de son efficacité.

4.4 Conclusion

Pour répondre à la problématique inhérente à l'interrogation de corpus de documents XML de structures hétérogènes, plusieurs pistes de recherche sont envisageables. De ce fait, nous avons proposé trois solutions.

Dans la première, nous proposons de résoudre le problème lié à la différence morphologique des noms de balises en proposant de construire un dictionnaire des balises synonymes de la collection. Puis de faire correspondre les conditions de structures exprimées dans la requête utilisateur avec les éléments effectivement présents dans la collection.
L'avantage principal de cette approche réside dans sa simplicité et son caractère totalement automatique. De plus, elle peut être utilisée dans les deux autres approches. Elle est en fait en partie incluse dans la seconde approche. Elle peut être utilisée dans la dernière approche pour aider à établir d'une façon automatique les couplages des nœuds feuilles dans les structures sources avec les étiquettes terminales de la structure cible. Cela en prenant en considération les relations de synonymie entre les étiquettes terminales des structures sources et les étiquettes terminales de la structure cible.
L'inconvénient principal que l'on peut lui attacher est qu'elle n'établit les correspondances qu'entre les noms des éléments, obligeant ainsi le système à faire des appels au dictionnaire pour chaque élément de la requête.

La seconde approche d'utilisation d'ontologie est plus générale dans la mesure où l'on s'intéresse aux deux problèmes de variation linguistique et de la variation de la

structuration des balises. Dans cette approche, nous faisons recours à l'ontologie pour concevoir une structure générique couvrant les différentes structures des documents de la collection. Cette structure une fois obtenue, servira d'interface pour une interrogation générique de la collection.
L'avantage de la méthode est qu'elle permet à l'utilisateur de ne gérer qu'une seule DTD lors de la formulation de ses requêtes. Sa contrainte forte est la disponibilité de l'ontologie.

Dans la troisième approche, nous avons proposé une méthode de conversion de documents XML de structures hétérogènes vers un schéma de médiation par classification probabiliste et arbre de dérivation. Nous avons considéré le cas général de conversion sans faire des restrictions sur les structures sources ou cible de la conversion.
L'intérêt de cette approche vient de l'automatisation de cette conversion en utilisant des techniques d'apprentissage pour découvrir des règles de transformation applicables pour une collection de documents. Sa contrainte est liée à la disponibilité des données d'apprentissage.

Toutes les approches semblent prometteuses. Seule une expérimentation de ces propositions avec un système de recherche d'information structurée dans une collection de test, nous permettra de vérifier leur faisabilité et efficacité, de juger de leur performance et de leur apport réel à la recherche d'information structurée.

Conclusion générale

Synthèse

Les travaux présentés dans ce mémoire se situent dans le contexte général de gestion automatisée de corpus de documents XML de structures hétérogènes. Ils se sont particulièrement intéressés à la mise en œuvre d'approches pour l'interrogation de documents ayant des structures hétérogènes.

Les documents représentés sous le format XML couvrent tous les domaines (les bases de données, les intranets, le e-commerce, etc.). Ils sont actuellement sur le point de conquérir tout le Web. Le langage de représentation de documents semi structurés XML doit principalement son succès à sa flexibilité : toute personne peut écrire une DTD (Document Type Definition) pour définir la structure de ses documents sous le format XML. Une structure qui représente les informations dans la forme que la personne désire. Cependant, cette liberté de conception de DTD conduit de fait à l'élaboration de structures décrivant souvent les mêmes éléments mais avec des noms de balises différents et/ou agencés différemment. Ceci cause de réels problèmes au niveau du stockage, de l'intégration et de l'interrogation de données dans ces larges collections de documents hétérogènes. On peut alors dire que l'avantage majeur d'XML (son extensibilité) est au même temps son handicap majeur.

Dans le domaine de la recherche d'information, l'accès à ce type de documents soulève de nouvelles problématiques : un utilisateur désirant interroger une collection de documents semi structurés formule des requêtes portant à la fois sur le contenu et la structure des documents. L'utilisateur ne pouvant avoir une idée sur toutes les structurations possibles des documents, se voit alors confronter à la diversité des DTDs (schémas). Il ressort alors, au système de recherche d'information de fournir des moyens adéquats pour permettre une interrogation simplifiée et une recherche efficace aboutissant à des résultats couvrant toute la collection quelque soit la DTD.

Nous nous sommes intéressés dans ce mémoire à proposer des solutions pour répondre à de telles problématiques. Dans ce cadre, nous avons présenté principalement trois contributions traduisant différentes pistes de recherche possibles :

1. Dans la première contribution, et afin de pallier le problème lié à l'utilisation de balises synonymes dans les DTD, nous proposons d'utiliser une ontologie pour construire automatiquement un dictionnaire des balises synonymes. Nous nous appuyons en cela sur l'exploitation de la sémantique portée par les balises XML. Dans la première phase, chaque balise d'une DTD est projetée sur l'ontologie. Les noms des balises sont ainsi attachés de façon univoque, aux concepts (les points d'entrée) de l'ontologie.
Comme une balise peut avoir plusieurs sens, une seconde phase de désambiguïsation globale est opérée. Elle permet de sélectionner les concepts relatifs aux noms des balises dans le contexte de chaque DTD où les noms des balises XML se désambiguïsent "mutuellement". Une fois ce traitement effectué sur toutes les DTDs du corpus, les concepts retenus sont regroupés pour construire le dictionnaire des balises synonymes. A chaque concept sera associé la liste des balises qui lui sont synonymes. L'utilisateur pourra interroger la collection par les termes d'une quelconque DTD, le système de recherche interrogera le dictionnaire pour avoir les synonymes des termes structurels de la requête, puis générera des requêtes dans l'ensemble des DTDs du corpus.

2. Une seconde façon d'utilisation d'une ontologie s'est matérialisée dans une seconde proposition qui se base en plus de l'exploitation de la sémantique portée par les balises XML, sur l'exploitation de la structure hiérarchique inhérente aux documents XML. Cette approche est plus générale que la précédente dans la mesure ou elle tente de pallier les deux problèmes de l'hétérogénéité à savoir : la variation linguistique et la différence de structuration des balises.
En effet, les liens reliant les éléments XML peuvent être organisés suivant la relation générique/spécifique (Is-a) et la relation de composition (part-of). Globalement cette approche comprend trois phases. Dans une première phase, les arbres des DTDs de la collection sont projetés sur la hiérarchie de concepts de l'ontologie et les noeuds (concepts isolés) identifiés dans les DTDs sont détectés. Dans cette étape, comme dans la première approche, une opération de désambiguïsation est effectuée pour sélectionner pour chaque balise le concept qui correspond le mieux à son sens dans son contexte. A l'issue de cette étape chaque DTD est donc représentée sous forme d'un sous arbre. La seconde étape consiste à regrouper les sous arbres similaires. Ceci revient à comparer ces sous arbres deux à deux pour déterminer éventuellement des groupements des arbres selon les domaines qu'ils décrivent. Dans la dernière étape, il s'agit de construire le sous arbre minimale qui englobe les éléments de chaque groupe construit précédemment. L'idée derrière cela est de représenter les sous arbres obtenus à l'étape précédente par rapport à un même référentiel, qui consiste en un sous arbre minimal de l'ontologie qui contient tous ces arbres. Il servira comme interface pour l'interrogation de la collection.

3. Dans la troisième contribution, nous proposons d'utiliser des techniques issues du domaine de l'apprentissage automatique pour faire face à notre problématique. Notre approche consiste en un processus de conversion automatique des documents XML hétérogènes dans un schéma de médiation qui servira d'interface pour l'interrogation du

corpus. Nous supposons donc, que nous disposons d'un schéma de médiation (qui peut être fourni par les experts du domaine, ou choisi parmi les schémas de la collection) et d'un ensemble de données d'apprentissage où les contenus textuels sont reliés à leurs étiquettes dans le schéma médian. Le processus de conversion se divise globalement en trois étapes. La première étape, classification probabiliste consiste à parcourir la séquence des feuilles du document et d'estimer une séquence d'étiquettes associée à ces feuilles. Les étiquettes possibles sont choisies parmi les éléments terminaux du schéma de médiation. Ces estimations se basent sur les résultats fournis par un ensemble de classifieurs probabilistes entraînés sur l'ensemble des données d'apprentissage. La séquence d'étiquette estimée sera utilisée comme point d'entrée à la phase suivante. La seconde étape consiste à estimer un arbre de dérivation associé à la séquence d'entrée en utilisant les contraintes grammaticales décrites par le schéma cible. Cela se fait en exploitant les propriétés des grammaires hors contexte probabilistes. Une dernière étape consiste à combiner les résultats des phases précédentes pour réussir la conversion automatique des documents en respectant à la fois la sémantique et la structure des documents.

Perspectives

Ce travail n'est qu'un petit pas d'un long périple, il ouvre une porte sur de longues recherches qui peuvent être riches en résultats.

En guise de perspectives nous proposons en premier lieu, de réaliser des prototypes implémentant les approches proposées en parallèle avec un système de recherche d'information sur les documents semi structurés, afin d'étayer le bien fondé de ces approches, de les tester et même de les comparer sur des collections de test standards (INEX par exemple).

Ensuite, penser à tirer profit des avantages de la première approche, construction du dictionnaire des balises synonymes, en l'exploitant dans les deux autres approches qui ont besoin toutes deux d'identifier les synonymies qui peuvent exister entre les balises XML des documents de la collection. En effet, dans l'approche d'interrogation par une structure générique, elle y fait déjà partie intégrante. Dans l'approche de classification probabiliste et arbre de dérivation pour conversion de documents XML, elle peut être utilisée pour permettre d'établir automatiquement les couplages des nœuds feuilles dans les structures sources avec les étiquettes terminales de la structure cible en prenant en considération les relations de synonymies entre les étiquettes terminales des structures sources et les étiquettes terminales de la structure cible.

Bibliographie

[Abiteboul et al, 04] S. Abiteboul, I. Manolescu, B. Nguyen, and N. prada. A test plateform for the INEX heterogeneous track.In Pre-proceedings of INEX 2004,Dagstuhl, Allemagne, pages177-182.

[Abolhassani et al, 04] M. Abolhassani and N. Fuhr. Applying the divergence from randomness approach for content-only search in XML documents. In *Proceedings of ECIR 2004, Sunderland*, pages 409-419, 2004.

[Agrawal et al, 94] R. Agrawal and R. Srikant, "Fast Algorthms for Mining Association Rules", In Proceedings of the 20th VLDB Conference, Santigo, Chile.

[Aho et al, 91] A. Aho, R. Sethi et J. Ullman, Compilateurs − Principes, techniques et outils, Addison−Wesley, Reading, Massachussets USA, 1991.

[Allan et al, 98] J. Allan, J. Callan, M. Sanderson, J. Xu, and S. Wegmann. INQERY at TREC-7. In *Proceedings of TREC-7*, pages 201–216, 1998.

[Amer-Yahia, 04] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. Texquery : A fulltext search extension to Xquery. In *Proceedings of WWW 2004*, 2004.

[Amini, 01] M.Amini. *Apprentissage Automatique et Recherche de l'Information* : *application à l'Extraction d'Information de surface et au Résumé de texte*. PhD thesis, LIP6, Université Paris 6, Paris, France, July 2001.

[Angeline et al, 93] P. J. Angeline and J. B. Pollack, Hierarchical RAAMs : a Uniform Modular Architectutre, Technical Report, Ohio State University, USA, 1993.

[Anh et al, 02] V. N. Anh and A. Moffat. Compression and an ir approach to XML retrieval. In *Proceedings of INEX 2002 Workshop, Dagstuhl, Germany*, 2002.

[Aussenac-Gilles et al, 00] Aussenac-Gilles N., Biébow B., Szulman N., Revisiting Ontology Design: a method based on corpus analysis. Knowledge engineering and knowledge management: methods, models and tools, Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Management. Juan-Les-Pins (F). Oct 2000. R Dieng and O. Corby (Eds). Lecture Notes in Artificial Intelligence Vol 1937. Berlin: Springer Verlag. pp. 172-188. 2000.

[Baeza-Yates et al, 99] R. Baea-Yates and B. Riberto-Neto. *Modern Information Retrieval*. New- York : ACP Press, Addison-Wesley, 1999.

[Bagy,] D.Bagy, An Efficient Schema Matching Algorithm for an Automated Transformation of XML Documents,

[Baum et al, 70] L. E. Baum, T. Petrie, G. Soules and N. Weiss:A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains, in Ann. Lath. Stat., Volume 41(1), pages 164-171, 1970.

[Baziz, 05] Mustapha BAZIZ, Indexation conceptuelle guidée par ontologie pour la recherche d'information, thèse de DOCTORAT DE L'UNIVERSITE PAUL SABATIER, Décembre 2005.

[Beale et al, 95] Beale, Stephen, Sergei Nirenburg und Kavi Mahesh (1995): ``Semantic Analysis in the Mikrokosmos Machine Translation Project." In Proceedings of the Second Symposium on Natural Language Processing, Bangkok, Thailand, 2.-4. August 1995. 294-307.

[Bickel et al, 01] P. Bickel and K. Doksum. Mathematical Statistics: Basic Ideas and Selected Topics. Prentice Hall, 2001.

[Bonhomme, 98] Stéphane Bonhomme, Transformation de documents structurés, une combinaison des approches explicite et automatique, thèse de doctorat de l'université Joseph Fourier, 1998.

[Borst, 97] Pim Borst, Hans Akkermans, Jan Top: Engineering ontologies. Int. J. Hum.-comput. Stud. 46(2): 365-406 (1997).

[Boughanem et al, 92] Mohand Boughanem, C. Soulé-Dupuy: A Connexionist Model for Information Retrieval. DEXA 1992: 260-265.

[Boukottaya et al, 04] A. Boukottaya, C. Vanoibeek, F. Paganelli et O. Abou Khaled, Automating XML documents transformation using a conceptual modelling based approach, Proceeding of the 1st Asian-Pacific Conference on Conceptual Modelling, 2004.

[Braga et al, 02] D. Braga, A. Campi, E. Damiani, P. Lanzi, and G. Pasi. FXpath : flexible querying of XML documents. In *Proceedings of Eurofuse 2002*, 2002.

[Brin et al, 97] S. Brin, R. Motwani and C. Silverstein, "Beyond market baskets : generalizing association rules to correlations", In proceedings of the International Conference of Management of Data, Tucson, Arizona, pages 265-276.

[Callan et al, 92] J. P. Callan, W. B. Croft and S. M. Harding : The INQUERY Retrieval System, in proceedings of the International Conference in Database and Expert Systems Applications, pages 78-83, 1992.

[Carmel et al, 03] D. Carmel, Y. Maarek, M. Mandelbrot, and A. Soffer. Searching xml documents via xml fragments. In *Proceedings of SIGIR 2003*, pages 151-158, 2003.

[Chakrabarti et al, 98] S. Charkrabarti, B. E. Dom and P. Indyk: Enhanced Hypertext Categorization using Hyperlinks, in proceedings of ACM International Conference on Management of DATA, SIGMOD, pages 307-318, 1998.

[Charniak, 91] E. Charniak: Bayesian Networks without Tears, in AI Magazine, pages 51-61,
1991.

[Chiaramella et al, 96] Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical report, Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, 1996.

[Chua et al, 04] Chua, S.; Kulathuramaiyer, N.; Semantic Feature Selection Using WordNet Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on 20-24 Sept. 2004 Page(s):166 – 172.

[Clark et al, 99] J. Clark and S. Derose. XML Path Language (XPath) , version 1.0.
Technical report, World Wide Web Consortium (W3C), W3C Recommendation, Novembre 1999.

[Cleverdon, 67] C. Cleverdon. The Cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173{192, 1967.

[Cline, 99] M. Cline : Utilizing HTML Structure and Linked Pages to Improve Learning for Text Categorization, Undergraduate Honor Thesis, Department of Computer Science, University of Texas, USA, 1999.

[Collins et al, 02] M. Collins and N. Duffy: Convolution Kernels for Natural Language, in proceedings of Advances in Neural Information Processing Systems 14, pages 625-632, 2002

[Cooper, 68] W. Cooper. Expected search length : a single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *Am Doc.*, 19 :30–41, 1968.

[Cormen et al, 01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms (Section Edition). MIT Press, 2001.

[Cristianinni et al, 00] N. Cristianini and J. Shawe-Taylor: An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods), Cambridge University Press,
2000.

[Crouch et al, 02] C. Crouch, S. Apte, and H. Bapat. An IR approach to XML retrieval based on the extended vector model. In *Proceedings of INEX 2002 Workshop, Dagstuhl, Germany*, pages 98–99, 2002.

[Cruz et al, 04] Isabel F.Cruz, H. Xiao et F.Hsu ; " An ontology-based Framework for XML Semantic Integration", in (IDEAS 2004).

[Delobel et al, 03] C. Delobel, C.Reynaud, M.C. Rousset , J.P. Sirot and D. Vodislav,"Semantic Integration in Xyleme: a Uniform Tree-based Approach", Journal on Data and Knowledge Engineering, 44(2):pp 267-298, 2003.

[Denoyer, 04] L. Denoyer, Apprentissage et inférence statistique dans les bases de documents structurés : Application aux corpus de documents textuels, thèse de doctorat de l'université de Paris 6, 2004.

[Denoyer at al, 04] L. Denoyer, G. Wisniewski, and P. Gallinari. Document structure matching for heterogeneous corpora. In Proceedings of XML and IR workshop,SIGIR 2004.

[Dignum et al, 04] V. Dignum and R. van Zwol. Guidelines for topic development in heterogeneous collections. Guidelines of INEX 2004, 2004.

[Diligenti et al, 01a] M. Diligenti, M. Gori, M. Maggini and F. Scarselli, Classification of HTML documents using Hidden Tree Markov Models, In proceedings of ICDAR, pages 849-853, 2001.

[Diligenti et al, 01b] M. Diligenti, P. Frasconi and M. Gori, Image Document Categorization using Hidden Tree Markov Models and Structured Representations, In proceedings of the International Conference on Application of Pattern Recognition, 2001.

[Diligenti et al, 03] M. Diligenti, P. Frasconi and M. Gori, Hidden Tree Markov Models for Document Image Classification, In IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 25(4), pages 519-523, 2003.

[Ding et al, 01] Ying Ding, Robert Engels: IR and AI: Using co-occurrence Theory to Generate Lightweight Ontologies. DEXA Workshop 2001: 961-965.

[Doan et al, 03] A. Doan, P. Domingos, and A. Halevy. Learning to match the database schemas: A multistrategy approach. *Machine Learning*, 2003. Special Issue on Multistrategy Learning.

[Dumais et al, 00] S. T. Dumais and H. Chen, "Hierarchical Classification of Web Content", in proceedings of SIGIR 2000.

[Eyheramendy et al, 03] S. Eyheramendy, D. D. Lewis and D. Madigan : On the Naïve Bayes Model for Text Categorization, in proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, 2003.

[Fensel et al, 01] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider: OIL: An Ontology Infrastructure for the Semantic Web. IEEE Intelligent Systems 16(2): 38-45, 2001.

[Fine et al, 98] S. Fine, Y. Singer and N. Tishby: The Hierarchical Hidden Markov Model : Analysis and Applications In Machine Learning, Volume 32(1), pages 41-62, 1998.

[Frilley, 89] F. Frilley, Différenciation d'ensembles structurés, Doctorat informatique, Université de Paris VII, mars 1989.

[Fuhr et al, 03a] N. Fuhr and K. Grossjohann. XIRQL : a query language for information retrieval in XML documents. In *In Proceedings of SIGIR 2001, Toronto, Canada*, 2003.

[Fuhr et al, 03b] N. Fuhr, M. Lalmas, and S. Malik. INEX 2003 workshop proceedings, 2003.

[Fuller et al, 93] M. Fuller, E. Mackie, R. Sacks-Davis, and R. Wilkinson. Structural answers for a large structured document collection. In *Proceedings of ACM SIGIR 1993, Pitthsburgh*, pages 204–213, 1993.

[Fuselier et al, 05] Chidlovskii B., Fuselier J., « A Probabilistic Learning Method for XML Annotation of Documents », *Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edimbourgh, Scotland, August, 2005.

[Gangemi et al, 99] Gangemi A, Pisanelli DM, Steve G: An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies. Data and Knowledge Engineering, 1999, 31, pp. 183-220 (1999).

[Gao et al, 05] Guihong Gao and Jian-Yun Nie and Jing Bai: Integrating word relationships into language models. SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil, 2005.

[Gardarin, 02] G. Gardarin. *XML : Des bases de données aux services Web*. Dunod -01 Informatique, Paris 2002, 2002.

[Gharamani, 97] Z. Ghahramani: Learning Dynamic Bayesian Networks, in Adaptative Processing of Sequences and Data Structures, pages 168-197, 1997.

[Goldfarb, 90] C. Goldfarb. *The SGML Handbook*. Oxford University Press, Oxford, 1990.

[Goller et al, 96] C. Goller and A. K uchler. Learning Task-Dependant Distributed Represntations by Backpropagation Through Structure. In *International Conference on Neural Networks (ICNN-96)*, 1996.

[Gonzalo et al, 98] Gonzalo, J., Verdejo, F., Chugur I., Cigarrán J.: Indexing with WordNet synsets can improve text retrieval, in Proc. the COLING/ACL '98 Workshop on Usage of WordNet for Natural Language Processing, 1998.

[Govert et al, 02] N. G¨overt, M. Abolhassani, N. Fuhr, and K. Grossjohann. Contentoriented XML retrieval with hyrex. In *Proceedings of the first INEX Workshop, Dagstuhl, Germany*, 2002.

[Govert et al, 03] Gövert, N., Kazai, G., Fuhr, N., Lalmas, M. (2003): Evaluating the effectiveness of content-orientated XML Retrieval. Technical Report Computer Science 6, Technischer bericht, University of Drotmund.

[Grabs, 03] T. Grabs. Storage and Retrieval of XML Documents within a Cluster of Database Systems. PhD thesis, Institut f´ed´eral Suisse de Technologie de Z¨urich, 2003.

[Grabs et al, 02] T. Grabs and H.-J. Scheck. Flexible information retrieval from xml with PowerDB XML. In *Proceedings in the First Annual Workshop for the Evaluation of XML Retrieval (INEX)*, pages 26–32, December 2002.

[Grossman et al, 98] David Grossman and Ophir Frieder, Ad Hoc Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, 1998.

[Gruber, 93] T. R. Gruber, "Toward Principles for the design of Ontologies used for Knowledge Sharing," in Proc of International Workshop on Formal Ontology, Padova, Italy, March 1993.

[Grust, 02] T. Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA*. In M. J. Franklin, B. Moon, and A. Ailamaki, editors, ACM Press, 2002.

[Guarino, 97] Nicola Guarino: Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. SCIE 1997: 139-170.

[Guarino et al, 99] Guarino, N., C. Masolo, and G. Vetere, OntoSeek: Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs, . 1999, National Research Council, LADSEBCNR: Padavo, Italy.

[Guarino et al, 00] Nicola Guarino, Christopher A. Welty: Ontological Analysis of Taxonomic Relationships. ER 2000: 210-224.

[Guarino et al, 02] Nicola Guarino, Christopher A. Welty: Evaluating ontological decisions with OntoClean. Commun. ACM 45(2): 61-65 (2002)

[Gutierrez et al, 00] A. Gutierrez, R. Motz, and D. Viera. Building databases with information extracted from web documents. In *Proceedings XX international conference of the Chilean computer sciences society*, pages 41–49, 2000.

[Hatano et al, 02] K. Hatano, H. Kinutani, M. Yoshikawa, and S. Uemura. Information Retrieval System for XML Documents. 2002.

[Haussler, 99] D. Haussler: Convolution Kernels on Discrete Structures, Technical Report UCSC-CRL-99-10, University of California, USA, 1999.

[He et al, 03] B. He and K. C.-C. Chang, "Statistical schema matching across web query interfaces". In *SIGMOD Conference*, 2003.

[Houtsma et al, 95] M.A.W. Houtsma and A.N.Swami,"Set-oriented Mining for Associtions rules in Relational Databases", in proceedings of the 11th International Conference on Data Engineering, pages 25-33, IEEE Computer Society, 1995.

[INEX] Initiative for the Evaluation of XML Retrieval: http://inex.is.informatik.uni-duisburg.de

[Ingwersen, 92] P. Ingwersen. Information retrieval interaction. London, Taylor Graham, 1992.

[Izabel et al, 04] Maria Izabel, M. Azevedo, Lucas Pantuza Amorim, Nivio Ziviani, A Universal Model for XML Information Retrieval, In Pre-proceedings of INEX 2004, Dagstuhl, Allemagne, pages158-162.

[Jaakkola et al, 99] S. Jaakkola, M. Diekhans and D. Haussler: Using the Fisher Kernel Method to Detect Remote Protein Homologies, in Intelligent Systems for Molecular Biology Conference, 1999.

[Jacquemin et al, 02] Jacquemin, C., Daille, B., Royanté, J., and Polanco, X. 2002. In vitro evaluation of a program for machine-aided indexing. *Inf. Process. Manage.* 38, 6 (Nov. 2002), 765-792.

[Jarvelin et al, 02] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4) :422–446, 2002.

[Jelinek, 82] F. Jelinek: Continuous Speech Recognition by Statistical Methods, in proceedings of the IEEE, Volume 64, pages 532-536, 1976.

[Jensen, 96] F. V. Jensen: An Introduction to Bayesian Netwroks, UCL Press, 1996.

[Jiang et al, 97] Jiang J.and Conrath D. "Semantic similarity based on corpus statistics and lexical taxonomy".In *Proceedings on International Conference on Research in Computational Linguistics*, Taiwan, 1997.

[Joachims, 98] T. Joachims: Text Categorization with Support Vector Machines: Learning with many Relevant Features, in proceedings of ECML-98, 1998.

[Kakade et al, 05] V. Kakade and P. Raghavan. Encoding XML in vector spaces. In *Proceedings of ECIR 2005, Saint Jacques de COmpostelle, Espagne*, 2005.

[Kamps et al, 04] J. Kamps, M. de Rijke, and B. Sigurbjornsson. Length normalization in XML retrieval. In *Proceedings of SIGIR 2004, Sheffield, England*, pages 80–87, 2004.

[Katz, 96] S. Katz: Distribution of Content Words and Phrases, in Text and Language Modelling Natural Language Engineering, pages 15-60, 1996.

[Kazai et al, 02] G. Kazai, M. Lalmas, and T. Roelleke. Focused document retrieval, In *9th International Symposium on string processing and information retrieval, Lisbon, Portugal*, September 2002.

[Kazai, 04] Kazai, G. (2004): Report of the INEX 2003 metrics working group. In Proceedings of the 2nd Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany, 184-190, ERCIM Publications.

[Kazai et al, 04a] G. Kazai, M. Lalmas, and A. P. de Vries. The overlap problem in contentoriented XML retrieval evaluation. In *Proceedings of SIGIR 2004, Shef-field, England*, pages 72–79, July 2004.

[Kazai et al, 04b] G. Kazai, M. Lalmas, and A. de Vries. Reliability tests for the XCG and inex-2002 metrics. In *Pre-Proceedings of INEX 2004*, pages 33–39, december 2004.

[Kazai et al, 05] Gabriella Kazai and Mounia Lalmas, INEX 2005 Evaluation Metrics, In Proceedings of INEX 2005 Workshop, pages 401-406, November 28-30, 2005

[Khan, 00] Latifur R. Khan, *Ontology-based Information Selection, Phd Thesis,* Faculty of the Graduate School, University of Southern California. August 2000.

[Kimpeläinen, 92] P. Kimpeläinen, "Tree Matching Problems with application to Structured Text Databases", PhD thesis, Department of Computer Science, University of Helsinki. Rapport A-1992-6.

[Knight et al, 94] Kevin Knight and S. Luk. Building a large-scale knowledge base for machine translation. In Proceedings of *AAAI'94*.

[krishnamurthi et al, 00] S.krishnamurthi, K.Gray and P.Grauke, transfomations by example for XML, in proceeding of PADL'00, 2000.

[Kwok, 89] K.L. Kwok, A neural network for probabilistic information retrieval. 12th International ACM SIGIR Conference on Research and Developpement in Information Retrieval, pp 21-30, 1989.

[Lalmas, 97] M. Lalmas. Dempster-shafer's theory of evidence applied to structured documents : modeling uncertainty. In *Proceedings of SIGIR'97, Philadelphia, USA*, pages 110–118, 1997.

[Lari et al, 90], Lari K., Young S. J.,"The Estimation of Stochastic Context-free Grammars using the Inside-Outside Algorithm", *Computer Speech and Language*, vol. 4, p. 35-56, 1990.

[Larson, 04] Ray R. Larson, Cheshire II at INEX 04: Fusion and Feedback for the Adhoc and Heterogeneous Tracks, In Pre-proceedings of INEX 2004, Dagstuhl, Allemagne, 163-170.

[Lassila et al, 99] O. Lassila and R. R. Swick. Resource Description Framework (RDF) model and syntax specification. Technical report, World Wide Web Consortium (W3C),W3C Recommendation, February 1999.

[Leacock et al, 98] Leacock, C., Miller, G. A., and Chodorow, M. "Using corpus statistics and WordNet relations for sense identification". Comput. Linguist.24, 1(Mar.98), 147-165.

[Lenat, 95] D.B. Lenat, CYC : A Large Scale Investment in Knowledge Infrastructure, Communications of the ACM, Vol. 38, N.11, pp 33-38, 1995.

[Levinson et al, 83] S. E. Levinson, L. R. Rabiner and M. M. Sondhi :An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition, in Bell System Tech. J., Volume 62(4), pages 1035-1047, 1983.

[Lewis, 98] D. D. Lewis. Naive (bayes) at forty : The independence assumption in information retrieval. In *European Conference on Machine Learning (ECML'98)*, 1998.

[Lin, 98] D. Lin. "An information-theoretic definition of similarity". In Proceedings of 15th International Conference on Machine Learning, 1998.

[List et al, 03] J. A. List, V. Mihajlovic, A. Vries, G. Ramirez, and D. Hiemstra. The TIJAH XML-IR system at Inex 2003. In *Proceedings of INEX 2003, Dagstuhl, Germany*, 2003.

[Liu et al, 04] Liu, S., Liu, F., Yu, C., and Meng, W. 2004. An effective approach to document retrieval via utilizing WordNet and recognizing phrases. In Proceedings of the 27th Annual international Conference on Research and Development in information Retrieval (Sheffield, United Kingdom, July 25 - 29, 2004). SIGIR '04. ACM Press, New York, NY, 266-272.

[Lodhi et al, 02] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Crestianini and C. Watkins, Text Classification using String Kernels, In Journal of Machine Learning Research, Volume 2, pages 419-444, 2002.

[Luhn, 58] Luhn, H. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, *2*(2), 159–165.

[Manning et al, 99] Manning CD and Schuetze H (1999) Foundations of Statistical Natural Language Processing. The MIT Press.

[Maron et al, 60] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery*, 7 :216–244, 1960.

[Marx et al, 02] M. Marx, J. Kamps, and M. de Rijke. The university of Amsterdam at INEX 2002. In *INEX 2002 Workshop Proceedings, Dagstuhl, Germany*, pages 23–28, 2002.

[Mass et al, 02] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. Maarek, and A. Soffer. JuruXML- an XML retrieval system at INEX'02. In *Proceedings of INEX 2002, Dagstuhl, Germany*, pages 73–80, 2002.

[Mass et al, 03] Y. Mass and M. Mandelbrod. Retrieving the most relevant XML components. In *Proceedings of INEX 2003, Dagstuhl, Germany*, 2003.

[Melnik et al, 90] O. Melnik, S. Levy and J. B. Pollack: RAAM for Infinite Context-Free Languages, 1990.

[Mihalcea et al, 00] Mihalcea, R. and Moldovan, D.: Semantic indexing using WordNet senses. In Proceedings of ACL Workshop on IR & NLP, Hong Kong, October 2000. http://www.seas.smu.edu/~rada/papers/acl00.nlp_ir.ps.gz

[Miller, 95] A.G. Miller, WordNet, A lexical Database for English, ACM 38 (11), 39-41, 1995.

[Miller et al, 99] D. R. H. Miller, T. Leek, and R. M. Schwartz. BBN at TREC7 : Using Hidden Markov Models for Information Retrieval. In *SIGIR*, 1999.

[Mizzaro, 97] S. Mizzaro. Relevance, the whole (hi) story. *Journal of the American society for information science*, 48(9) :810–832, 1997.

[Moldovan et al, 99] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. 1999. LASSO: A tool for surfing the answer net. In Proceedings of the Text Retrieval Conference (TREC-8).

[Murphy et al, 01] K. Murphy and M. Paskin, Linear Time Inference in Hierarchical HHMs, in proceedings of Neural Information Processing Systems, 2001.

[Ng et al, 01] A. Y. Ng and M. I. Jordan: On Discriminative vs. Generative Classifiers: a Comparison of Logistic Regression and Naïve Bayes, in Advances in Neural Information Processing Systems 14, 2001.

[Nie et al, 99] Fuji Ren, Lixin Fan, Jian-Yun Nie, SAAK Approach: How to Acquire Knowledge in an Actual Application System, IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, 1999, pp.136-140.

[Niles et al, 03] Niles, I., and Pease, A., (2003). Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology, Proceedings of the IEEE International Conference on Information and Knowledge Engineering, pp 412-416.

[Noy et al, 00] F.Noy and M.A.Musen, PROMPT : Algorithm and Tool For Automated Ontology Merging And Alignment. In Proceedings of the (AAAI/IAAI 2000), pages 450-455.

[Ogilvie et al, 03] P. Ogilvie and J. Callan. Using language models for flat text queries in XML retrieval. In *Proceedings of INEX 2003 Workshop, Dagstuhl, Germany*, pages 12–18, December 2003.

[Papakonstantinou et al, 00]. Papakonstantinou Y., Vianu V., « DTD Inference for Views of XML Data », *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Dallas, Texas, p. 35-46, May, 2000.

[Patwardhan et al, 03]. S. Patwardhan,S. Banerjee, and T. Pedersen : Using measures of semantic relatedness for word sense disambiguation. In Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics CICLING, Mexico City, 2003.

[Pearl, 88] J. Pearl. *Probabilistic reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[Pietriga et al, 01] E.Pietriga, J-Y.Vion-Dury and V.Quint, Vxt: a visual approach to XML transformations, in proceeding of ACM, 2001.

[Piwowarski et al, 02] B. Piwowarski, G.-E. Faure, and P. Gallinari. Bayesian networks and INEX. In *Proceedings in the First Annual Workshop for the Evaluation of XML Retrieval (INEX)*, December 2002.

[Piwowarski, 03a] B. Piwowarski. *Techniques d'apprentissage pour le traitement d'information structurées : application à la recherche d'information*. PhD thesis, Paris : Université Paris 6, 2003.

[Piwowarski, 03b] B. Piwowarski. Working group report: the assessment tool. In *Proceedings of INEX 2003, Dagstuhl, Germany*, pages 181–183, December 2003.

[Piwowarski et al, 04] B. Piwowarski and M. Lalmas. Interface pour l'évaluation de syst`emes de recherche sur des documents XML. In *Actes de CORIA 2004, Toulouse, France*, pages 109–121, 2004.

[Piwowarski et al, 05] B. Piwowarski, P. Gallinari, and G. Dupret. An extension of precision-recall with user modeling (PRUM): Application to structured information retrieval, ACM Transaction On Information System (TOIS), volume 25, 2007.

[Pollack, 90] J. B. Pollack : Recursive Distributed Representations, in Artificial Intelligence, volume 46(2), 1990.

[Plotkin, 70] G. Plotkin, "A note on inductive generation", Machine Intelligence, 5:153-163, (1970).

[Ponte et al, 98] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM conference on research anddevelopment in information retrieval (SIGIR 98)*, 1998.

[Porter, 80] M. F. Porter. An algorithm for suffix stripping. Program 14, 1980.

[Quek, 97] C. Y. Quek Classification of World Wide Web Documents, Senior Honor Thesis, School of Computer Science, CMU, 1997.

[Rabiner, 89] L. R. Rabiner A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition In proceedings of the IEEE, volume 77(2), pages 257-285, 1989.

[Rada et al, 89] Rada, R., Mili, H., Bicknell, E., and Blettner, M. "Development and application of a metric on semantic nets". IEEE Transaction on Systems, Man, and Cybernetics, 19(1):17–30.

[Raghavan et al, 89] V. V. Raghavan, S. J. Gwang, and peter Bollmann. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3) :205–229, july 1989.

[Rahm et al, 01] Erhard Rahm and Philip A. Bernstein, A survey of approaches to automatic Schema Matching, VLDB Journal 20 (4) (2001) 334-350.

[Raina et al, 03] R. Raina, Y. Shen, A. Y. Ng and A. McCallum :Classification with Hybrid Generative/Discriminative Models, in IPS2003.

[Resnik, 99] Resnik, P., "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language", Journal of Artificial Intelligence Research (JAIR), 11, pp. 95-130, 1999.

[Reynaud et al, 04] C.Reynaud, G.Giraldo, « Vers l'automatisation de la construction de systèmes de médiation pour le commerce éléctronique », le projet PICSEL.

[Ribeiro et al, 96] B. A. Ribeiro-Neto and R. Muntz. A belief network model for IR. In *Proceedings Of the 19th annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Suisse*, pages 253–260, 1996.

[Rijsbergen, 79] Van Rijsbergen, C. J. Information retrieval. London: Butterworth, (1975).

[Robertson, 77] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4) :294–304, 1977.

[Robertson et al, 94a] S. Robertson, S. Walker, S. Jones, and M. H.-B. and M.Gatford. Okapi at TREC 3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126, 1994.

[Robertson et al, 94b] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR 1994*, pages 232–241, 1994.

[Robertson et al, 97] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In Proceedings of the 20th annual international ACM SIGIR

conference on Research and development in information retrieval, pages 16–24. ACM Press, 1997.

[Roelleke et al, 02] T. Roelleke, M. Lalmas, G. Kazai, J. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. In *Proceedings of ECIR* 2002.

[Rosenblatt, 58] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386-408, 1958.

[Rumelhart et al, 86] D. E. Rumelhart, G. E. Hinton, and R. Williams. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing*, 1, 1986.

[Salton, 70] G. Salton, *The SMART retrieval system : Experiments in automatic document processing*. Prentice Hall, 1970.

[Salton, 75] G. Salton, A. Wong, and A. C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18 :229-237, 1975.

[Salton et al, 83] Salton, G., & McGill, M.. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.

[Sauvagnat, 05] Karen Sauvagnat, Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés, Doctorat d'informatique de l'Université Paul Sabatier, 2005.

[Sauvagnat et al, 05] Karen Sauvagnat, Lobna Hlaoua, and Mohand Boughanem, "XFIRM at INEX 2005: ad-hoc, heterogeneous and relevance feedback tracks. Preliminary work", In Proceedings of INEX 2005 Workshop, pages 72-83, November 28-30, 2005.

[Savasere et al, 95] A. Savasere, E.Omiecinski and S.B.Navathe, "An efficient Algorithm for Mining Association Rules in Large Databases", In the VLDB Journal, pages 432-444, 1995.

[Schenkel et al, 05] Ralf Schenkel, Gerhard Weikum et Jens Graupmann, The SphereSearch engine for unified ranked retrieval of heterogeneous XML and Web documents, Proceedings of the 31th VLDB conference, Trondheim, Norway, 2005.

[Schlieder et al, 02] T. Schileder and H. Meuss. Querying and ranking XML documents. *Journal of the American Society for Information Science and Technology*, 53(6) :489–503, 2002.

[Shafer, 76] G. Shafer. *A mathematical theory of evidence*. Princeton, NJ : Princeton University Press, 1976.

[Shaw et al, 97] W. Shaw, R. Burgin, and P. Howell. Performance standards and evaluations in IR test collections : Cluster-based retrieval models. *Information Processing and Management*, 33(1) :1–14, 1997.

[Shawe et al, 04] J. Shawe-Taylor and N. Cristianini: Kernel Methods for Pattern Analysis Cambridge University Press, 2004.

[SIGIR] Special Interest Group of Information Retrieval : http://www.acm.org/sigs/sigir/

[Sigurbjörnsson et al, 03] B. Sigurbj¨ornsson, J. Kamps, and M. de Rijke. An element-based approach to XML retrieval. In *Proceedings of INEX 2003 workshop, Dagstuhl, Germany*, december 2003.

[Singhal et al., 95] Amit Singhal, Chris Buckley, Mandar Mitra: New Retrieval Approaches Using SMART: TREC 4. TREC 1995.

[Skounakis et al, 03] M. Skounakis, M. Craven and S. Ray: Hierarchical hidden Markov models for information extraction, in proceedings of the 18th International Joint Conference on Artificial Intelligence, 2003.

[Smeaton et al, 95] A.F. Smeaton & I. Quigley (1996). Experiments on Using Semantic Distances Between Words in Image Caption Retrieval, in Proceedings of ACM SIGIR Conference, 19: 174-180.

[Song et al, 99] Fei Song, W. Bruce Croft: A General Language Model for Information Retrieval. CIKM 1999: 316-321.

[Sperduti , 93] A. Sperduti :Labelling RAAM, Technical Report TR-93-029, International Computer Science Institute, USA, 1993.

[Staab et al, 01] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, York Sure: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1): 26-34 (2001).

[Su et al, 01] Hong Su, Harumi Kuno, Elke A. Rundensteiner, Automating the Transformation of XML Documents, Proceedings of WIDM'01, pages 68-75, Atlanta 2001.

[Szalik et al, 04] Z. Szalik and T. Roelleke. Building and experimenting with a heterogeneous
collection. In *Pre-proceedings of INEX 2004, Dagstuhl, Allemagne*, pages 24–32, 2004.

[Tang et al, 01] X.Tang ann F.Tompa, specifying transformations for structured documents, In proceeding of WebDB, 2001.

[Termier, 04] A. Termier, « Extraction d'arbres fréquents dans un corpus hétérogènes de données semi-structurées : Application à la fouille de documents XML », Thèse de doctorat de l'université PARIS-SUD, Avril 2004.

[Termier et al, 02] A. Termier, M.C. Rousset and M. Sebag : « TreeFinder :a First Step towards XML Data Mining». ICDM'02, Maebashi, Japan.

[Theobald et al, 02] A. Theobald and G. Weikum. The index-based XXL search engine for querying XML data with relevance ranking. In EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, pages 477–495, 2002.

[Theobald et al, 05] A.Theobald, R.Schenkel and G.Weikum, "Semantic Similarity Search on Semistructured Data with the XXL Search Engine", Information Retrieval, 8, 521–545, 2005.

[Toivonen, 96] Hannu Toivonen, "Sampling Large Databases for Association Rules", In Proceedings of International Conference on Very Large Data Bases, pages 134-145, Morgan Kofman.

[TREC] http://www.trec.nist.gov

[tree-tagger] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[Trotman, 05] A. Trotman. Choosing document structure weights. *Information Processing and Management*, 41(2) :243–264, March 2005.

[Trotman et al, 04a] A. Trotman and B. Sigurbjörnsson. Narrowed extended XPath I (NEXI). In *INEX 2003 proceedings, Dagstuhl, Allemagne*, pages 219–237, December 2004.

[Trotman et al, 04b] A. Trotman and B. Sigurbj¨ornsson. NEXI, now and next. In *INEX 2003 proceedings, Dagstuhl, Allemagne*, pages 10–15, December 2004

[Tsuda et al, 04] K. Tsuda, S. Akaho, M. Kawanabe and K. R. Muller, "Asymptotic Properties of the Fisher Kernel", in proceedings of Neural Computation, 2004.

[Turtle, 91] H. Turtle. *Inference Networks for Document Retrieval*. PhD thesis, University of Massachussetts, Amhesrt, 1991.

[Turtle et al, 90] H. Turtle and W. Croft. Inference networks for document retrieval. In *Proceedings of ACM SIGIR 90*, pages 1–24, 1990.

[Uschold, 98] M.Uschold, "Knowledge Level Modelling: Concepts and Terminology", the knowledge engineering review, 13 (1), 5-29, 1998.

[Vapnik, 95] N. V. Vapnik The Nature of Statistical Learning Theory Springer-Verlag, 1995

[Vernet, 02] A. Vernet 2002, XML transformation langages, http://www.scdi.org/~avernet/misc/xml-transformation

[Viterbi, 67] A.Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm- IEEE Transactions on Information Theory, 1967.

[Vittaut et al, 04] J.-N. Vittaut, B. Piwowarski, and P. Gallinari. An algebra for structured queries in bayesian networks. In *INEX 2004 Pre-proceedings, Dagstuhl, Allemagne*, pages 58–65, 2004.

[W3C, 98a] W3C. EXtensible Markup Language (XML) 1.0. Technical report, World Wide Web Consortium (W3C), Technical report, february 1998.

[W3C, 98b] W3C. DOM Level 1 (Document Object Model). Technical report, World Wide Web Consortium (W3C), W3C standard, october 1998.

[W3C, 01a] W3C. XML Schema. Technical report, World Wide Web Consortium (W3C),W3C Recommendation, 2001.

[W3C, 01b] eXtensible Stylesheet Language (XSL), version 1.0. Technical report, World Wide Web Consortium (W3C),W3C Recommendation, October 2001.

[Walker et al, 97] S. Walker, S. Robertson, M. Boughanem, G. Jones, and K. S. Jones. Okapi at TREC-6 automatic and ad hoc, VLC, routing, filtering and QSDR. In *Proceedings of TREC-6*, pages 125–136, 1997.

[Watkins, 02] C. Watkins Dynamic Alignment Kernels In Advances in Large Margin Classifiers, pages 39-50, 2002.

[Weigel et al, 04] F. Weigel, K. U. Shulz, and H. Meuss. Ranked retrieval of structured documents with the STerm vector space model. In *Pre-Proceedings of INEX 2004, Dagstuhl, Allemagne*, pages 126–133, 2004.

[Widrow et al, 60] B. Widrow and M. Hoff. Adaptive switching circuits. *IRE WESCON Convention Record*, 4 :96{104, 1960.

[Wolff et al, 00] J. Wolff, H. Flörke, and A. Cremers. Searching and browsing collections of structural information. In *Proceedings of IEEE advances in digital libraries, Washington, 2000*, pages 141–150, 2000.

[Xie et al, 03] L. Xie, S. Chang, A. Divakaran and H. Sun :Unsupervised Discovery of Multilevel Statistical Video Structures using HHMMs, in IEEE Intl. Conf. Multimedia Expo (ICME), 2003.

[XSLWIZ, 01] XSLWIZ 2001. http://www.induslogic.com/products/xslwiz.html

[Yang et al, 02] Y. Yang, S. Slattery and R. Ghani, "A Study of Approaches to Hypertext Categorization", in Journal of Intelligent Systems, Volume 18(2/3), pages 219-241, 2002.

[Yi et al, 00] J. Yi and N. Sundaresan, A Classifier for Semi-Structured Documents, in Proceedings of the 6th ACM SIGKDD, pages 640-644, 2000.

[Yoo, 02] S. Yoo. An XML retrieval model based on structural proximities. In *INEX 2002 Workshop Proceedings, Dagstuhl, Allemagne*, pages 60–64, 2002.

[Yu et al, 76] C. Yu and G. Salton. Precision-weighting- an effective automaic indexing method. *J. ACM*, 23 :76–88, 1976.

[Zadeh, 65] L. Zadeh. Fuzzy sets. *Information and control*, 8 :338–353, 1965.

[Zaki, 02] M.J.Zaki. "Efficiently Mining Frequent Trees in a Forest", In proceedings of 8th ACM SIGDD International Conference On Knowledge Discovery and Data Mining, 2002.

[Zaki et al, 02] M.J.Zaki and C.J. Hsiao, « CHARM: An efficient Algorithm for closed Itemset Mining", In proccedings of the 2nd SIAM International Conference on Data Mining, Arlington, 2002.

[Zaki et al, 03] M.J.Zaki and K. Gouda, « Fast Vertical Mining Using Diffsets », in proceedings of the 9th International Conference on Knowlzdge Discovery and Data Mining, Washington, DC, 2003.

[Zaragoza, 99] Zaragoza : Modèles dynamiques d'apprentissage numérique pour l'accès à l'information textuelle. PhD Thesis, LIP6, University of Paris 6, France, 1999.

[Zipf, 49] G. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, 1949.