République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'hamed BOUGARA de BOUMERDES

Faculté des sciences

Département Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de Magister

Spécialité:

Informatique

Option:

Spécification de logiciels et traitement de l'information

Par : **MATAOUI M'hamed**

Thème

Reformulation de requêtes dans les systèmes de recherche d'information dans des documents XML

Soutenu le : 29/04/2007, devant le jury composé de:

Mr Mohamed Mezghiche, Professeur à l'Université de Boumerdes	Président
Mr Mohand Boughanem, Professeur à l'université Paul Sabatier de Toulouse	Rapporteur
Mr Amar Balla, Maître de conférences à l'INI	Examinateur
Mr Azzedine Chikh, Maître de conférences à l'Université de Tlemcen	Examinateur

Année universitaire 2006-2007

Résumé

Notre travail se situe dans le contexte de la *recherche d'information (RI)*, plus particulièrement la recherche d'information dans des documents semi structurés de type XML.

La reformulation de requêtes est une phase importante dans les systèmes de recherche d'information. Elle permet en effet de récrire la requête de l'utilisateur selon les informations retrouvées par la requête initiale. De manière générale, ceci consiste, dans le cas notamment de la réinjection de la pertinence, d'extraire à partir des documents jugés pertinents par l'utilisateur, les mots-clés importants puis les rajouter à la requête initiale.

L'objectif de ce projet est de proposer une solution pour adapter ce processus bien connu et bien établi dans les systèmes de recherche d'information plein texte, à la recherche d'information dans des documents XML. L'utilisation de la technique de réinjection de pertinence dans le contexte de la RI structurée nécessite la prise en charge de la dimension structurelle en plus de la dimension textuelle.

Dans ce travail nous avons tenté d'apporter des réponses aux différentes questions posées, à savoir : *Comment effectuer une reformulation de requêtes par réinjection de pertinence dans ce contexte? Comment extraire les meilleurs termes à partir d'unités d'information jugées pertinentes et non pertinentes par l'utilisateur, sachant que ces unités peuvent avoir des sémantiques différentes (ex : un paragraphe, une section, un titre), et peuvent être imbriquées les unes dans les autres? Quels poids doit-on assigner à ces différents termes dans ces différents cas de figures? Est-il opportun, par exemple, d'assigner le même poids à un terme provenant d'un titre et d'une section? Comment intégrer l'information structurelle dans la formation de la nouvelle requête ?*

Nos propositions concernent les catégories de stratégies : le ré-ordonnancement de la liste des résultats; et puis l'expansion de requêtes. Concernant la stratégie de ré-ordonnancement, nous proposons deux méthodes : le *ré-ordonnancement contextuel* et le *ré-ordonnancement par nom de Journal*. En ce qui concerne l'expansion de requêtes, nous proposons deux méthodes : expansion par ajout de termes et expansion par ajout de contraintes structurelles. L'évaluation effectuée porte sur les méthodes de ré-ordonnancement appliquées sur des résultats renvoyés par le système de recherche d'information XFIRM en utilisant des jugements de pertinence issus de la campagne INEX. L'évaluation des formules proposées nous a permis de constater que les résultats obtenus après ré-ordonnancement sont meilleurs que ceux de l'exécution de base.

Mots-clés : *recherche d'information, XML, réinjection de pertinence, reformulation de requête, expansion de requêtes, ré-ordonnancement contextuel.*

Abstract

Our work is in the context of the *information retrieval*, more particularly the information retrieval in semi structured documents of type XML.

The query reformulation is an important phase in the information retrieval systems. It allows rewrite the user query according to information's found by the initial query. In a general way, this consists, in the case in particular of the relevance feedback, to extract from the documents judged by the user as relevant, the important keywords then to add them at the initial query.

The aim of this project is to propose a solution to adapt this well-known and established process in full text information retrieval systems, to the information retrieval in XML documents. In addition to textual dimension, the use of relevance feedback in the context of the structured Information retrieval requires the responsibility assumption of structural dimension.

In this work we tried to bring answers to the various put questions, namely: *How to use query reformulation by relevance feedback in this context? How to extract the best terms starting from units of information considered as relevant and nonrelevant by the user, knowing that these units can have the semantic different ones (ex: a paragraph, a section, a title), and be overlapping the ones in the others? Which weights does one have to assign in these terms in these various cases? Is it convenient, to assign the same weight to term coming from a title and a section? How to integrate structural information in the formation of the new query?*

Our proposals relate to the categories of strategies: the re-ranking of the results list; and the query expansion. Concerning the re-ranking strategy, we propose two methods: *the contextual re-ranking* and the *re-ranking by Journal Name*. With regard to the query expansion, we propose two methods: expansion by addition of terms and expansion by addition of structural constraints. The evaluation carried out relates to the re-ranking methods applied to results returned by the XFIRM information retrieval system by using INEX relevance assessments. The evaluation of the formulas suggested enabled us to note that the results obtained after re-ranking are better than those of the base-line run.

Keywords: *information retrieval, XML, relevance feedback, query reformulation, query expansion, contextual re-rank.*

Dédicaces

Remerciements

J'exprime mes grandes reconnaissances et mes vifs remerciements à mon encadreur le professeur Mohand BOUGHANEM pour le temps et l'attention qu'il a bien voulu consacrer au bon déroulement de ce projet.

Je remercie très vivement le Professeur MEZGHICHE, responsable de l'école doctorale en spécification de logiciels et traitement de l'information, pour les efforts qu'il a bien voulu consacrer à ses étudiants.

Je remercie vivement les membres de jury pour avoir accepter de juger ce modeste travail.

Un grand merci à Mme IGHEZOU pour le temps et l'effort qu'elle a consacré pour moi.

Je remerci également Mme Karen SAUVAGNAT de l'I.R.I.T pour son aide précieuse.

Merci à tous mes amis qui m'ont encouragé et soutenu tout au long de ce travail, en particulier : Tayeb KENAZA, Lhouari ROUANE, Abdelghani BAKHTOUCHI, reda IGHEZOU et Sofiane BOUZNAD.

Enfin, je remercie tous ceux qui de près ou de loin ont bien voulu m'encourager pour que ce travail puisse être achevé.

Table des matières

Liste des figures

Liste des tableaux

Liste des Acronymes

AQE	Automatic Query Expansion
BD	Base de données
CAS	Content And Structure
CLEF	Cross Language Evaluation Form
CML	Chemical Markup Language
CO	Content Only
CO+S	Content Only + Structure
DARPA	Defense Advanced Research Project Agency
DOM	Document Object Model
DTD	Document Type Definition
FLWR	For, Let, Where, Return
GED	Gestion Electronique Documentaire
HTML	HypertText Markup Language
IDF	Inverse Document Frequency
IEF	Inverse Element Frequency
INEX	Initiative for the Evaluation of XML Retrieval
IQE	Interactive Query Expansion
IRIT	Institut de Recherche en Informatique de Toulouse
ITDF	Inverse Tag and Document Frequency
MATHML	Mathematical Markup Language
NEXI	Narrowed Extended XPath I
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OFX	Open Financial eXchange
OQL	Object Query language
PRP	Probability Ranking Principle
RF	Relevance Feedback
RI	Recherche d'Information
RSV	Retrieval Status Value
SFI	Système de Filtrage d'Information
SGBD	Système de Gestion de Bases de Données
SGML	Standard Generalized Markup Language
SMART	Salton's Magical Automatic Retriever of Text
SMIL	Synchronized Multimedia Integration Language
SRI	Système de Recherche d'Information
TF	Term Frequency
TREC	Text Retrieval Conference
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XCG	XML Cumulative Gain
XFIRM	XML Flexible Information Retrieval Model
XML	eXtensible Markup Language
XML-QL	XML Query Language
XQL	XML Query Language

Introduction Générale

L'homme, assoiffé de savoir, persiste à conquérir les champs de l'intrigante connaissance. Au temps de *Denis Diderot*, l'encyclopédie fut l'œuvre sainte rassemblant l'extrait du savoir humain, mais le siècle de lumière, où l'on inscrivit la connaissance sur du papier bouffant, tourna en une ère d'informatisation de toutes les formes du savoir. Les domaines, en perpétuelle expansion, ont engendré un flux informationnel ingérable avec les moyens classiques, ainsi le moment s'avère opportun pour l'emploi de techniques permettant une gestion lucrative de la connaissance humaine. De nos jours, et se conformant aux exigences de la technologie, les documents numériques sont les principaux véhicules de l'information.

En effet, le nombre d'e-mails envoyés chaque année représente par exemple 400 000 tera-octets d'information, et la quantité totale d'information produite en 2002 avoisinerait les 5 exa-octets.

Plus d'aiguilles plus de foin, en effet, plus la quantité d'information augmente, plus il devient très ardu de satisfaire à un besoin particulier. La Recherche d'Information ***(RI)*** s'attribue la tâche de trouver, justement, l'aiguille dans la botte de foin. Il s'agit de la mise en œuvre d'outils automatisés permettant un accès efficace à cette quantité gigantesque d'information numérique. Le processus de recherche d'information est le processus qui permet de mettre en relation l'ensemble des informations disponibles d'une part et les besoins de l'utilisateur d'une autre part.

Contexte du travail

Notre travail se situe dans le contexte de la *recherche d'information*, plus particulièrement la recherche d'information dans des documents semi structurés de type XML. Actuellement, la recherche d'information a évolué de l'accès à un document ou un ensemble de documents vers l'accès à des informations répondant à un intérêt particulier de l'utilisateur.

L'environnement le plus populaire aujourd'hui en recherche d'information est l'Internet. Depuis leur création, les documents semi-structurés ont présenté un intérêt particulier dans le domaine de gestion électronique des documents. On assiste aujourd'hui à un format standard et universel d'échange de données connu sous le nom de XML (*eXtensible Markup Language*) qui est une recommandation (février 1998) du W3C (*World Wide Web Consortium*) **[W3C_XML, 1998]**. Ce dernier permet de combiner l'information structurelle avec le contenu.

Une exploitation efficace des documents structurés et semi-structurés disponibles et qui ne cessent d'augmenter de façon exponentielle en terme de nombre et de volume, nécessite la prise en compte de la dimension structurelle. Cette dimension a engendré l'apparition d'une toute nouvelle problématique en RI, qui est la *RI structurée*. La RI dans des documents XML permet à un utilisateur d'exprimer ses besoins par deux types de requêtes : requêtes orientées contenu, et requêtes orientées contenu et structure.

La reformulation de requêtes est une phase importante dans les systèmes de recherche d'information ***(SRI)***. Elle permet en effet de récrire la requête de l'utilisateur selon les informations retrouvées par la requête initiale. De manière générale, ceci consiste, dans le cas notamment de **la réinjection de la pertinence,** d'extraire à partir des documents jugés pertinents par l'utilisateur, les mots clés importants puis les rajouter à la requête.

L'objectif de nos travaux est de proposer une solution pour adapter ce processus bien connu et bien établi dans les systèmes de recherche d'information plein texte, à la recherche d'information dans les documents XML. L'utilisation de la technique de réinjection de pertinence dans le contexte de la RI structurée nécessite la prise en charge de la dimension structurelle en plus de la dimension textuelle.

Problématique

Aujourd'hui, le nombre de documents structurés ou semi-structurés mis à notre disposition est en perpétuelle croissance.

La co-existence de l'information structurelle et de l'information de contenu dans ce type de documents a engendré une nouvelle problématique appelée RI structurée. Dans le cadre de cette nouvelle problématique de nouveaux modèles ont été proposés afin d'exploiter efficacement l'information structurelle.

Compte tenu de la forme des documents XML, l'interrogation des ces documents peut être effectuée en n'utilisant que des mots clés (on parlera des requêtes CO (*Content Only*)) ou des requêtes comportant des mots clés et des contraintes de structures (CAS (*Content And Structure*)), la notion de réinjection de la pertinence doit donc intégrer également à la fois les notions de contenu et de structure.

En RI traditionnelle, l'unité documentaire jugée et donc à partir de laquelle les termes sont extraits, est le document entier. Or, dans le contexte de la recherche d'information structurée, recherche dans les documents XML par exemple, l'unité documentaire peut avoir différentes formes. Elle peut être le document entier ou encore tout élément du document. Un élément est un sous arbre d'un document XML. La première problématique posée dans le cadre de ce mémoire est : comment effectuer une reformulation de requêtes par réinjection de pertinence dans ce contexte? Plus précisément, la question majeure posée est : comment extraire les meilleurs termes à partir d'unités d'information jugées pertinentes et non pertinentes par l'utilisateur, sachant que ces unités peuvent avoir des sémantiques différentes (ex : un paragraphe, une section, un titre), et peuvent être imbriquées les unes dans les autres? La seconde question primordiale dans toute technique de reformulation est: quels poids doit-on assigner à ces différents termes dans ces différents cas de figures? Est-il opportun, par exemple, d'assigner le même poids à un terme provenant d'un titre et d'une section?

En plus de l'information textuelle, les éléments jugés par l'utilisateur contiennent aussi de l'information structurelle qui peut être exploitée dans le processus de reformulation de requêtes. Une fois que nous disposons de cette information, la question qui se pose est alors: Comment intégrer l'information structurelle dans la formation de la nouvelle requête ? Que doit-on faire dans le cas des requêtes orientées contenu, et dans le cas des requêtes orientées contenu et structure?

L'objectif de ce mémoire est de tenter d'apporter des réponses (solutions) aux différentes questions posées ci-dessus.

Contribution

Selon **[Ruthven and Lalmas, 2003]**, l'application de la technique de réinjection de pertinence a deux effets : l'effet d'ordonnancement (*Ranking effect*); et l'effet du feedback (*Feedback*

effect). Le premier effet va simplement faire un ré-ordonnancement de la liste des résultats en mettant (*push*) les documents jugés pertinents à la tête de la liste. Le deuxième effet représente la capacité d'un système à extraire les documents qui n'existaient pas dans la liste initiale.

Nos propositions concernent les catégories de stratégies : le ré-ordonnancement de la liste des résultats; et puis l'expansion de requêtes.

Concernant la stratégie de ré-ordonnancement, nous proposons deux méthodes. La première dite *"ré-ordonnancement contextuel"* a été proposée en partant de l'hypothèse suivante : « *un élément qui appartient à un document contenant beaucoup d'éléments jugés par l'utilisateur comme étant pertinents doit être mieux classé qu'un élément se trouvant dans un document qui contient peu d'éléments jugés pertinents*». Plus précisément, elle consiste à recalculer pour chaque élément retrouvé un nouveau poids basé sur l'importance du document (contexte) auquel il appartient. Ces poids vont augmenter ou bien diminuer l'importance des éléments de la liste renvoyée. La deuxième méthode de ré-ordonnancement proposée se base, d'après les études effectuées par Mihajlovic et al. **[Mihajlovic et al., 2005]** et Ramirez et al. **[Ramirez et al., 2004]**, sur l'hypothèse suivante: « *si un élément est jugé pertinent alors le « Journal » auquel appartient cet élément peut contenir d'autres éléments qui ont un contenu similaire* ». L'idée de base pour appliquer cette proposition est de calculer certains poids liés aux journaux auxquels appartiennent les éléments jugés.

En ce qui concerne l'expansion de requêtes, nous proposons deux stratégies : expansion par ajout de termes et expansion par ajout de contraintes structurelles. Notre approche pour l'expansion de requêtes par ajout de termes entre dans la même lignée que celle proposée par **[Hlaoua and Boughanem, 2005]**. Elle est basée sur la formule de Rocchio, de telle sorte à permettre de pallier le problème d'imbrication des éléments renvoyés pour jugements et aussi d'éliminer si possible l'inclusion des nœuds non pertinents dans le choix des termes à rajouter. Concernant l'expansion structurelle, nous nous sommes tout d'abord poser la question de l'intérêt d'ajouter une contrainte de structure à une requête. Nous avons pour cela effectué une étude statistique sur un échantillon de résultats (ensemble d'éléments) retournés en réponse à des requêtes prises dans le cadre d'INEX[1]. Puis, à l'issue de cette analyse nous avons proposé une technique permettant d'identifier la structure pertinente pouvant être rajoutée à la requête.

Organisation du Mémoire

Ce mémoire est organisé en cinq chapitres :

Le chapitre 1, ***Notions de Base sur la Recherche d'Information***, traite des concepts, approches, techniques et modèles étudiés dans le domaine de la recherche d'information. Une première section est consacrée à la description du processus de la RI (section 1.2.), dans laquelle nous définissons les notions de : indexation (section 1.2.3), appariement (section 1.2.4) et reformulation de requêtes (section 1.2.5). Dans la section (1.3.), nous passons en revue les principaux modèles de RI. La section (1.5.) traite l'aspect évaluation des modèles et systèmes de RI. Enfin, la reformulation de requêtes est détaillée dans la section (1.4.).

Le chapitre 2 présente un état de l'art sur la ***Recherche d'Information Structurée***. La section (2.2.) présente les documents semi-structurés tout en définissant la notion de structure;

[1] INEX : Initiative for the Evaluation of XML Retrieval

La section (2.3.) montre les problématiques spécifiques à la RI structurée, deux classes d'approches sont à distinguer : premièrement les approches orientées BD, et deuxièmement les approches orientées RI. La section (2.4.) présente les différentes techniques d'indexation des documents semi-structurés. La section (2.5.) donne un aperçu sur les langages de requêtes qui prennent en compte l'aspect structurel des documents XML. La section (2.6.) présente les différents modèles de recherche proposés dans la littérature. La section (2.8.) décrit l'aspect évaluation dans un contexte de RI structurée. Enfin, la section (2.7.) présente deux exemples de systèmes de recherche dans les documents XML.

Le chapitre 3, ***RF en RI structurée : état de l'art***, dans lequel, nous allons donner un aperçu sur les principaux travaux de recherche menés pour l'application de la technique de réinjection de pertinence dans le contexte de la RI structurée. En premier lieu, nous donnons la motivation et les problématiques qui ont poussé les chercheurs à investir ce thème. Les approches et méthodes proposées dans la littérature seront détaillées selon chacune des approches : 1) ré-ordonnancement de la liste des résultats ; 2) expansion de requêtes. Quelques travaux fournissent un ensemble de données concernant la collection de tests utilisée dans la campagne INEX, afin d'argumenter le choix de l'information structurelle utilisée, ces données seront expliquées dans la section (3.5) « statistiques ». Enfin, la section (3.6) présente une discussion dans laquelle nous montrons quelques limites des approches proposées pour l'application de la technique de réinjection de pertinence dans le contexte de la RI structurée.

Le chapitre 4, ***Propositions pour l'intégration de la technique de RF en RI structurée***, est divisé en deux grandes parties : la section (4.2) traite nos propositions pour la stratégie de ré-ordonnancement; Nos propositions pour l'expansion de requêtes sont représentées dans la section (4.3).

Nous terminons avec le chapitre 5, ***Expérimentation et résultats***, présente les résultats des expérimentations effectuées pour évaluer nos propositions portant essentiellement sur les méthodes de ré-ordonnancement.

En annexe on trouvera un listing des requêtes de la tâche CO+S de la campagne INEX 2005.

Chapitre 1 :

Notions de Base sur la Recherche d'Information

1.1. Introduction

La Recherche d'Information ***(RI)*** est un domaine de l'informatique qui s'intéresse à l'organisation, au stockage et à la sélection d'informations répondant aux besoins des utilisateurs **[Salton, 1970] [Salton, 1984]**. Ce domaine manipule différents concepts : la requête, le besoin en information, les documents, la pertinence, etc. Deux types de systèmes peuvent donner accès à l'information : La collecte active à travers les systèmes de recherche d'information ***(SRI)*** et la collecte passive à travers les systèmes de filtrage d'information ***(SFI)***. Dans ce chapitre nous nous intéressons aux systèmes de recherche d'information qui sont définis comme étant un ensemble de programmes informatiques ayant pour but de sélectionner dans une collection de documents ceux qui sont susceptibles de répondre à un besoin utilisateur.

Ce chapitre traite des concepts, approches, techniques et modèles étudiés dans le domaine de la recherche d'information. Il est organisé comme suit :

Une première section est consacrée à la description du processus de la RI (section 1.2.), dans laquelle nous définissons les notions de : indexation (section 1.2.3), appariement (section 1.2.4) et reformulation de requêtes (section 1.2.5). Dans la section (section 1.3.) nous passons en revue les principaux modèles de RI. La section (section 1.5.) traite l'aspect évaluation des modèles et systèmes de RI. Etant donné que la reformulation de requêtes représente le cœur du sujet, nous la détaillons dans la section 1.4.

1.2. Le processus de RI

Le processus de recherche d'information est le processus qui permet de mettre en relation l'ensemble des informations disponibles d'une part et les besoins de l'utilisateur d'une autre part. L'expression de ces besoins se fait par le biais de requêtes. Ces requêtes seront envoyées au système de recherche d'information afin qu'il extrait les documents pertinents répondant au besoin de l'utilisateur. Cette notion de pertinence est fortement subjective, car elle dépend de l'utilisateur, donc très difficile à automatiser.

Le processus de recherche d'information comprend plusieurs concepts :

- La collection de documents ;
- Le besoin en information ;
- La fonction d'indexation ;
- La fonction d'appariement requête-document ;
- La fonction de modification de requête qui se traduit généralement par un mécanisme de reformulation de requêtes.

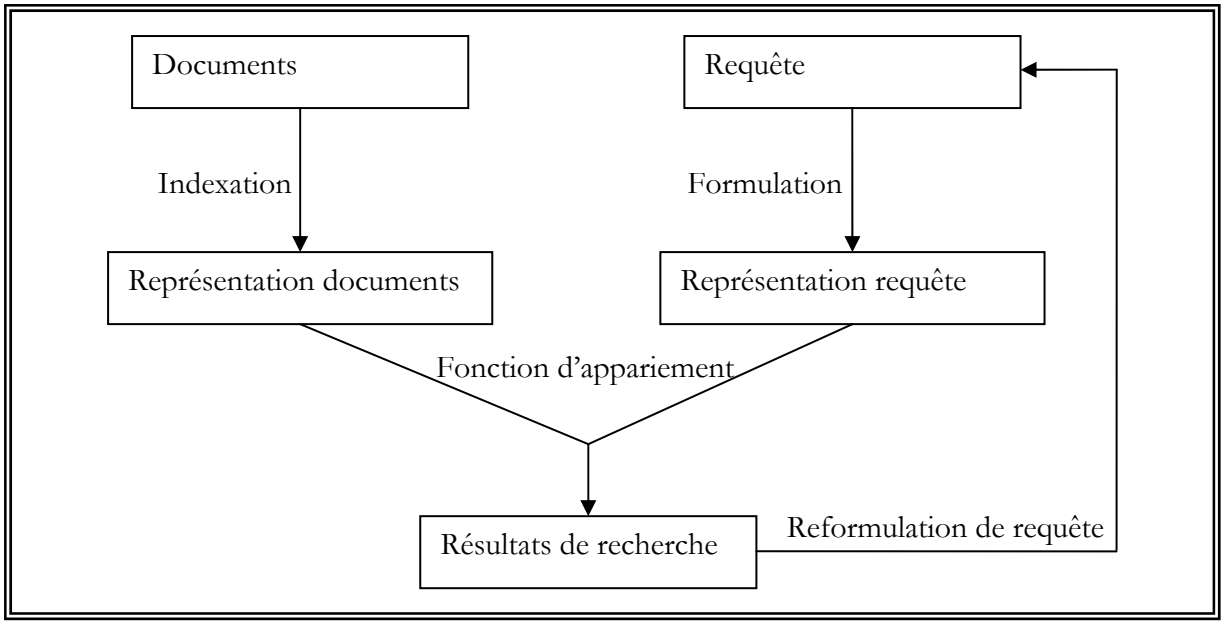


Figure 1.1 : Le processus de recherche d'information **[Rijsbergen, 1979]**

1.2.1. Collection de documents (corpus)

Un corpus de documents est un ensemble de granules documentaires qui peuvent être des documents entiers ou bien des parties de documents. Dans la RI traditionnelle l'unité d'information utilisée et recherchée lors du processus de recherche est le document.

1.2.2. Besoin en information

Ce besoin est l'expression mentale de ce que l'utilisateur recherche. L'expression d'un besoin se fait par une requête qui permet l'interrogation d'un système de recherche d'information. L'interrogation d'un SRI peut prendre plusieurs formes, à savoir : interrogation en langage naturel, interrogation en langage booléen ou interrogation graphique. Selon Kleinberg **[Kleinberg, 1999]** il existe trois différentes formes de requêtes :

- Requêtes spécifiques
- Requêtes larges
- Requêtes par similarité

D'une manière générale, les requêtes sont composées d'un ensemble de mots clés. Ces mots clés peuvent être reliés entre eux par des opérateurs booléens, comme ils peuvent être aussi organisés sous forme d'expressions.

Un autre type de requêtes est celui spécifique à la RI structurée, ce dernier type prend en compte en plus de l'information textuelle des contraintes de structure. Ce type de requêtes sera détaillé dans le chapitre 2 sur la RI structurée.

1.2.3. La fonction d'indexation

L'indexation est le processus permettant de créer une représentation des documents et des requêtes facilement manipulable par un système de recherche d'information. Elle consiste à analyser les documents afin d'extraire un ensemble de mots clés servant comme descripteurs des documents. Il existe trois types d'indexation :

a) Indexation manuelle. L'extraction et le choix des descripteurs s'effectuent par un documentaliste ou un spécialiste du domaine.

b) Indexation automatique. L'extraction et le choix des descripteurs s'effectuent d'une façon totalement automatisée.

c) Indexation semi-automatique. L'extraction des descripteurs s'effectue par le système et le choix des descripteurs est laissé au spécialiste.

Une étude comparative entre l'indexation automatique et l'indexation manuelle a été réalisée par Anderson et Perez-Carballo **[Anderson and Pérez-Carballo, 2001]**. Le résultat de l'étude montre que les avantages et les inconvénients de chacune des deux méthodes d'indexation ont une tendance à s'équilibrer. Autrement dit, le choix de l'une ou de l'autre est en fonction du domaine, de la collection et de l'application considérés.

Dans les sections suivantes nous allons décrire en détail les différentes étapes de l'indexation automatique : de l'extraction des mots à partir des documents jusqu'à la création de l'index.

1.2.3.1. L'analyse lexicale

L'étape de l'analyse lexicale permet d'extraire l'ensemble des termes appartenant à un document. Cette extraction est effectuée en tenant compte des espaces de séparation entre mots, des chiffres et des ponctuations. Un terme peut être un mot simple (pomme) ou un mot composé (pomme de terre) mais en RI on utilise souvent les mots simples.

1.2.3.2. L'élimination des mots vides

Les mots vides sont des mots peu significatifs et porteurs de peu de sens, augmentant ainsi la taille de l'index et rendant la recherche plus lente. De ce fait, leur élimination est impérative.

L'élimination des mots vides est importante dans la mesure où il représente un facteur qui a une grande influence sur la précision de la recherche. En effet, le fait de ne pas éliminer les mots vides provoque inévitablement du bruit (les documents non pertinents à la requête). Si une requête comporte le terme « de » alors tous les documents du corpus correspondent ! On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste de mots vides
- L'élimination des mots dépassant un certain nombre d'occurrences dans la collection.

1.2.3.3. La lemmatisation

La lemmatisation est l'opération qui consiste à réduire les formes fléchies des mots à leur racine grammaticale. Car un mot donné peut avoir plusieurs formes dans un texte dont le sens est presque similaire. Plusieurs méthodes de lemmatisation ont été proposées dans la littérature, parmi lesquelles **[Frakes, 1992]** : la troncature, la méthode des n-grammes **[Adamson and Boreham, 1974]**, les dictionnaires ou l'élimination des affixes (exemple : algorithme de Porter **[Porter, 1980]**).

L'étape de lemmatisation permet d'éviter à l'utilisateur de faire introduire les différentes formes d'un mot (exemple : pluriel, singulier,…) lors d'une recherche. La lemmatisation permet ainsi d'augmenter le rappel, ce qui diminue en pratique le taux de précision. Ceci est dû à la perte de la sémantique originale du mot lors du passage à la forme finale (canonique ou fléchie).

1.2.3.4. La pondération des termes

Cette étape permet de mesurer l'importance d'un terme dans un document. Elle a pour but de trouver les meilleurs termes représentant le contenu d'un document. L'importance d'un terme est généralement mesurée par des méthodes statistiques (et quelques fois linguistiques).

La plupart des formules de pondération proposées dans la littérature de RI se basent sur deux facteurs, à savoir : la pondération locale et la pondération globale. La première quantifie la représentativité locale d'un terme dans le document *(*on parle de *tf : Term Frequency)*, et la deuxième quantifie la représentativité du terme vis-à-vis de la collection des documents *(*on parle de *idf : Inverse of Document Frequency).*

- ***tf (Term Frequency)*** : cette mesure indique l'importance du terme dans le document. Cette importance est proportionnelle à la fréquence du terme. Plusieurs formules de pondération locale ont été proposées, parmi lesquelles : la fonction brute (nombre d'occurrences), la fonction binaire, la fonction logarithmique et la fonction normalisée.

- ***idf (Inverse of Document Frequency)*** : ce facteur mesure l'importance d'un terme dans toute la collection (pondération globale). Il traduit l'impact d'un terme selon son nombre d'apparitions dans la base documentaire. La formule qui exprime l'importance d'un terme dans sa collection peut être vue comme suit : $log(N/df)$, où df représente le nombre de documents contenant le terme et N représente le nombre total de documents de la base documentaire.

La combinaison des deux mesures (tf et idf) donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de tailles homogènes. Les fonctions de pondération sont souvent référencées sous le nom de TF-IDF.

Un autre facteur a été proposé pour pallier aux effets négatifs de la taille des documents sur le facteur tf. Robertson **[Robertson et al., 1994a]** et Singhal et al. **[Singhal et al., 1996]** proposent d'intégrer la taille des documents à la formule de pondération, ce facteur est appelé facteur de normalisation.

1.2.3.5. La création de l'index

L'index, défini comme étant la structure de stockage utilisée pour mémoriser les informations sélectionnées lors du processus d'indexation cette structure permet de sélectionner pour n'importe quel terme tous les documents où il apparaît. Plusieurs solutions de stockage ont

été proposées parmi lesquelles : les fichiers inverses *(inverted files)*, les fichiers de signatures *(signature files)* et les tableaux de suffixes *(suffix arrays)*.

La solution la plus utilisée actuellement est celle des fichiers inverses. Ces fichiers sont composés de deux éléments principaux : Le dictionnaire et le fichier posting. Le vocabulaire consiste en la liste de tous les mots distincts extraits du texte. Pour chaque mot est assigné l'ensemble des positions dans lesquelles ce dernier apparaît.

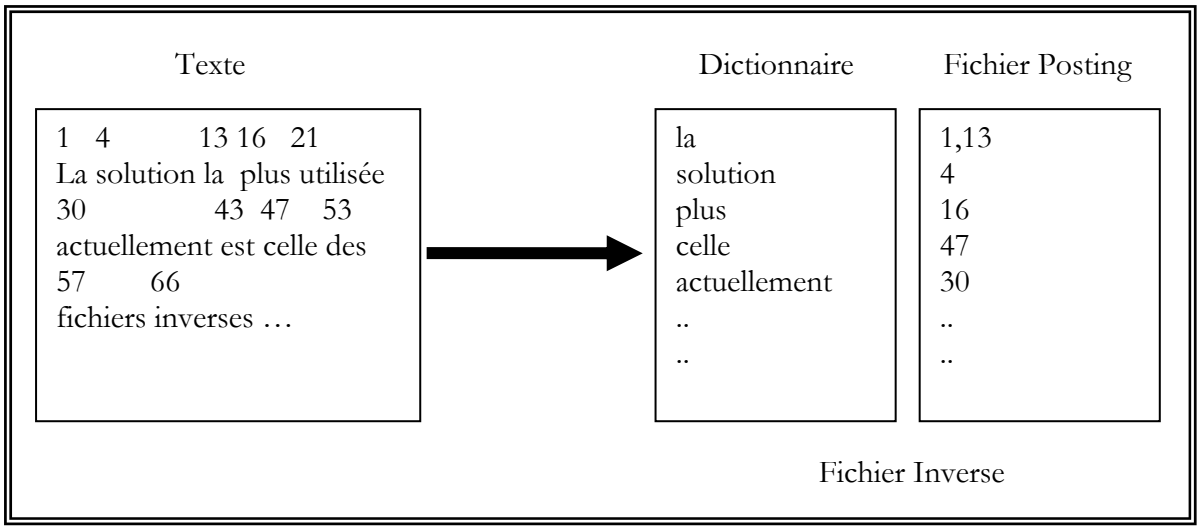


Figure 1.2 : Représentation en un fichier inverse d'un texte

1.2.4. La fonction d'appariement requête-document

Cette fonction est définie afin de mesurer la pertinence d'un document vis-à-vis d'une requête. Elle est vue comme une probabilité ou une similarité vectorielle, notée RSV (Q,d) *(Retrieval Status Value)*, où Q représente la requête et d représente un document. La fonction de similarité permet d'ordonner les documents renvoyés à l'utilisateur par ordre de pertinence.

1.2.5. La fonction de modification de requêtes

L'expression du besoin en information d'un utilisateur sous forme de requête est souvent une chose difficile. Par conséquent, les documents trouvés par la requête initiale peuvent ne pas accomplir le besoin en information de l'utilisateur. C'est pour cette raison que le système de recherche d'information fait appel à la fonction de modification de requêtes afin de corriger le chemin de la recherche. La reformulation de requêtes peut s'effectuer selon deux stratégies : L'extension de la requête avec de nouveaux termes, et la repondération des termes de la requête initiale. La modification de la requête peut être manuelle, automatique ou bien semi-automatique. Cette fonction sera détaillée dans la section 1.4.

1.3. Les modèles de RI

Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence **[Salton et al., 1983]**. On distingue trois principaux modèles :

- Les modèles ensemblistes
- Les modèles algébriques
- Les modèles probabilistes

La figure 1.3 présente une taxonomie des principaux modèles de RI.

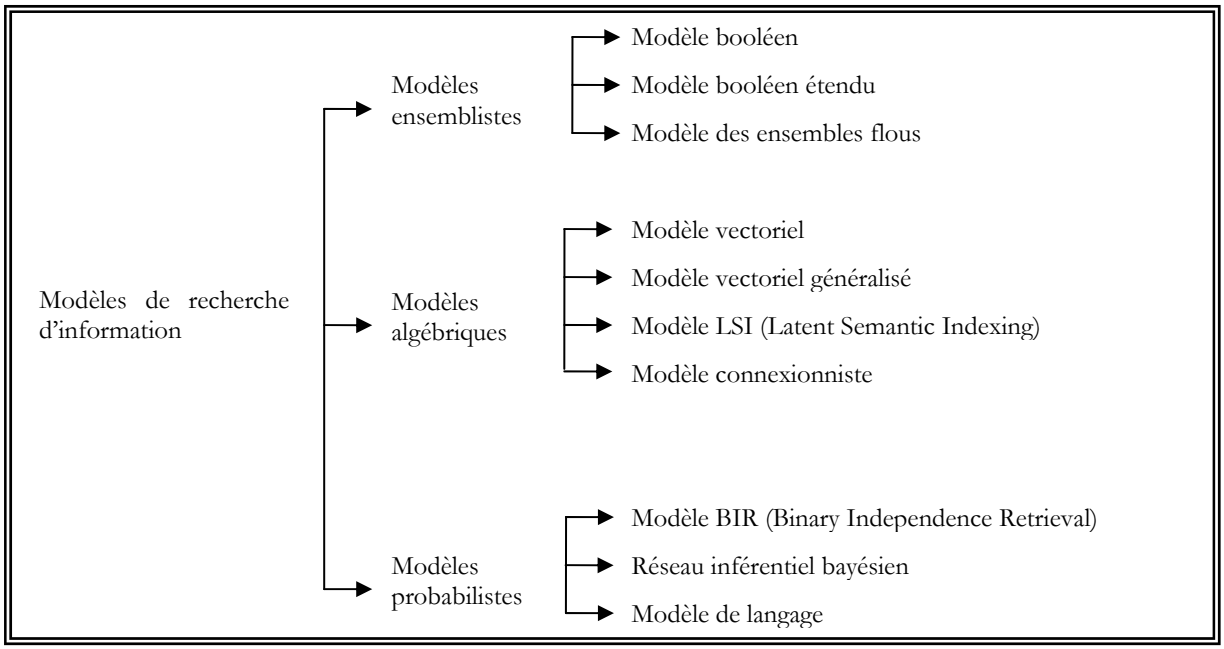


Figure 1.3 : Taxonomie des modèles de recherche d'information

Les modèles ensemblistes reposent sur la théorie des ensembles. Dans ces modèles, les termes de la requête sont séparés par des opérateurs logiques : conjonction (ET), disjonction (OU) et négation (NON). Ces opérateurs permettent d'effectuer des opérations d'union « OU », d'intersection « ET » et de différence « NON » entre les ensembles de résultats associés à chaque terme.

Les modèles algébriques se basent sur la théorie algébrique. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance (ou similarité) dans un espace vectoriel.

Enfin, les modèles probabilistes se basent sur la théorie des probabilités. Pour ces modèles, la pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

Dans les sections suivantes, nous décrivons pour chacun de ces courants le modèle le plus représentatif (à savoir : le modèle booléen, le modèle vectoriel et le modèle probabiliste).

1.3.1. Le modèle booléen

Le modèle booléen est le premier modèle qui s'est imposé dans le monde de la recherche d'information. Il se base sur la manipulation des ensembles et l'algèbre de Boole. Dans ce modèle une requête est une expression logique composée de termes séparés par des opérateurs logiques (ET, OU et NON). Les poids des termes dans l'index sont binaires, c'est-à-dire que les termes

sont présents ou absents du document ($w_{ij} \in \{0,1\}$). Le modèle booléen utilise l'appariement exact, c'est-à-dire qu'il ne permet de restituer que les documents appartenant à l'ensemble décrit par la requête. La similarité entre un document et une requête est définie par :

$$\begin{cases} RSV(q,d) = 1 & si\ d\ appartient\ à\ l'ensemble\ décrit\ par\ la\ requête \\ 0 & sinon \end{cases} \quad \ldots (1.1)$$

Ainsi, un document est considéré dans le modèle booléen comme étant pertinent, ou bien non pertinent. Les résultats de la fonction de similarité ne permettent pas de renvoyer à l'utilisateur une liste ordonnée de documents, ce qui empêche le modèle d'avoir de bonnes performances.

1.3.2. Le modèle vectoriel

Le modèle vectoriel est l'un des modèles qui utilisent l'approche statistique. Il consiste à représenter les documents et les requêtes sous forme de vecteurs de termes pondérés. L'idée de base du modèle vectoriel est d'utiliser une représentation géométrique pour classer les documents par ordre de pertinence par rapport à une requête, c'est-à-dire que les documents et les requêtes sont représentés sous forme de vecteurs dans l'espace vectoriel engendré par les termes extraits de tous les documents de la collection. Cette idée a été développée par Gérard Salton et son équipe **[Salton, 1970]** dans leur projet SMART *(Salton's Magical Automatic Retriever of Text)*. Salton a proposé un modèle basé sur la mesure de similarité par le produit scalaire.

Contrairement au modèle booléen où les termes de la requête doivent être reliés par des connecteurs logiques, le modèle vectoriel permet à l'utilisateur d'exprimer son besoin en information sous forme d'une liste de mots clés ou en langage naturel.

Formellement, dans le modèle vectoriel, la représentation d'un document est vue comme un vecteur $\vec{d}_j = \{w_{1,j}, w_{2,j}, \ldots, w_{t,j}\}$ où $w_{i,j}$ représente le poids des termes dans le document, t étant le nombre total de termes de l'index, et la requête aussi est vue comme un vecteur $\vec{q} = \{w_{1,q}, w_{2,q}, \ldots, w_{t,q}\}$. Une des plus simples mesures de similarité est celle du produit scalaire :

$$RSV(\vec{d}_j, \vec{q}) = \sum_{i=1}^{t} w_{i,j} * w_{i,q} \quad \ldots (1.2)$$

Cette mesure de similarité consiste à mesurer le nombre de termes partagés entre la requête et les documents, car les poids des termes sont binaires.

Plusieurs fonctions de similarité ont été proposées dans la littérature. Nous citons les fonctions les plus répandues : les mesures de Cosinus, Jaccard et Dice.

Mesure du cosinus : $$Sim(D_j, Q_k) = \frac{\sum_{i=1}^{N} (wd_{ij} * wq_{ik})}{\sqrt{\sum_{i=1}^{N} wd_{ij}^2 * \sum_{i=1}^{N} wq_{ik}^2}} \quad \ldots (1.3)$$

Mesure de Jaccard : $$Sim(D_j, Q_k) = \frac{\sum_{i=1}^{N}(wd_{ij} * wq_{ik})}{\sum_{i=1}^{N} wd_{ij}^2 + \sum_{i=1}^{N} wq_{ik}^2 - \sum_{i=1}^{N}(wd_{ij} * wq_{ik})} \quad \dots (1.4)$$

Mesure de Dice : $$Sim(D_j, Q_k) = 2 * \frac{\sum_{i=1}^{N}(wd_{ij} * wq_{ik})}{\sum_{i=1}^{N}(wd_{ij}^2 + wq_{ik}^2)} \quad \dots (1.5)$$

Les avantages du modèle vectoriel sont nombreux : il permet la pondération des termes, ce qui augmente les performances du système; il permet de renvoyer des documents qui répondent approximativement à la requête et effectivement de trier les documents répondant à une requête. Les documents sont en effet restitués dans un ordre décroissant de leur degré de similarité avec la requête. Plus le degré de similarité d'un document est élevé, plus le document ressemble à la requête et plus il est susceptible d'être pertinent pour l'utilisateur.

Théoriquement, le modèle vectoriel présente le principal inconvénient lié à l'indépendance mutuelle des termes d'indexation. Wong et al. **[Wong et al., 1985]** ont proposé un modèle vectoriel généralisé *(Generalized Vector Space Model)* qui lève l'hypothèse d'indépendance des termes.

Aujourd'hui le modèle vectoriel est le plus populaire en recherche d'information, et malgré sa simplicité, il donne de bons résultats par rapport aux autres méthodes d'ordonnancement des résultats.

1.3.3. Le modèle probabiliste

Le premier modèle probabiliste a été proposé par Maron et Kuhns **[Maron and Kuhns, 1960]** au début des années 1960 qui proposent de modéliser le processus de sélection des documents dans un SRI en se basant sur la théorie des probabilités. Le principe de base du modèle probabiliste consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête. Robertson **[Robertson, 1977]** résume ce critère d'ordre par le "principe de classement probabiliste", désigné aussi par PRP *(Probability Ranking Principle)*.

Dans le modèle probabiliste, répondre à une requête revient à spécifier les propriétés d'un certain ensemble de documents appelé : *« ensemble de réponse idéal »*. Cet ensemble contient exactement les documents pertinents et aucun autre. Au moment de la requête, les propriétés de l'ensemble idéal ne sont pas connues, donc il faut d'abord qu'il y ait une première tentative permettant de générer une première description probabiliste de cet ensemble. Ensuite, il faut une interaction avec l'utilisateur pour améliorer cette description probabiliste **[Robertson, 1977]**.

Pour mesurer cette pertinence le modèle probabiliste se base sur la distribution des termes dans un échantillon représentatif de documents d'apprentissage. Les hypothèses posées sont les suivantes :

- la distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents ;

- les variables *« document pertinent »*, *« document non pertinent »* sont indépendantes.

Le processus de recherche se traduit par le calcul du degré (ou probabilité) de pertinence d'un document vis-à-vis à une requête. Deux probabilités conditionnelles sont utilisées dans le processus de décision :

- $P(w_{ij}/Pert)$: probabilité que le terme t_i occurre dans le document d_j sachant que ce dernier est pertinent pour la requête.

- $P(w_{ij}/NonPert)$: probabilité que le terme t_i occurre dans le document d_j sachant que ce dernier n'est pas pertinent pour la requête.

Le modèle probabiliste a été implémenté par Robertson et Walker **[Robertson et al., 1994a] [Robertson et al., 1994b]** dans le système Okapi.

1.3.4. Autres modèles

1.3.4.1. Le modèle booléen étendu

Le modèle booléen étendu, appelé aussi modèle *P_Norm*, a été introduit en 1983 par Salton et al. **[Salton et al., 1983]**. Ce modèle étend le modèle booléen de base afin de supporter l'appariement approché, ceci en assignant des poids aux termes de la requête et des documents et en mesurant un score de pertinence. Le modèle booléen étendu interprète les opérateurs de l'équation de la requête comme des distances entre requêtes et documents.

Considérons un ensemble de termes $\{t_1, \ldots, t_N\}$ et soit wd_{ij} le poids du terme t_i dans le document D_j, où $D_j = (wd_{1j}, \ldots, wd_{Nj})$, avec $1 \le i \le N$ et $0 \le wd_{ij} \le 1$. La similarité entre le document D_j et une requête Q_k décrite sous une forme conjonctive ou disjonctive est donnée comme suit :

Opérateur ***OU*** :
$$Sim(D_j, Q_k) = \left(\frac{\sum_{i=1}^{N} wq_{ik}^p . wd_{ij}^p}{\sum_{i=1}^{N} wq_{ik}^p} \right)^{\frac{1}{p}} \quad \ldots (1.6)$$

Opérateur ***ET*** :
$$Sim(D_j, Q_k) = 1 - \left(\frac{\sum_{i=1}^{N} wq_{ik}^p . (1 - wd_{ij}^p)}{\sum_{i=1}^{N} wq_{ik}^p} \right)^{\frac{1}{p}} \quad \ldots (1.7)$$

Où p /$0 \le p \le 1$ est une constante, et wq_{ik}^p le poids du terme t_i dans la requête Q_k.

Le modèle booléen étendu est un modèle hybride qui inclut les propriétés des modèles ensembliste et algébrique, mais qui n'a pas été beaucoup utilisé par la suite, même s'il donne un cadre nouveau à la recherche d'information.

1.3.4.2. Le modèle vectoriel généralisé

Wong et al. **[Wong et al., 1985]** ont proposé en 1985 le modèle vectoriel généralisé dans le but de représenter les dépendances entre termes de l'index.

Tous les modèles cités précédemment traitent les termes de l'index d'une manière indépendante. Ce modèle a essayé d'établir un cadre formel dans lequel les dépendances entre les termes peuvent être facilement représentées. Le modèle vectoriel généralisé est caractérisé par sa complexité et sa lenteur par rapport au modèle vectoriel classique. On trouve plus de détails sur ce modèle dans **[Wong et al., 1985].**

1.3.4.3. Le modèle de langage

Les modèles de langage sont des modèles probabilistes. Ils désignent une fonction de probabilité qui assigne à chaque séquence de mots une probabilité. Leur objectif est de capter les régularités linguistiques d'une langue, en observant la distribution des mots, successions de mots, dans une langue donnée.

Les modèles de langage sont basés sur l'hypothèse suivante : « *un utilisateur en interaction avec un système de recherche fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver* ». Dans un modèle de langage un document est considéré pertinent lorsque la requête de l'utilisateur ressemble à celle inférée par le document. Le principe est alors : chercher à estimer la probabilité qu'une requête soit inférée par le document **[Boughanem et al., 2004]**. Cette probabilité (notée $P(Qk \mid D)$) sera utilisée pour ordonner la liste des résultats.

La mesure suivante permet d'ordonner les documents :

$$P(T_1, T_2, ..., T_n \mid D) = \prod_{i=1}^{n} ((1 - \lambda_i) P(T_i) + \lambda_1 P(T_i \mid D)) \qquad \dots (1.8)$$

Tels que T_1, T_2, …, T_n sont des variables aléatoires indépendantes représentant les termes de la requête, et λ_i estime l'importance du terme T_i de la requête. $P(T_i \mid D)$ représente la probabilité de pertinence du terme T_i dans le document D et $P(T_i)$ représente la probabilité que ce terme soit non important. Ces deux probabilités sont définies comme suit :

$$P(Ti \mid D) = \frac{tf(Ti \mid D)}{\sum_T tf(T, D)} \text{ , terme important} \qquad \dots (1.9)$$

$$P(Ti) = \frac{df(Ti)}{\sum_T df(T)} \text{, terme sans importance} \qquad \dots (1.10)$$

Où $tf(T_i \mid D)$ est la fréquence du terme T_i dans le document D et $df(T)$ est le nombre de documents dans lesquels T apparaît.

1.4. La reformulation de requêtes

Le processus de la recherche d'information est incertain. Les utilisateurs peuvent avoir des idées moins que bien développées de ce qu'ils recherchent; ils peuvent ne pas pouvoir exprimer un besoin en information en termes de requête appropriée. Les chercheurs ont conclu que bien que les utilisateurs aient souvent la difficulté d'exprimer leurs besoins informationnels avec précision, ils pourraient identifier l'information utile quand elle leurs sera présentée. C'est-à-dire, qu'une fois que le système présente un premier ensemble de documents, ils peuvent facilement différencier entre les documents qui contiennent de l'information utile et ceux qui ne la contiennent pas. C'est ce qu'on appelle communément la réinjection de pertinence.

1.4.1. La réinjection de pertinence

La *réinjection de pertinence* (*relevance feedback*) est l'une des techniques de modification de requêtes les plus utiles dans le domaine de la recherche d'information.

Cette méthode est mise en pratique quand l'utilisateur doit améliorer la requête qu'il a formulée au système de recherche d'information parce que les documents trouvés à l'étape initiale de la recherche ne répondent pas de manière pertinente aux besoins en information de l'utilisateur.

La technique fonctionne comme suit:

- l'utilisateur soumet une requête au SRI, qui produit une liste ordonnée des documents selon leurs degrés correspondants de pertinence à la requête.
- l'utilisateur examine cette liste triée et détermine quels sont les documents pertinents et non pertinents.
- avec cette information, le SRI modifie la requête initiale, en donnant plus d'importance aux termes apparaissant dans les documents pertinents, et affaiblissant la force de ceux qui appartiennent aux non pertinents.
- ce processus est répété jusqu'à ce que l'utilisateur soit complètement satisfait de l'ensemble des documents trouvés.

1.4.2. Les modèles de recherche d'information et la réinjection de pertinence

Nous allons dans cette section détailler les approches de reformulation de requêtes utilisées dans chacun des modèles.

1.4.2.1. Modèle vectoriel

Contrairement au modèle booléen qui permet d'extraire les documents qui satisfont exactement la requête, le modèle vectoriel permet d'extraire tous les documents qui contiennent au moins un des termes de la requête. Le classement de ces documents se fait par le biais des fonctions de mesure de similarité.

La première formalisation de la technique de réinjection de pertinence dans le modèle vectoriel est celle de Rocchio **[Rocchio, 1971]**. Ce dernier a défini le problème par la détermination de la requête optimale. Cette détermination peut être faite par l'ajout de nouveaux

termes ou/et la repondération des termes de la requête initiale. L'équation originale de Rocchio est :

$$Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{n1} R_i - \frac{1}{n_2}\sum_{i=1}^{n2} S_i \qquad \ldots (1.11)$$

Où :

$\boldsymbol{Q_0}$: Représente le vecteur de la requête initiale;
$\boldsymbol{Q_1}$: Représente le vecteur de la nouvelle requête;
$\boldsymbol{n_1}$: Représente le nombre des documents pertinents;
$\boldsymbol{n_2}$: Représente le nombre des documents non pertinents;
$\boldsymbol{R_i}$: Représente le vecteur du i ème document pertinent;
$\boldsymbol{S_i}$: Représente le vecteur du i ème document non pertinent.

La nouvelle requête contient de nouveaux termes (à partir des documents jugés pertinents), et de nouveaux poids pour les termes de la requête initiale. Une variante de cette formule a été implémentée sous le système SMART **[Rocchio, 1971]**. Cette formule a obtenu de bons résultats.

Ide **[Ide, 1971]** a testé plusieurs aspects de la technique de réinjection de pertinence sur le système SMART **[Salton, 1970]**. Parmi ces aspects : l'utilisation des documents pertinents uniquement dans le processus du feedback et la variation du nombre de documents utilisés dans le feedback. Une variante de la formule de Rocchio est celle appelée : « IDE-DEC-HI ». Cette formule n'utilise que le premier document non pertinent trouvé (Formule 1.12). Mais aucune performance n'a été mise en évidence.

$$Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{nr} R_i - S_i \qquad \ldots (1.12)$$

Où :

$\boldsymbol{nr}$: Représente le nombre des documents pertinents;

Une nouvelle variante de la formule de Rocchio a été proposée. Elle permet de donner un degré d'importance (α, β et δ) à chacun des composants utilisés dans le processus de reformulation de requêtes (formule 1.13).

$$Q_1 = \alpha.Q_0 + \frac{\beta}{n_1}\sum_{i=1}^{n1} R_i - \frac{\delta}{n_2}\sum_{i=1}^{n2} S_i \qquad \ldots (1.13)$$

1.4.2.2. Modèle probabiliste

Dans son principe, le modèle probabiliste est dédié à la reformulation de requêtes. Au départ de la recherche, aucune information de pertinence n'est disponible pour estimer qu'un document est pertinent pour un des termes de la requête. La solution de ce problème est d'utiliser une des fonctions de pondération (exemple : tf.idf) pour la recherche initiale. Ensuite, après avoir obtenu la première liste ordonnée de documents, la fonction suivante peut être utilisée.

$$w_{x_i} = \log \frac{P_q(x_i \mid rel) / Pq(\overline{x_i} \mid rel)}{P_q(x_i \mid \overline{rel}) / Pq(\overline{x_i} \mid \overline{rel})} = \log \frac{r/(R-r)}{(n-r)/(N-n-R+r)} \quad \ldots (1.14)$$

$$w_i = \log \frac{r_i/(R-r_i)}{(n_i-r_i)/(N-n_i-R+r_i)} \bullet \left(\frac{r_i}{R} - \frac{n_i - r_i}{N-R} \right) \quad \ldots (1.15)$$

Où :

ri : le nombre de documents pertinents contenants le terme **i**
ni : le nombre de documents contenants le terme **i**
R : le nombre de documents pertinents à la requête
N: le nombre de documents dans la collection

1.4.3. La technique de réinjection de pertinence sans jugements utilisateur

La technique de réinjection de pertinence, décrite précédemment dépend des jugements de l'utilisateur sur la liste des documents trouvés. Une autre approche alternative, appelée ***pseudo*** ou ***blind*** RF emploie la technique de réinjection de pertinence automatiquement sans utiliser l'information issue des jugements de l'utilisateur.

Dans cette technique le système génère une liste triée de documents pour la première requête initiale. Ensuite, il sélectionne un certain nombre de documents (petit ensemble) à partir des tops documents du classement. Une nouvelle itération RF sera exécutée en supposant que les documents sélectionnés sont pertinents. La nouvelle requête générée par le processus de reformulation de requêtes est utilisée pour produire une nouvelle liste triée de documents.

Le principe de base du *pseudo* RF est que : « une itération feedback basée sur les premiers documents sélectionnés pendant l'exécution de la requête initiale, va engendrer un meilleur classement pour les documents pertinents ».

Selon Croft & Harper **[Croft and Harper, 1979]**, cette technique souffre d'un problème majeur qui est appelé: « *query drift* ». Ce problème apparaît lorsque l'ensemble des documents utilisés pour le feedback ne contient que peu de documents pertinents (ou ne contient pas du tout). Donc, cette technique ne fonctionne bien que dans le cas où la requête initiale permet d'extraire de bons résultats (beaucoup de documents pertinents).

1.4.4. Technique interactive pour la modification de requêtes

La plupart des méthodes de modification de requêtes procèdent d'une façon automatique pour le choix des termes à ajouter à la requête initiale. Une alternative qui apparaît naturelle (logique) est de donner la possibilité à l'utilisateur de choisir ces termes. Cette méthode est appelée « *interactive query expansion (IQE)* ». Selon Harman **[Harman, 1988]**, la sélection de termes par l'utilisateur peut améliorer les résultats de la reformulation mieux que le processus automatique.

1.4.5. Types du feedback

Spink **[Spink, 1997]** a proposé une classification constituée de 5 types de feedback :

1- ***Réinjection de pertinence basée sur le contenu*** : Dans ce type, l'utilisateur juge directement les contenus des documents (qui peuvent être soit positifs ou négatifs).

2- ***Réinjection de pertinence basée sur les termes*** : Dans ce type l'utilisateur essaye de trouver de nouveaux termes qui seront ajoutés à sa requête initiale à partir des documents supposés pertinents.

3- ***Feedback basé sur la grandeur*** : Ce type de feedback s'intéresse à la quantité de l'information (nombre de documents) issue de la première recherche. L'utilisateur juge les documents en choisissant une des options :

- Ensemble très grand de documents;
- Ensemble très petit de documents;
- Ensemble satisfaisant de documents.

4- ***Feedback basé sur la révision*** : Ce type utilise la notion de « historique de recherche » pour que l'utilisateur puisse avoir une idée sur la manière d'effectuer une recherche. Ce type est généralement utilisé par les utilisateurs non expérimentés à la recherche.

5- ***Feedback basé sur la révision de terminologie*** : D'une manière générale, ce type est utilisé par les utilisateurs qui n'ont pas une idée sur la terminologie utilisée par le système. Donc, en examinant le fichier inverse (ou bien un dictionnaire), l'utilisateur peut choisir les termes à utiliser dans sa recherche.

1.4.6. Types de capture

Selon **[Hanglin, 2004]**, il existe deux types de capture de l'information feedback :

- Capture binaire : l'utilisateur ne peut juger un élément dans la liste des résultats que selon deux valeurs qui sont : + ou -.
    - « + » : qui veut dire que l'élément est jugé par l'utilisateur comme étant pertinent
    - « - » : qui veut dire que l'élément est jugé par l'utilisateur comme étant non pertinent
- Capture non binaire : l'utilisateur peut juger un élément dans la liste des résultats selon plusieurs degrés de pertinence. Voici un exemple d'échelle :
    - (+2) : très pertinent
    - (+1) : marginalement pertinent
    - (-2) : non pertinent du tout

1.4.7. Conclusion sur la reformulation de requêtes

La technique de réinjection de pertinence a prouvé qu'elle peut être très utile pour la correction d'expression d'un besoin utilisateur. Son utilisation est conceptuellement simple, car l'utilisateur n'a qu'à marquer les documents pertinents et non pas les décrire.

Selon **[Ruthven and Lalmas, 2003]**, la réinjection de pertinence n'est pas uniquement une technique utile pour améliorer la qualité des résultats de recherche, mais aussi une méthode qui permet d'étudier le comportement de l'utilisateur en utilisant des systèmes de recherche d'information.

1.5. Evaluation des systèmes de Recherche d'Information

Le domaine de la RI n'est pas une science exacte, car les approches font en sorte que la pertinence du système soit la plus proche possible de celle de l'utilisateur. Même si le temps de réponse et l'espace utilisé pour le stockage d'informations sont plus ou moins importants dans l'évaluation des SRI, la qualité des résultats renvoyés (appelée aussi efficacité) par un système reste le critère le plus important. Cette évaluation permet de comparer les SRI entre eux.

Les mesures d'évaluation doivent être effectuées pour les différents SRI sur les mêmes bases documentaires afin de rendre valable cette comparaison. Pour cela, plusieurs campagnes d'évaluation ont été créées. L'évaluation des SRI repose généralement sur trois éléments principaux :

- une collection de documents de test;
- des requêtes de test;
- une liste des documents pertinents pour chaque requête.

Nous décrivons ci-dessous les mesures d'évaluation de SRI les plus courantes.

1.5.1. Rappel et précision

Les mesures de rappel et précision permettent d'évaluer la capacité d'un SRI à répondre aux deux objectifs principaux qui sont : retrouver tous les documents pertinents et rejeter tous les documents non pertinents. Afin de présenter ces deux mesures, nous introduisons (voir figure 1.4) le partitionnement de l'ensemble des documents restitués (noté B) par le SRI en deux sous ensembles : un sous-ensemble de documents pertinents et un sous-ensemble de documents non pertinents.

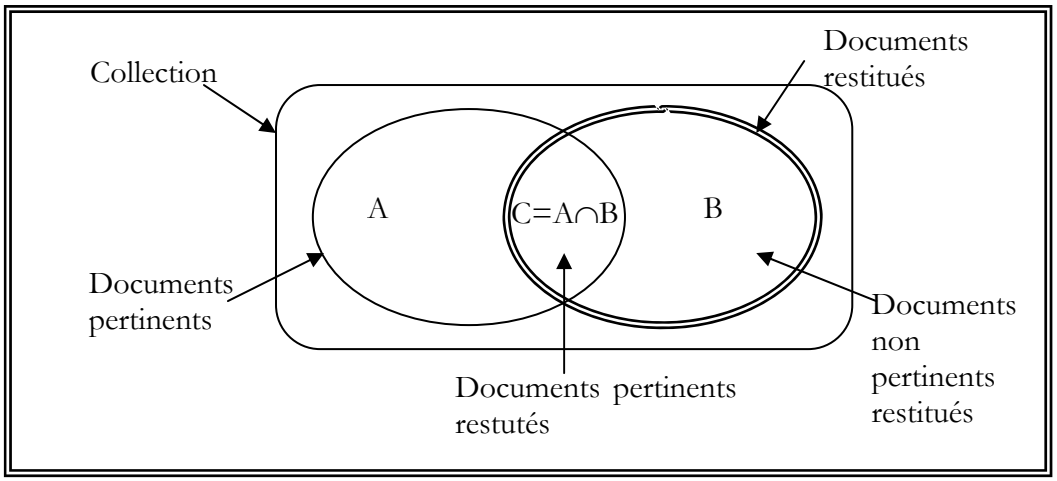


Figure 1.4 : Partition de la collection pour une requête

Les taux de rappel et de précision sont définis comme suit :

Taux de rappel : le rappel mesure la capacité du système de retrouver tous les documents pertinents répondant à une requête. Autrement dit, il mesure la proportion de documents pertinents restitués par le système relativement à l'ensemble des documents pertinents contenus dans la collection. Il est exprimé par :

$$\text{Rappel} = \frac{|C|}{|A|} \qquad \ldots (1.16)$$

Taux de précision : la précision mesure la capacité du système à rejeter tous les documents non pertinents à une requête. Autrement dit, elle mesure la proportion de documents pertinents relativement à l'ensemble des documents restitués par le système. Elle est exprimée par :

$$\text{Précision} = \frac{|C|}{|B|} \qquad \ldots (1.17)$$

Courbes Rappel-Précision

Pour pouvoir examiner les résultats efficacement, on doit calculer les paires des mesures (taux de rappel, taux de précision) à chaque document restitué. Afin d'illustrer les calculs de rappel et de précision, nous donnons un exemple (tableau 1.1) qui représente les résultats de recherche renvoyés pour une requête (Q1) par deux systèmes (S1, S2) dans une collection contenant 10 documents pertinents. Les courbes de rappel-précision associées sont tracées sur la figure 1.5.

	Système S1			Système S2		
Rang	pertinent	rappel	précision	pertinent	rappel	précision
1	√	0.100	1.000	√	0.100	1.000
2	**X**	0.100	0.500	√	0.200	1.000
3	**X**	0.100	0.333	**X**	0.200	0.666
4	√	0.200	0.500	√	0.300	0.750
5	√	0.300	0.600	**X**	0.300	0.600

6	X	0.300	0.500	√	0.400	0.666
7	X	0.300	0.428	X	0.400	0.571
8	√	0.400	0.500	√	0.500	0.625
9	X	0.400	0.444	√	0.600	0.666
10	X	0.400	0.400	X	0.600	0.600

Tableau 1.1 : Exemple de calcul de rappel et précision pour les systèmes S1 et S2

Nous pouvons remarquer que pour un même point de rappel correspondent plusieurs valeurs de précision. Une manière de rendre plus simple la lecture des courbes est de ne représenter que la précision calculée à chaque point de rappel.

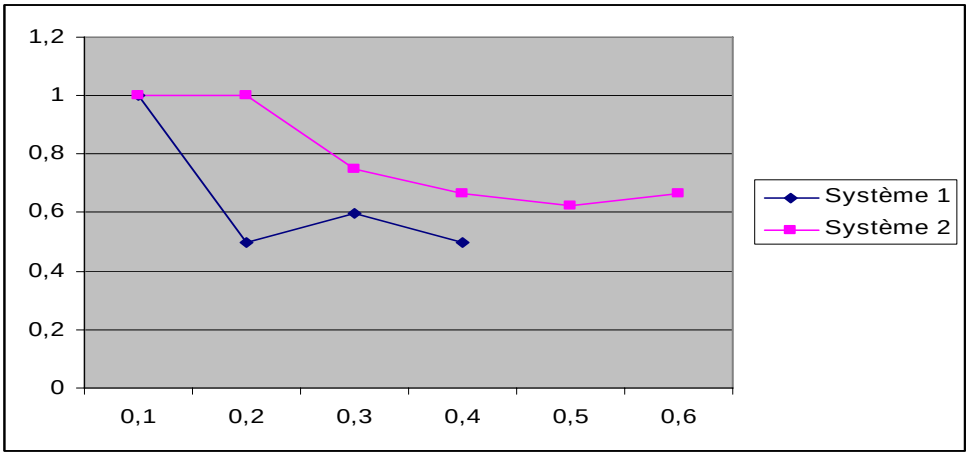


Figure 1.5 : Courbes de rappel-précision pour les systèmes S1 et S2

On dit qu'un système est parfait s'il ne restitue que les documents pertinents, avec un rappel et une précision de 100%. Dans la pratique, les deux mesures varient inversement, la précision diminue au fur et à mesure que le rappel augmente, ce qui signifie que la courbe rappel-précision est décroissante. Un système S1 est dit plus performant qu'un système S2 si sa courbe de rappel-précision est élevée par rapport à celle de S2. Donc de la figure ci-dessus nous pouvons déduire que le système S2 est plus performant que S1.

1.5.2. Mesures alternatives

En se basant sur le principe des mesures de rappel et de précision, les chercheurs ont été amenés à investir dans d'autres mesures qui pourront être plus représentatives. Ces nouvelles mesures essayent de combiner les mesures de rappel et de précision afin de sortir avec une seule valeur. Parmi les mesures proposées nous pouvons citer : la mesure harmonique et la mesure d'évaluation appelée E.

1.5.2.1. Mesure harmonique

La mesure harmonique H est une fonction qui combine les valeurs de rappel et de précision en une seule valeur incluse dans l'intervalle [0,1] **[Shaw et al., 1997]**.

$$H(j) = \frac{2}{\frac{1}{R(j)} + \frac{1}{P(j)}} \qquad \ldots (1.18)$$

où : R(j) et P(j) représentent respectivement le rappel et la précision du j ème document renvoyé par le système.

Cette mesure tend vers 0 lorsqu'aucun document pertinent n'est restitué et égale à 1 lorsque tous les documents restitués sont pertinents.

On peut constater que la fonction H prend des valeurs élevées quand les valeurs de rappel et de précision sont élevées.

1.5.2.2. Mesure d'évaluation « E »

Cette mesure a été proposée par Van Rijsbergen **[Rijsbergen, 1979]**. Son but est de permettre à l'utilisateur de spécifier laquelle des valeurs de précision et de rappel est plus intéressante. La mesure est ainsi définie par :

$$E(j) = 1 - \frac{1+b^2}{\frac{b^2}{R(j)} + \frac{1}{P(j)}} \qquad \ldots (1.19)$$

La variable **b** est un paramètre de l'utilisateur qui permet de spécifier l'importance du rappel et précision. Si b=1, E(j) va prendre la valeur du complément de la mesure harmonique H(j). Si (b < 1), l'utilisateur privilège le rappel et si (b > 1), il privilège la précision.

1.5.3. Collections de référence

Depuis les années 70, plusieurs projets dont le but est de construire des collections de test et des protocoles d'évaluation ont vu le jour. Ces projets permettent d'établir un cadre pour la comparaison des différents systèmes. Ces projets ont fourni plusieurs collections de référence en RI : la collection CACM, la collection CLEF (*Cross Language Evaluation Form*) et la collection ISI.

Un des plus importants projets d'évaluation est celui initié par la DARPA (*Defense Advanced Research Project Agency*) co-organisé avec le NIST (*National Institute of Standards and Technology*). Ce projet est appelé : La campagne d'évaluation TREC (*Text Retrieval Conference*) **[TREC]**. Cette campagne a commencé en 1992, son but est d'encourager le domaine de la recherche documentaire sur des grandes collections de test. Elle fournit un ensemble de documents et de requêtes aux participants qui vont ensuite renvoyer les résultats de recherche de leurs systèmes au NIST. Le NIST prend les N premiers documents de chaque système et les donne aux juges qui décident de la pertinence de chaque document.

Au départ, TREC comportait deux tâches : la tâche ad hoc et la tâche routage. Par la suite, de nouvelles tâches ont apparu, exemple : la tâche Terabyte, la tâche QA (Question-Réponse), la tâche multimédia et la tâche Web.

1.5.4. Evaluation des algorithmes de reformulation de requêtes

Selon **[Ruthven and Lalmas, 2003]**, le but de la reformulation de requêtes est d'améliorer le rappel et la précision par l'utilisation de l'information contenue dans les documents jugés par l'utilisateur. L'application de la technique de réinjection de pertinence a deux effets :

- l'effet d'ordonnancement (*Ranking effect*)
- l'effet du feedback (*Feedback effect*)

Le premier effet « *Ranking effect* » va simplement faire un ré-ordonnancement de la liste des résultats en mettant (*push*) les documents jugés pertinents à la tête de la liste. Le deuxième effet représente la capacité d'un système d'extraire les documents qui n'existaient pas dans la liste initiale.

Plusieurs techniques d'évaluation ont été proposées pour mesurer la performance des systèmes dans le deuxième type d'effet « *feedback* » **[Chang et al., 1971]** :

- ***Residual ranking*** (les collections résiduelles). Dans cette technique les documents utilisés dans le processus de reformulation de requêtes sont enlevés avant l'évaluation. Ceci veut dire que tout document utilisé sera enlevé (pertinent ou non pertinent). Les valeurs de rappel et de précision seront calculées sur le reste de la collection.

L'avantage de cette méthode est qu'elle ne considère l'effet du feedback que sur les éléments pertinents qui non pas été déjà vus. Mais son inconvénient est que les résultats du feedback ne peuvent pas être comparés avec le classement original parce que la collection résiduelle ne contient que peu de documents (inclus les documents pertinents) par rapport à la collection originale. Un autre problème arrive lorsque les documents pertinents sont enlevés de la collection. Après quelques itérations il n'y aura pas dans la collection des documents pertinents qui vont permettre de calculer les courbes rappel-précision. Donc il est conseillé d'utiliser cette méthode avec un petit nombre d'itérations feedback.

- ***Freezing.*** Cette méthode est basée sur les positions des documents dans la liste des résultats. Il existe deux types de « *freezing* » : « *full freezing* » et « *modified freezing* ». Le premier type permet de fixer (*froze*) les positions des *n* tops documents utilisés dans le processus de feedback. Le deuxième type fixe les positions à la position du dernier document jugé pertinent.

- ***Test and control groups.*** Dans cette technique la collection de test est divisée en deux ensembles :

- Test group ;
- Control group.

Le premier ensemble « *test group* » sert à évaluer la requête initiale. La requête modifiée (le processus de reformulation de requêtes) sera ensuite exécutée sur la collection « *control group* ».

Chacune de ces techniques possède des avantages et des inconvénients, pourtant, elles comparent la performance des différents algorithmes de reformulation dans des conditions idéalistes. Par exemple, souvent le nombre de documents utilisés dans le processus de feedback est le même, ce qui n'est pas le cas pour un utilisateur simple. Aussi, la reformulation de requêtes suppose une connaissance parfaite de la collection de test (l'ensemble des documents pertinents

pour une requête est connu à l'avance), ce qui n'est pas le cas dans un système de recherche interactive.

1.6. Conclusion

Dans ce chapitre, nous avons présenté les concepts de base de la recherche d'information traditionnelle à travers l'étude du processus de recherche d'information et des modèles de recherche. Chacun des modèles proposés tente de résoudre des problèmes inhérents à la recherche d'information. Nous avons aussi parlé de l'évaluation des systèmes de recherche à travers l'introduction des mesures d'évaluation.

Vu que la notion de reformulation de requêtes représente le fond du sujet traité dans ce mémoire, nous lui avons consacré une partie importante de ce chapitre, en particulier la technique de réinjection de pertinence (RF).

Ce chapitre a été consacré aux techniques, modèles et approches proposés dans la RI manipulant des documents plein texte. On assiste aujourd'hui de plus en plus à la prolifération de documents structurés ou semi structurés. Compte tenu de leur nature, mélange de contenu et de structure, ces documents apportent ou réactualisent un certain nombre de problèmes de la RI traditionnelle. C'est le sujet du chapitre 2.

Chapitre 2 :

Recherche d'Information Structurée

2.1. Introduction

Depuis leur création, les langages de balisage permettent d'archiver les documents électroniques sous une forme structurée ou semi-structurée. De SGML (*Standard Generalized Markup Language*) **[SGML]**, au HTML (*HypertText Markup Language*) qui est une exploitation adaptée au WWW (*World Wide Web*), et qui permet de décrire le formatage des données sur un browser Web **[Amoussou]**, on arrive aujourd'hui à un format standard et universel d'échange de données connu sous le nom de XML (*eXtensible Markup Language*) qui est une recommandation (février 1998) du W3C (*World Wide Web Consortium*) **[W3C_XML, 1998]** ; ce dernier permet de combiner l'information structurelle avec le contenu.

Une exploitation efficace des documents structurés et semi-structurés disponibles et qui ne cessent d'augmenter d'une façon exponentielle en termes de nombre et de volume nécessite la prise en compte de la dimension structurelle. Cette dimension a engendré l'apparition d'une toute nouvelle problématique en RI, qui est la *RI structurée*.

Dans ce chapitre, nous allons présenter les différents aspects traités par la RI structurée, notamment les approches et solutions citées dans la littérature. Le présent chapitre est organisé comme suit :

- La section (2.2.) présente les concepts de base liés aux documents semi-structurés;
- la section (2.3.) décrit les problématiques spécifiques à la RI structurée, deux classes d'approches sont à distinguer : les approches orientées BD, et les approches orientées RI ;
- la section (2.4.) présente les différentes techniques d'indexation des documents semi-structurés ;
- la section (2.5.) donne un aperçu des langages de requêtes qui prennent en compte l'aspect structurel des documents XML ;
- la section (2.6.) présente les différents modèles de recherche proposés dans la littérature ;
- la section (2.7.) traite deux exemples de systèmes de recherche dans les documents XML ;
- Enfin, la section (2.8.) décrit l'aspect évaluation dans un contexte de RI structurée.

2.2. Documents semi-structurés

Depuis leur création, les documents semi-structurés ont présenté un intérêt particulier dans le domaine de gestion électronique des documents. De SGML pour l'archivage structuré au HTML la forme la plus populaire, utilisée spécialement pour la présentation (notamment sur Internet), on assiste aujourd'hui au format connaissant la plus grande expansion à savoir XML.

Chacun de ces formats intègre la notion de structure avec un degré de restriction qui diffère. Nous nous concentrons dans le cadre de ce travail sur les documents XML.

2.2.1. Les Documents XML

XML (*Langage à balises étendues*, ou *Langage à balises extensibles*) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de structurer des documents grâce à des balises.

Contrairement à HTML, qui est considéré comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises. XML est un sous ensemble de SGML, défini par le standard ISO8879 en 1986 **[St. Laurent, 2000]**, utilisé dans le milieu de la GED (*gestion électronique documentaire*). XML reprend la majeure partie des fonctionnalités de SGML. Il s'agit donc d'une simplification de SGML.

La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il permet de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir. En réalité les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML). Ainsi, XML permet de séparer le contenu de la présentation. XML a été mis au point par le XML Working Group sous l'égide du W3C dès 1996 **[St. Laurent, 2000]**. Depuis le 10 février 1998, les spécifications *XML 1.0* ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu. Tous les documents liés à la norme XML sont consultables et téléchargeables sur le site web du W3C **[W3C_XML, 1998]**.

2.2.2. La notion de structure

La structure des documents XML est définie par des balises encadrant les portions d'informations. Une balise (ou tag ou label) est une suite de caractères encadrés par "<" et ">", comme par exemple <nom_balise>. Un élément est une unité sémantique identifiée, délimitée par des balises de début <A> et de fin </A>, comme par exemple : <ma_balise> texte </ma_balise>. Les éléments peuvent être imbriqués :

```
<Document type= « Livre »>
    <Titre> Introduction to modern information retrieval </Titre>
    <Auteur>
        <Nom>Gerard</ Nom >
        <Prenom>Salton</ Prenom >
    </Auteur>
    <Annee>1984</ Annee >
</Document>
```

Tableau 2.1 : Les éléments d'un document XML

Les attributs des balises sont spécifiés au début de l'élément et après le nom de la balise, en utilisant la syntaxe nom_attribut=valeur_ attribut. Par exemple :

<ma_balise nom_attribut = valeur_ attribut >texte </ma_balise>.

2.2.2.1. Structure des documents XML

XML fournit un moyen pour vérifier la syntaxe d'un document grâce aux DTD (*Document Type Definition*). Il s'agit d'un fichier décrivant la structure (grammaire) des documents. Ainsi un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles. Un document suivant les règles de XML est appelé *document bien formé*. Un document XML possédant une DTD et étant conforme à celle-ci est appelé *document valide*.

2.2.2.2. Décodage d'un document XML

XML permet de définir un format d'échange selon les besoins de l'utilisateur et offre des mécanismes pour vérifier la validité du document produit. Il est donc essentiel pour l'utilisateur d'un document XML de pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé analyseur (en anglais parser, parfois francisé en *parseur*). Le parseur permet d'une part d'extraire les données d'un document XML (on parle d'*analyse* du document ou de *parsing*) et d'autre part de vérifier la validité du document.

2.2.2.3. Les avantages de XML

Le langage XML possède plusieurs avantages **[St. Laurent, 2000]**, parmi lesquels nous pouvons citer les suivants :

- *La lisibilité* : aucune connaissance ne doit théoriquement être nécessaire pour comprendre le contenu d'un document XML ;
- *Une structure arborescente* : permettant de modéliser la majorité des problèmes informatiques ;
- *Universalité et portabilité* : les différents jeux de caractères sont pris en compte ;
- *Déployable* : il peut être facilement distribué par n'importe quel protocole (exemple : HTTP);
- *Intégrabilité* : un document XML est utilisable par toute application pourvue d'un parseur;
- *Extensibilité* : un document XML doit pouvoir être utilisable dans tous les domaines d'applications. Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

2.2.3. Les standards du monde XML

2.2.3.1. DOM

DOM **[W3C_DOM, 1998]** (*Document Object Model*) est une interface de programmation API permettant d'accéder à la structure et au contenu d'un document XML **[Olivier]**. Cette API permet de générer l'arbre d'objets correspondant à un document XML, les objets peuvent être des éléments ou des attributs, ces derniers ont des valeurs qui sont sauvegardés au niveau des nœuds feuilles. Ainsi, un arbre DOM est composé de : nœud racine ; attributs ; nœuds internes ; et nœuds feuilles.

Par exemple, si on considère le tableau suivant, extrait d'un document XML :

```
<DOC>
   <CORPS>
```

```
        <SECTION>
                <TITRE>Recherche d'information </TITRE>
                <PAR>Le domaine de la recherche…. </PAR>
        </ SECTION >
        < SECTION >
                <TITRE>Intelligence Artificielle </TITRE>
                <PAR>Ce domaine…</PAR>
        </ SECTION >
   </CORPS>
</DOC>
```

Tableau 2.2 : Extrait d'un document XML

DOM représente l'extrait du tableau 2.2 comme suit (voir Figure 2.1) :

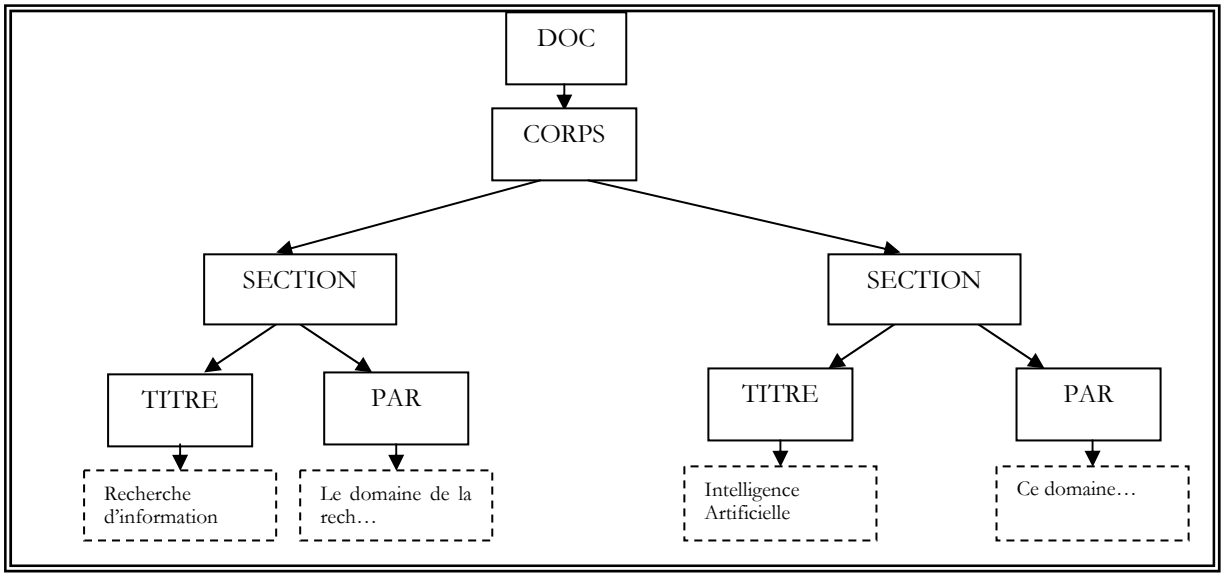


Figure 2.1 : Représentation DOM de l'extrait du tableau 2.1

2.2.3.2. XPath

Xpath est une recommandation du W3C **[Clark and Derose, 1999]**. Il joue un rôle essentiel dans l'identification des sections de documents XML en permettant l'adressage des différentes parties du document XML par l'intermédiaire de chemins d'accès. Un chemin d'accès fournit les instructions permettant de se rendre jusqu'à un emplacement précis du document XML **[Sturm, 2000]**. XPath utilise une syntaxe compacte et non-XML pour faciliter son utilisation dans des URI (*Uniform Resource Identifier*) et des attributs de balises XML. XPath agit sur les structures abstraites et logiques d'un document XML, plutôt que sur sa syntaxe apparente. Le nom XPath vient de l'utilisation d'une écriture de type "chemins d'accès", comme les URL (*Uniform Resource Locator*), pour se déplacer à l'intérieur de la structure hiérarchique d'un document XML **[Clark and Derose, 1999]**.

Pour l'expression des chemins, on utilise la notion d'axes. Ces derniers permettent de se rendre à un emplacement précis dans l'arbre du document. Les axes suivants sont disponibles :

- L'axe enfant *(**child**)* contient les enfants directs du nœud contextuel ;

- l'axe descendant *(**descendant**)* contient les descendants du nœud contextuel; un descendant est un enfant ou un petit enfant etc... Aussi, l'axe descendant ne contient jamais de nœud de type attribut ou des noms d'espace ;

- l'axe parent *(**parent**)* contient le parent du nœud contextuel, s'il y en a un ;

- l'axe ancêtre *(**ancestor**)* contient les ancêtres du nœud contextuel; cela comprend son parent et les parents des parents etc... Aussi, cet axe contient toujours le nœud racine, sauf si le nœud contextuel est lui-même la racine ;

- l'axe cible suivante *(**following-sibling**)* contient tous les nœuds cibles du nœud contextuel; si celui-là est un attribut ou un espace de noms, le suivant ciblé est vide ;

- l'axe cible précédente *(**preceding-sibling**)* contient tous les prédécesseurs du nœud contextuel; si le nœud contextuel est un attribut ou un espace de noms, la cible précédente est vide ;

- l'axe suivant *(**following**)* contient tous les nœuds du même document que le nœud contextuel qui sont après le nœud contextuel dans l'ordre du document, à l'exclusion de tout descendant, des attributs et des espaces de noms ;

- l'axe précédent *(**preceding**)* contient tous les nœuds du même document que le nœud contextuel, qui sont avant lui dans l'ordre du document, à l'exclusion de tout ancêtre, des attributs et des espaces de noms ;

- l'axe attribut *(**attribute**)* contient les attributs du nœud contextuel; l'axe est vide quand le nœud n'est pas un élément ;

- l'axe des espaces de noms *(**namespace**)* contient les nœuds des espaces de noms du nœud contextuel; l'axe est vide quand le nœud contextuel n'est pas un élément ;

- l'axe réflexif *(**self**)* contient seulement le nœud contextuel ;

- l'axe descendant-ou-réflexif *(**descendant-or-self**)* contient le nœud contextuel et ses descendants ;

- l'axe ancêtres-ou-réflexif *(**ancestor-or-self**)* contient le nœud contextuel et ses ancêtres; aussi, cet axe contiendra toujours le nœud racine **[W3C_XPATH, 1999]**.

2.2.3.3. XQuery

Depuis octobre 1999, le W3C travaille sur le problème d'interrogation des documents XML **[W3C_QUERY]**. Le fruit des efforts du consortium est le langage XML Query ou bien XQuery. Ce langage a donc été conçu pour permettre de créer des requêtes précises tout en pouvant s'adapter à tout type de source de données XML, qu'elles soient question de bases de données, documents XML ou autres.

XQuery peut être utilisé avec des documents XML validés par des schémas[2], des DTD ou encore simplement des documents XML bien formés. XQuery est un langage basé sur les expressions de type FLWR (for, let, where, return). Un script ou programme XQuery contiendra

[2] XML Schema contient des informations de structure et de typage des données du document XML. Il présente de nombreuses améliorations par rapport aux DTD, notamment une plus grande flexibilité et un typage plus important des données. XML Schema est une recommandation du W3C depuis 2001.

toujours une ou plusieurs expressions et optionnellement des fonctions et des définitions **[Bilodeau]**.

a) Expressions de chemin

Les expressions de chemins ressemblent beaucoup à celles que l'on retrouve dans le langage XPath. Prenons par exemple l'extrait du document suivant, tableau 2.3, dans lequel l'attribut «num» représente un numéro associé à un élève et la valeur de la note de celui-ci.

```
<Examen>
        <note num="001">80</note>
        <note num="012">75</note>
        <note num="525">99</note>
        <note num="601">60</note>
</Examen>
```

Tableau 2.3 : Extrait d'un document XML

L'expression de chemin suivante permet de retourner le texte contenu dans le nœud dont la valeur de l'attribut num est égale à celui de la variable $a : //examen/note[@num=$a]/text()

b) Les expressions FLWR

Cette syntaxe est utilisée dans différents langages d'interrogation des documents XML. Le nom provient de for, let, where et return.

for : fournit un mécanisme d'itération ;

let : permet l'assignation de variable ;

where : les clauses for et let génèrent un ensemble de nœuds qui peuvent être filtrés par un ou plusieurs prédicats dans une clause where ;

return : génère le résultat de l'expression FLWR.

Voici un exemple simple d'une requête FLWR. Cette requête a pour but de présenter une comparaison des prix des livres similaires (ayant le même titre) dont l'auteur est Mohamed Dib, dans deux librairies affichant leurs produits sur le web.

```
<livres>
{for $a in document("livres.xml")//livres/livre[auteur='Mohamed Dib'], $b in
document("produits.xml")//produits/livre[@auteur=' Mohamed Dib '] where $a/titre = $b/titre
return
<livre>
<prix1>{$a}</prix1>
<prix2>{$b}</prix2>
<livre>
}
</livres>
```

Tableau 2.4 : Exemple d'une requête XQuery

2.2.4. Autres formats

L'intérêt de disposer d'un format commun d'échange d'information dépend du contexte professionnel dans lequel les utilisateurs interviennent. C'est pourquoi, de nombreux formats de données issus de XML apparaissent (il en existe plus d'une centaine) :

OFX : Open Financial eXchange, pour les échanges d'informations dans le monde financier ;

MathML : Mathematical Markup Language, permet de représenter des formules mathématiques ;

CML : Chemical Markup Language, permet de décrire des composés chimiques ;

SMIL : Synchronized Multimedia Integration Language, permet de créer des présentations multimédia en synchronisant diverses sources : audio, vidéo, texte,...etc.

2.3. Les enjeux de la RI structurée

2.3.1. La granularité de l'information recherchée

La RI traditionnelle permet de renvoyer des résultats de recherche aux utilisateurs sous forme d'une liste de documents. Ces derniers peuvent être composés de contenus hétérogènes. Ce qui oblige l'utilisateur à chercher l'information qu'il désire avoir à l'intérieur du document.

La RI structurée offre à un utilisateur la possibilité d'avoir des réponses sous une autre forme plus significative. Les réponses aux requêtes des utilisateurs sont des unités d'information auto explicatives **[Sauvagnat, 2005]**. Cela signifie que l'information contenue ne dépend pas d'une autre pour être comprise. Les unités d'information sont des sous arbres des documents XML.

L'évaluation de la pertinence d'un élément (sous arbre) vis-à-vis d'une requête se fait selon deux dimensions, qui sont :

- Exhaustivité;
- Spécificité.

L'exhaustivité décrit jusqu'à quel point l'élément discute du sujet de la requête. Alors que, la spécificité décrit jusqu'à quel point l'élément se focalise sur le sujet de la requête.

Le principe de recherche dans les documents structurés est défini par : « *Un système devrait toujours retrouver l'unité d'information la plus exhaustive et spécifique répondant à une requête* ».

2.3.2. Les problématiques spécifiques à la RI structurée

La dimension structurelle des documents structurés a soulevé plusieurs problématiques relatives à chaque phase du processus de recherche.

La première problématique spécifique à ce contexte est l'indexation de ces documents. L'indexation doit prendre en compte l'information structurelle qui s'ajoute au contenu. Quant à

ce contexte plusieurs questions se posent **[Sauvagnat, 2005]** : Que doit-on indexer de la structure des documents ? Comment relier cette structure au contenu même du document ? Comment pondérer les termes d'indexation, c'est à dire comment évaluer l'importance d'un terme au sein de l'élément, du document et de la collection ?

La deuxième problématique est celle de l'interrogation des corpus de documents semi structurés. Un système doit pouvoir permettre à un utilisateur d'exprimer ses besoins d'une manière simple et en exploitant les deux types d'information contenue dans les documents semi structurés (textuelle et structurelle).

La dernière problématique est celle des modèles de recherche et de tri des unités d'informations. Les systèmes de recherche doivent pouvoir décider de la granularité de l'information à renvoyer s'il s'agit d'une requête orientée contenu seulement. S'il s'agit d'une requête orientée contenu et structure deux cas sont envisageables :

- l'utilisateur spécifie le type d'éléments à renvoyer ;
- l'utilisateur ne spécifie pas le type d'éléments à renvoyer.

Dans le deuxième cas, c'est au système de décider de la granularité de l'information à renvoyer.

Ce sont trois problématiques spécifiques à la RI structurée et elles restent toujours d'actualité, vue la jeunesse de ce domaine d'intérêt.

2.3.3. Approches pour la RI structurée

La RI structurée englobe deux approches qui tentent de proposer des méthodes pour : l'indexation, l'interrogation, la recherche et le tri des documents XML. Ces deux approches sont :

- ***L'approche orientée données*** : Elle utilise des techniques développées par la communauté des bases de données, et voit les documents XML comme des collections de données, typées et relativement homogènes.
- ***L'approche orientée document*** : Cette approche est prise en charge par la communauté de la recherche d'information, et se focalise sur des applications considérant les documents XML d'une manière traditionnelle, c'est-à-dire que les balises servent uniquement à décrire la structure logiques des documents.

Le tableau 2.5 résume quelques points propres à chaque approche pour chacune des phases du processus de recherche.

	Approches orientées BD	Approches orientées RI
Indexation	- Confondent les notions d'indexation et de stockage : toute l'information textuelle et structurelle des documents est stockée au sein de tables dans des BD. - Ceci pose un problème pour les recherches orientées contenu,	- Utilisent des techniques traditionnelles pour l'extraction des termes d'indexation. - De nouvelles problématiques sont soulevées concernant la structure : Que doit-on indexer de la structure des documents ? Comment relier cette structure au

	puisque le contenu textuel est indexé en tant que chaîne de caractères. - Proposent des schémas de stockage optimaux pour la structure des documents.	contenu même du document ?
Langages d'interrogation	- Historiquement les premiers à proposer des langages pour l'interrogation des documents XML. - Ces langages sont basés sur des syntaxes proches de SQL, et permettent à l'utilisateur d'exprimer des conditions très précises sur la structure des documents. - Les requêtes doivent toujours porter sur des conditions de structure bien définies. L'utilisateur doit de plus spécifier le type d'élément qu'il désire voir retourné par le système, alors qu'il n'a pas forcément d'idée précise sur la question.	- Cherchent à simplifier ces langages en ce qui concerne les conditions de structure. - Proposent de nouvelles fonctionnalités concernant la recherche sur le contenu (Utilisation du prédicat 'about' au lieu de 'contains', ou bien encore d'opérateurs booléens dans des conditions de contenu).
Traitement des requêtes	- Evaluent de façon exacte des expressions de type *attribut=valeur*. - Le traitement est effectué d'une manière booléenne et il n'est pas possible de renvoyer à l'utilisateur une liste triée de résultats.	- Cherchent à évaluer le degré de pertinence entre la requête et les unités d'informations et attribuent à ces derniers un score de pertinence. - L'intérêt est double : tout d'abord sélectionner les unités d'informations qui répondent le mieux au besoin de l'utilisateur, et lui proposer ensuite une liste triée de résultats.

Tableau 2.5 : Caractéristiques des deux approches pour chacune des phases du processus de RI

Les solutions proposées par la communauté de la RI peuvent être utilisées comme « *surcouche* » aux solutions orientées BD. Cette surcouche sert essentiellement à intégrer la notion de pertinence dans la recherche, en complétant les approches proposées par la communauté de BD pour le stockage et l'interrogation des documents **[Sauvagnat, 2005]**.

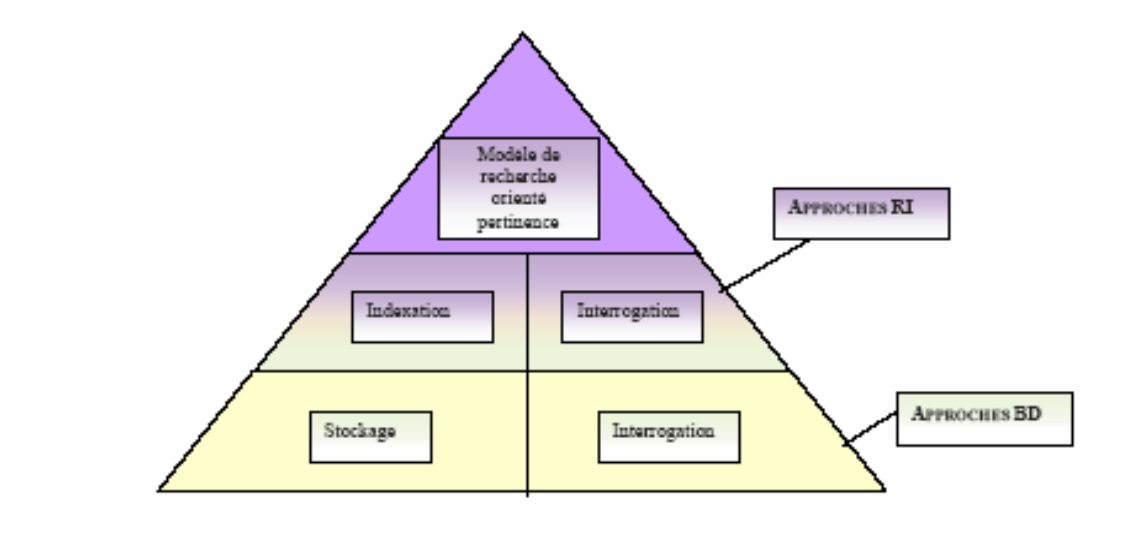


Figure 2.2 : Domaines de compétence de la BD et de la RI **[Sauvagnat, 2005]**

2.4. Indexation des documents semi-structurés

L'indexation des documents semi-structurés diffère de celle utilisée pour des documents textes « plats » par le fait de la prise en compte de la dimension structurelle qui s'ajoute au contenu. Quant à ce contexte plusieurs questions se posent : Que doit-on indexer de la structure des documents ? Comment relier cette structure au contenu même du document ? Comment pondérer les termes d'indexation, en d'autres termes comment évaluer l'importance d'un terme au sein de l'élément, du document et de la collection ? Nous essayons de répondre à chacune de ces questions dans les sections qui suivent.

2.4.1. Que faut-il indexer

Comme nous l'avons déjà mentionné, les documents semi-structurés comportent deux types d'informations à savoir l'information textuelle (le contenu) et l'information structurelle (la structure). A première vue, nous pensons que la manière la plus simple pour indexer ces documents est de ne considérer que l'information textuelle, ce qui rend ce processus similaire à celui utilisé en RI traditionnelle. On relève deux faiblesses de cette manière de faire : premièrement aucune recherche sur la structure n'est possible, et deuxièmement la granularité utilisée reste toujours le document dans son intégralité.

Selon **[Sauvagnat, 2005]**, plusieurs aspects doivent être couverts par le schéma d'indexation :

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

Deux dimensions caractérisent les approches d'indexation des documents semi-structurés :

- le schéma de stockage des documents ;
- les types de transformations possibles entre les documents XML et les structures de stockage.

Concernant le schéma de stockage, on trouve deux types d'approches :

- les approches orientées SGBD (ou middleware de transformation) ;
- les modèles de stockage XML natifs qui stockent des documents complets ou des parties de documents.

Pour la dimension types de transformations possibles, on distingue :

- les approches de transformation basées sur un modèle : ces approches créent un schéma générique de base de données qui reflète le modèle de données du format XML. Ces approches sont considérées comme extensibles et n'ont pas besoin de la DTD des documents ;
- les approches de transformations basées sur la structure : ces approches construisent un schéma d'index qui prend en compte la sémantique de l'application.

Dans **[Sauvagnat, 2005],** l'auteur adopte une classification qui porte sur le type d'information, cette classification permet de mieux comprendre les enjeux soulevés par chaque type d'indexation.

2.4.2. Indexation de l'information textuelle

La problématique d'indexation de l'information textuelle reste d'actualité dans le domaine de la RI structurée. Les approches orientées BD considèrent le texte complet des nœuds feuilles comme étant l'unité textuelle d'indexation, alors que les approches orientées RI considèrent le terme. Deux problèmes s'avèrent intéressants à détailler dans la partie indexation textuelle, premièrement le problème de la portée des termes et deuxièmement le problème de leur pondération.

2.4.2.1. Portée des termes d'indexation

Nous tentons dans cette partie de trouver une réponse à la question : « comment rattacher les termes à l'information structurelle ? ». Deux solutions ont été proposées dans la littérature :

- l'approche d'indexation des sous-arbres imbriqués qui cherche à agréger le contenu des nœuds ;
- l'approche d'indexation des unités disjointes qui indexe tous les contenus des nœuds séparément.

a) Sous arbres imbriqués

Les approches de ce premier groupe considèrent que le texte complet de chaque nœud de l'index est un document atomique et propagent donc les termes des nœuds feuilles dans l'arbre des documents.

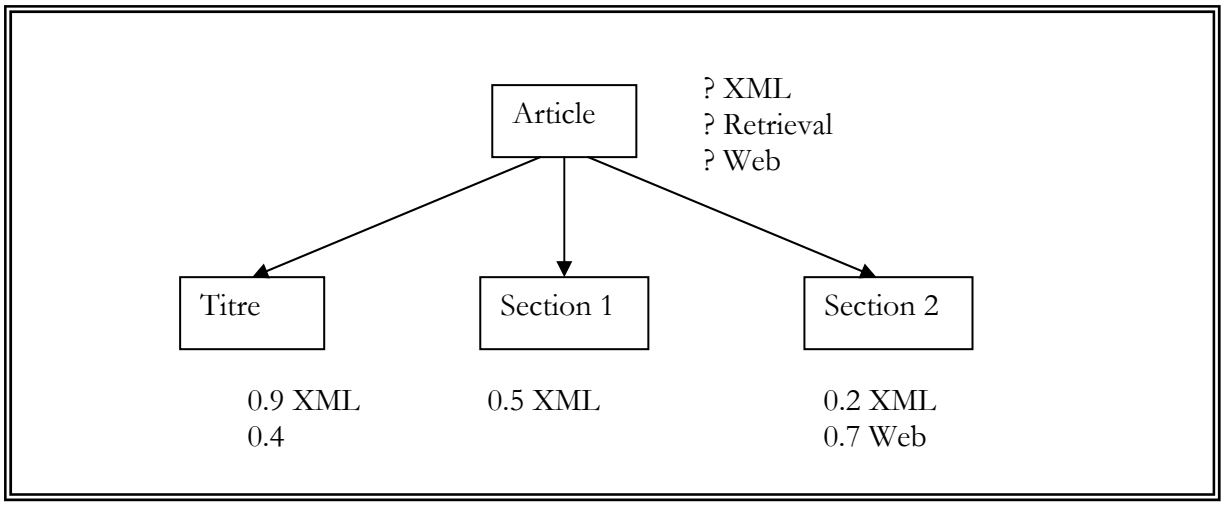


Figure 2.3 : L'approche sous-arbres imbriqués

b) Unités disjointes

Dans ces approches le document XML est décomposé en unités disjointes, de telle façon que le texte de chaque nœud de l'index est l'union d'une ou de plus de ces parties disjointes**.**

2.4.2.2. Pondération des termes d'indexation

Dans le cas des approches orientées BD aucune pondération des termes n'est effectuée parce que le texte des nœuds feuilles est considéré comme étant une seule unité. Contrairement aux approches orientées RI, la pondération est une étape primordiale. Cependant, elle doit être vue d'une autre façon. De nouvelles formules permettant d'évaluer l'importance des termes au sein de l'élément, du document et de la collection s'avèrent plus appropriées, ***idf*** (*Inverse Document Frequency*) utilisé en RI traditionnelle a été adapté pour la RI structurée sous le nom d'***ief*** (*Inverse Element Frequency*). Ainsi la formule ***tf.idf*** a été remplacée par ***tf.itdf*** (*Term Frequency-Inverse Tag and Document Frequency*), qui permet de calculer la force discriminatoire d'un terme ***t*** pour une balise ***b*** relative à un document ***d***.

Plusieurs autres paramètres peuvent être pris en compte en plus de la fréquence du terme dans l'élément, on trouve aussi la longueur de l'élément, la longueur moyenne des éléments de la collection, etc.….

2.4.3. Indexation de l'information structurelle

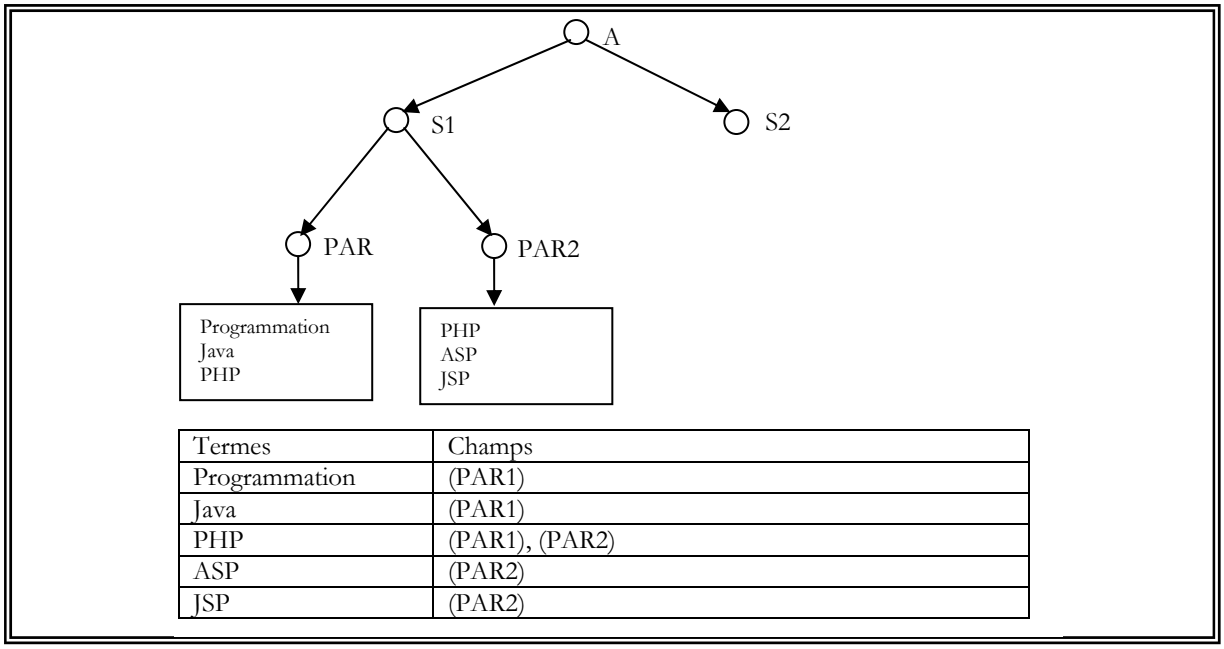
Dans la littérature on distingue trois classes d'approches pour l'indexation de l'information structurelle, la première dite basée sur des champs, la seconde dite basée sur des

chemins, et la dernière dite basée sur des arbres. Ces trois approches sont indépendantes de la manière d'utiliser l'information textuelle (l'approche BD ou bien l'approche RI).

2.4.3.1. Indexation basée sur des champs

Dans cette méthode d'indexation, le document est représenté comme un ensemble de champs et du contenu associé à chaque champ**.** Plusieurs façons permettent d'obtenir les différents champs d'un document XML :

- ils peuvent être codés en tant que métadonnées dans les fichiers XML, par exemple en utilisant RDF ;
- dans le cas d'un document d'un format quelconque transformé en XML, ils peuvent provenir du document dans son format original ;
- ils peuvent être retrouvés à l'aide de différentes techniques d'extraction ;
- ils sont simplement extraits de la DTD ou du schéma XML associé.



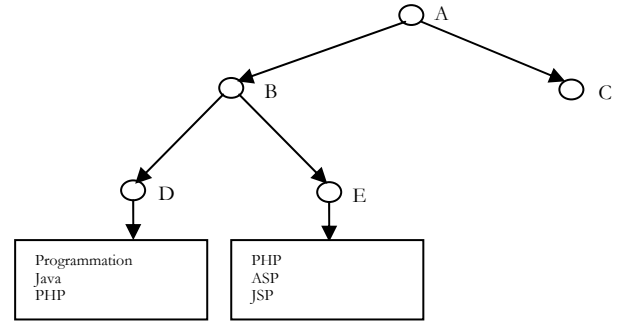
Termes	Champs
Programmation	(PAR1)
Java	(PAR1)
PHP	(PAR1), (PAR2)
ASP	(PAR2)
JSP	(PAR2)

Figure 2.4 : Exemple d'indexation basée sur des champs

Les termes de l'index sont construits en combinant le nom du champ avec les termes du contenu pour permettre une recherche restreinte à certains champs.

2.4.3.2. Indexation basée sur des chemins

Ce type de techniques facilite la navigation dans les documents en permettant la résolution des expressions XPath, il permet aussi de retrouver des documents ayant des valeurs connues pour certains éléments ou attributs, et utilise des index pleins textes sur les contenus. Ces techniques souffrent de la difficulté de retrouver les relations ancêtre-descendant entre les différents nœuds des documents. La figure suivante (figure 2.5) représente une illustration de ce type d'indexation.



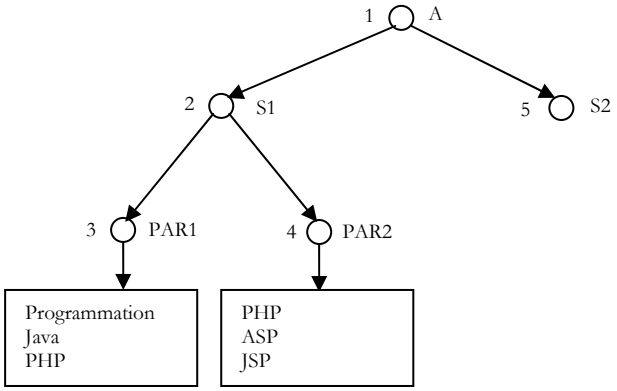
Termes	Chemins
Programmation	(/A/B/D)
Java	(/A/B/D)
PHP	(/A/B/D), (/A/B/E)
ASP	(/A/B/E)
JSP	(/A/B/E)

Chemins	Documents
(/A)	Doc1,…
(/A/B)	Doc1,…
(/A/B/D)	Doc1,…
(/A/B/E)	Doc1,…
….	……..

Figure 2.5 : Exemple d'indexation basée sur des chemins

2.4.3.3. Indexation basée sur des arbres

Ce type d'indexation permet de résoudre la difficulté de la technique basée sur des chemins (à savoir : retrouver les relations ancêtre-descendant) car les nœuds de l'arbre sont numérotés dans la structure arborescente des documents.



Termes	Noeuds
Programmation	3
Java	3
PHP	3, 4
ASP	4
JSP	4

Figure 2.6 : Exemple d'indexation basée sur des arbres

Plusieurs techniques appartiennent à cette classe, nous citons parmi elles :

- l'index ANOR **[Lee et al., 1996]** ;
- l'approche EDGE **[Florescu and Kossmann, 1999]**;
- l'approche BINARY **[Florescu and Kossmann, 1999]**;
- l'index Xpath Accelerator **[Grust, 2002]**;
- l'approche XFIRM **[Sauvagnat, 2005].**

2.5. Langages de requêtes

Les langages de requêtes permettent de donner un moyen d'expression du besoin utilisateur. L'aspect structurel des documents XML permet d'étendre les possibilités d'expression, ce qui a donné naissance à une nouvelle forme d'interrogation. En plus des requêtes orientées contenu, l'utilisateur peut ajouter des conditions de structure pour créer des requêtes dites orientées contenu et structure.

Dans cette section nous parlons des principaux langages de requêtes proposés dans la littérature. Ces langages sont principalement issus de la communauté de bases de données, et leur syntaxe est souvent dérivée de SQL. Récemment, quelques langages orientés RI ont fait leur apparition (exemple : le langage XFIRM).

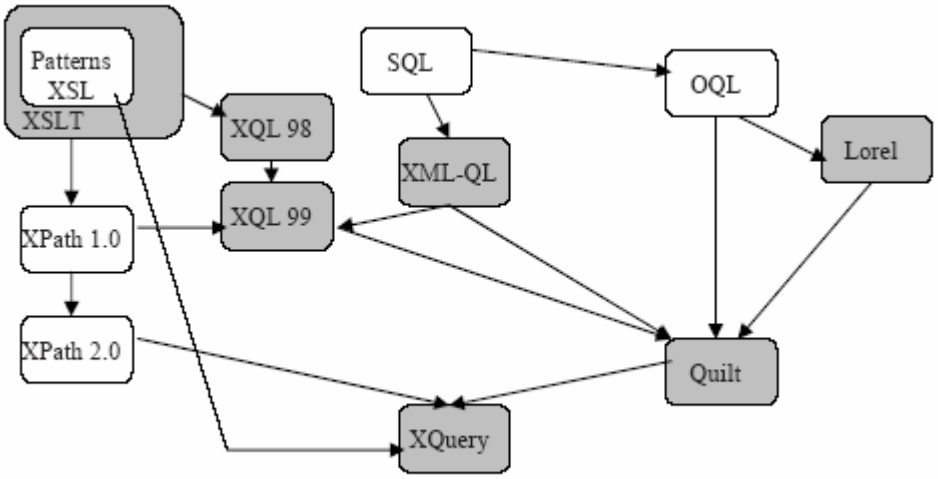


Figure 2.7 : Historique des langages d'interrogation XML **[Sauvagnat, 2005]**

2.5.1. XQL

XQL (*XML Query Language*) est un langage d'interrogation de documents XML dont la sémantique est proche de celle de XQuery et de XPath. Soumis au W3C par webmethods et Microsoft, XQL est resté au stade de la proposition. L'objectif de XQL était de rester simple tout en offrant plus de possibilités que XPath.

2.5.2. Quilt

C'est un langage d'interrogation XML conçu pour interroger des sources de données hétérogènes, et inspiré du design XML-QL, SQL et OQL. Ce langage est créé par des membres du groupe de travail « Query » du W3C **[W3C_QUERY]**.

2.6. Traitement de requêtes

Dans le contexte de la RI structurée nous trouvons deux approches de traitements de requêtes :

- l'approche orientée BD, qui traite le contenu des documents d'une façon booléenne ;
- l'approche orientée RI, qui essaye d'attribuer des scores de pertinence aux nœuds d'un document XML.

Les requêtes sont de deux types : requêtes orientées contenu et requêtes orientées contenu et structure (voir section 2.8.3.1). Afin de traiter ces requêtes plusieurs modèles ont été proposés. Ces modèles sont à la base des modèles de RI traditionnelle adaptés au contexte de la RI structurée. Dans ce qui suit, nous donnons un aperçu sur quelques modèles proposés dans la littérature.

2.6.1. Modèle vectoriel étendu

Ce modèle tente de mesurer la similarité de chaque élément vis à vis la requête. La plupart des approches utilise la méthode de propagation et indexe les documents sous la forme de sous arbres imbriqués. Ce qui veut dire que les termes sont propagés dans l'arbre du document. Le résultat de la requête est une liste ordonnée des éléments renvoyés.

Une des premières adaptations du modèle vectoriel à un contexte de documents structurés est celle de Fuller et al. **[Fuller et al., 1993]**. Dans leur modèle, la similarité d'un nœud vis-à-vis une requête est exprimée par la formule suivante :

$$Sim(q,n) = \alpha(T).\cos m(q,n) + \sum_{k=1}^{s} \frac{\cos m(q,n_k)}{\beta^{k-1}} \qquad \ldots(2.1)$$

Où :

$\alpha(T)$: facteur représentant le type du nœud.

s : le nombre de nœuds descendants n_k de n.

β : paramètre qui assure la non-introduction d'un biais par le nombre des nœuds descendants.

La fonction *cosm* est définie comme suit :

$$\cos m(q,n) = \sum_{i=1}^{T} \frac{w_i^q * w_i^n}{|n|} \qquad \ldots(2.2)$$

Où w_i^q et w_i^n représentent respectivement le poids du terme t_i dans la requête q et dans le nœud n, et $|n|$ le nombre de termes dans le nœud n.

La pertinence d'un nœud peut être calculée et combinée avec la pertinence des nœuds descendants. Pour le traitement des requêtes orientées contenu et structure le modèle peut être appliqué récursivement à chaque sous-arbre de la hiérarchie. Par la suite un agrégat des scores est effectué.

2.6.2. Modèle probabiliste

L'adaptation du modèle probabiliste au contexte de la RI structurée doit prendre en compte l'aspect structurel des documents XML. Fuhr et al. **[Fuhr et al., 2002]** ont proposé une méthode d'augmentation basée sur le modèle probabiliste. Elle a été implémentée sous le moteur de recherche « *Hyrex* » et utilise le langage de requêtes XIRQL. Dans leur modèle les nœuds feuilles ne sont pas indexés et les termes sont propagés jusqu'aux nœuds indexables les plus proches.

La pertinence d'un nœud est calculée grâce à la propagation des poids des termes dans la hiérarchie du document. Un facteur nommé « facteur d'augmentation » est utilisé pour diminuer le poids d'un terme lors de la propagation.

2.7. Exemples de systèmes développés

Dans cette section nous allons présenter deux exemples de systèmes de recherche d'information dans des documents XML : Le système TIJAH **[Mihajlovic et al., 2004]** qui appartient à l'approche orientée données, et le système XFIRM **[Sauvagnat, 2005]** qui appartient à l'approche orientée document.

2.7.1. Système utilisant l'approche orientée BD : TIJAH

2.7.1.1. Définition

TIJAH **[Mihajlovic et al., 2004] [Mihajlovic et al., 2005]** est un système de recherche d'information dans des documents XML basé sur le noyau de base de données MonetDB. Il s'agit d'un système conçu en couches (ou bien niveaux), ce qui lui donne l'air d'être un SGBD (Système de Gestion de Bases de Données) relationnel. TIJAH a été développé par une équipe composé de : Johan List, Vojkan Mihajlovic, Georgina Ramirez, Thijs Westerveld, Djoerd Hiemstra, Henk Ernst Blok, Arjen P. de Vries. Sa première participation à la campagne INEX été en 2003. Ensuite dans les campagne d'INEX 2004 et 2005 le système TIJAH a été doté d'un ensemble de techniques, parmi lesquelles : la technique de réinjection de pertinence.

2.7.1.2. Architecture du système TIJAH

Le système TIJAH est constitué des trois niveaux traditionnels d'un SGBD qui sont : les niveaux : conceptuel (voir figure 2.8), logique et physique. Les concepteurs du système TIJAH ont effectué quelques modifications sur cette architecture afin d'établir le lien entre les SGBDs et les SRIs.

a) Niveau conceptuel

Le système TIJAH utilise comme base à son niveau conceptuel le langage NEXI **[Trotman and Sigurbjörnsson, 2004]**. Les expressions NEXI sont codées sous une forme qui ressemble à la requête originale, et appelée : représentation interne. Comme résultat de ce traitement ils obtiennent une représentation conceptuelle de la requête.

b) Niveau logique

Ce niveau est basé sur l'algèbre probabiliste de région. L'idée de base des approches de l'algèbre de région est de présenter le texte sous forme d'un ensemble de zones, où chaque zone est définie par une position de début et une position de fin. L'application de cette idée sur les documents XML est très simple **[Mihajlovic et al., 2004]**.

c) Niveau physique

Le système TIJAH utilise, dans le niveau physique, le noyau MonetDB La dernière étape du niveau logique est la translation vers une requête MIL (*Monet Interpreter Language*). Les requêtes MIL sont exécutées en utilisant les primitives MIL qui définissent les manipulations sur les BAT's (*Monet Binary Tables*).

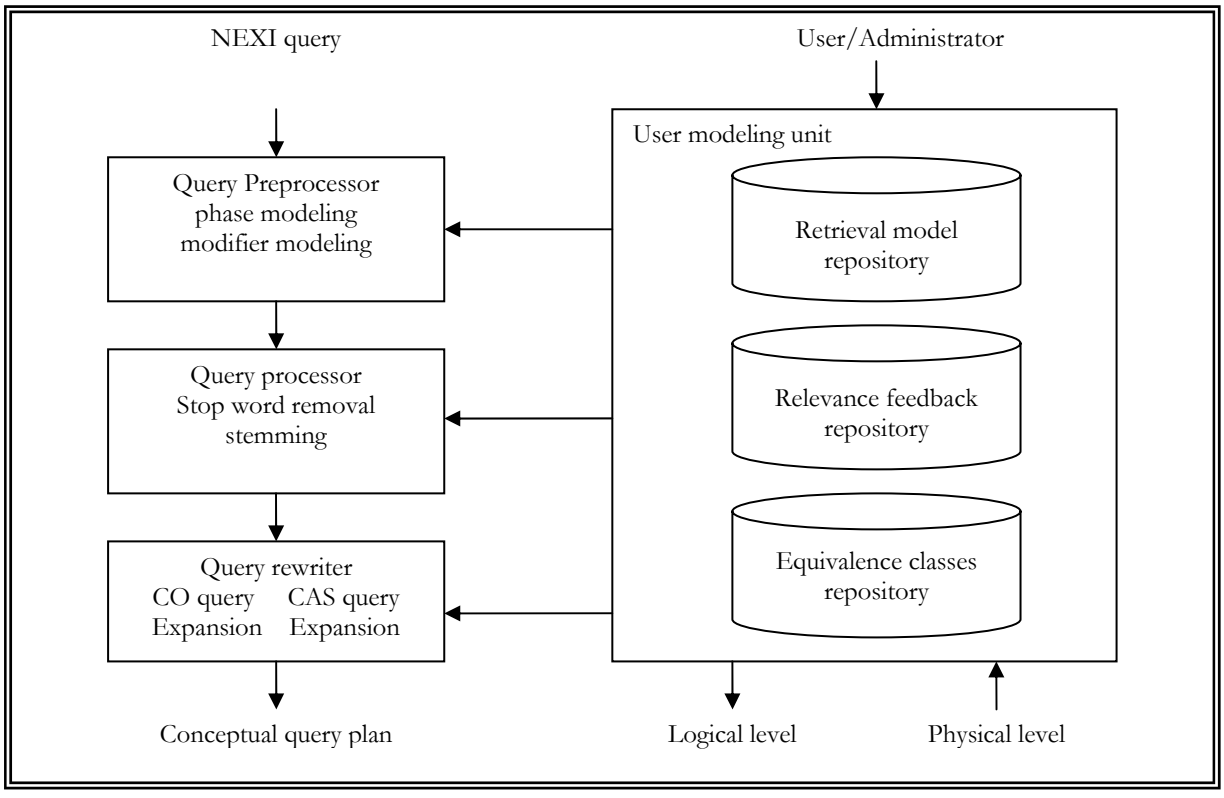


Figure 2.8 : Le niveau conceptuel du système TIJAH **[Mihajlovic et al., 2004]**

2.7.2. Système utilisant l'approche orientée RI : XFIRM

2.7.2.1. Définition

Le système XFIRM **[Sauvagnat, 2005]** repose sur le modèle XFIRM (*XML Flexible Information Retrieval Model*) qui veut dire : un modèle flexible pour la recherche d'information dans des documents XML. Ce modèle a été proposé afin de répondre à certaines limites des approches proposées dans la littérature. Il utilise une technique de propagation de la pertinence des nœuds dans l'évaluation des requêtes. XFIRM définit ainsi : un modèle de représentation des documents, un langage de requêtes et un processus d'évaluation de requêtes.

2.7.2.2. Modèle de représentation des documents

Le modèle de représentation proposé permet de naviguer dans la structure arborescente des documents XML et de représenter le contenu de cette structure. Il s'agit d'une simplification du modèle XPATH **[W3C_XPATH, 1999]**. Dans XFIRM, un document structuré ***ds*** est un arbre défini par les ensembles : N, F, A et L.

$$ds = (N, F, A, L)$$

où :

N = $\{n_1, n_2, \ldots\}$ représente l'ensemble des nœuds internes

F = $\{nf_1, nf_2, \ldots\}$ représente l'ensemble des nœuds feuilles

A = $\{a_1, a_2, \ldots\}$ représente l'ensemble des attributs

L représente l'ensemble des arcs orientés

Cette représentation permet de gérer efficacement des collections de documents hétérogènes. L'information textuelle est située au niveau des nœuds feuilles. Un nœud feuille est défini par :

$$nf_i = \{(t_1, w_1^i), (t_2, w_2^i), \ldots\} = \{(t_j, w_j^i)\}$$ où : t_j est un terme et w_j^i est son poids

Plusieurs formules de pondération peuvent être utilisées dans le système XFIRM. Dans **[Sauvagnat et Boughanem, 2006]**, nous trouvons une étude sur l'impact des différentes formules utilisées pour le calcul du poids des termes d'indexation. Ces formules sont des combinaisons des fonctions tf (*term frequency*), idf (*inverse document frequency*) et ief (*inverse element frequency*).

2.7.2.3. Langage de requêtes

Selon **[Sauvagnat, 2005]**, le langage de requêtes XFIRM se caractérise par :

- syntaxe simple qui peut être vue comme une simplification de XPath;
- formulation de requêtes à base de simples mots-clés;
- possibilité de formuler des contraintes sur la structure des documents;

- possibilité de formuler des requêtes plus complexes, en introduisant la notion de hiérarchie entre les différentes contraintes de structure;

- possibilité d'étendre les requêtes grâce à un dictionnaire des noms de balises des différents nœuds rencontrés dans le corpus.

L'expression du besoin utilisateur peut être effectuée avec le langage XFIRM selon quatre degrés de précision :

- requêtes avec de *simples mots-clés* (degré de précision P1);
- requêtes avec des *conditions sur la structure* des documents (degré de précision P2);
- requêtes avec des *conditions sur la structure* avec ajout de la notion de *hiérarchie* entre les différentes conditions de structures (degré de précision P3);
- requêtes avec spécification de l'unité d'information que l'utilisateur désire voir retournée (degré de précision P4).

2.7.2.4. Evaluation des requêtes

Il existe dans le système XFIRM deux types d'évaluation relatifs à chaque type de requêtes (requêtes orientées contenu et requêtes orientées contenu et structure).

L'évaluation des requêtes orientées contenu s'effectue en deux étapes :

- Premièrement, évaluer la similarité des nœuds feuilles de l'index vis-à-vis la requête.
- Deuxièmement, rechercher les sous arbres pertinents et informatifs. Ceci est effectué par :
  - la propagation vers le haut du score des nœuds feuilles dans l'arbre du document;
  - la propagation vers le bas du score du document dans sa globalité (pertinence contextuelle).

La fonction de similarité pour l'évaluation des requêtes orientées contenu est :

$$RSV_m(q, nf) = \sum_{i=1}^{T} w_i^q * w_i^{nf} \qquad \dots (2.3)$$

Où : w_i^q représente le poids du terme *i* dans la requête et w_i^{nf} représente le poids du terme *i* dans le nœud feuille. La pertinence $\boldsymbol{P_n}$ d'un nœud $\boldsymbol{n}$ est définie par la formule suivante :

$$P_n = \rho * \left|F_n^{\,p}\right| . \sum_{nf_k \in F_n} \alpha^{dist(n,nf_k)-1} * \beta(nf_k) * RSV(q, nf_k) + (1-\rho) * P_{racine} \qquad \dots (2.4)$$

Où : $\boldsymbol{F_n}$: ensemble des nœuds feuilles descendants de $\boldsymbol{n}$;

$\boldsymbol{F}$: ensemble des nœuds feuilles du document;

Et $\beta(nf_k)$: représente un paramètre introduisant la longueur du nœud feuille.

En ce qui concerne l'évaluation des requêtes orientées contenu et structure, un seul traitement est effectué. En effet, les requêtes de type P3 et P4 seront décomposées en requêtes de type P2 :

$P3=//P2_1//P2_2//\ldots//P2_n$

$P4=//P2_1//P2_2//\ldots//ec :P2_i//\ldots//P2_n$

Ainsi, les requêtes de types P3 et P4 seront représentées par un ensemble de sous requêtes élémentaires. L'évaluation s'effectue en trois étapes :

- traitement des sous requêtes élémentaires
- traitement des requêtes de types P2 à partir des résultats des sous requêtes élémentaires
- traitement des conditions de hiérarchie de la requête à partir des résultats des requêtes de type P2.

2.8. La campagne d'évaluation INEX

L'évaluation des SRI est une phase très importante pour comparer leurs performances. Comme TREC (*Text Retrieval Conference*) **[TREC]** pour la RI en plein texte, INEX **[INEX, 2006]** est considérée comme étant la seule campagne d'évaluation (depuis 2002 à ce jour) des SRI dans des documents XML. L'évaluation de l'efficacité des SRI dans des documents XML nécessite une collection de test (documents XML, requêtes et jugements de pertinence), cette collection est fournie aux différents participants pour arriver à effectuer un classement de leurs systèmes au niveau de la campagne.

2.8.1. Collection de test

Les collections de test en RI traditionnelle sont composées de trois parties : un ensemble de documents, un ensemble de besoins utilisateurs exprimés sous forme de requêtes (Topics) et un ensemble de jugements de pertinence représenté par une liste des documents pertinents correspondant à chaque requête. En RI structurée, la collection de test diffère sur plusieurs points de celle utilisée en RI traditionnelle. Bien qu'elle consiste toujours des mêmes parties, la nature de ces parties est fondamentalement différente. Les éléments XML constituent l'unité de recherche atomique. En plus des mots clés, les requêtes peuvent contenir des conditions de structure. En conséquence les jugements de pertinence doivent prendre en compte la nature structurelle des documents XML.

En 2005, la collection de test été composée de 16819 articles provenant des revues de la « *IEEE computer society* », d'une taille totale avoisinant les 700 Mo, et sauvegardés sous le format XML. L'ensemble d'articles contient 8 millions d'éléments représentés par 192 balises distinctes **[Lalmas, 2005]**. En plus de la collection originale, une autre collection de documents XML a été ajoutée en 2005 spécialement pour la tâche Multimédia. Cette collection est basée sur « *The Lonely Planet WorldGuide* » **[LONELY]** qui consiste en 462 documents XML **[Van Zwol et al., 2005].**

Equivalences des balises. La collection IEEE actuelle d'INEX contient plusieurs balises utilisées d'une manière équivalente (pour des raisons de publication). Les balises appartenant aux groupes suivants sont considérées comme équivalentes et peuvent être utilisées interchangeables dans une requête **[Sigurbjörnsson et al., 2005]**.

- Paragraphes: ilrj, ip1, ip2, ip3, ip4, ip5, item-none, p, p1, p2, p3;
- Sections: sec, ss1, ss2, ss3;
- Listes: dl, l1, l2, l3, l4, l5, l6, l7, l8, l9, la, lb, lc, ld, le, list, numeric-list, numeric-rbrace, bullet-list;
- Entêtes: h, h1, h1a, h2, h2a, h3, h4.

2.8.2. Requêtes (Topics)

Les requêtes sont créées en collaboration par les différents participants et elles représentent des besoins des utilisateurs en information. Dans la campagne INEX, deux grandes catégories de requêtes existent :

- Les requêtes CO (*Content Only*) : ce sont des requêtes en langage naturel, similaires à celles utilisées dans TREC. Les mots-clés de la requête peuvent être éventuellement 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 requêtes CAS (*Content And Structure*) : ces requêtes permettent d'exprimer des contraintes sur la structure des documents.

Chaque topic INEX (CO ou CAS) est caractérisée par un ensemble de champs, qu'on peut résumer dans : le champ *Title* qui donne une définition formelle de la requête, le champ *Keywords* qui contient un ensemble de mots-clés, et les champs *Description* et *Narrative*, explicités en langage naturel, indiquent les intentions de l'auteur **[Sigurbjörnsson et al., 2005]**. En créant des topics, un certain nombre de facteurs devraient être pris en compte. Ainsi, les topics devraient:

- être écrites par un expert en matière (ou quelqu'un de familier avec) dans les domaines couverts par la collection ;
- refléter des besoins réels des systèmes opérationnels ;
- être diverses ;
- être différents dans leur couverture ;
- être évaluées par l'auteur du sujet.

Le langage utilisé pour l'expression des requêtes CAS (le langage NEXI) est une variante de Xpath **[W3C_XPATH, 1999]** et est détaillé dans **[Trotman and Sigurbjörnsson, 2004]**. La formulation des requêtes est étroitement liée à la tâche de recherche associée. Nous donnons donc quelques exemples de requêtes dans la section suivante.

2.8.3. Tâches

La campagne INEX regroupe plusieurs tâches. Parmi ces tâches figure la tâche ad hoc, qui est considérée comme étant une simulation de l'utilisation d'une bibliothèque électronique (*Digital Library*). Cette bibliothèque contient un ensemble de documents sur lesquels les systèmes exécutent des requêtes utilisateurs. Les requêtes sont de multiples types, aujourd'hui, la tâche ad hoc compte les sous tâches : CO, CO+S et CAS dans ses différentes formes : SSCAS, VSCAS, SVCAS et VVCAS. Nous allons décrire chacune des sous tâches en détails. En plus de la tâche ad hoc, nous trouvons la tâche interactive, la tâche NLP (Natural Language Processing), la tâche hétérogène, la tâche multimédia, la tâche document mining, et la tâche RF (relevance feedback).

2.8.3.1. Tâche ad hoc

Sous tâche CO (*Content Only Task*). Le but de cette tâche est de répondre à des requêtes utilisateurs de type CO par des éléments/documents XML. Aucune indication sur la granularité de la réponse à renvoyer n'est exprimée, la requête CO ne contient que des mots clés. Le tableau suivant (tableau 2.6) présente une requête CO issue de la campagne INEX :

```
<inex_topic topic_id="98" query_type="CO">
<title> "Information Exchange" +"XML" "Information Integration"</title>
<description> How to use XML to solve the information exchange (information integration) problem, especially in heterogeneous data sources ? </description>
<narrative> Relevant documents/components must talk about techniques of using XML to solve information exchange (information integration) among heterogeneous data sources where the structures of participating data sources are different although they might use the same ontologies about the same content. </narrative><keywords> Information exchange, XML, information integration, heterogeneous data sources </keywords>
</inex_topic>
```

Tableau 2.6 : Exemple d'une requête CO issue de la campagne INEX

Sous tâche CO+S (*Content Only + Structure*). Ce type de requêtes tente de simuler un utilisateur qui ne connaît pas (ou ne veut pas utiliser) la structure réelle des documents XML. Ce profil est susceptible d'adapter la plupart des utilisateurs recherchant dans des bibliothèques digitales de documents XML. Lors de découvrir que sa requête CO a renvoyé beaucoup d'éléments non pertinents, l'utilisateur pourrait décider d'ajouter des contraintes de structure. C'est semblable à un utilisateur ajoutant + et – à une requête Web quand trop de pages non pertinentes sont renvoyées. À INEX, ces contraintes structurelles ajoutées (+S) sont indiquées en utilisant la syntaxe formelle de NEXI **[Sigurbjörnsson et al., 2005]**. Le tableau suivant (tableau 2.7) présente une requête CO+S issue de la campagne INEX 2005 :

```
<inex_topic query_type="CO+S">
<title>formal logic reason UML diagrams</title>
<castitle>//article[about(.//bb, Rumbaugh Jacobson Booch) and about(.//abs, formal methods)]//sec[about(.,formal logic reason UML diagrams)]</castitle>
<description>I want to know about the application of formal methods and logics to reason about UML diagrams. Relevant items probably cite Rumbaugh, Jacobson, or Booch. </description>
<narrative>My main interest is the application of formal methods and logics in software development. I choose to search for its application to UML diagrams because I think it is an interesting application area. To be relevant, a document/component must discuss the use of formal logics, such as first-order-, temporal-, or descriptionlogics, to model or reason about UML diagrams. I'm only interested in proper formal logics, Business-logics and Client-logics do not have a proof system and are therefore not considered to be formal logics. I think that sections are the most appropriate unit of retrieval for this fairly specific topic, since I'm not really interested in reading a lot about UML stuff in general. I want to focus in on the document parts that talk about logic. I think it is useful for the search engine to look for citation to the UML trio: Rumbaugh, Jacobson and Booch. Similarly think that it might be usefu to put the formal methods constraints on the abstract to stress that I'm only interested in this particular subset of UML articles. Of course a relevant article need not have this sort of reference or abstract, therefore the relevance of an element will be judged on basis of how well it explains the use of formal logics to model or reason about UML diagrams.</narrative>
```

```
</inex_topic>
```

Tableau 2.7 : Exemple d'une requête CO+S issue de la campagne INEX 2005

Sous tâche CAS (*Content And Structure*). Une requête CAS contient deux types de contraintes structurelles : où regarder (éléments de support), et quels sont les éléments à retourner (éléments cibles). Dans les ateliers antérieurs d'INEX les contraintes sur les éléments cibles ont été interprétées strictement ou vaguement, alors que les éléments de support ont toujours été interprétés vaguement **[Sigurbjörnsson et al., 2005]**. Ces ateliers ont conduit à créer une discussion sur la façon d'interpréter les éléments de support. Il y a la vue de base de données: toutes les contraintes structurelles doivent être suivies strictement (appariement exact), et la vue de la recherche d'information : un élément est considéré comme étant pertinent s'il satisfait au besoin en l'information, indépendamment des contraintes structurelles.

A INEX 2003 et 2004, il y avait deux interprétations pour les requêtes CAS qui sont : SCAS et VCAS, à partir d'INEX 2005 quatre ensembles de jugements automatiques ont été proposés **[Sigurbjörnsson et al., 2005] [Kazai and Lalmas, 2005] [Trotman and Lalmas, 2005]** :

VVCAS: Les évaluations de RI sur le topic entier ;

SVCAS: Le sous-ensemble d'évaluations de VVCAS qui satisfont strictement la contrainte d'élément cible ;

VSCAS: Les évaluations de VVCAS qui satisfont toutes les contraintes d'élément de support, indépendamment de la contrainte d'élément cible. Les opérateurs booléens entre les éléments de support sont suivis strictement ;

SSCAS: Les évaluations qui satisfont strictement toutes les contraintes d'élément de support aussi bien que la contrainte d'élément cible. Le tableau suivant (tableau 2.8) présente une requête CAS issue de la campagne INEX 2005 :

```
<inex_topic query_type="CAS">
<castitle>//article[about(.//au,"Jiawei Han")]//abs[about(.,"data mining")]</castitle>
<description> a synopsis of data mining papers by Jiawei Han</description>
<narrative>I'm writing a short article about the impact of Jiawei Han on the field of data mining. Therefore I'm interested in finding a short and concise overview of his papers. I believe this is to be found in the abstracts of his papers. To be relevant, the component has to be the abstract, written by Jiawei Han, about "data mining". Any topics of data mining (e.g. association rules, data cube etc.) should be considered as relevant.</narrative>
</inex_topic>
```

Tableau 2.8 : Exemple d'une requête CAS issue de la campagne INEX 2005

2.8.3.2. Tâche RF (Relevance Feedback)

Le but de cette tâche est d'étudier la reformulation de requêtes dans le contexte de la recherche d'information dans des documents XML. La reformulation de requêtes devrait idéalement considérer non seulement le contenu mais également la structure des documents XML.

INEX utilise la technique de collection résiduelle pour évaluer l'efficacité de la RF **[RF_TASK]**. Dans cette technique, tous les éléments XML examinés (ou employés) dans le processus de reformulation de requêtes doivent être enlevés de la collection avant que l'évaluation ait lieu. Sous des directives d'INEX, ceci signifie non seulement que chaque élément utilisé ou observé dans le processus de reformulation de requêtes mais également tous les descendants de cet élément doivent être enlevés de la collection. Alors que tous les antécédents de cet élément seront maintenus dans la collection résiduelle **[RF_TASK]**. Notre travail va se concentrer sur cette tâche, à savoir la tâche RF (reformulation de requêtes par réinjection de pertinence).

2.8.3.3. Autres tâches

Tâche NLP. Cette tâche simule un utilisateur posant sa question au SRI en langage naturel. Elle examine la capacité d'un SRI de satisfaire le besoin en information, exprimé en langage naturel. Le but des expérimentations de la tâche NLP dans INEX 2005 est de comparer la performance des techniques CO, CO+S et NLP **[Sigurbjörnsson et al., 2005]**.

Tâche Hétérogène. Cette tâche se divise en deux sous tâches « HET.CO » et « HET.CAS », le but de la première sous tâche est de rechercher des éléments dans diverses collections en se basant sur leur contenu. Alors que la deuxième doit utiliser non seulement le contenu mais également la structure, ou la structure implicite. Cette tâche comporte plusieurs (7 en 2005) collections qui possèdent des DTDs différentes **[Sigurbjörnsson et al., 2005]**, et essaye de répondre aux questions suivantes :

- Pour les requêtes CO, quelles sont les méthodes faisables pour déterminer les éléments qui peuvent être considérés comme de bonnes réponses ?
- Quelles sont les méthodes qui peuvent être utilisées pour transformer les contraintes structurelles vers d'autres DTDs ?
- La transformation doit focaliser sur les noms d'éléments seulement ou bien sur le contenu aussi ?
- Quels sont les critères d'évaluation pour les collections hétérogènes ?

Tâche Multimédia. L'objectif principal de la tâche multimédia d'INEX 2005 est de fournir une plateforme d'évaluation pour les SRI qui n'incluent pas que du texte dans le processus de recherche **[Sigurbjörnsson et al., 2005]**. Beaucoup de collections de documents structurés contiennent aujourd'hui également d'autres types de medias, tels que des images, la parole, et la vidéo. La tâche Multimédia est basée sur « *The Lonely Planet WorldGuide* » **[LONELY]** qui consiste en 462 documents XML **[Van Zwol et al., 2005]**.

2.8.4. Jugements de pertinence

Pendant les exécutions, les participants évaluent les topics et produisent une liste (ou un ensemble) d'éléments XML pour chacune des requêtes. Les tops 1500 éléments trouvés sont envoyés à la campagne INEX. Les soumissions reçues seront ensuite redistribuées aux participants (autant que possible aux auteurs du topic) pour jugement. Notant que les jugements concernant un topic doivent être effectués par une seule personne (par exemple l'auteur du topic) **[Lalmas and Piwowarski, 2005].**

Pertinence dans INEX. La pertinence est définie dans INEX par les deux dimensions suivantes **[Lalmas and Piwowarski, 2005]** :

- Exhaustivité : qui décrit jusqu'à quel point l'élément discute du sujet de la requête ;
- Spécificité : qui décrit jusqu'à quel point l'élément se focalise sur le sujet de la requête.

La campagne INEX met à la disposition des différents participants un accès au système de jugement en ligne, ce système est disponible sur le site : http://inex.lip6.fr/2005/xrai **[Kazai and Lalmas, 2005] [Lalmas and Piwowarski, 2005]**.

2.8.5. Evaluation

Pour évaluer les performances des différents systèmes de recherche dans des documents semi-structurés développés, la campagne INEX a adopté des méthodes basées sur les mesures de rappel et précision. Elle a été inspirée en grande partie du travail sur l'évaluation des SRI développé dans les expérimentations de cranfield et plus tard dans TREC **[Hiemstra, 2005]**.

2.8.6. Mesures d'évaluation

Nous donnons dans cette section un aperçu sur les mesures d'évaluation utilisées depuis INEX 2002 à INEX 2005. Nous avons déjà mentionné que la pertinence dans INEX est définie par deux dimensions : Exhaustivité et spécificité. La plupart des mesures utilisent les fonctions d'agrégation pour produire le résultat final de l'évaluation (formule 2.5).

$$F_{quant}(e,s) : ES \rightarrow [0,1] \qquad \ldots(2.5)$$

Où ES représente l'ensemble des valeurs possibles pour les paires (e,s) : ES = {(0,0), (1,1), (1,2), (1,3),(2,1), (2,2), (2,3), (3,1), (3,2), (3,3)}

Chaque élément peut être marginalement (1), assez (2) ou très (3) exhaustif ou spécifique, ou bien non pertinent (dénoté par la paire (0,0)).

2.8.6.1. La mesure INEX 2002 : inex_eval

La mesure d'INEX 2002 (appelée aussi inex_eval) calcule la mesure « *precall* », proposée par Kakade and Raghavan **[Kakade and Raghavan, 2005]**, en utilisant la probabilité qu'un élément XML vu par l'utilisateur soit pertinent (P (rel/retr)) (formule 2.6):

$$P(rel/retr)(x) = \frac{x.n}{(x.n + esl_{x.n})} \qquad \ldots(2.6)$$

Où $esl_{x.n}$ représente la longueur de recherche supposée (*expected search length*), c'est à dire le nombre d'éléments supposés non-pertinents retournés jusqu'à ce qu'un point de rappel x soit atteint **[Kazai, 2003] [Hiemstra, 2005].**

La longueur de recherche supposée est spécifiée par la formule suivante :

$$esl_{x.n} = j + \left(\frac{s.i}{r+1}\right) \qquad \ldots(2.7)$$

Où ***j*** représente le nombre total d'éléments non-pertinents dans tous les niveaux précédant le niveau final ; ***s*** est le nombre d'éléments pertinents requis dans le niveau final pour satisfaire le point de rappel ; ***i*** représente le nombre d'éléments non-pertinents dans le niveau final ; et ***r*** représente le nombre d'éléments pertinents dans le niveau final **[Hiemstra, 2005]**. L'application de cette mesure nécessite l'agrégation des deux dimensions de pertinence (E & S) pour avoir une seule valeur. Deux types de fonctions ont été utilisés:

- Une agrégation « stricte » pour évaluer si un système est capable de retrouver des éléments très spécifiques

$$f_{stricte}(s,e) = \begin{cases} 1 & Si \quad e = 3 \, et \; s = 3 \\ 0 & Sinon \end{cases} \qquad \ldots (2.8)$$

- Une agrégation « généralisée » pour évaluer les éléments selon leur degré de pertinence.

$$f_{gen}(s,e) = \begin{cases} 1 & Si \quad (e,s) = (3,3) \\ 0.75 & Si \quad (e,s) \in \{(2,3),(3,\{2,1\})\} \\ 0.5 & Si \quad (e,s) \in \{(1,3),(2,\{2,1\})\} \\ 0.25 & Si \quad (e,s) \in \{(1,2),(1,1)\} \\ 0 & Si \quad (e,s) = (0,0) \end{cases} \qquad \ldots (2.9)$$

2.8.6.2. La mesure INEX 2003 : inex_eval_ng

Les mesures d'INEX 2003 (appelées aussi inex_eval_ng) tentent de surmonter le problème causé par la mesure de 2002 qui est l'imbrication d'éléments dans les résultats en incorporant la taille des éléments et l'imbrication dans la définition du rappel et de précision. Toutefois elle ne traite pas le problème d'imbrication des éléments XML dans le résultat d'évaluation. L'imbrication est surpassée en considérant seulement l'incrémentation de la taille du texte des éléments déjà vus. Cette mesure suppose que l'information pertinente est distribuée uniformément à l'intérieur d'un élément, ce qui n'est pas prouvé dans la pratique **[Hiemstra, 2005].**

Les formules du rappel et de précision pour la mesure « inex_eval_ng » sont :

$$recall_0 = \frac{\sum_{i=1}^{k} e(c_i).\frac{|c_i'|}{|c_i|}}{\sum_{i=1}^{N} e(c_i)} \qquad \ldots (2.10)$$

$$precision_0 = \frac{\sum_{i=1}^{k} s(c_i).|c_i'|}{\sum_{i=1}^{N} |c_i'|} \qquad \ldots (2.11)$$

où $\mathbf{c_1, c_2, ...c_n}$ représentent une liste ordonnée des résultats ; **N** est le nombre total des éléments de la collection ; $\mathbf{e(c_i)}$ et $\mathbf{s(c_i)}$ dénotent respectivement les valeurs de l'exhaustivité et de spécificité attribuées à l'élément $\mathbf{c_i}$; $\mathbf{|c_i|}$ représente la taille de l'élément $\mathbf{c_i}$; et $\mathbf{|c_i'|}$ est la taille de l'élément vu précédemment par l'utilisateur, cette dernière peut être calculée comme suit :

$$\left|c_i^{'}\right| = \left| c_i - \bigcup_{c \in C[1,n-1]} (c) \right| \qquad \ldots (2.12)$$

Où n est le rang de l'élément **ci** et **c[1,n-1]** est l'ensemble des éléments à retourner entre les positions **[1,n-1]**. Des fonctions d'agrégation sont définies de telle manière qu'elles fournissent une transformation séparée de l'exhaustivité et de la spécificité :

$$f_{quant}^{'}(e) : E \to [0,1] \quad and \quad f_{quant}^{'}(s) : S \to [0,1] \qquad \ldots (2.13)$$

Pour le cas de l'agrégation stricte, la seule fonction prend comme valeurs e=3 et s=3 respectivement. Pour le cas de l'agrégation généralisée, les fonctions sont définies comme suit :

$$f'_{general}(e) = e/3 \qquad \ldots (2.14)$$

Et

$$f'_{general}(s) = s/3 \qquad \ldots (2.15)$$

Le problème de cette mesure réside dans la séparation des deux dimensions de la pertinence **[Hiemstra, 2005]**.

2.8.6.3. La mesure 2004 : fonctions orientées spécificité et orientées exhaustivité

Basée sur la discussion effectuée pendant INEX 2003 sur les fonctions d'agrégation et les inconvénients des mesures de cette campagne, deux classes de fonctions d'agrégation ont été introduites (définies).

Les fonctions orientées exhaustivité appliquées à l'agrégation stricte :

$$f_{e3\_s321}(s,e) = \begin{cases} 1 & Si \quad s \in \{3,2,1\} \; et \; e = 3 \\ 0 & Sinon \end{cases} \qquad \ldots (2.16)$$

$$f_{e3\_s32}(s,e) = \begin{cases} 1 & Si \quad s \in \{3,2\} \; et \; e = 3 \\ 0 & Sinon \end{cases} \qquad \ldots (2.17)$$

Et similairement des fonctions orientées spécificité, appliquées à l'agrégation stricte :

$$f_{s3\_e321}(s,e) = \begin{cases} 1 & Si \quad e \in \{3,2,1\} \; et \; s = 3 \\ 0 & Sinon \end{cases} \qquad \ldots (2.18)$$

$$f_{s3\_e32}(s,e) = \begin{cases} 1 & Si \quad e \in \{3,2\} \; et \; s = 3 \\ 0 & Sinon \end{cases} \qquad \ldots (2.19)$$

Cependant, cette mesure souffre du problème d'imbrication **[Hiemstra, 2005]**.

2.8.6.4. XCG: Extended Cumulated Gain

Les mesures XCG (XML Cumulative Gain) sont des extensions du gain cumulatif (CG) proposé par Järvelin et Kekäläinen **[Järvelin and Kekäläinen, 2002]**. Ce gain cumulatif peut être calculé à chaque position ***i*** selon la formule suivante :

$$CG[i] = \sum_{j=1}^{i} G[j] \qquad \ldots (2.20)$$

Dans la relation précédente le gain cumulatif à la position ***i*** est la somme des scores de pertinence de ses positions inférieures **[Hiemstra, 2005]**. La fonction qui permet de définir le score de pertinence pour un élément en utilisant la mesure XCG est :

$$rv(c_i) = f(quant(assess(c_i))) \qquad \ldots (2.21)$$

où assess($\mathbf{c_i}$) est la fonction qui permet de retourner la paire de jugement pour l'élément $\mathbf{c_i}$, et quant(assess($\mathbf{c_i}$)) représente la fonction d'agrégation.

La fonction ***f*** possède trois variantes :

- dans le cas où l'élément $\mathbf{c_i}$ n'est pas encore jugé : f(x) = x = quant(assess(c_i)) ;
- dans le cas où l'élément $\mathbf{c_i}$ est déjà vu : f(x) = (1-α)*x ;

où α représente un facteur qui simule le comportement de l'utilisateur en respectant les éléments déjà vus.

- Enfin, dans le cas où une partie de $\mathbf{c_i}$ est déjà vue :

$$f(x) = \alpha . \frac{\sum_{j=1}^{m} (rv(c_j) . |c_j|)}{|c_j|} + (1-\alpha) * x \qquad \ldots (2.22)$$

Où **m** est le nombre de nœud fils du nœud $\mathbf{c_i}$ jugés pertinents **[Hiemstra, 2005]**.

2.9. Conclusion

Dans ce chapitre, nous avons présenté les différents éléments de la RI structurée. En premier lieu, nous avons donné un aperçu sur les documents semi-structurés. Ainsi, nous avons présenté les enjeux soulevés par la RI structurée relatifs à l'aspect structurel de ces documents. Différents approches et modèles d'indexation ont été proposés dans la littérature, ainsi que des langages d'interrogation des corpus de documents XML. Ces langages permettent aux utilisateurs d'exprimer leurs besoins à travers deux types de requêtes : les requêtes orientées contenu et les requêtes orientées contenu et structure. Le deuxième type de requête suppose que l'utilisateur possède une idée sur la structure du document, donc il peut indiquer le type de l'unité d'information qu'il désire voir renvoyée. Alors que dans le premier type, c'est au système de décider de la granularité de l'information à renvoyer.

Ces dernières années, plusieurs systèmes ont été développés. Chacun de ces systèmes a sa manière de traiter les requêtes utilisateur. Ces systèmes participent chaque année dans une campagne d'évaluation qui s'appelle INEX.

L'utilisation de la technique de réinjection de pertinence dans le contexte de la RI structurée nécessite la prise en charge de la dimension structurelle en plus de la dimension textuelle. Peu de travaux ont été effectués dans ce domaine, le chapitre suivant sera consacré à l'étude des différents travaux de recherche menés pour l'application de la réinjection de pertinence à la RI structurée.

Chapitre 3 :

RF en RI structurée : état de l'art des travaux

3.1. Introduction

Dans ce chapitre nous allons donner un aperçu sur les principaux travaux menés pour l'application de la technique de réinjection de pertinence dans le contexte de la RI structurée.

En premier lieu, nous donnons la motivation et les problématiques qui ont poussé les chercheurs à investir dans ce domaine.

Les approches et méthodes proposées dans la littérature seront détaillées séparément selon chacune des approches : 1) ré-ordonnancement de la liste des résultats ; 2) expansion de requêtes.

Quelques travaux fournissent un ensemble de données concernant la collection de test utilisée dans la campagne INEX, afin d'argumenter le choix de l'information structurelle utilisée, ces données seront expliquées dans la section « statistiques » (section 3.5).

La dernière section (section 3.6) présente une discussion dans laquelle nous montrons quelques limites des approches proposées pour l'application de la technique de réinjection de pertinence dans le contexte de la RI structurée.

3.2. Motivation

Comme nous l'avons mentionné dans le chapitre 1, la requête initiale exprimée par l'utilisateur peut ne pas trouver des documents pertinents (ou peu de documents pertinents). Une méthode qui peut corriger ceci est la technique de la réinjection de pertinence. Utilisée depuis longtemps en RI traditionnelle, la réinjection de pertinence consiste à extraire de l'information à partir des documents jugés (pertinents ou bien non pertinents) par l'utilisateur, pour modifier la requête initiale, afin d'améliorer la qualité des résultats.

En RI traditionnelle, l'unité documentaire jugée et donc à partir de laquelle les termes sont extraits, est le document entier. Or, dans le contexte de la recherche d'information structurée, recherche dans les documents XML par exemple, l'unité documentaire peut avoir différentes formes. Elle peut être le document entier ou encore tout élément du document. Un élément est un sous arbre d'un document XML. La problématique posée dans le cadre de ce mémoire est : comment effectuer une reformulation de requête par réinjection de requêtes dans ce contexte? Plus précisément, la question majeure qui est posée est : comment extraire les meilleurs termes à partir d'unités d'information jugées pertinentes et non pertinentes par l'utilisateur, sachant que ces unités peuvent avoir des sémantiques différentes (ex : un paragraphe, une section, un titre), peuvent être imbriquées les unes dans les autres. La seconde question primordiale dans toute technique de reformulation est : quels poids doit-on assigner à ces

différents termes dans ces différents cas de figures? Est-il opportun par exemple d'assigner le même poids à un terme provenant d'un titre et d'une section ?

En plus de l'information textuelle, les éléments jugés par l'utilisateur contiennent aussi de l'information structurelle qui peut être exploitée dans le processus de reformulation de requêtes. Une fois que nous disposons de cette information, la question qui se pose est : Quelle est la meilleure façon pour modifier la requête initiale en utilisant l'information structurelle issue des jugements utilisateur ? (Que doit-on faire dans le cas des requêtes orientées contenu, et dans le cas des requêtes orientées contenu et structure ?).

Nous allons décrire dans ce qui suit les travaux phares de la RF dans les documents XML.

3.3. Résumé des travaux relatifs

Un des premiers travaux dans ce domaine a été celui de Mass & Mandelbrod **[Mass and Mandelbrod, 2004]**. Ces auteurs ont décrit une approche basée sur l'information feedback qui est ajoutée à leur algorithme de recherche. Cette approche détermine les types d'éléments (composants) les "plus informatifs" dans la collection (ex: dans INEX : paragraphes, sections, articles), et crée pour chaque type son index. Le processus de reformulation de requêtes se déroule comme suit : pour chaque requête les tops N éléments de la liste ordonnée sont examinés afin de choisir les éléments pertinents et ceux non pertinents. Ils utilisent deux méthodes pour la construction de la requête feedback (*Rocchio Algorithm & Lexical Affinity*). Selon **[Crouch, 2005]**, les travaux de Mass et Mandelbrod basés sur les deux approches n'ont pas montré d'améliorations par rapport à l'exécution de base (*baseline run*).

Dans INEX 2004, et ensuite dans INEX 2005, l'équipe TIJAH **[Mihajlovic et al., 2004] [Mihajlovic et al., 2005]** a proposé une approche qui permet d'exploiter l'information structurelle contenue dans les éléments examinés. En 2004, Mihajlovic et al **[Mihajlovic et al., 2004] [Ramirez et al., 2005]** ont utilisé trois types d'information structurelle : *Journal Name, XML Tag Name et Element Size*. Ce choix n'a pas donné lieu à de bonnes performances. En 2005, l'équipe TIJAH a adopté une nouvelle approche de réinjection de pertinence basée seulement sur les informations *Journal Name et XML Tag Name*. Dans **[Mihajlovic et al., 2004] [Mihajlovic et al., 2005]** on trouve les détails de l'approche proposée par l'équipe TIJAH.

En parallèle, Hlaoua & Boughanem **[Hlaoua and Boughanem, 2005]** ont proposé une nouvelle approche qui traite les deux aspects de la reformulation de requêtes dans le contexte de la RI structurée à savoir : l'information textuelle et l'information structurelle. Cette approche utilise la notion de "plus proche ancêtre commun" pour l'intégrer dans la requête CO initiale afin d'obtenir une requête de type CAS. La proposition de Hlaoua & Boughanem dans **[Hlaoua and Boughanem, 2005] [Sauvagnat et al., 2005]** est l'une des rares propositions qui permettent la modification des requêtes orientées contenu vers des requêtes orientées contenu et structure en utilisant les jugements de l'utilisateur sur la liste initiale des résultats.

En plus de ces travaux, quelques autres auteurs ont investi dans ce domaine. On peut citer par exemple : Hanglin Pan **[Hanglin, 2004]** qui utilise le calcul de similarité basé sur une ontologie pour l'application du feedback. R. Schenkel et M. Theobald **[Schenkel and Theobald, 2005]** proposent un modèle pour la génération des requêtes CAS à partir des jugements utilisateur et utilisent ce qu'ils appellent classes de candidats (*classes of candidates*). Ces travaux vont être détaillés dans les sections suivantes.

3.4. Les approches de relevance feedback

Nous pouvons classer les solutions proposées pour l'intégration de la technique de réinjection de pertinence dans le contexte de la RI structurée selon deux types d'approches :

- ré-ordonnancement de la liste des résultats;
- expansion de requête.

3.4.1. Ré-ordonnancement de la liste des résultats

Ce type d'approches consiste à extraire à partir des éléments jugés par l'utilisateur comme étant pertinents (ou non pertinents) des informations qui vont permettre de réordonner la liste des résultats en attribuant de nouveaux scores (ou priorités) aux différents éléments. Le système TIJAH propose une méthode qui permet le calcul de certaines priorités. Ces informations seront ajoutées dans la requête sous forme de poids (ou bien priorité) :

- *Journal Name, Tag Name et Element Size* pour TIJAH 2004 **[Mihajlovic et al., 2004]**;
- *Journal Name et Tag Name* pour TIJAH 2005 **[Mihajlovic et al., 2005]**;

Par exemple, si on considère la caractéristique *"Journal Name"*, les éléments appartenant aux journaux considérés par le système (après jugements de l'utilisateur) comme étant pertinents seront favorisés par rapport à ceux qui n'appartiennent pas à cet ensemble de journaux. La formule qui permet de calculer la priorité des 20 premiers journaux est définie par :

$$P(J) = a + b.\frac{\sum_{r \in top_{20} \subseteq J} E_r}{3.\left|\{\, r \in top_{20} \,|\, E_r \rangle\ 0 \,\}\right|} + (1 - a - b).\frac{\left|J \supseteq top_{20}\right|}{20} \quad \ldots (3.1)$$

Où :

Er : représente la valeur d'exhaustivité des tops 20 éléments pertinents ;

J : représente le journal pour lequel on veut calculer la priorité (voir tableau 3.8 pour la liste des journaux de la campagne INEX) ;

a et ***b*** : représentent des paramètres de pondération.

Comme nous pouvons le constater, cette formule favorise les journaux qui contiennent beaucoup d'éléments pertinents et aussi les journaux auxquels appartiennent des éléments avec des valeurs d'exhaustivité élevées. Notons que cette caractéristique est appliquée seulement à la campagne INEX.

L'autre caractéristique structurelle avec laquelle TIJAH effectue le ré-ordonnancement de la liste des résultats est « *ElementSize* ». L'équipe TIJAH suppose que les tailles des éléments pertinents sont similaires (proches). Pour cela, ils définissent une fonction « *DesiredSize* » qui permet de combiner les tailles des tops éléments retournés pour estimer la taille désirée. Cette fonction est définie par :

$$DesiredSize = \frac{\sum_{r \in top20} size(r) * SizeModifier_r}{\sum_{r \in top20} \text{sgn}(SizeModifier_r)} \quad \ldots (3.2)$$

Le paramètre « $SizeModifier_r$ » est défini par :

$$SizeModifier_r = \begin{cases} 1 & Si\ (E_r, S_r) \in \{(2,2),(3,3)\} \\ 0 & Si\ (E_r, S_r) \in \{(1,1),(0,0)\} \\ \dfrac{3 - E_r + S_r}{3} & Sinon \end{cases} \quad \ldots (3.3)$$

Où : Er, Sr représentent respectivement la valeur d'exhaustivité et de spécificité de l'élément ***r***.

Une autre formule pour l'estimation de « $SizeModifier_r$ » a été expérimentée, elle est définie ainsi par :

$$SizeModifier_r = \begin{cases} 0 & Si\ (E_r, S_r) \in \{(1,1),(0,0)\} \\ \dfrac{S_r}{E_r} & Sinon \end{cases} \quad \ldots (3.4)$$

Pour l'utilisation de la caractéristique « *TagName* » dans le système TIJAH, nous allons donner un exemple qui illustre cet aspect. Considérons la requête initiale suivante : trouver les sections ou bien les paragraphes portant sur « *XML retrieval* » :

[sec or p] [about (.,+xml +retrieval)]

En utilisant le processus de reformulation de requêtes, le système calcule les priorités à attribuer à chaque composant de la requête, par exemple il attribue la valeur 0.2 au composant structurel « sec » et 0.16 au composant structurel « p », la nouvelle requête s'écrit alors comme suit :

[MATCH(sec,0.2) or MATCH(p,0.16)] [about(.,+xml +retrieval)]

Ce qui veut dire que les éléments de type « *sec* » vont être favorisés par rapport aux éléments de type « *p* ». La formule qui permet de calculer le poids de chaque « *TagName* » est donnée par :

$$P(e) = a + b.\frac{\sum_{r \in top_{20} \subseteq e} E_r + S_r}{6.\left|\{\, r \in top_{20} \,|\, E_r.S_r \rangle\ 0 \,\}\right|} + (1 - a - b).\frac{|e \supseteq top_{20}|}{20} \quad \ldots (3.5)$$

Cependant, l'approche TIJAH ne permet de prendre en considération que le nom de la balise. Selon **[Schenkel and Theobald, 2005]** le chemin de cette balise reflète le degré de pertinence de l'élément. Par exemple, la balise « p » dans le chemin « */article/bdy/sec/p* » joue un rôle totalement différent de celle qui se trouve dans le chemin « */article/fm/cr/p* ».

L'équipe TIJAH **[Mihajlovic et al., 2004]** a présenté les résultats obtenus par leur approche basée sur la réinjection de pertinence structurelle (voir le tableau 3.1).

baseline	S1	S1+J	S1+XT	S2	S2+J	J	S1+J+XT	XT	XT+J
0.0405	0.0406	**0.0416**	0.0406						
0.0431				0.0429	0.0448	**0.0450**			
0.0456							0.0482	**0.0486**	0.0468

Tableau 3.1 : Résultats officiels des exécutions du système TIJAH dans la tâche RF **[Mihajlovic et al., 2004]**

Le tableau précédent montre l'utilisation d'un ensemble de combinaisons des trois propriétés structurelle de l'approche TIJAH **[Mihajlovic et al., 2004]** qui sont :

- S1 : propriété de la taille d'un élément selon l'équation (Voir la formule 3.3);
- S2 : propriété de la taille d'un élément selon l'équation (Voir la formule 3.4);
- J : propriété « *Journal* »;
- XT : propriété « *XML Tag Name* ».

Selon les résultats du tableau 3.1, aucune des combinaisons utilisées n'a montré une amélioration significative de la performance du système TIJAH.

Un autre système (ou bien méthode) qui utilise cette stratégie (*query re-weighting*) est celui de Hanglin **[Hanglin, 2004]**. Ce dernier exprime les requêtes en utilisant le langage XXL qui, dans sa syntaxe, donne la possibilité d'attribuer des poids. Ces poids sont calculés par le biais d'une mesure de similarité basée sur une ontologie.

Un autre exemple est celui de R. Schenkel et M. Theobald **[Schenkel and Theobald, 2005]**. Dans leur approche, ils utilisent ce qu'ils appellent classes de caractéristiques ou de propriétés (*classes of features*). A partir de la liste d'éléments trouvés pendant la première exécution (*baseline run*), ils essayent de calculer le score de chaque élément correspondant à chaque classe de caractéristiques après avoir été jugé par l'utilisateur. Cet ensemble contient :

- C-features : qui représentent tous les termes d'un élément ;
- P-features : qui représentent le chemin d'un élément ;
- D-features : qui représentent les paires ***balise-terme*** de l'élément racine du document.

Pour le calcul des poids des différentes classes de caractéristiques, les auteurs utilisent la formule standard de Rocchio. La méthode proposée par les auteurs donne la possibilité de définir d'autres classes de caractéristiques. Ces mêmes classes de caractéristiques peuvent être utilisées dans l'approche "expansion de requête".

Le tableau 3.2 résume les résultats obtenus par la première approche (Ré ordonnancement de la liste des résultats). La première colonne des résultats représente les valeurs obtenues durant l'exécution de base et les autres colonnes représentent les valeurs de la combinaison (*configurations*) des différentes classes de propriétés (*C-features, P-features, D-features*).

Les résultats de cette approche ont montré de bonnes performances du système par rapport à l'exécution de base (*baseline run*).

evaluation	baseline	C	P	C+P	D	C+D	D+P	C+D+P
plain	0.0367	0.0465	0.1008	0.0534	0.0911	0.0492	**0.1120**	0.0563
resColl-result	0.0262	0.0343	**0.0581**	0.0216	0.0412	0.0312	0.0579	0.0228
resColl-anc	0.0267	0.0340	0.0581	0.0198	0.0400	0.0297	**0.0589**	0.0219
resColl-desc	0.0330	0.0180	0.0489	0.0142	0.0284	0.0132	**0.0498**	0.0151
resColl-doc	0.0309	0.0140	**0.0480**	0.0114	0.0249	0.0097	0.0468	0.0126
freezeTop	0.0367	0.0367	0.0371	0.0353	**0.0373**	0.0369	0.0362	0.0358

Tableau 3.2 : Résultats des exécutions avec différentes configurations et méthodes d'évaluation, pour l'approche ré-ordonnancement de la liste des résultats **[Schenkel and Theobald, 2005]**

3.4.2. Expansion de requêtes

Ce type d'approches permet l'expansion de la requête initiale, soit par ajout de nouveaux termes, ou bien par ajout de nouvelles conditions de structure. Elle peut être appliquée aux deux types de requêtes (CO ou CAS). Dans cette catégorie se situent la plupart des travaux proposés dans la littérature.

3.4.2.1. Expansion des requêtes orientées contenu

Cette approche est considérée comme étant l'axe de recherche, dans le domaine de la reformulation de requêtes dans des documents XML, le plus exploré. Deux types d'expansion sont possibles :

- Requêtes CO en requêtes CO : ce qui veut dire que l'expansion se fait seulement par ajout de nouveaux termes à la requête initiale ;
- Requêtes CO en requêtes CAS : ce qui veut dire qu'en plus des termes, des conditions de structure seront ajoutées à la requête initiale.

a) Expansion des requêtes CO par ajout de nouveaux termes

Dans **[Hanglin, 2004]**, l'auteur utilise une ontologie comme base pour l'expansion d'une requête. Ceci consiste à extraire à partir de l'ontologie les termes (ou plutôt les concepts), proches de ceux de la requête et de les ajouter pour créer une nouvelle requête. L'auteur définit aussi dans son modèle un ensemble de type de conditions qui vont être utilisées dans l'expansion de la requête (C-conditions, T-conditions, P-conditions, V-conditions). Ces travaux ont été appliqués sur le système XXL **[Schenkel et al., 2005]**.

Pour l'expansion de la requête avec de nouveaux termes, Hlaoua et Boughanem **[Hlaoua and Boughanem, 2005]** utilisent une approche basée sur la formule de Rocchio, en donnant de l'importance aux termes qui se répètent souvent dans les éléments jugés pertinents. Donc, le poids des termes est proportionnel au nombre d'occurrences dans les éléments jugés pertinents.

Ces deux auteurs proposent dans **[Sauvagnat et al., 2005]** une nouvelle méthode pour la sélection des meilleurs termes à ajouter dans la requête initiale. Cette méthode consiste à attribuer des scores selon une suite de formules (formules : 3.6, 3.7, 3.8) calculant l'importance de ces

termes dans les éléments jugés pertinents. La formule suivante permet de calculer le score d'un terme au sein d'un nœud feuille. Il est défini comme suit :

$$score(t_{ij}, ln_j^k) = \frac{tf_i^{\ j}}{size(ln_j)} \qquad \ldots (3.6)$$

où $tf_i^{\ j}$ représente la fréquence du terme t_i dans le nœud feuille $\ln_j^k$

La première étape consiste à calculer pour chaque terme la somme des scores des nœuds feuilles des éléments pertinents. La formule qui permet ce calcul est définie par :

$$score\ (t_i, e_k^r) = \sum_{\ln \in e_k^r} score\ (t_i, \ln_{\ j}) \qquad \ldots (3.7)$$

Puis un score global du terme est calculé en considérant tous les éléments pertinents, en calculant la somme des scores de ce terme dans les éléments jugés pertinents. Elle est définie par :

$$score(t_i) = \sum_{e_k^r \in E^r} score(t_i, e_k^r) \qquad \ldots (3.8)$$

Les meilleurs termes choisis selon l'ordre décroissant des scores sont rajoutés à la requête.

b) Expansion des requêtes CO par ajout de conditions de structure

L'ajout de nouveaux termes à la requête peut permettre à un système de sélectionner de nouveaux éléments pertinents. Cependant, les documents XML possèdent un autre type d'informations qui peut être ajouté à la requête initiale : la contrainte structurelle.

Selon **[Schenkel and Theobald, 2005]**, une requête orientée contenu et structure spécifie avec plus de précision les conditions à satisfaire par les éléments pertinents. A partir de ce principe, plusieurs approches ont été investies pour intégrer la contrainte structurelle issue des éléments jugés pertinents dans une requête CO.

Deux approches ont été proposées dans ce contexte d'expansion. La première est celle relevant des travaux de Hlaoua et Boughanem **[Hlaoua and Boughanem, 2005] [Sauvagnat et al., 2005]** appliqués au modèle XFIRM et La seconde concerne les travaux de R. Schenkel et M. Theobald **[Schenkel and Theobald, 2005]**.

- Travaux de Hlaoua et Boughanem

Dans **[Hlaoua and Boughanem, 2005]**, les auteurs proposent une fonction pour le calcul de score pour chaque structure candidate afin de choisir celle qui a le meilleur score (voir formule 3.9). L'idée principale de l'approche proposée par Hlaoua et Boughanem est de trouver, dans la liste des éléments jugés pertinents, une structure définie comme étant « la structure générique appropriée » (*appropriate generative structure*). Intuitivement cette structure correspond à une balise fréquente dans les éléments pertinents. Nous aurons donc de forte chance que l'information pertinente se trouve dans ce type de balise. Cette structure sera ajoutée à la requête CO initiale pour produire une requête de type CAS. Pour cela, ils ont défini la notion de « plus

petit ancêtre commun » (*Smallest Common Ancestor*). Le choix de la meilleure structure SCA qui peut être ajoutée à la requête initiale se fait par l'application de la formule suivante :

$$S_{score} = \sum_{i}^{n} S_i . \alpha^{d} \qquad \ldots (3.9)$$

Avec :

Si : le score de l'élément pertinent qui a une base commune avec l'élément candidat.;

n : le nombre d'éléments jugés comme étant pertinents;

α : constante qui varie entre ***0*** et ***1***;

d : la distance qui sépare l'élément de l'élément candidat.

Les auteurs proposent dans **[Sauvagnat et al., 2005]** tout un algorithme qui permet le calcul de score des différents SC (*Common Structure*) en utilisant la fonction SCA. Après ce calcul, la structure choisie sera celle qui aura le meilleur score.

SCA(e_i,e_j)
begin
If sp_i.last = sp_j.last, then $w_i \leftarrow w_i + w_j$
If $\exists e_p(sp_p,w_p)$, with $sp_p \in SC/sp_p$.last = sp_i.last then $w_p \leftarrow w_p + w_i$
else $SC \leftarrow sp_i$
If sp_i.last $\neq$ sp_j.last, then $sp_j \leftarrow tail(sp_j)$
$w_j \leftarrow w_j/2$
SCA(e_i,e_j)
end

Avec
sp.last : la dernière balise du chemin sp
& tail(sp) la fonction qui permet d'enlever la dernière balise du chemin sp, ex :tail(/A/B/C)=/A/B/

Tableau 3.3 : Algorithme de calcul des SC (*Common Structure*) **[Sauvagnat et al., 2005]**

Le résultat de l'algorithme de calcul des SC sera, dans la plupart des cas, des éléments proches de la racine du document XML (exemple: *article*, *bdy* et *sec* pour les documents de la campagne INEX) ce qui veut dire que la dimension de spécificité de la SC choisie sera d'une valeur faible. Ceci va conduire dans certains cas à une diminution des performances si on utilise la mesure stricte pour l'évaluation des résultats du système.

Le tableau 3.4 présente les résultats obtenus (valeurs des améliorations relatives aux résultats de base obtenus par le système XFIRM) par les deux approches proposées par **[Sauvagnat et al., 2005]**. L'extension *cs* (resp. *c10s*) désigne les résultats obtenus en appliquant la RF par structure (resp. RF combinée en ajoutant les 10 premiers termes par l'algorithme de Rocchio).

	MAep strict	MAep gen	MAnxCG@1500 strict	MAnxCG@1500 gen
AR-VVCAS-RF-cs	-0.0063	-0.1810	-0.4200	-0.2914
AR-VVCAS-RF-c10s	0.0985	0.0451	0.01031	0.0319
AR-COS-RF-cs	1.5790	1.0821	0.3630	0.8394
AR-COS-RF-c10s	0.4588	0.5537	-0.1067	0.4773
AR-CO-RF-cs	-0.5391	-0.3397	-0.3540	-0.3685
AR-CO-RF-c10s	-0.7736	-0.7451	-0.8441	-0.6684

Tableau 3.4 : Résultats des deux approches proposées par **[Sauvagnat et al., 2005]** (approche orientée structure et approche orientée structure et contenu)

La reformulation combinée s'avère intéressante dans des requêtes VVCAS et COS mais pas dans le cas des requêtes CO. En effet, d'après le tableau 3.4, la valeur de AR est de l'ordre 40% pour les requêtes COS et environ 7% pour les requêtes VVCAS ce qui confirme à la fois les résultats de la reformulation des requête en RI classique et le fait que la structure générique permet d'affiner une requête initiale. Elle est négative pour les requêtes CO ce qui pousse à chercher d'autres méthodes d'extraction et/ou d'autres façons d'injection de la structure et des mots clés.

- **Travaux de R. Schenkel & M. Theobald**

Leur méthode consiste à générer une nouvelle requête CAS en utilisant un certain nombre de propriétés (caractéristiques), qu'ils appellent « classes de candidats » (*Classes of candidates*). Ces classes sont :

- C-candidates : qui représentent tous les termes d'un élément avec leurs scores ;
- D-candidates : qui représentent toutes les paires ***balise-terme*** (avec leurs scores) des descendants d'un élément ;
- A-candidates : qui représentent toutes les paires ***balise-terme*** (avec leurs scores) des ancêtres d'un élément ;
- AD-candidates : qui représentent toutes les paires ***balise-terme*** (avec leurs scores et la balise de l'ancêtre) des descendants des ancêtres d'un élément ;

Selon **[Schenkel and Theobald, 2005]**, le système peut être étendu par de nouvelles classes. La sélection des différents candidats se fait par le biais de la formule suivante :

$$w_{RSJ}(c) = \log \frac{r_c + 0.5}{R - r_c + 0.5} + \log \frac{E - ef_c - R + r_c + 0.5}{ef_c - r_c + 0.5} \qquad \ldots (3.10)$$

Où :

c : représente un candidat ;

$w_{RSJ}(c)$: poids Robertson-Sparck-Jones (RSJ) du candidat **c**

r_c : représente le nombre d'éléments pertinents que contient le candidat **c** ;

R : correspond au nombre d'éléments pertinents ;

E : représente le nombre d'éléments dans la collection ;

ef$_c$: représente la fréquence d'éléments d'un candidat.

En utilisant les tops ***b*** candidats, ils génèrent la requête CAS à partir de la requête CO initiale. La forme générale de la nouvelle requête est **[Schenkel and Theobald, 2005]**:

//ancestor_tag[A + AD constraints]//[keywords + C + D constraints]*

Comme exemple, supposons que la requête initiale était « XML », que le candidat A choisi est « *(anc,article,'IR')* », que le candidat AD choisi est *« (article,bib,'index') »* et que le candidat D choisi est « *(desc,p,'index')* ». La nouvelle requête sera:

//article[about(.,'IR') and about(//bib,'index')]//[about(.,'XML') and about(//p,'index')]*

Cette solution semble bonne (selon les résultats obtenus dans INEX) dans des conditions où l'on connaît la structure des documents de la collection. Mais dans le cas où la collection comporte des structures hétérogènes, il y aura un problème. Supposons dans l'exemple précédent que la balise « p » issue du candidat D n'appartienne pas à l'arborescence de « article ». Quoi doit-on faire dans ce cas ? On peut donc dire que cette solution est propre à INEX.

Le tableau 3.5 montre les résultats obtenus par l'application de l'approche expansion de requête. La première colonne des résultats représente les valeurs obtenues durant l'exécution de base et les autres colonnes représentent les valeurs correspondantes aux combinaisons (configurations) des différentes classes de candidats utilisées pour l'expansion de requête (A, C, D et AD).

La combinaison donnant le meilleur rapport de performance est celle qui utilise toutes les classes de candidats (A, C, D et AD). En général, les résultats ont montré une amélioration de qualité qui varie entre 5% à 20% **[Schenkel and Theobald, 2005]**.

evaluation	baseline	C	D	C+D	A	AD	A+AD	A+C+D+AD
plain	0.0367	0.0419	**0.0707**	0.0406	0.0646	0.0663	0.0654	0.0513
resColl-result	0.0262	0.0294	0.0300	0.0295	0.0309	0.0306	0.0294	**0.0324**
resColl-anc	0.0267	0.0350	0.0356	0.0305	0.0298	0.0294	0.0296	**0.0366**
resColl-desc	0.0330	0.0353	0.0355	0.0355	0.0363	0.353	0.0346	**0.0365**
resColl-doc	0.0309	0.0317	0.0313	0.0314	0.0321	0.0310	0.0315	**0.0325**
freezeTop	0.0367	0.0378	0.0374	0.0375	0.0384	0.0378	0.0380	**0.0387**

Tableau 3.5 : Résultats des exécutions avec différentes configurations et méthodes d'évaluation, pour l'approche expansion de requête **[Schenkel and Theobald, 2005]**

3.4.2.2. Expansion des requêtes orientées contenu et structure

Les seuls qui ont investi ce type d'expansion sont Hlaoua et Boughanem **[Hlaoua and Boughanem, 2005]** et Sauvagnat et al. **[Sauvagnat et al., 2005]**. Selon **[Hlaoua and Boughanem, 2005]**, l'expansion des requêtes orientées contenu et structure est une sorte de correction des conditions structurelles (contraintes de structure).

Toujours basée sur les SCA, la méthode proposée par les auteurs a aussi été appliquée pour l'expansion des requêtes orientées contenu et structure. Dans **[Hlaoua and Boughanem,**

2005], les auteurs donnent un exemple de correction d'une requête CAS (exprimée en XFIRM). La correction peut être effectuée au niveau de l'élément support de la requête après calcul de score pour chaque structure candidate.

//A[about(.,X)] // ce : B[Y]

Elément support élément cible

Dans la campagne INEX 2005, les auteurs ont changé de stratégie, en utilisant l'opérateur « OR » pour ajouter la nouvelle requête CAS à l'initiale **[Sauvagnat et al., 2005]**. Cette stratégie a été appliquée sur les tâches CO+S & VVCAS. La forme de la nouvelle requête sera donc :

Requête CAS initiale **OR** SCA choisie

Notons que les propositions des auteurs comportent le traitement de l'information textuelle (contenu) et de l'information structurelle. Dans les exécutions d'INEX 2005, les auteurs ont fixé le nombre de termes à ajouter à 15 termes. Notons aussi que la syntaxe d'un SCA peut être vue selon deux formes :

- Forme complexe (ex : /article/bdy/sec[about(.,+texte)])
- Forme simple (ex : sec[about(.,+texte)])

Pour l'expansion des requêtes CAS, les auteurs ont choisi d'utiliser la forme simple.

3.5. Quelques statistiques

Dans cette section, nous présentons quelques études effectuées sur la collection de test INEX. Ces études ont tenté de tirer profit de la nature de composition de cette collection afin d'exploiter cette information (statistiques) dans le processus de reformulation de requêtes.

3.5.1. Classification des types de topics

Selon **[Ramirez and P. de Vries]**, les requêtes d'INEX peuvent être divisées en deux classes selon leur but principal décrit dans le champ « *narrative description* » :

- informationnel : collection d'informations à propos d'un sujet (généralement une information sur un contenu spécialisé).
- ressource : recherche un type spécifique de ressource (ex : références, algorithmes, figures, …).

Voici le type d'expressions utilisé pour les requêtes de classe « informationnel » :

- « I am looking for information about .... »
- « I am interested in articles about... »

Les expressions utilisées dans la classe « ressource » peuvent être du genre :

- « Find experimental results on ... »
- « I am looking for definitions of ... »

Dans **[Ramirez and P. de Vries]**, on trouve une classification qui permet de prendre en compte le degré de connaissance de l'utilisateur sur la collection qu'il interroge. Cette classification comporte six catégories définies dans le tableau 3.6. Selon les auteurs**,** cette classification peut aider le système à choisir les meilleurs éléments qui satisfont la requête.

	Collection Familiarity		
Task type	None	Some	High
Informational	A	B	C
Resource	D	E	F

Tableau 3.6 : Classification des requêtes selon le degré de connaissance d'un utilisateur **[Ramirez and P. de Vries]**

Les requêtes d'INEX 2004 (47 de 75 requêtes) ont été classées en utilisant cette classification, le tableau résultant est :

	Collection Familiarity		
Task type	None	Some	High
Informational	A (17)	B (6)	C (3)
Resource	D (8)	E (5)	F (8)

Tableau 3.7 : Classification des requêtes INEX 2004 selon le degré de connaissance d'un utilisateur **[Ramirez and P. de Vries]**

3.5.2. Etude statistique sur l'information structurelle

Dans **[Ramirez et al., 2004]**, les auteurs présentent une étude détaillée sur quelques caractéristiques (ou bien propriétés) structurelles d'un élément :

- *Journal* : représente le nom du journal d'un élément ;
- *Element type* : représente le nom de balise d'un élément ;
- *Size* : représente la taille d'un élément.

Les résultats de cette étude vont être expliqués dans les sections suivantes.

3.5.2.1. Journal

Le contenu de la collection INEX consiste en 24 différents Journaux (18 en 2004, en 2005 le nombre était 24). Chacun de ces journaux contient un ensemble d'articles traitant un des domaines relatifs à l'informatique. Le tableau suivant résume les journaux et leurs abréviations utilisés dans la collection INEX.

L'hypothèse des auteurs de **[Ramirez et al., 2004]** peut être traduite par : « si un élément est jugé comme étant pertinent alors le « Journal » auquel appartient cet élément peut contenir d'autres éléments qui ont un contenu similaire ». Cette information peut être ensuite utilisée pour donner plus de poids aux éléments appartenant à des journaux contenant des éléments jugés pertinents.

an	IEEE Annals of the History of Computing
cg	IEEE Computer Graphics and Applications
co	Computer
cs	Computing in Science & Engineering
dt	IEEE Design & Test of Computers
ex	IEEE Intelligent Systems
ic	IEEE Internet Computing
it	IT Professional
mi	IEEE Micro
mu	IEEE Multimedia
pd	IEEE Parallel & Distributed Technology
so	IEEE Software
tc	IEEE Transactions on Computers
td	IEEE Transactions on Parallel & Distributed Systems
tg	IEEE Transactions on Visualization and Computer Graphics
tk	IEEE Transactions on Knowledge and Data Engineering
tp	IEEE Transactions on Pattern Analysis and Machine Intelligence
ts	IEEE Transactions of Software Engineering

Tableau 3.8 : La liste des journaux de la campagne INEX 2004 **[Ramirez et al., 2004]**

Le tableau 3.9 représente des statistiques sur le nombre de journaux par requête. Ceci a permis aux auteurs de conclure que l'utilisation de cette information permet d'améliorer considérablement la qualité des résultats. La figure 3.1 représente le nombre de journaux utilisés pour quelques requêtes de la campagne INEX.

Source	Avg	Median	Max	Min
Relevant (E3S3)	3.6	2	9	0
Relevant (all)	7.15	7	16	2
Results	16.65	17	18	12

Tableau 3.9 : Le nombre de journaux par requête **[Ramirez et al., 2004]**

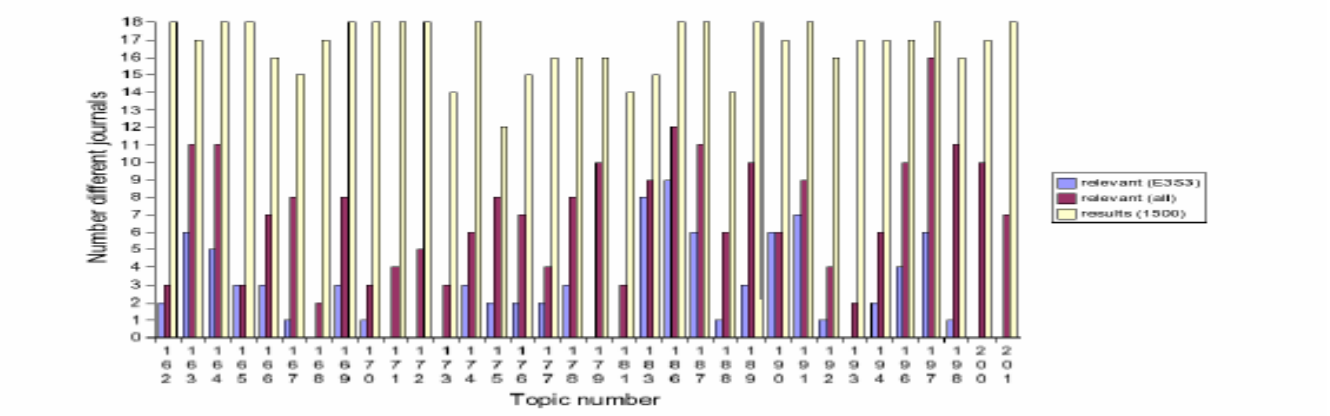


Figure 3.1 : Nombre de journaux par requête ; éléments pertinents vs. résultats **[Ramirez et al., 2004]**

Sur la figure 3.1, nous pouvons remarquer que le nombre des journaux comportant des éléments pertinents est inférieur à huit pour la plupart des requêtes (sauf quelques requêtes, exemple : 197, 186). De ce fait, et proprement à la campagne INEX nous pouvons dire que l'information structurelle « *Journal* » peut jouer un rôle important dans la sélection des bonnes sources d'information.

3.5.2.2. Element type (Type d'élément)

La collection INEX contient plus de 150 types d'éléments. Le tableau 3.10 reproduit le nombre de types d'éléments dans l'ensemble des éléments pertinents et l'ensemble des résultats (issus du système utilisé par les auteurs de **[Ramirez et al., 2004]**).

Source	Avg	Median	Max	Min
Relevant (E3S3)	8.6	4	31	0
Relevant (all)	22.32	19	60	6
Results	35.03	35	51	12

Tableau 3.10 : Le nombre de types d'éléments par requête **[Ramirez et al., 2004]**

En tenant compte des résultats obtenus dans le tableau 3.10, nous remarquons que le nombre de types de balises (ou bien d'éléments) est élevé (en moyenne 22.32 (pertinents) / 35.03 (résultats) sont pertinents sachant que la collection INEX contient plus de 150 types d'éléments) ce qui montre la difficulté de choisir une structure (balise ou chemin) permettant d'améliorer la qualité des résultats.

3.5.2.3. Size (Taille)

Les auteurs ont observé que la collection contient beaucoup d'éléments de petites tailles. Alors que la taille des éléments pertinents tend généralement vers de grandes tailles. Cependant, il existe quelques requêtes qui ne suivent pas cette tendance.

3.5.3. Relation entre propriétés structurelles et type de requête

Dans **[Ramirez and P. de Vries]**, les auteurs présentent une étude sur la relation entre quelques propriétés structurelles et le type de requête (ou bien type d'information) voulu par l'utilisateur. Les propriétés structurelles pour lesquelles ils ont effectué l'étude sont les mêmes que celles définies précédemment : « *Element size* », « *Element type* » et « *Journal information* ».

L'étude effectuée sur la propriété « *Element size* » a montré qu'il n'y a pas une grande différence entre les deux types de requêtes (à savoir informationnel et ressource). La figure 3.2 montre que les requêtes de type « informationnel » requièrent des éléments qui sont généralement de grande taille (>2000 mots) **[Ramirez and P. de Vries]**.

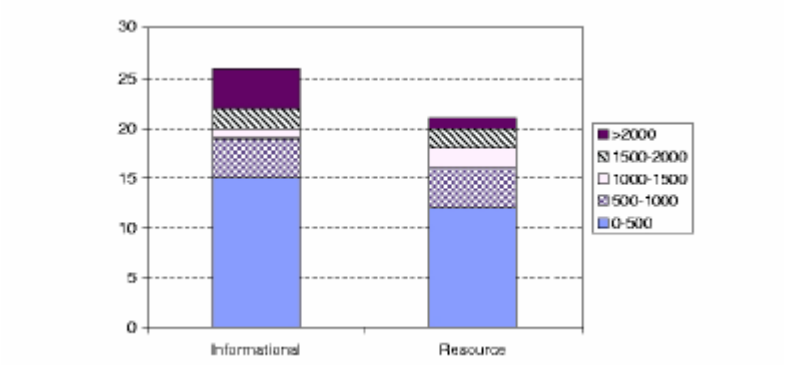


Figure 3.2 : Histogramme de la taille des éléments selon leur classe de requête **[Ramirez and P. de Vries]**

La deuxième propriété qui est « *Element type* », a aussi été étudiée, et cette étude a montré que le type « ressource » requiert des éléments plus spécifiques, comme par exemple : les paragraphes. Alors que, le type « informationnel » requiert des éléments plus génériques, comme par exemple : les sections, les bodys et les articles.

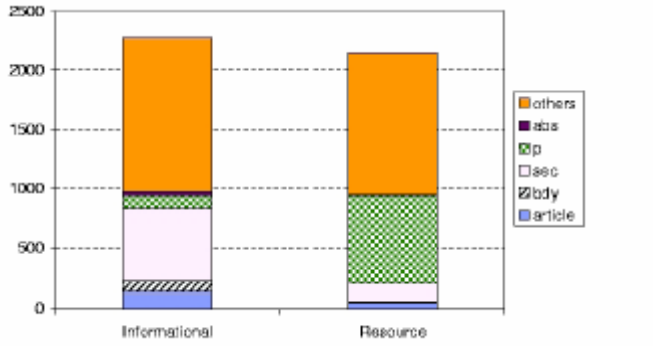


Figure 3.3 : Histogramme des types d'éléments par classe de requête **[Ramirez and P. de Vries]**

La dernière propriété étudiée est : « *Journal Information* ». Ces études ont montré que l'utilisation de cette propriété peut améliorer d'une manière significative la qualité des résultats. En général le type « ressource » ne requiert que peu de journaux dans la liste des résultats. Alors que, le type « informationnel » apparaît dans un ensemble plus large de journaux. Ce dernier type de propriété (à savoir « *Journal Information* ») est propre à la collection INEX.

3.6. Discussion et conclusion

Dans ce chapitre nous avons essayé de présenter les différentes approches et méthodes proposées dans la littérature pour l'application de la technique de réinjection de pertinence dans un contexte de RI structurée. Ces approches se divisent en deux catégories :

- Approches pour le ré-ordonnancement de la liste des résultats ;
- Approches pour l'expansion de requêtes ;

La deuxième catégorie se divise quant à elle en deux sous catégories : - les approches qui permettent l'expansion de requêtes orientées contenu et les approches qui permettent l'expansion de requêtes orientées contenu et structure.

Pour l'expansion de requêtes avec ajout de contraintes structurelles peu de travaux ont proposé des solutions, et les résultats obtenus sont généralement non convaincants. Ce qui amène à poser de nouveau la question : est ce que l'ajout d'une contrainte de structure peut améliorer la qualité des résultats (voir commentaires du tableau 3.10) ? D'abord la tâche CO+S a été définie pour ce but, c'est-à-dire étudier le comportement de la recherche en ajoutant des contraintes de structure à la requête CO initiale. La réponse à cette question nécessite une étude sur les résultats obtenus par un des systèmes pour des requêtes initiales et aussi pour des requêtes modifiées.

Notons aussi que la plupart (si ce n'est pas toutes) des propositions sont propres à la campagne INEX et leur application à des environnements de documents XML hétérogènes nécessite sûrement une révision. Beaucoup d'études statistiques, notamment celles de **[Ramirez et al., 2004]** et **[Ramirez and P. de Vries]** ont été effectuées afin de profiter des caractéristiques structurelles des documents de la campagne INEX.

Dans le cadre de notre travail, nous allons essayer de proposer des solutions permettant l'intégration de la technique de réinjection de la pertinence dans un SRI dans des documents XML, afin d'améliorer la pertinence des résultats. Ces propositions sont relatives à chacune des deux types d'approches (ré-ordonnancement de la liste des résultats et expansion de requêtes).

Chapitre 4 :

Propositions pour l'intégration de la technique de RF en RI structurée

4.1. Introduction

Nous allons décrire dans ce chapitre nos propositions sur l'intégration de la reformulation de requêtes en RI structurée. Ces propositions concernent les deux stratégies décrites dans le chapitre précédent, à savoir : la stratégie de ré-ordonnancement de la liste des résultats, et la stratégie d'expansion de requêtes.

Ce chapitre est divisé en deux grandes parties : la section (4.2) traite nos propositions pour la stratégie de ré-ordonnancement, la section (4.3) traite nos propositions pour l'expansion de requêtes.

4.2. Ré-ordonnancement de liste des résultats

Comme nous l'avons déjà mentionné (voir section 1.4.3**,** chapitre 1), l'application de la technique de réinjection de pertinence a deux effets : l'effet d'ordonnancement (*ranking effect*) et l'effet du feedback (*feedback effect*). Dans cette section nous allons nous intéresser au premier type d'effet pour lequel nous présentons deux propositions. La première appelée ré-ordonnancement contextuel et la seconde dite ré-ordonnancement par nom de journal.

4.2.1. Ré-ordonnancement contextuel

En partant de l'hypothèse suivante : « *un élément qui appartient à un document qui contient beaucoup d'éléments jugés par l'utilisateur comme étant pertinents doit être mieux classé qu'un élément se trouvant dans un document qui ne contient que peu d'éléments jugés pertinents*». Ceci peut être traduit par le fait que le concepteur d'un document XML suit une certaine unité dans ces idées.

La première approche que nous proposons consiste donc à recalculer pour chaque élément retrouvé un nouveau score de pertinence. Ce score de pertinence est fonction du score de l'élément considéré et d'un coefficient de pertinence du document (contexte) comportant cet élément. La formule (4.1) que nous proposons pour calculer le nouveau score de pertinence de chaque élément est la suivante :

$$Nouv\_Score\ (Ei) = Anc\_Score(Ei) * [\ 1 + Coef(Document(Ei))] \quad \ldots (4.1)$$

Ceci signifie que le nouveau score de pertinence d'un élément est égal à l'ancien score (de l'itération feedback précédente) multiplié par un coefficient de pertinence qui reflète l'importance du document auquel il appartient.

Où :

Ei : représente un élément qui appartient à la liste renvoyée;

Nouv_Score(Ei) : représente le nouveau score d'un élément Ei;

Anc_Score(Ei) : représente l'ancien score d'un élément Ei (nous pouvons par exemple citer la formule 2.4 du chapitre 2, c'est-à-dire celle utilisée par le système XFIRM).

Coef(Document(Ei)) : représente le coefficient de pertinence du document comportant l'élément Ei.

Document(Ei) : représente le document auquel l'élément Ei appartient

Exemple :

$$Document\ (\ co/2004/r4034/article[1]/bm[1]\ ) = co/2004/r4034$$

Plusieurs formules peuvent être envisagées pour le calcul de *Coef(Document(Ei))* :

Premièrement la formule (4.2) permet la prise en compte du nombre d'éléments pertinents ainsi que le nombre d'éléments non pertinents.

Formule 1 :

$$Coef(D) = \frac{NBR\_Elements\_Pert(D) - NBR\_Elements\_NPert(D)}{NBR\_Total\_Elements\_Juges} \quad \ldots (4.2)$$

Pour étudier l'impact de la non prise en compte du facteur « *NBR_Elements_NPert* » nous proposons une deuxième formule (4.3) qui ne tient compte que des éléments jugés pertinents.

Formule 2 :

$$\begin{cases} Coef(D) = \dfrac{NBR\_Elements\_Pert(D)}{NBR\_Total\_Elements\_Pert\_Juges} & si \quad NBR\_Total\_Elements\_Pert\_Juges > 0 \\ Coef(D) = 0 & sinon \end{cases} \quad \ldots (4.3)$$

La troisième formule (4.4) que nous proposons permet de favoriser les documents selon le pourcentage des éléments pertinents qu'ils contiennent.

Formule 3 :

$$Coef(D) = \frac{NBR\_Elements\_Pert(D)}{NBR\_Elements\_Pert(D) + NBR\_Elements\_NPert(D)} \quad \ldots (4.4)$$

Où :

NBR_Elements_Pert(D) : est le nombre d'éléments jugés pertinents appartenant au document D.

NBR_Elements_NPert(D) : est le nombre d'éléments jugés non pertinents appartenant au document D.

NBR_Total_Elements_Juges : correspond au nombre total des éléments jugés (généralement les 20 premiers éléments de la liste des résultats).

NBR_Total_Elements_Pert_Juges : représente le nombre total d'éléments jugés par l'utilisateur comme étant pertinents.

- **Exemple**

Nous allons donner dans ce qui suit un exemple pour illustrer le fonctionnement de cette méthode de ré-ordonnancement. Le tableau (4.1) représente un extrait d'un fichier XML comportant les résultats renvoyés par le système XFIRM. En combinant ces résultats avec les jugements de la campagne INEX (voir le tableau 4.9) nous pouvons appliquer notre méthode de ré-ordonnancement.

```
<?xml version="1.0"?>
<inex-submission participant-id="16" run-id="xfirm.co_th.06.09.idfief" task="CO.Thorough" query="automatic">
<description> Using XFIRM with tf-idf-ief as weighting formula, alpha=0.6, rho=09 </description>
<topic topic-id="202">
<collections>
        <collection>ieee</collection>
</collections>
        <result>
                <file>tk/2003/k0442</file>
                <path>/article[1]/bdy[1]</path>
                <rsv>33.01136</rsv>
        </result>
        <result>
                <file>ex/2003/x2026</file>
                <path>/article[1]/bdy[1]</path>
                <rsv>31.883337</rsv>
        </result>
        <result>
…
        </result>
</topic>
<topic topic-id="203">
…
</topic>
…
<inex-submission>
```

Tableau 4.1 : Un extrait du fichier XML des résultats renvoyés par le système XFIRM (la tâche CO, stratégie « *Thorough* »)

La balise *« topic »* représente la requête pour laquelle le système a renvoyé les résultats ; l'attribut *« topic_id »* représente le numéro de la requête ; la balise *« collection »* représente la collection de test utilisée par le système ; la balise *« result »* regroupe l'ensemble d'information d'un élément renvoyé ; la balise *« file »* est le nom du document auquel l'élément renvoyé appartient ; la balise *« path »* représente le chemin de l'élément renvoyé dans l'arborescence ; enfin, la balise *« rsv »* représente la valeur RSV calculée par le système de recherche.

Le tableau suivant (4.2) représente les 20 premiers éléments renvoyés par le système XFIRM pour la requête 237 de la tâche « *CO.Thourough* » :

Rang	Elément	RSV	Jugement_User
1	ex/1999/x5068/article[1]	1.7382883	+
2	tp/2003/i0550/article[1]/bm[1]	1.4279556	-
3	ex/1998/x1026/article[1]/bm[1]	1.3183781	+
4	ex/2002/x5042/article[1]/bm[1]	1.2012767	+
5	tk/2002/k0850/article[1]/bm[1]	1.0975381	+
6	ex/2001/x2060/article[1]/bm[1]	1.0917804	+
7	ex/1999/x5068/article[1]/bm[1]	0.98268116	-
8	ex/1998/x1074/article[1]/bdy[1]	0.9252556	+
9	co/2003/r5056/article[1]/bm[1]/vt[2]/p[1]	0.9216	+
10	tp/2003/i0550/article[1]	0.8924724	-
11	ex/1998/x1026/article[1]	0.8239863	+
12	ex/2002/x5042/article[1]	0.7507981	+
13	tk/2002/k0850/article[1]	0.6859613	+
14	ex/2001/x2060/article[1]	0.68236274	+
15	ex/1998/x1074/article[1]	0.6677102	+
16	tp/2003/i0550/article[1]/bm[1]/vt[2]/p[1]	0.6219593	-
17	cg/1999/g3038/article[1]	0.60953176	-
18	cg/1999/g3056/article[1]	0.6083152	+
19	tk/2002/k0850/article[1]/bm[1]/bib[1]/bibl[1]	0.5827028	+
20	cg/1999/g3056/article[1]/bdy[1]	0.5822445	+

Tableau 4.2 : Liste des résultats avec jugements utilisateur pour les 20 premiers éléments examinés de la requête 237

+ : signifie un élément pertinent

- : signifie un élément non pertinent

Notre objectif est donc de réordonner ces résultats (nombre d'éléments renvoyés est égal à 914 dont 359 pertinents) selon notre approche. Nous remarquons que le nombre d'éléments pertinents dans la liste initiale de résultats était de 15 sur les 20 premiers éléments renvoyés.

La première étape consiste à extraire la liste des documents (distincts) auxquels appartiennent les éléments jugés (20 premiers dans notre cas). La liste des documents est la suivante :{ *ex/1999/x5068, tp/2003/i0550, ex/1998/x1026, ex/2002/x5042, tk/2002/k0850, ex/2001/x2060, ex/1998/x1074, co/2003/r5056, cg/1999/g3038, cg/1999/g3056* }.

Les calculs suivants sont effectués selon la formule (4.2), c'est-à-dire la formule qui permet de prendre en compte le nombre d'éléments non pertinents dans le calcul du coefficient:

Coef (ex/1999/x5068) = (1–1) / 20 = 0 / 20 = 0
Coef (tp/2003/i0550) = (0–3) / 20 = -3 / 20 = -0.15
Coef (ex/1998/x1026) = (2–0) / 20 = 2 / 20 = 0.1
Coef (ex/2002/x5042) = (2–0) / 20 = 2 / 20 = 0.1
Coef (tk/2002/k0850) = (3–0) / 20 = 3 / 20 = 0.15
Coef (ex/2001/x2060) = (2–0) / 20 = 2 / 20 = 0.1

Coef (ex/1998/x1074) = (2–0) / 20 = 2 / 20 = 0.1
Coef (co/2003/r5056) = (1–0) / 20 = 1 / 20 = 0.05
Coef (cg/1999/g3038) = (0–1) / 20 = -1 / 20 = -0.05
Coef (cg/1999/g3056) = (2–0) / 20 = 2 / 20 = 0.1

De ce fait, nous pouvons dire que les éléments appartenant aux documents ou bien aux arborescences (par ordre décroissant d'importance) : *tk/2002/k0850, ex/1998/x1026, ex/2002/x5042, ex/2001/x2060, ex/1998/x1074, cg/1999/g3056, co/2003/r5056, ex/1999/x5068* auront plus de chance d'apparaître en tête de liste. Les éléments appartenant aux documents *tp/2003/i0550* et *cg/1999/g3038* seront moins classés par rapport à leurs positions dans la liste initiale.

Exemple de calcul de score pour quelques éléments :

- L'élément du rang 5 :

Nouv_Score (/article[1]/bm[1]) = Anc_Score(/article[1]/bm[1]) * [1 + Coef(tk/2002/k0850)] = Anc_Score(/article[1]/bm[1]) * 1.15 = 1.0975381* 1.15 = 1,262168815

Ce qui signifie que cet élément aura un score qui va augmenter de 15%. Après calcul, l'élément aura la position 4 dans la nouvelle liste.

- L'élément du rang 17 :

Nouv_Score (/article[1]) = Anc_Score(/article[1]) * [1 + Coef(cg/1999/g3038)] = Anc_Score(/article[1]) * 0.95 = 0.60953176 * 0.95 =0,579055172

Ce qui signifie que cet élément aura un score qui va diminuer de 5 %. Après calcul, l'élément aura la position 20 dans la nouvelle liste.

Voici le nouvel ordre (tableau 4.3) de la liste renvoyée après la première itération de l'application de la formule (4.2):

Rang	Elément	RSV	Jugement_User
1	ex/1999/x5068 /article[1]	1,7382883	+
2	ex/1998/x1026 /article[1]/bm[1]	1,45021591	+
3	ex/2002/x5042 /article[1]/bm[1]	1,32140437	+
4	tk/2002/k0850 /article[1]/bm[1]	1,262168815	+
5	tp/2003/i0550 /article[1]/bm[1]	1,21376226	-
6	ex/2001/x2060 /article[1]/bm[1]	1,20095844	+
7	ex/1998/x1074 /article[1]/bdy[1]	1,01778116	+
8	ex/1999/x5068 /article[1]/bm[1]	0,98268116	-
9	co/2003/r5056 /article[1]/bm[1]/vt[2]/p[1]	0,96768	+
10	ex/1998/x1026 /article[1]	0,90638493	+
11	ex/2002/x5042 /article[1]	0,82587791	+
12	tk/2002/k0850 /article[1]	0,788855495	+
13	tp/2003/i0550 /article[1]	0,75860154	-
14	ex/2001/x2060 /article[1]	0,750599014	+
15	ex/1998/x1074 /article[1]	0,73448122	+

16	tk/2002/k0850 /article[1]/bm[1]/bib[1]/bibl[1]	0,67010822	+
17	cg/1999/g3056 /article[1]	0,66914672	+
18	cg/1999/g3056 /article[1]/bdy[1]	0,64046895	+
19	co/2003/r5056 /article[1]/bm[1]/vt[2]	0,58968	+
20	cg/1999/g3038 /article[1]	0,579055172	-

Tableau 4.3 : Ordre de la liste du tableau (4.2) après application du ré-ordonnancement contextuel (formule 4.2) pour la requête 237 de la tâche « *CO.Thorough* »

Nous remarquons, du tableau 4.3, que le nombre d'éléments pertinents dans la nouvelle liste est passé de 15 à 16 sur les 20 premiers éléments, et que les quatre éléments top classés sont devenus pertinents.

Si nous appliquons la formule (4.3) à la même liste (tableau 4.2), nous aurons, après la première itération feedback, les résultats présentés dans le tableau (4.4). Les calculs suivants sont effectués selon cette formule :

Coef (ex/1999/x5068) = 1/ 15 = 0.06
Coef (tp/2003/i0550) = 0/ 15 = 0
Coef (ex/1998/x1026) = 2/ 15 = 0.13
Coef (ex/2002/x5042) = 2/ 15 = 0.13
Coef (tk/2002/k0850) = 3/ 15 = 0.2
Coef (ex/2001/x2060) = 2/ 15 = 0.13
Coef (ex/1998/x1074) = 2/ 15 = 0.13
Coef (co/2003/r5056) = 1/ 15 = 0.06
Coef (cg/1999/g3038) = 0/ 15 = 0
Coef (cg/1999/g3056) = 2/ 15 = 0.13

De ces calculs, nous pouvons dire que les éléments appartenant au document : *tk/2002/k0850* auront plus de chance d'apparaître en tête de liste. Les éléments appartenant aux documents *tp/2003/i0550* et *cg/1999/g3038* seront moins classés par rapport à leurs positions dans la liste initiale.

Rang	Elément	RSV	Jugement_User
1	ex/1999/x5068 /article[1]	1,85417418666	+
2	ex/1998/x1026 /article[1]/bm[1]	1,49416184666	+
3	tp/2003/i0550 /article[1]/bm[1]	1,4279556	-
4	ex/2002/x5042 /article[1]/bm[1]	1,36144692666	+
5	tk/2002/k0850 /article[1]/bm[1]	1,31704572	+
6	ex/2001/x2060 /article[1]/bm[1]	1,23735112	+
7	ex/1998/x1074 /article[1]/bdy[1]	1,04862301333	+
8	ex/1999/x5068 /article[1]/bm[1]	1,04819323733	-
9	co/2003/r5056 /article[1]/bm[1]/vt[2]/p[1]	0,98304	+
10	ex/1998/x1026 /article[1]	0,93385114	+
11	tp/2003/i0550 /article[1]	0,8924724	-
12	ex/2002/x5042 /article[1]	0,85090451333	+
13	tk/2002/k0850 /article[1]	0,82315356	+
14	ex/2001/x2060 /article[1]	0,77334443866	+

15	ex/1998/x1074 /article[1]	0,75673822666	+
16	tk/2002/k0850 /article[1]/bm[1]/bib[1]/bibl[1]	0,69924336	+
17	cg/1999/g3056 /article[1]	0,68942389333	+
18	cg/1999/g3056 /article[1]/bdy[1]	0,6598771	+
19	tp/2003/i0550/article[1]/bm[1]/vt[2]/p[1]	0,6219593	-
20	cg/1999/g3038 /article[1]	0,60953176	-

Tableau 4.4 : Ordre de la liste du tableau (4.2) après application du ré-ordonnancement contextuel (formule 4.3) pour la requête 237 de la tâche « *CO.Thorough* »

L'application de la formule (4.4) sur la même liste du tableau (4.2) a permis d'obtenir les résultats montrés dans le tableau (4.5). Les calculs des coefficients sont les suivants :

Coef (ex/1999/x5068) = 1 / (1+1) = 0.5
Coef (tp/2003/i0550) = 0 / (0+3) = 0
Coef (ex/1998/x1026) = 2 / (2+0) = 1
Coef (ex/2002/x5042) = 2 / (2+0) = 1
Coef (tk/2002/k0850) = 3 / (3+0) = 1
Coef (ex/2001/x2060) = 2 / (2+0) = 1
Coef (ex/1998/x1074) = 2 / (2+0) = 1
Coef (co/2003/r5056) = 1 / (1+0) = 1
Coef (cg/1999/g3038) =0 / (0+1) = 0
Coef (cg/1999/g3056) = 2 / (2+0) = 1

Selon le calcul des coefficients, les éléments appartenant aux documents : *ex/1998/x1026, ex/2002/x5042, ex/2002/x5042, tk/2002/k0850, ex/2001/x2060, ex/1998/x1074, co/2003/r5056, cg/1999/g3056* auront plus de chance d'apparaître en tête de liste. Les éléments appartenant aux documents : *ex/1999/x5068, tp/2003/i0550* et *cg/1999/g3038* seront moins classés par rapport à leurs positions dans la liste initiale.

Rang	Elément	RSV	Jugement_User
1	ex/1998/x1026 /article[1]/bm[1]	2,6367562	+
2	ex/1999/x5068 /article[1]	2,60743245	+
3	ex/2002/x5042 /article[1]/bm[1]	2,4025534	+
4	tk/2002/k0850 /article[1]/bm[1]	2,1950762	+
5	ex/2001/x2060 /article[1]/bm[1]	2,1835608	+
6	ex/1998/x1074 /article[1]/bdy[1]	1,8505112	+
7	co/2003/r5056 /article[1]/bm[1]/vt[2]/p[1]	1,8432	+
8	ex/1998/x1026 /article[1]	1,6479726	+
9	ex/2002/x5042 /article[1]	1,5015962	+
10	ex/1999/x5068 /article[1]/bm[1]	1,4279556	-
11	tp/2003/i0550 /article[1]/bm[1]	1,47402174	-
12	tk/2002/k0850 /article[1]	1,3719226	+
13	ex/2001/x2060 /article[1]	1,36472548	+
14	ex/1998/x1074 /article[1]	1,3354204	+
15	cg/1999/g3056 /article[1]	1,2166304	+
16	tk/2002/k0850 /article[1]/bm[1]/bib[1]/bibl[1]	1,1654056	+

17	cg/1999/g3056 /article[1]/bdy[1]	1,164489	+
18	co/2003/r5056 /article[1]/bm[1]/vt[2]	1,1232	+
19	ex/2001/x2060 /article[1]/bm[1]/vt[1]/p[1]	1,0349038	+
20	cg/1999/g3056 /article[1]/bdy[1]/sec[1]/p[4]	0,99068474	+

Tableau 4.5 : Ordre de la liste du tableau (4.2) après application du ré-ordonnancement contextuel (formule 4.4) pour la requête 237 de la tâche « *CO.Thorough* »

Nous pouvons remarquer à partir du tableau (4.5) que le nombre d'éléments pertinents dans la nouvelle liste a augmenté de 15 à 18 sur les 20 premiers éléments, et que les neuf éléments top classés sont pertinents. L'exécution du feedback (pour la 4ème itération) en appliquant la formule (4.4), nous a permis de constater que les vingt premiers éléments sont devenus pertinents. Des expérimentations plus détaillées sur un ensemble de requêtes sont décrites dans le chapitre suivant.

4.2.2. Ré- ordonnancement par nom de journal

La collection de documents de la campagne INEX comprenait 24 différents journaux (en 2005), ces derniers correspondent aux revues d'IEEE Computer Society. Les noms de journaux peuvent être considérés comme une caractéristique structurelle spécifique à la campagne INEX.

L'hypothèse que nous considérons dans cette approche est proche des travaux de **[Mihajlovic et al., 2005]**. En effet, les études effectuées par les auteurs ont montré que : « *si un élément est jugé comme étant pertinent alors le « Journal » auquel il appartient peut contenir d'autres éléments qui ont un contenu similaire* ». Cette information peut être ensuite utilisée pour donner plus de poids aux journaux contenant des d'éléments jugés pertinents.

L'approche que nous proposons est une adaptation de notre approche précédente (ré-ordonnancement contextuel) mais en se basant cette fois-ci sur le nom de journal auquel appartient un élément. Ceci revient donc à calculer un nouveau score de pertinence de chaque élément en fonction d'un coefficient de pertinence du journal. Notons que cette proposition est propre à la campagne INEX, c'est-à-dire qu'elle ne peut être utilisée qu'avec la collection d'INEX. La formule que nous proposons pour calculer le nouveau score de pertinence de chaque élément est la suivante :

$$Nouv\_Score\ (Ei) = Anc\_Score(Ei) * [\ 1 + Coef(Journal(Ei))] \qquad \ldots(4.5)$$

La formule (4.5) signifie que le nouveau score de pertinence d'un élément est égal à l'ancien score (itération feedback précédente) multiplié par un coefficient de pertinence du journal auquel appartient l'élément.

Où :

Ei : représente un élément qui appartient à la liste renvoyée;

Nouv_Score(Ei) : est le nouveau score d'un élément Ei;

Anc_Score(Ei) : correspond à l'ancien score d'un élément Ei (par exemple nous citons la formule (2.4) du chapitre 2, c'est-à-dire celle utilisée par le système XFIRM);

Coef(Journal (Ei)) : représente le coefficient de pertinence du « Journal » comportant l'élément Ei;

avec : *Journal(Ei)* : représente le nom du « Journal » auquel l'élément *Ei* appartient.

Exemple:

$$Journal(\ co/2004/r4034/article[1]/bm[1]\ ) = co \qquad \text{(co : Computer)}$$

De la même manière que le coefficient de pertinence d'un document; plusieurs formules peuvent être envisagées pour le calcul de *Coef(Journal(Ei))* : Nous proposons tout d'abord une formule (4.6) qui permet la prise en compte du nombre d'éléments pertinents ainsi que le nombre d'éléments non pertinents. La deuxième formule (4.7) proposée permet d'étudier l'impact de la non prise en compte du facteur « *NBR_Elements_NPert* ». En fin, la formule (4.8) permet de favoriser les journaux selon le pourcentage des éléments pertinents qu'ils contiennent.

Formule 1 :

$$Coef(J) = \frac{NBR\_Elements\_Pert(J) - NBR\_Elements\_NPert(J)}{NBR\_Total\_Elements\_Juges} \qquad \ldots (4.6)$$

Formule 2 :

$$\begin{cases} Coef(J) = \dfrac{NBR\_Elements\_Pert(J)}{NBR\_Total\_Elements\_Pert\_Juges} & si \quad NBR\_Total\_Elements\_Pert\_Juges > 0 \\ Coef(J) = 0 & sinon \end{cases} \qquad \ldots (4.7)$$

Formule 3 :

$$Coef(J) = \frac{NBR\_Elements\_Pert(J)}{NBR\_Elements\_Pert(J) + NBR\_Elements\_NPert(J)} \qquad \ldots (4.8)$$

Où :

NBR_Elements_Pert(J) : représente le nombre d'éléments jugés pertinents appartenant au journal *"J"*.

NBR_Elements_NPert(J) : est le nombre d'éléments jugés non pertinents appartenant au journal *"J"*.

NBR_Total_Elements_Juges : correspond au nombre total des éléments jugés.

NBR_Total_Elements_Pert_Juges : représente le nombre total d'éléments jugés comme étant pertinents.

- Exemple

Prenons, à titre d'exemple, les résultats du tableau (4.2). La première étape consiste à extraire la liste des journaux (distincts) auxquels appartiennent les éléments jugés. La liste des

journaux est la suivante : {*ex, tp ,tk, co, cg*}. Les calculs suivants sont effectués selon la formule (4.6) :

Coef (ex) = (9–1) / 20 = 8 / 20 = 0.40
Coef (tp) = (0–3) / 20 = -3 / 20 = -0.15
Coef (tk) = (3–0) / 20 = 3 / 20 = 0.15
Coef (co) = (1–0) / 20 = 1 / 20 = 0.05
Coef (cg) = (2–1) / 20 = 1 / 20 = 0.05

Nous pouvons dire que les éléments appartenant aux journaux (par ordre décroissant d'importance) : « *ex* », « *tk* », « *co* », « *cg* » auront plus de chance d'apparaître en tête de liste. Les éléments appartenant à « *tp* » seront moins classés par rapport à leurs positions dans la liste initiale (Voir tableau 4.6).

Exemple de calcul de score pour quelques éléments :

- L'élément du rang 11 (voir tableau 4.2):

Nouv_Score (article[1])= Anc_Score(article[1]) * [1 + Coef(ex)] = Anc_Score(article[1]) * 1.40 = 0.8239863 * 1.40 = 1,15358082

Ce qui signifie que cet élément aura un score qui va augmenter de 40%.

- L'élément du rang 2 (voir tableau 4.2):

Nouv_Score (article[1]/bm[1]) = Anc_Score(article[1]/bm[1]) * [1 + Coef(tp)] = Anc_Score(article[1]/bm[1]) * 0.85 = 1.4279556 * 0.85 = 1,21376226

Ce qui signifie que cet élément aura un score qui va diminuer de 15 %.

Dans le tableau ci-dessous (tableau 4.6) est présenté le nouvel ordre de la liste renvoyée après la première itération de l'application de la formule (4.6):

Rang	Elément	RSV	Jugement_User
1	ex/1999/x5068 /article[1]	2,43360362	+
2	ex/1998/x1026 /article[1]/bm[1]	1,84572934	+
3	ex/2002/x5042 /article[1]/bm[1]	1,68178738	+
4	ex/2001/x2060 /article[1]/bm[1]	1,52849256	+
5	ex/1999/x5068 /article[1]/bm[1]	1,375753624	-
6	ex/1998/x1074 /article[1]/bdy[1]	1,29535784	+
7	tk/2002/k0850 /article[1]/bm[1]	1,262168815	+
8	tp/2003/i0550 /article[1]/bm[1]	1,21376226	-
9	ex/1998/x1026 /article[1]	1,15358082	+
10	ex/2002/x5042 /article[1]	1,05111734	+
11	co/2003/r5056 /article[1]/bm[1]/vt[2]/p[1]	0,96768	+
12	ex/2001/x2060 /article[1]	0,955307836	+
13	ex/1998/x1074 /article[1]	0,93479428	+
14	tk/2002/k0850 /article[1]	0,788855495	+
15	tp/2003/i0550 /article[1]	0,75860154	-

16	ex/1999/x5068 /article[1]/fm[1]	0,7350133	-
17	ex/2001/x2060 /article[1]/bm[1]/vt[1]/p[1]	0,72443266	+
18	ex/1999/x5068 /article[1]/bm[1]/vt[1]/p[1]	0,71048418	-
19	tk/2002/k0850 /article[1]/bm[1]/bib[1]/bibl[1]	0,67010822	+
20	cg/1999/g3038 /article[1]	0,60953176	-

Tableau 4.6 : Ordre de la liste du tableau (4.2) après application du ré-ordonnancement par nom de journal (formule 4.6) pour la requête 237 de la tâche « *CO.Thorough* »

Les résultats de l'application de la formule (4.7), c'est-à-dire celle qui ne prend en compte que les éléments pertinents sont montrés dans le tableau suivant :

Rang	Elément	RSV	Jugement_User
1	ex/1999/x5068 /article[1]	2,78126128	+
2	ex/1998/x1026 /article[1]/bm[1]	2,10940496	+
3	ex/2002/x5042 /article[1]/bm[1]	1,92204272	+
4	ex/2001/x2060 /article[1]/bm[1]	1,74684864	+
5	ex/1999/x5068 /article[1]/bm[1]	1,572289856	-
6	ex/1998/x1074 /article[1]/bdy[1]	1,48040896	+
7	tp/2003/i0550 /article[1]/bm[1]	1,4279556	-
8	ex/1998/x1026 /article[1]	1,31837808	+
9	tk/2002/k0850 /article[1]/bm[1]	1,31704572	+
10	ex/2002/x5042 /article[1]	1,20127696	+
11	ex/2001/x2060 /article[1]	1,091780384	+
12	ex/1998/x1074 /article[1]	1,06833632	+
13	co/2003/r5056 /article[1]/bm[1]/vt[2]/p[1]	0,98304	+
14	tp/2003/i0550 /article[1]	0,8924724	-
15	ex/1999/x5068 /article[1]/fm[1]	0,8400152	-
16	ex/2001/x2060 /article[1]/bm[1]/vt[1]/p[1]	0,82792304	+
17	tk/2002/k0850 /article[1]	0,82315356	+
18	ex/1999/x5068 /article[1]/bm[1]/vt[1]/p[1]	0,81198192	-
19	tk/2002/k0850 /article[1]/bm[1]/bib[1]/bibl[1]	0,69924336	+
20	cg/1999/g3038 /article[1]	0,69080266133	-

Tableau 4.7 : Ordre de la liste du tableau (4.2) après application du ré-ordonnancement par nom de journal (formule 4.7) pour la requête 237 de la tâche « *CO.Thorough* »

De façon générale, nous remarquons que les résultats obtenus par le ré-ordonnancement contextuel sont meilleurs que ceux obtenus par le ré-ordonnancement par nom de journal pour notre exemple (requête 237). Pour confirmer cette constatation nous devons procéder à une expérimentation portant sur plusieurs requêtes (voir les résultats du chapitre 5).

4.3. Expansion de requêtes

La seconde approche que nous proposons repose sur le deuxième effet de l'application de la technique de réinjection de pertinence, à savoir « *feedback effect* ». Ceci peut être effectué par l'utilisation de l'information textuelle ou structurelle issue des jugements de l'utilisateur. Dans nos propositions, nous nous focalisons sur les requêtes de type CO.

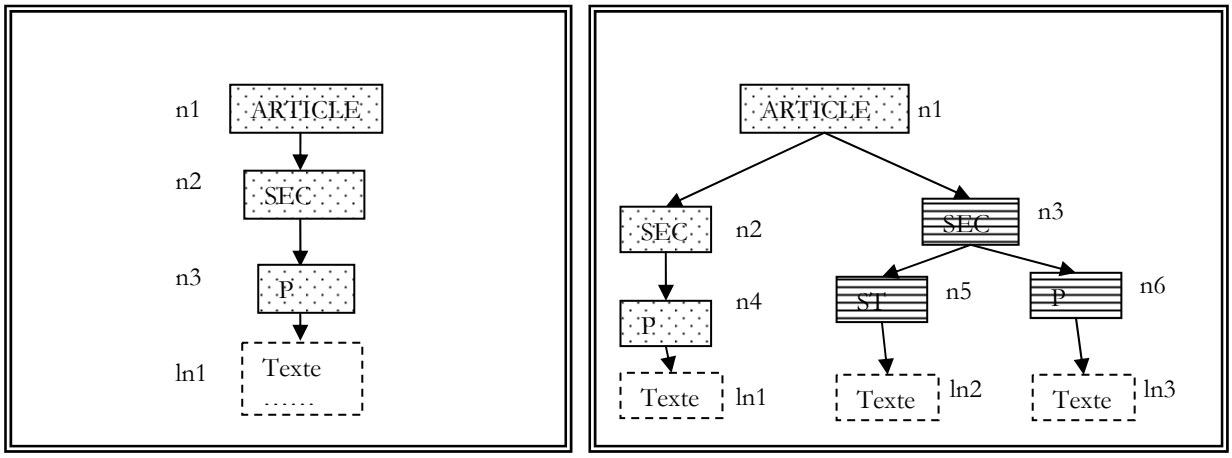
Dans cette partie nous allons décrire nos propositions concernant l'expansion de requêtes. Ces propositions ont été divisées selon le type d'information utilisée pour l'expansion. Nous trouvons donc l'expansion de requêtes par ajout de nouveaux termes et l'expansion de requêtes par ajout de contraintes structurelles.

4.3.1. Expansion par ajout de nouveaux termes

Le but de l'approche que nous décrivons dans cette section est de définir une façon de choisir les meilleurs termes à rajouter dans la requête initiale.

Notre approche rentre dans la même lignée que celle proposée par **[Hlaoua and Boughanem, 2005]**. Notre objectif est de pallier deux inconvénients de l'approche proposée par ces auteurs, en l'occurrence, le problème d'imbrication des éléments renvoyés pour jugements et la faiblesse d'inclusion des nœuds non pertinents dans le choix des termes à rajouter.

Afin d'illustrer ces propos nous donnons un exemple pour chacun de ces deux inconvénients. La figure suivante (figure 4.1) montre deux arbres jugés par l'utilisateur. Le premier arbre (arbre de la figure 4.1.a) contient trois éléments imbriqués *n1, n2 et n3* qui sont jugés pertinents. Le deuxième arbre (arbre de la figure 4.1.b) contient trois éléments imbriqués jugés pertinents *n1, n2 et n4* et trois d'autres éléments jugés non pertinents *n3, n5 et n6*.



: (a) : Arbre N°1 (b) : Arbre N°2

Figure 4.1 : Exemples d'arborescence XML

Les éléments avec motif par points représentent les éléments jugés pertinents par l'utilisateur, alors que les éléments avec motif par lignes représentent les éléments jugés non pertinents. Prenons l'arbre de la figure 4.1.a. Selon les formules proposées dans **[Hlaoua and Boughanem, 2005]** le score d'un terme ***t1*** dans l'élément n1 est égal à Score(t1,ln1); le score d'un

terme ***t1*** dans l'élément n2 est égal à Score(t1,ln1); ainsi, le score d'un terme ***t1*** dans l'élément n3 est égal à Score(t1,ln1). De cela, nous pouvons dire que le score final du terme ***t1*** sera : Score(t1) = Score(t1,n1) + Score(t1,n2) + Score(t1,n3) = 3 * Score(t1,ln1). Alors que ce terme (c'est-à-dire ***t1***) ne doit être considéré qu'une seule fois dans le nœud feuille ln1. Ce problème, que nous appelons "*problème d'imbrication*", n'est pas résolu par la solution de Hlaoua et Boughanem.

Prenons l'arbre de la figure 4.1.b. Si nous effectuons les calculs selon la méthode proposée par Hlaoua et Boughanem, on aura : n4 = {ln1}, n2 = {ln1}, n1 = {ln1, ln2, ln3}. Supposons qu'on veuille calculer le score pour un terme ***t1***: Score(t1,n4) = Score(t1,ln1); Score(t1,n2) = Score(t1,ln1); Score(t1,n1) = Score(t1,ln1) + Score(t1,ln2) + Score(t1,ln3). Ce calcul donne plus d'importance à un terme ***t1*** existant dans les éléments (*n3, n5 ou n6*) qui sont supposés être non pertinents. Donc, nous devons faire attention aux termes issus des éléments non pertinents qui sont dans l'arborescence d'un élément pertinent (exemple : dans la figure 4.1.b **n3** est non pertinent mais il appartient à l'arbre de l'élément **n1** qui est pertinent).

Notre méthode focalise sur les nœuds feuilles parce qu'ils contiennent l'information textuelle. Elle consiste à définir (ou bien extraire), à partir des éléments jugés par l'utilisateur, deux ensembles de nœuds feuilles, un premier ensemble appelé *"ensemble pertinent des nœuds feuilles (noté EP)"* et un second ensemble appelé *"ensemble des nœuds feuilles appartenant à des éléments jugés non pertinents (noté ENP)"*. De telle sorte que l'intersection des deux ensembles sera égale à l'ensemble vide. Ainsi, un nœud feuille appartenant à l'ensemble *EP* ne doit jamais figurer dans l'ensemble *ENP*. A partir de ces deux ensembles nous pouvons calculer le poids de chaque terme afin de choisir les meilleurs qui vont être utilisés dans le processus de reformulation de requêtes

Pour cela, nous allons définir les éléments qui vont intervenir dans le processus de choix des termes. Comme nous venons de le signaler les éléments jugés par l'utilisateur sont divisés en deux ensembles : ensemble des éléments jugés pertinents (appelé : ensemble P) et ensemble des éléments jugés non pertinents (appelé : ensemble NP).

P_k : élément jugé pertinent, tel que $P_k \in P$

NP_h : élément jugé non pertinent, tel que $NP_h \in NP$

$$P_k = \{nf_1^k, ..., nf_n^k\},\ NP_h = \{nf_1^h, ..., nf_m^h\}$$

Où : nf représente les nœuds feuilles ayant des scores positifs (c'est-à-dire supérieur à 0) pour un des termes de la requête.

Nous définissons un ensemble L qui représente tous les nœuds feuilles des éléments jugés pertinents (ayant des scores positifs pour un des termes de la requête). L est l'*union* des nœuds feuilles des éléments jugés pertinents (P_k), tel que : $L = \bigcup P_k$.

Les formules que nous proposons sont basées sur le principe de la formule de Rocchio **[Rocchio, 1971]** : La première formule (4.9) consiste à utiliser les termes issus des éléments jugés pertinents ainsi que ceux jugés non pertinents.

$$poids\_final(t_i) = poids(t_i, EP) - poids(t_i, ENP) \qquad \ldots (4.9)$$

La deuxième formule (4.10) consiste à utiliser les termes issus seulement des éléments jugés pertinents.

$$poids\_final(t_i) = poids(t_i, EP) \quad \ldots (4.10)$$

où :

EP : représente l'ensemble pertinent des nœuds feuilles

ENP : représente l'ensemble des nœuds feuilles appartenant à des éléments jugés non pertinents.

Pour le calcul du poids d'un terme au sein d'un nœud feuille, nous utilisons la même formule proposée dans **[Sauvagnat et al., 2005]** (voir formule 3.6, chapitre 3) :

$$score(t_{ij}, nf_j) = \frac{tf_i^{\,j}}{size(nf_j)} \quad \ldots (4.11)$$

Où : $tf_i^{\,j}$ représente la fréquence du terme t_i dans le nœud feuille nf_j.

Le poids d'un terme au sein de l'ensemble EP (respectivement ENP) est défini par la formule 4.12 (respectivement 4.13) :

$$poids(t_i, EP) = \sum_{nf_j \in EP} score(t_{ij}, nf_j) \quad \ldots (4.12)$$

$$poids(t_i, ENP) = \sum_{nf_j \in ENP} score(t_{ij}, nf_j) \quad \ldots (4.13)$$

Pour avoir ces deux ensembles (c'est-à-dire EP et ENP) nous appliquons l'algorithme suivant :

1- Extraire tous les nœuds feuilles (appartenant à des éléments jugés pertinents) ayant un score positif ; on appelle cet ensemble « L ».
2- Extraire l'ensemble des nœuds feuilles (appartenant à des éléments jugés non pertinents) ayant un score positif ; on appelle cet ensemble « ENP »
3- Appliquer l'algorithme de suppression des nœuds feuilles appartenant à des éléments jugés pertinents à partir des éléments jugés non pertinents (voir tableau 4.8)
4- EP = L – ENP (En d'autres termes enlever de l'ensemble L tous les nœuds feuilles appartenant à ENP)

```
Tableau_1 = {Nœud_Pert_i/ Nœud_Pert_i ∈ P}
Tableau_2 = {Nœud_NPert_i/ Nœud_NPert_i ∈ NP}
Pour tout (Nœud_NPert_i) faire
    Pour tout (Nœud_Pert_i) faire
        Si Ancêtre (Nœud_Pert_i)=Nœud_NPert_i alors
            Enlever (Tous les nœuds feuilles appartenant à l'ensemble
```

Nœud_Pert$_i$ de l'ensemble Nœud_NPert$_j$)
FinSi
FinPour
FinPour

Tableau 4.8 : Algorithme de suppression des nœuds feuilles appartenant à des éléments jugés pertinents à partir des éléments jugés non pertinents

Deux cas peuvent survenir lors de l'application de l'algorithme. Ces deux cas sont montrés dans les figures (4.2) et (4.3).

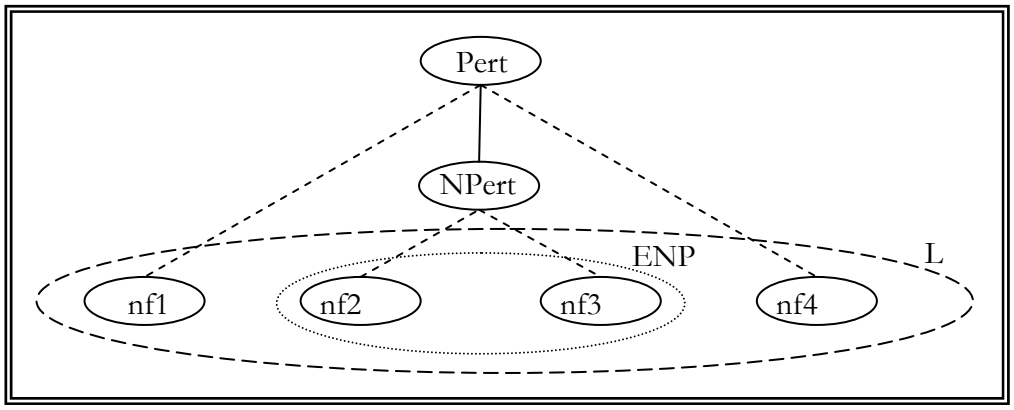


Figure 4.2 : Cas où un élément pertinent comporte dans son arborescence un élément non pertinent

Le deuxième cas (la figure 4.3) n'arrive qu'avec l'utilisation de la mesure stricte (e=3 et s=3) dans l'évaluation.

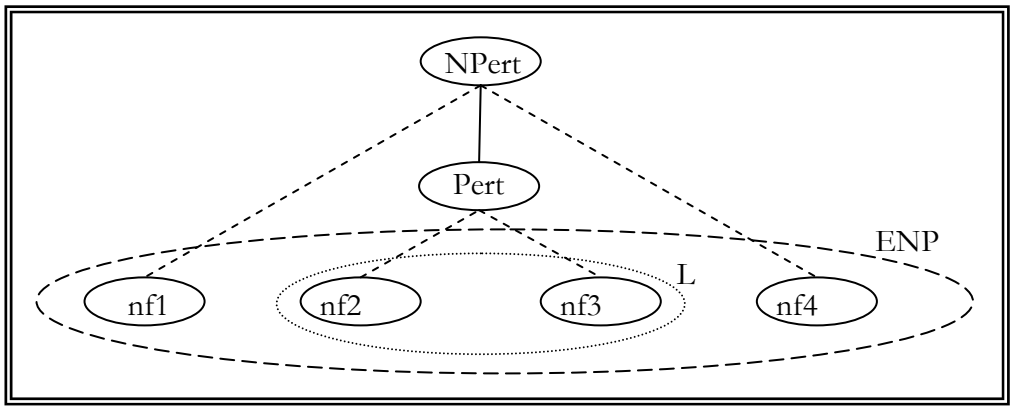


Figure 4.3 : Cas où un élément non pertinent comporte dans son arborescence un élément pertinent

Après ces quatre étapes, nous obtenons les deux ensembles EP et ENP, tels que : EP∩ENP=∅. Pour illustrer le déroulement du processus de choix des meilleurs termes à rajouter à la requête initiale nous donnons l'exemple suivant : Supposons qu'une requête portant

sur le thème « Programmation Java Héritage Encapsulation ». Les éléments jugés par l'utilisateur sont mentionnés sur l'arbre de la figure (4.4).

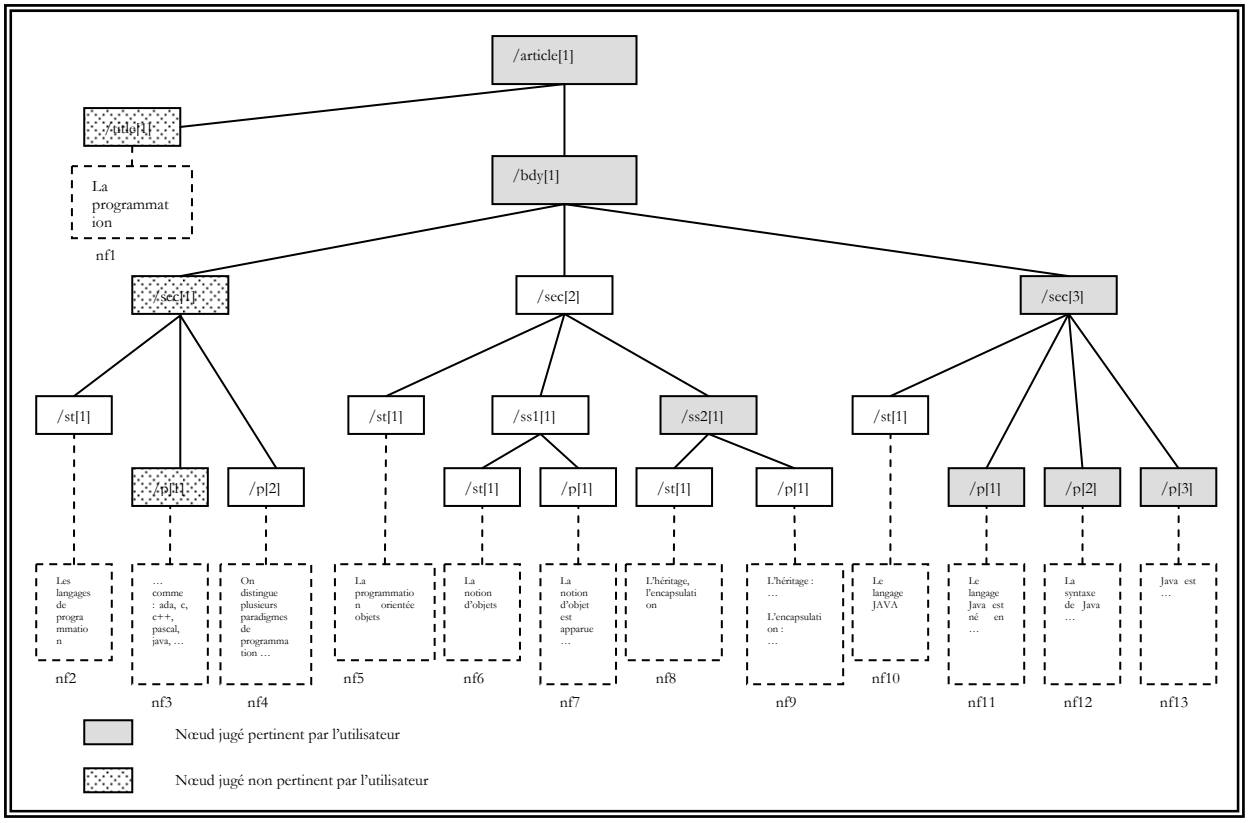


Figure 4.4 : Arbre d'un document XML avec jugements utilisateur

Suite aux jugements de l'utilisateur nous procédons (première étape de l'algorithme) à la définition des deux ensembles (L et ENP).

Eléments jugés pertinents /article[1]/bdy[1]/sec[3]={nf10, nf11, nf12, nf13} /article[1]/bdy[1]/sec[3]/P[1] ={nf11} /article[1]/bdy[1]/sec[3]/P[3] ={nf13} /article[1]/bdy[1]/sec[3]/P[2] ={nf12} /article[1]/bdy[1]/sec[2]/ss2[1] ={nf8, nf9} /article[1]/bdy[1]/ ={nf2, ..., nf13}-{nf6} /article[1]/ ={nf1, ..., nf13}-{nf6} Le nœud nf6 ne sera pas utilisé dans les calculs parce que son score est égal à 0 L = {nf1, ..., nf13}-{nf6}
Eléments jugés non pertinents /article[1]/bdy[1]/sec[1] ={nf2, nf3, nf4} /article[1]/title[1] ={nf1} /article[1]/bdy[1]/sec[1]/p[1] ={nf3} ENP = {nf1, nf2, nf3, nf4}

Dans notre cas (arbre de la figure 4.4) nous ignorons l'étape 3 de l'algorithme car nous n'avons aucun élément jugé non pertinent ancêtre d'un autre élément jugé pertinent. A partir des deux ensembles L et ENP nous pouvons déduire l'ensemble EP qui va être utilisé dans le processus de choix des termes :

EP = L – ENP

= {{*nf1, ..., nf13*}-{*nf6*}}- {*nf1, nf2, nf3, nf4*} = {*nf5, nf7, nf8, nf9, nf10, nf11, nf12, nf13*}

4.3.2. Expansion par ajout de contraintes de structure

Nous allons présenter dans cette partie une étude statistique effectuée sur les résultats renvoyés par le système XFIRM à la campagne INEX pour la tâche CO+S. Cette étude va nous permettre de répondre à la question suivante : « *est ce que l'ajout des contraintes de structure permet d'améliorer la qualité des résultats ?* ».

La première étape de cette étude statistique consiste à définir les différentes composantes qui vont intervenir. En combinant les fichiers des résultats renvoyés avec les fichiers comportant les jugements de pertinence issus de la campagne INEX (tableau 4.9), nous avons conçu une sorte de base de données qui contient l'ensemble des données des deux types de fichiers. Le schéma de la figure suivante (figure 4.5) représente les étapes pour lesquelles nous sommes passés pour réaliser cette étude.

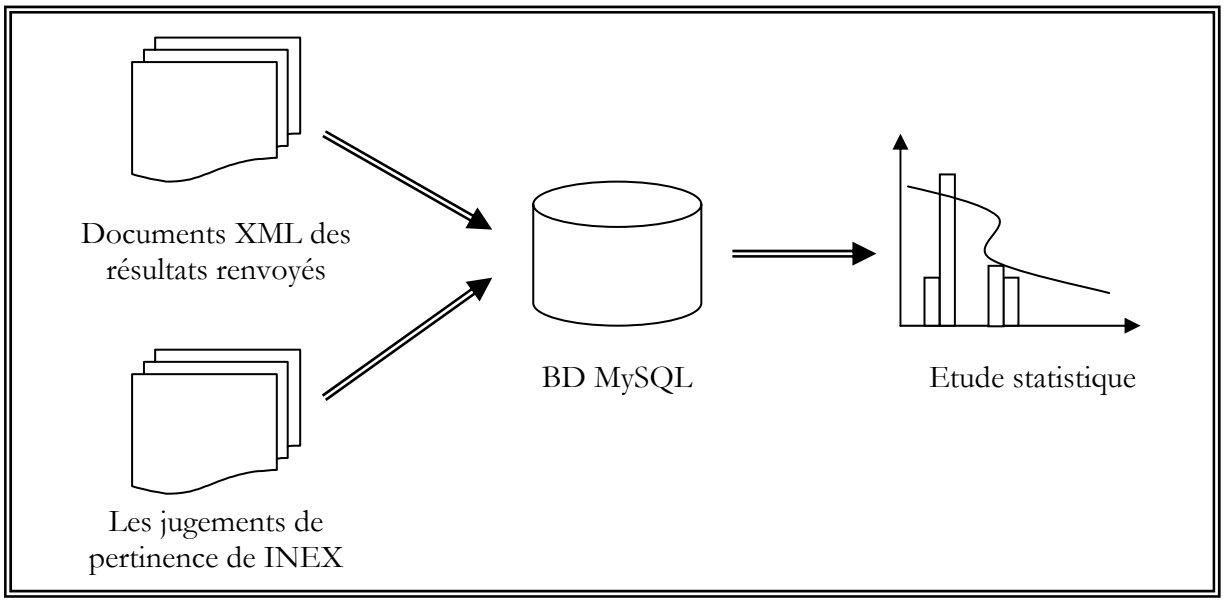


Figure 4.5 : Schéma d'exploitation des données utilisées dans l'étude statistique

Après avoir obtenu les données des deux types de fichiers (résultats et jugements), nous allons essayer de répondre à deux questions : *Est ce qu'il y a des types d'éléments qui sont souvent jugés pertinents ? Est ce qu'il y a des types d'éléments qui sont souvent jugés non pertinents ?*. Notons que les éléments que nous considérons comme étant pertinents sont ceux qui ont des valeurs d'exhaustivité >= 1 (la mesure généralisée).

```
<?xml version="1.0"?>
<!DOCTYPE assessments SYSTEM "../assessments.dtd">
```

```
<assessments pool="309" topic="212" version="2">
<!-- Topic definition -->
<inex_topic topic_id="212" query_type="CO+S" ct_no="49">
<InitialTopicStatement>I'm writing a paper that uses hidden markov models and I'd like to see how others write the markov model equations, so that my notation may be somewhat consistent with the notation in other literature. I'm searching for components containing equations of hidden markov models and the corresponding text that describes the equation. I expect this topic to be fairly difficult to express in NEXI, yet still an important information need.</InitialTopicStatement>
<title>HMM "hidden Markov model" equation</title>
<castitle>//*[(about(., HMM equation) OR about(., "hidden Markov model" equation)) AND .//en >0]</castitle>
<description>Find text containing equations for a hidden Markov model (HMM).</description>
<narrative>I'm writing a paper that uses hidden markov models and I'd like to see how others write the markov model equations, so that my notation may be somewhat consistent with the notation in other literature. I'm searching for components containing equations of hidden Markov models (HMMs) and the corresponding text that describes the equation. </narrative>
</inex_topic>
<!-- Topic assessments (only completed files) -->
<file collection="ieee" name="tp/2002/i1291">
  <passage start="/article[1]/bdy[1]/sec[4]/ss1[1]/st[1]" end="/article[1]/bdy[1]/sec[4]/ss1[6]/ip1[5]/text()[5].128" size="16592"/>
  <element path="/article[1]/bdy[1]/sec[4]" exhaustivity="2" size="16606" rsize="16592"/>
  <element path="/article[1]/bdy[1]" exhaustivity="2" size="29249" rsize="16592"/>
  <element path="/article[1]" exhaustivity="2" size="33744" rsize="16592"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[1]" exhaustivity="2" size="3486" rsize="3486"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[1]/st[1]" exhaustivity="?" size="28" rsize="28"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[1]/p[1]" exhaustivity="1" size="238" rsize="238"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[2]" exhaustivity="2" size="2020" rsize="2020"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[2]/st[1]" exhaustivity="?" size="17" rsize="17"/>
  <element path="/article[1]/bdy[1]/sec[4]/ss1[2]/p[1]" exhaustivity="1" size="1149" rsize="1149"/>
…
</file>
…
<file collection="ieee" name="…..">
</file>
…
</assessments>
```

Tableau 4.9 : Un extrait d'un fichier de jugements des résultats du système XFIRM (issus de la campagne INEX, requête 212 de la tâche CO+S)

Les résultats que nous avons obtenus pour les requêtes de la tâche CO+S montrent que les balises qui appartiennent à l'ensemble des éléments pertinents sont en particulier : *« /p », « /sec », « /bdy », « /article »* et leurs balises équivalentes (c'est-à-dire *« /ip », « /p1 », « /ss1 »*, …). Dans le tableau 4.10 (respectivement 4.11), nous trouvons les statistiques sur le nombre d'éléments pertinents et non pertinents pour les balises : *« /p », « /sec », « /bdy », « /article »* (respectivement les balises *« /p », « /sec »* et leurs balises équivalentes).

N° Topic	Les éléments de type « /p »		Les éléments de type « /sec »		Les éléments de type « /bdy »		Les éléments de type « /article »	
	Pert	***NPert***	***Pert***	***NPert***	***Pert***	***NPert***	***Pert***	***NPert***
202	2	119	7	0	7	0	7	0
203	53	34	28	0	20	0	20	0
205	29	0	17	0	16	0	16	0

206	288	23	99	0	65	0	66	0
207	168	25	87	9	58	2	69	0
208	186	320	64	0	34	0	37	0
209	142	1182	231	1	53	0	53	0
210	103	17	58	0	35	0	36	0
212	434	12	71	0	41	0	41	0
213	111	13	84	1	41	0	43	0
216	68	1094	180	8	97	0	99	0
218	213	60	62	0	34	0	35	0
221	121	23	41	1	31	0	34	0
222	259	203	152	60	99	7	108	0
223	200	193	180	33	115	0	124	0
227	77	2	52	0	28	0	28	0
228	169	6	60	0	37	0	40	0
229	46	40	34	0	29	0	30	0
230	21	0	25	0	15	0	15	0
232	15	9	21	0	12	0	12	0
233	21	0	8	0	4	0	4	0
235	237	8	106	0	64	0	70	0
236	171	123	66	9	37	0	50	0
239	3	455	56	0	19	0	20	0
241	20	0	16	0	7	0	7	0
242	75	809	132	0	48	0	50	0

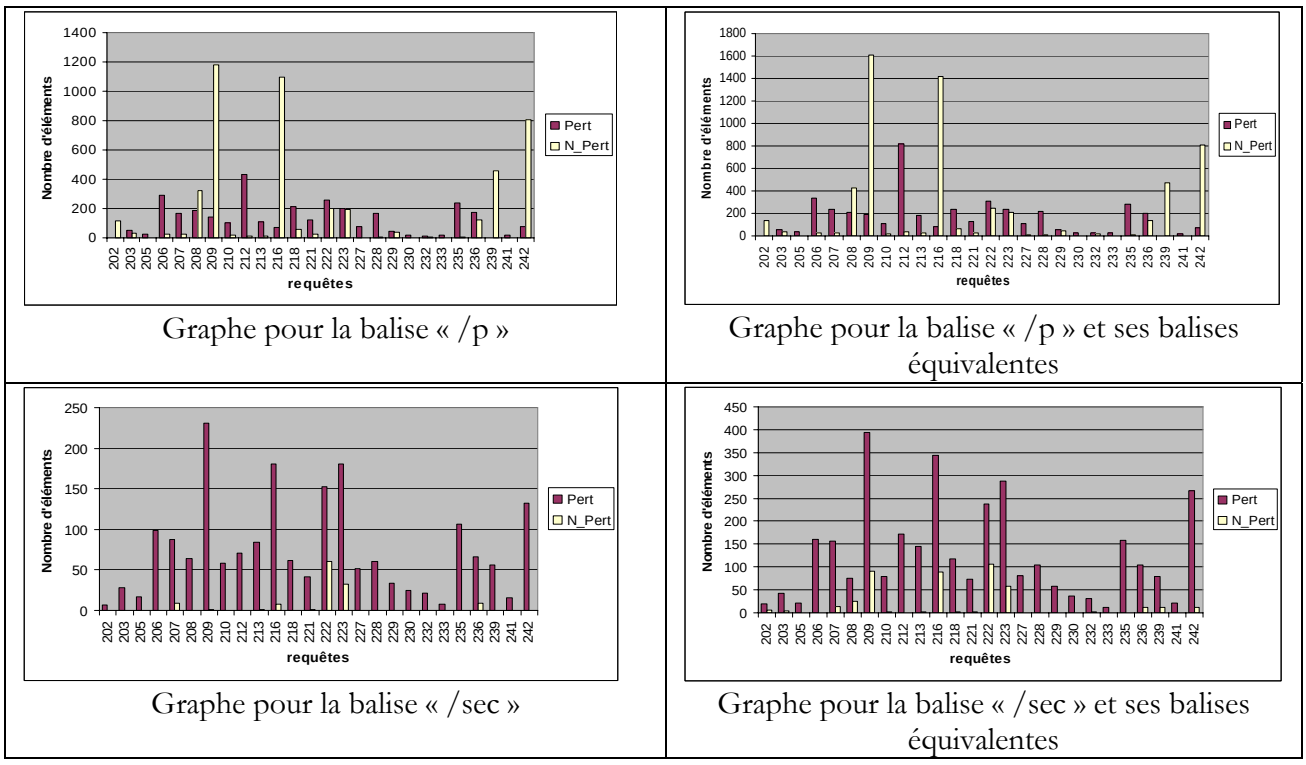
Tableau 4.10 : Nombre d'éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX (pour les requêtes de la tâche CO+S)

N° Topic	Les éléments de type « /p » et éléments équivalents (ex:/ip, p1, p2, …)		Les éléments de type « /sec » et éléments équivalents (ex:/ss1, /ss2, /ss3)	
	Pert	***NPert***	***Pert***	***NPert***
202	3	133	20	6
203	57	35	42	3
205	32	0	21	0
206	334	26	160	0
207	239	31	157	14
208	209	429	76	26
209	194	1611	394	90
210	113	21	79	2
212	821	33	172	0
213	185	30	145	1
216	79	1421	343	89
218	236	64	117	1
221	127	23	73	1
222	306	244	237	107
223	239	209	287	57
227	105	7	81	0

228	217	7	105	0
229	52	50	58	0
230	31	0	36	0
232	27	20	30	1
233	24	0	12	0
235	286	8	159	0
236	200	135	105	11
239	3	471	80	11
241	20	0	22	0
242	75	812	266	12

Tableau 4.11 : Nombre d'éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX (en tenant compte des équivalences de balises (voir section 2.8.1, chapitre 2), pour les requêtes de la tâche CO+S)

La figure 4.6 montre la distribution des éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX pour les requêtes de la tâche CO+S.



Graphe pour la balise « /p »

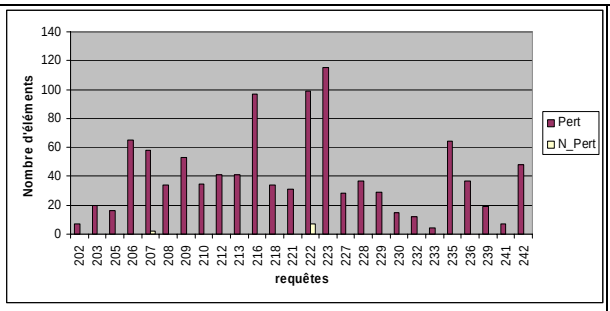
Graphe pour la balise « /p » et ses balises équivalentes

Graphe pour la balise « /sec »

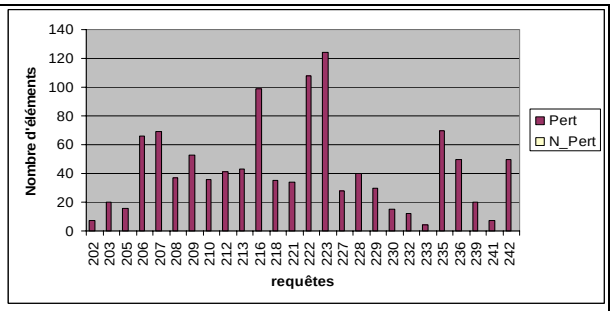
Graphe pour la balise « /sec » et ses balises équivalentes

Figure 4.6 : Distribution des éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX pour les requêtes de la tâche CO+S

De la figure (4.6) précédente nous pouvons remarquer que les éléments de type « /p » sont pertinents pour la plupart des topics (20 sur 26) et que leurs balises équivalentes (*p1, p2, ip1,* …) sont pertinentes pour la plupart des topics (sauf quelques topics, exemple : 208, 209, 216). Il en est de même pour les éléments de type « /sec » qui sont pertinents pour la plupart des topics (25 sur 26) et que leurs balises équivalentes (*ss1, ss2, ...*) sont pertinentes pour la plupart des topics (sauf quelques topics, exemple : *ss2* et *ss3* en 208, *ss2* en 216)



Graphe pour la balise « /bdy »



Graphe pour la balise « /article »

Figure 4.7 : Distribution des éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX (balises de type « */bdy* » et « */article* ») pour les requêtes de la tâche CO+S

De la figure précédente nous pouvons remarquer que les éléments de type « */bdy* » sont pertinents dans tous les topics (26 sur 26) et aussi les éléments de type « */article* » sont pertinents dans toutes les topics (26 sur 26) à un point où aucune des balises « */article* » n'est considérée comme non pertinente.

Les résultats ont montré aussi que la plupart des autres types de balises sont souvent jugés non pertinents dans la totalité des topics (exemple : */art, /tt, /fig, /row,* ...). Prenons à titre d'exemple les résultats des éléments de type « */art* », « */st* », « */fig* », « */b* » qui sont montrés dans tableau suivant (4.12) :

N° Topic	Les éléments de type « /art »		Les éléments de type « /st »		Les éléments de type « /fig »		Les éléments de type « /b »	
	Pert	**NPert**	**Pert**	**NPert**	**Pert**	**NPert**	**Pert**	**NPert**
202	0	10	0	21	0	12	0	8
203	-	-	0	12	-	-	0	2
205	-	-	-	-	-	-	-	-
206	2	16	0	31	14	0	0	29
207	1	2	11	0	13	3	3	0
208	0	1	0	46	0	1	2	148
209	0	1936	2	409	0	73	0	139
210	0	1	0	2	0	1	-	-
212	37	742	0	115	40	0	4	123
213	3	109	0	15	10	0	0	108
216	14	620	1	271	31	134	0	95
218	0	6	0	12	15	8	2	1
221	0	8	0	24	7	3	0	5
222	10	5	3	2	8	7	5	0
223	-	-	1	10	3	0	1	1
227	0	90	-	-	-		0	4
228	-	-	-	-	1	0	-	-
229	0	51	0	12	0	3	0	1
230	-	-	0	1	-	-	-	-
232	5	0	4	2	-	-	3	5
233	-	-	0	7	-	-	-	-
235	-	-	0	1	-	-	0	26

236	0	11	0	8	6	5	0	19
239	0	49	1	30	0	35	0	18
241	-	-	1	0	-	-	-	-
242	0	331	0	228	31	82	0	59

Tableau 4.12 **:** Nombre d'éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX (pour les éléments de type : « */art* », « */st* », « */fig* », « */b* »), pour les requêtes de la tâche CO+S)

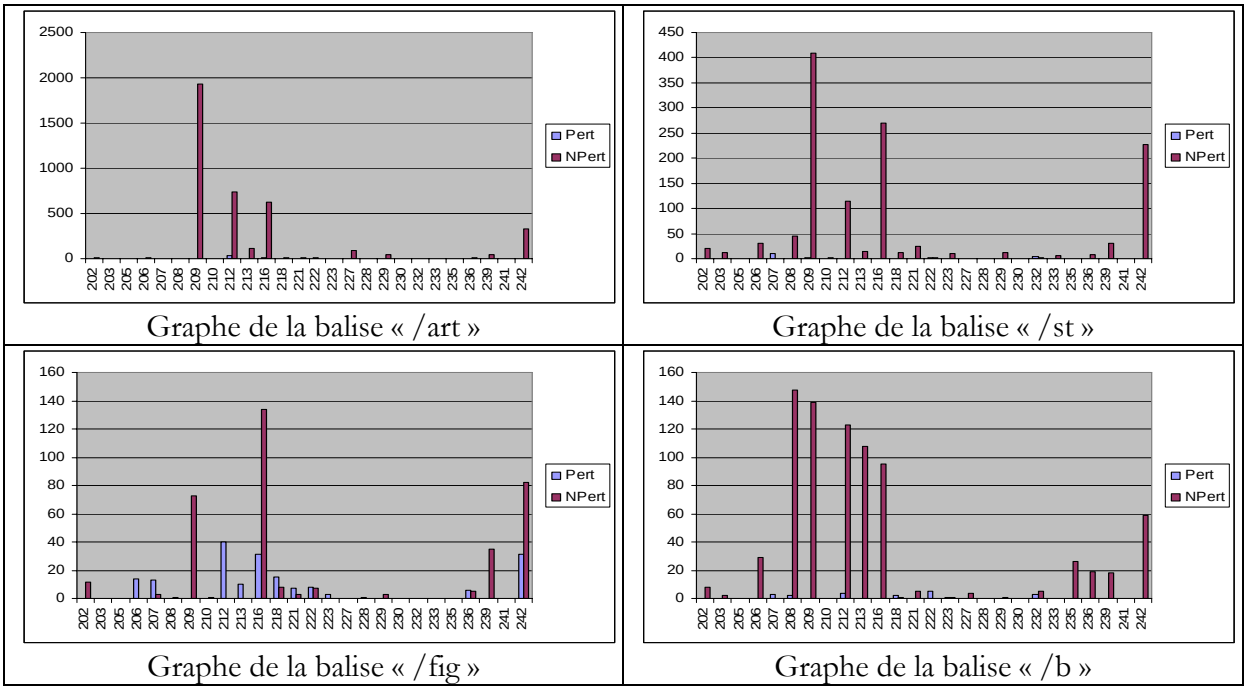


Figure 4.8 : Distribution des éléments pertinents et non pertinents renvoyés par le système XFIRM selon les jugements issus d'INEX (balises de type « */art* », « */st* », « */fig* » , « */b* ») pour les requêtes de la tâche CO+S

De l'étude statistique précédente, nous pouvons dire qu'il existe effectivement certains types de balises qui sont souvent jugés pertinents et d'autres types qui sont souvent jugés non pertinents.

Après avoir effectué cette étude statistique, nous pouvons répondre à la première question posée : « *Est ce que l'ajout des contraintes de structure permet d'améliorer la qualité des résultats ?* ». Pour cela nous donnons quelques exemples de résultats renvoyés :

- **La requête 210** : Le champ « title » (voir section 2.8.2) de cette requête est : « +multimedia "document models" "content authoring" ». Le champ « CAStitle » est : « //article//(abs|sec)[about(.,+multimedia "document models" "content authoring")] ». La première exécution portant seulement sur le champ « title » (c'est-à-dire qu'elle n'inclue pas des contraintes de structure) a permis de renvoyer 451 éléments dont 157 non pertinents. Alors qu'avec l'intégration d'une contrainte de structure (c'est-à-dire le champ « CAStitle » du topic) le nombre d'éléments renvoyés était de 86 dont 2 seulement non pertinents.

- **La requête 228** : Le champ « title » de cette requête est : « "IPv6 deployment" "IPv6 support" ». Le champ « CAStitle » est : « //article[about(.//abs, IPv6)]//sec[about(., "IPv6

deployment") or about(., "IPv6 support")] ». La première exécution portant seulement sur le champ « title » a permis de renvoyer 510 éléments dont 80 non pertinents. Alors qu'avec l'inclusion d'une contrainte de structure le nombre d'éléments renvoyés était de 105 tous pertinents.

- **La requête 230** : Le champ « title » de cette requête est le suivant : « +brain research +"differential geometry" ». Le champ « CAStitle » est : « //article//sec[about(.,brain research "differential geometry")] ». La première exécution portant seulement sur le champ « title » a permis de renvoyer 125 éléments dont 20 non pertinents. Alors qu'avec l'inclusion de la contrainte structurale du champ « CAStitle » le nombre d'éléments renvoyés était de 36 tous pertinents.

Nous pouvons conclure que l'introduction des contraintes structurelles dans une requête de type CO permet d'améliorer la qualité des résultats. En se basant sur cette étude statistique nous proposons une méthode permettant l'ajout d'une contrainte structurelle à la requête initiale.

La méthode que nous proposons repose sur la probabilité qu'un type élément soit souvent jugé pertinent. Cette probabilité peut être calculée par la formule suivante :

$$Prob(Tag) = \frac{NBR\_Elements\_Pert(Tag)}{NBR\_Element\_Juges\_Pert} \qquad \dots (4.14)$$

où :

NBR_Elements_Pert(Tag) : représente le nombre d'éléments jugés pertinents ayant comme type de balise « *Tag* ».

NBR_Elements_Juges_Pert : correspond au nombre total d'éléments jugés pertinents.

Cette formule nous permet d'avoir une idée sur les types d'éléments qui jouent un rôle important dans la liste renvoyée des éléments pertinents.

L'exploitation de cette information, c'est-à-dire les types d'éléments jugés souvent pertinents, s'effectue selon l'algorithme :

Faire le classement des différentes « *Tag* » par ordre décroissant selon la formule (4.14)
Sauvegarder les « Tags » avec leurs % dans le tableau « TAB_Tags »
Calculer le SCA selon l'algorithme défini dans **[Sauvagnat et al, 2005]** (voir tableau 3.3)
Extraire le % de la balise du SCA choisi (ex : Tag(« /article/bdy/sec ») = « /sec », donc extraire le % de « /sec »)
ENS <- SCA ;
Pour i = 1 jusqu'à Taille(TAB_Tags) **faire**
  Si ($\sum$ *(%ENS)* < *%DEF des éléments pertinents*) & TAB_Tags[i]<>SCA) **Alors**
    ENS <- ENS $\cup$ TAB_Tags[i] ;
  Sinon
    //SCA[about(., mots clés)] ***OR*** *(Tag_1|…|Tag_n) [about(., mot clés)]*
    Sortir
      Tel que : $Tag_1|\dots|Tag_n \in$ ENS. ENS comporte les balises qui ne sont ni descendants ni ascendants de SCA. Si Tag_i appartient, dans les résultats, à l'arborescence du SCA et aussi à une autre arborescence on préserve son ancêtre directe, exemple : « */sec/p* » et « */abs/p* », si SCA = « */sec* » alors on garde seulement « */abs* » et non pas « */p* »).

%DEF : c'est un pourcentage à définir avec lequel nous fixons la qualité des résultats qu'on peut accepter.
ENS : l'ensemble de balises qui sont jugés pertinents et qui vont être rajoutés comme contraintes structurelles à la nouvelle requête.
Fsi
FPour

Tableau 4.13 : Algorithme d'ajout des contraintes structurelles à une requête de type CO

Pour illustrer le fonctionnement de l'algorithme d'ajout des contraintes de structure, nous donnons l'exemple suivant : supposons qu'on utilise la requête 210 (la tâche CO de la campagne INEX) qui est définie par les mots clés : « +*multimedia "document models" "content authoring"* ». La liste des 50 premiers éléments renvoyés comporte 33 éléments jugés pertinents dont 8 de type « */sec* », 8 de type « */p* », 6 de type « */bdy* », 6 de type «*/article* » et 5 des autres types (3 de type « /abs » et 2 de type « fm »). Supposons que le SCA calculé par le système XFIRM est : « */article/bdy/sec* ». Le tableau « TAB_Tags » se présente comme suit :

« /sec »	« /p »	« /bdy »	« /article »	« /abs »	« /fm »
8/33	8/33	6/33	6/33	3/33	2/33

En fait, dans ces résultats nous avons remarqué que la balise « /p » existe dans l'arborescence de « /sec » et de « /abs », ainsi nous allons choisir la balise « /abs » pour remplacer « /p ». En fin de calcul ENS sera égal à {« /sec », « /abs », «/fm »}.

La nouvelle requête CAS se présente sous la forme suivante :

//article//bdy//sec [about*(.,+multimedia "document models" "content authoring"*)]

OR

(abs | fm) [about(.,+*multimedia "document models" "content authoring"*)]

4.4. Conclusion

Nous avons présenté dans ce chapitre un ensemble de propositions pour l'intégration de la technique de réinjection de pertinence dans le contexte de la RI structurée. Ces propositions sont divisées selon les deux types de stratégies : stratégie de ré-ordonnancement et stratégie d'expansion.

Dans le premier type de stratégies nous avons décrit une méthode basée sur le contexte des éléments jugés et une autre méthode basée sur l'information structurelle « *Nom de journal* ». Dans chacune des propositions, nous avons envisagé l'utilisation de plusieurs formules afin d'étudier l'impact de chacun des facteurs utilisés.

En ce qui concerne l'expansion de requêtes, nous proposons deux stratégies : expansion par ajout de termes et expansion par ajout de contraintes structurelles. Notre approche pour l'expansion de requêtes par ajout de termes entre dans la même lignée que celle proposée par **[Hlaoua and Boughanem, 2005]**. Elle est basée sur la formule de Rocchio, de telle sorte à permettre de pallier le problème d'imbrication des éléments renvoyés pour jugements et aussi

d'éliminer si possible l'inclusion des nœuds non pertinents dans le choix des termes à rajouter. Concernant l'expansion structurelle, nous nous sommes tout d'abord poser la question de l'intérêt d'ajouter une contrainte de structure à une requête. Nous avons pour cela effectué une étude statistique sur un échantillon de résultats (ensemble d'éléments) retournés en réponse à des requêtes prises dans le cadre d'INEX. Puis, à l'issue de cette analyse nous avons proposé une technique permettant d'identifier la structure pertinente pouvant être rajoutée à la requête.

Le prochain chapitre consiste à expérimenter les différentes méthodes de ré-ordonnancement présentées lors de ce chapitre et d'étudier l'impact des différentes formules envisagées pour le calcul de coefficient.

Chapitre 5 :

Expérimentation et résultats

5.1. Introduction

Dans ce chapitre, nous présentons les résultats d'expérimentation effectuée pour évaluer les différentes propositions du chapitre précédent. Les évaluations portent essentiellement sur les méthodes de ré-ordonnancement proposées.

En premier lieu nous décrivons l'environnement d'évaluation et les conditions expérimentales (collection, données, outils et mesures utilisés).

Nos expérimentations sont organisées en trois parties: la première partie concerne l'évaluation des différentes formules proposées pour le ré-ordonnancement contextuel; la deuxième partie concerne l'évaluation des formules proposées pour le ré-ordonnancement par nom de journal; enfin, la troisième partie porte sur l'étude de l'impact des facteurs suivants pour les deux types de ré- ordonnancement :

- introduction des éléments non pertinents dans le calcul de coefficient ;
- variation du nombre d'éléments jugés ;
- nombre d'itérations.

En fin de chapitre, nous présentons une analyse et discussion des résultats obtenus. Ainsi, nous pourrions juger la qualité des résultats de chacune de nos propositions.

5.2. Environnement d'évaluation

Dans cette section nous allons décrire l'environnement et les conditions expérimentales qui vont nous permettre d'évaluer les différentes formules du chapitre précédent. Tout d'abord, nous décrivons les mesures, la collection, les données utilisées dans l'expérimentation. Ensuite, nous présentons l'application de ré-ordonnancement réalisée. Enfin, nous détaillons le processus de ré-ordonnancement et d'évaluation suivi.

5.2.1. Les mesures utilisées

Dans notre expérimentation nous avons utilisé les mesures officielles d'INEX 2005 qui sont : les mesures XCG (*XML Cumulative Gain*). Les mesures XCG sont des extensions du gain cumulatif (CG) proposé par Järvelin et Kekäläinen **[Järvelin and Kekäläinen, 2002]**.

Pour comparer les résultats obtenus après feedback avec ceux de l'exécution de base, nous utilisons la formule de l'amélioration relative (AR) définie par :

$$AR = \frac{Valeur(Execution\_RF) - Valeur(Execution\_base)}{Valeur(Execution\_base)} \qquad \dots (5.1)$$

5.2.2. Collection, données et outils

Toutes les données utilisées pour évaluer les formules des deux types de ré-ordonnancement sont issues du système XFIRM (exécution de base) et de la campagne INEX (jugements utilisateur). Le premier type de données (résultats de l'exécution de base) suit le format défini dans le chapitre précédent (voir tableau 4.1). Le deuxième type de données (jugements INEX) suit le format défini aussi dans le chapitre précédent (voir le tableau 4.9). Le nombre de requêtes avec lesquelles nous avons effectué l'expérimentation est égal à 29 requêtes "CO+S". Les résultats employés pour le ré-ordonnancement sont ceux de la tâche "*CO.Thourough*".

5.2.3. Application de ré-ordonnancement

Pour effectuer l'expérimentation, nous avons conçu et réalisé une application dans laquelle toutes les formules de ré-ordonnancement sont implémentées. Cette application permet de fixer différents paramètres (numéro de requête, nombre d'éléments jugés, type de ré-ordonnancement et la formule de calcul de coefficient) avant d'exécuter le ré-ordonnancement.

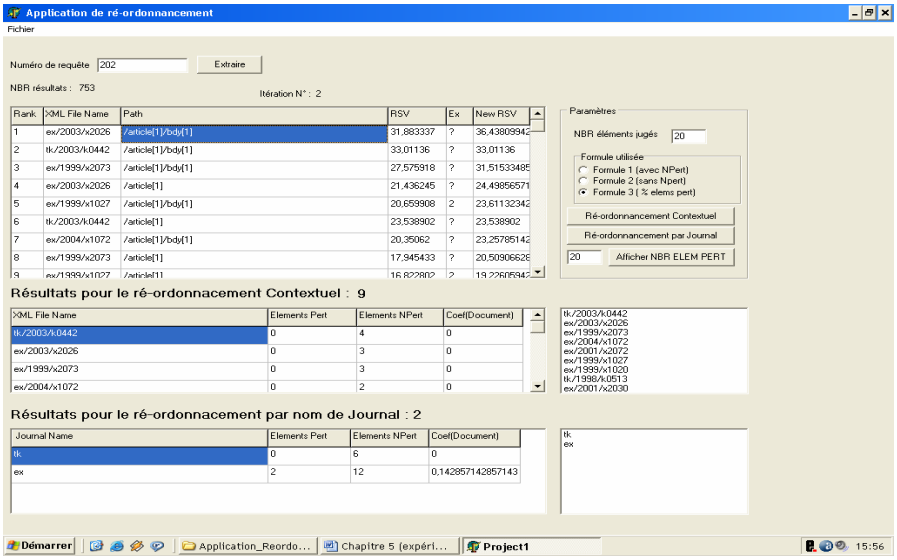


Figure 5.1 : Capture d'écran de l'application de ré-ordonnancement

5.2.4. Processus de ré-ordonnancement et évaluation

Le processus de ré-ordonnancement et d'évaluation (figure 5.2) se déroule comme suit:

- Réception des résultats renvoyés par le système XFIRM (exécution de base);
- Combinaison de ces résultats avec les jugements issus d'INEX;
- Application des différentes techniques de ré-ordonnancement (contextuel et par nom de journal);

- Evaluation des résultats du ré-ordonnancement (l'exécution feedback) par l'outil EvalJ[3];
- Comparaison des résultats du ré- ordonnancement avec les résultats initiaux.

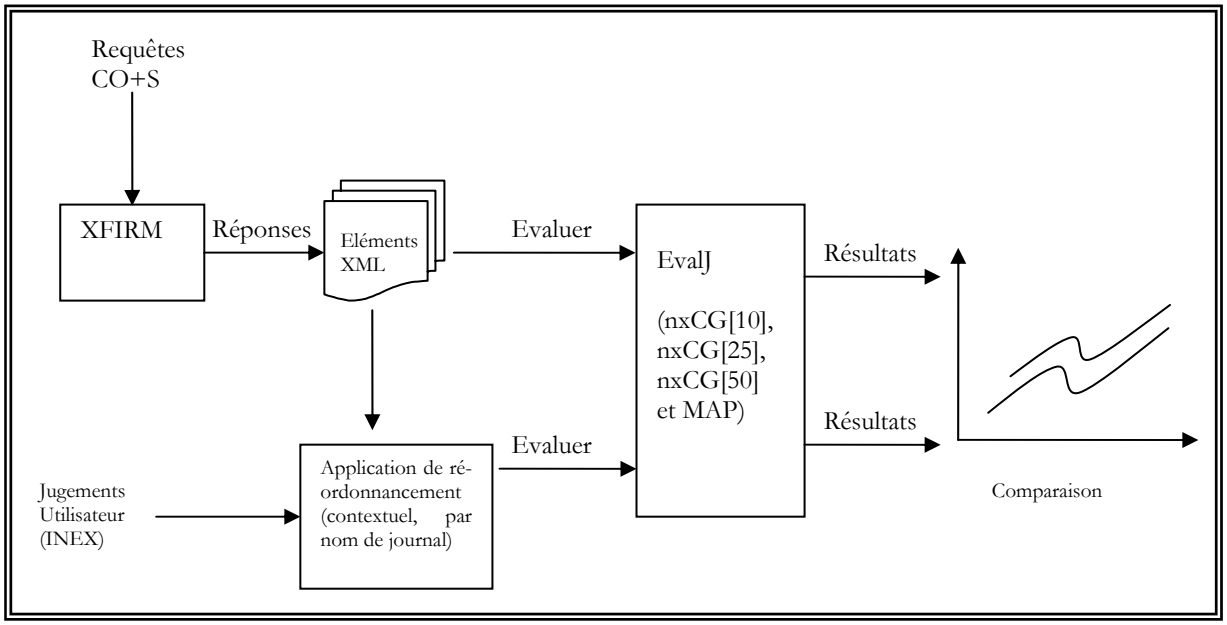


Figure 5.2 : Processus de ré-ordonnancement et d'évaluation

Dans la suite du mémoire, nous allons utiliser la notation suivante pour représenter les formules:

	Ré-ordonnancement Contextuel	Ré-ordonnancement par nom de journal
Formule 1	Formule (4.2)	Formule (4.6)
Formule 2	Formule (4.3)	Formule (4.7)
Formule 3	Formule (4.4)	Formule (4.8)

Tableau 5.1 : Notation des formules

Lors de notre expérimentation, nous avons étudié plusieurs aspects (type de ré-ordonnancement, formule utilisée, nombre d'éléments jugés et nombre d'itérations). Ainsi, les résultats se présentent sous la forme de combinaisons comportant chacune une des variations possibles des 4 aspects déjà cités. Pour le ré-ordonnancement, nous avons deux types (contextuel et par nom de journal); les formules peuvent prendre l'une des valeurs: formule 1, formule 2 ou formule 3; pour le nombre d'éléments jugés, nous l'avons fixé à 20 et 50; enfin, pour le nombre

[3] EvalJ : Projet JAVA dans lequel toutes les mesures utilisées durant INEX 2005 ont été implémentées par Benjamin Piwowarski. Cette application peut être téléchargée à partir de l'adresse : https://sourceforge.net/projects/evalj

d'itérations, nous avons utilisé 1, 2 et 3 itérations. De ce fait, l'expérimentation va comporter 36 combinaisons (2*3*2*3 = 36).

5.3. Evaluation des méthodes de ré-ordonnancement

Dans cette section, nous allons présenter les résultats d'évaluation obtenus par chacune des méthodes de ré-ordonnancement pour les différentes combinaisons envisagées. Nous considérons trois parties : 1) résultats d'expérimentation des formules de ré-ordonnancement contextuel; 2) résultats d'expérimentation des formules de ré-ordonnancement par nom de journal; 3) étude de l'impact des différents facteurs (introduction des éléments non pertinents dans le calcul de coefficient, variation du nombre d'éléments jugés et nombre d'itérations). Chaque combinaison se présente sous forme d'un tableau organisé comme suit : les lignes indiquent le numéro d'itération et les formules utilisées; les colonnes indiquent les niveaux de coupures pour lesquelles ont été prises les mesures (nxCG[10], nxCG[25], nxCG[50] et MAP). Chaque tableau suit un des types de mesure (stricte ou généralisée) et est propre à un nombre fixé d'éléments jugés (20 ou 50). Ainsi, tous les tableaux contiennent les valeurs de AR calculées pour chaque niveau de coupure. ARmin représente l'amélioration relative minimale et ARmax représente l'amélioration relative maximale.

5.3.1. Ré-ordonnancement contextuel

Dans cette section nous présentons les résultats obtenus par l'application des différentes formules proposées pour le ré-ordonnancement contextuel. Ces résultats sont évalués selon les deux types de mesures (stricte et généralisée).

5.3.1.1. Mesure généralisée

Le tableau 5.2 présente les résultats obtenus avec les trois formules du ré-ordonnancement contextuel en utilisant la mesure généralisée avec un nombre d'éléments jugés fixé à 20.

		nxCG[10]	**nxCG[25]**	**nxCG[50]**	**MAP**
	Base_Run	0,166	0,1737	0,1831	0,0541
Itération 1	**formule_1**	0,1742	0,1779	0,188	0,0555
	formule_2	0,1747	0,1808	0,1901	0,05534
	formule_3	0,1945	0,1916	0,2075	0,0571
Itération 2	**formule_1**	0,1947	0,1901	0,1951	0,0572
	formule_2	0,1958	0,1932	0,1987	0,05661
	formule_3	0,2114	0,221	0,2217	0,0593
Itération 3	**formule_1**	0,2065	0,2044	0,1983	0,0587
	formule_2	0,2027	0,2085	0,2026	0,0574
	formule_3	0,2174	0,2422	0,2271	0,0609

min	0,1742	0,1779	0,188	0,05534
max	0,2174	0,2422	0,2271	0,0609
ARMin(%)	4,93	2,41	2,67	2,29
ARMax(%)	30,96	39,43	24,03	12,56

Tableau 5.2 : Résultats pour 20 éléments jugés (mesure généralisée)

Nous remarquons que les résultats des trois formules de calcul de coefficient sont meilleurs que ceux de l'exécution de base (résultats initiaux obtenus par XFIRM). Ceci est valide pour les différents points de coupure (nxCG[10], nxCG[25] et nxCG[50]) ainsi que pour le MAP. La meilleure AR est obtenue pour nxCG[25] (formule 3) dans la troisième itération (AR=39.43%).

Le tableau 5.3 présente les résultats d'application du ré-ordonnancement contextuel en utilisant la mesure généralisée avec un nombre d'éléments jugés fixé à 50.

		nxCG[10]	**nxCG[25]**	**nxCG[50]**	**MAP**
	Base_Run	0,166	0,1737	0,1831	0,0541
Itération 1	**formule_1**	0,1758	0,18	0,1917	0,0558
	formule_2	0,1732	0,1816	0,1875	0,0553
	formule_3	0,1887	0,2032	0,2192	0,059
Itération 2	**formule_1**	0,179	0,1919	0,1987	0,0572
	formule_2	0,1823	0,1984	0,1974	0,0562
	formule_3	0,2117	0,2299	0,2446	0,0631
Itération 3	**formule_1**	0,1902	0,2023	0,2055	
	formule_2	0,1935	0,2102	0,1998	0,0574
	formule_3	0,2275	0,2457	0,2588	0,0662

min	0,1732	0,18	0,1875	0,0553
max	0,2275	0,2457	0,2588	0,0662
ARMin(%)	4,33	3,62	2,40	2,21
ARMax(%)	37,04	41,45	41,34	22,36

Tableau 5.3 : Résultats pour 50 éléments jugés (mesure généralisée)

Les résultats des trois formules sont tous meilleurs que ceux obtenus durant l'exécution de base. Nous remarquons que la formule 3 permet d'obtenir la meilleure AR (AR=41.45%). Cette amélioration est obtenue après 3 itérations pour la coupure nxCG[25]. Pour le MAP, nous avons obtenu une ARMax égale à 22.36% à la 3ème itération.

5.3.1.2. Mesure stricte

Le tableau 5.4 présente les résultats obtenus par les trois formules du ré-ordonnancement contextuel en utilisant la mesure stricte avec un nombre d'éléments jugés fixé à 20.

		nxCG[10]	**nxCG[25]**	**nxCG[50]**	**MAP**
	Base_Run	0,0115	0,0248	0,0444	0,01064
Itération 1	**formule_1**	0,0115	0,0232	0,0452	0,0109
	formule_2	0,0115	0,0282	0,0451	0,0109
	formule_3	0,0115	0,0345	0,0543	0,0114
Itération 2	**formule_1**	0,0115	0,0324	0,0428	0,011
	formule_2	0,0115	0,0339	0,0441	0,01113
	formule_3	0,0115	0,0465	0,0622	0,01229
Itération 3	**formule_1**	0,0115	0,0355	0,0403	0,0114
	formule_2	0,0115	0,037	0,0442	0,0115
	formule_3	0,0115	0,0542	0,0645	0,0129

min	0,0115	0,0232	0,0403	0,0109
max	0,0115	0,0542	0,0645	0,0129
ARMin(%)	0	-6,45	-9,23	2,44
ARMax(%)	0	118,54	45,27	21,24

Tableau 5.4 : Résultats pour 20 éléments jugés (mesure stricte)

Nous remarquons que la valeur n'a pas changé sur toute la colonne nxCG[10], ce qui veut dire que le changement s'effectue surtout en terme d'ordre et que les éléments très spécifiques et très exhaustifs restent toujours parce que l'utilisateur les a jugés pertinents.

Le fait d'avoir des AR négatives peut être interprété de la manière suivante : Quand on possède un ensemble d'éléments (appartenant à un document X1) jugés pertinents, mais qui ne contient qu'un seul élément très spécifique et très exhaustif (mesure stricte), les éléments appartenant à X1 seront favorisés ; alors que les éléments appartenant à un autre document X2, qui contient 2 éléments très spécifiques et très exhaustifs et quelques autres éléments non pertinents, seront moins favorisés par rapport à ceux appartenant à X1. (Dans ce même exemple nous allons remarquer une amélioration en utilisant la mesure généralisée et une baisse en utilisant la mesure stricte).

Nous remarquons que la formule 3 permet d'obtenir les meilleures AR pour tous les niveaux de coupure (nxCG[25], nxCG[50] et MAP). La meilleure AR correspond à nxCG[25] (AR=118.54%).

Le tableau 5.5 présente les résultats obtenus par les formules du ré-ordonnancement contextuel en utilisant la mesure stricte avec un nombre d'éléments jugés fixés à 50.

		nxCG[10]	**nxCG[25]**	**nxCG[50]**	**MAP**
	Base_Run	0,0115	0,0248	0,0444	0,01064
Itération 1	**formule_1**	0,0115	0,0232	0,0459	0,0108
	formule_2	0,0115	0,0282	0,0405	0,0107
	formule_3	0,0115	0,0302	0,0571	0,0123
Itération 2	**formule_1**	0,0115	0,0281	0,0508	0,011
	formule_2	0,0115	0,0355	0,0365	0,0111
	formule_3	0,0115	0,0391	0,0658	0,0137
Itération 3	**formule_1**	0,0115	0,0355	0,0469	0,0113
	formule_2	0,0115	0,037	0,038	0,01165
	formule_3	0,0115	0,052	0,0673	0,01465

min	0,0115	0,0232	0,0365	0,0107
max	0,0115	0,052	0,0673	0,01465
ARMin(%)	0	-6,45	-17,79	0,56
ARMax(%)	0	109,67	51,57	37,68

Tableau 5.5 : Résultats pour 50 éléments jugés (mesure stricte)

La même remarque sur la colonne nxCG[10] du tableau tableau 5.4 est valable pour le tableau 5.5.

Nous pouvons aussi constater les valeurs négatives de AR qui sont dues au phénomène déjà expliqué. Nous remarquons que la formule 3 donne les meilleures AR sur tous les niveaux (AR=109.67% pour nxCG[25]).

5.3.2. Ré-ordonnancement par nom de journal

Dans cette section nous présentons les résultats obtenus par l'application des différentes formules du ré-ordonnancement par nom de journal. Ces résultats sont évalués selon les mesures stricte et généralisée.

5.3.2.1. Mesure généralisée

Le tableau 5.6 présente les résultats obtenus avec les trois formules de calcul de coefficient du ré-ordonnancement par nom de journal en utilisant la mesure généralisée avec un nombre d'éléments jugés fixé à 20.

		nxCG[10]	nxCG[25]	nxCG[50]	MAP
	Base_Run	0,166	0,1737	0,1831	0,0541
Itération 1	**formule_1**	0,1747	0,1781	0,1841	0,0554
	formule_2	0,1737	0,1811	0,1877	0,0554
	formule_3	0,1783	0,1887	0,1997	0,0568
Itération 2	**formule_1**	0,1735	0,1893	0,1976	0,0564
	formule_2	0,1734	0,1968	0,1966	0,0558
	formule_3	0,191	0,2049	0,2116	0,0591
Itération 3	**formule_1**	0,1844	0,1987	0,2087	0,0573
	formule_2	0,1747	0,2015	0,2012	0,0558
	formule_3	0,2108	0,2159	0,2245	0,0608

min	0,1734	0,1781	0,1841	0,0554
max	0,2108	0,2159	0,2245	0,0608
ARMin(%)	4,45	2,53	0,54	2,40
ARMax(%)	26,98	24,29	22,61	12,38

Tableau 5.6 : Résultats pour 20 éléments jugés (mesure généralisée)

Nous remarquons que toutes les exécutions feedback sont meilleures que l'exécution de base. La meilleure AR constatée est celle de la formule 3 pour la coupure nxCG[10] (AR=26.98%). Nous remarquons que les meilleurs résultats ont été obtenus par la formule 3.

Le tableau 5.7 présente les résultats d'application des formules de ré-ordonnancement par nom de journal en utilisant la mesure généralisée avec un nombre d'éléments jugés fixé à 50.

		nxCG[10]	nxCG[25]	nxCG[50]	MAP
	Base_Run	0,166	0,1737	0,1831	0,0541
Itération 1	**formule_1**	0,1748	0,1808	0,1898	0,0556
	formule_2	0,1657	0,1774	0,1885	0,0551
	formule_3	0,1758	0,1893	0,205	0,0581
Itération 2	**formule_1**	0,173	0,1883	0,202	0,0568
	formule_2	0,1586	0,193	0,1958	0,0555

	formule_3	0,1883	0,1956	0,2181	0,0613
Itération 3	**formule_1**	0,1809	0,1969	0,2126	0,0579
	formule_2	0,1676	0,2113	0,2	0,0558
	formule_3	0,1974	0,2064	0,2264	0,0638

min	0,1586	0,1774	0,1885	0,0551
max	0,1974	0,2113	0,2264	0,0638
ARMin(%)	-4,45	2,13	2,94	1,84
ARMax(%)	18,91	21,64	23,64	17,92

Tableau 5.7 : Résultats pour 50 éléments jugés (mesure généralisée)

Nous remarquons que les résultats obtenus après feedback sont tous meilleurs que ceux obtenus durant l'exécution de base. La meilleure AR est obtenue par l'application de la formule 3 (après 3 itérations) sur nxCG[50] (AR=23.64%). Pour le MAP nous avons mesuré une amélioration de 17.92%.

5.3.2.2. Mesure stricte

Le tableau 5.8 présente les résultats obtenus par le ré-ordonnancement par nom de journal en utilisant la mesure stricte avec un nombre d'éléments jugés fixé à 20.

		nxCG[10]	**nxCG[25]**	**nxCG[50]**	**MAP**
	Base_Run	0,0115	0,0248	0,0444	0,01064
Itération 1	**formule_1**	0,0115	0,0177	0,0388	0,0108
	formule_2	0,0115	0,0192	0,0405	0,0107
	formule_3	0,0115	0,0302	0,0567	0,0116
Itération 2	**formule_1**	0,0115	0,025	0,0418	0,0112
	formule_2	0,0115	0,0339	0,0411	0,011
	formule_3	0,0115	0,0391	0,059	0,01257
Itération 3	**formule_1**	0,0115	0,0324	0,0746	0,0117
	formule_2	0,0115	0,0385	0,0443	0,011
	formule_3	0,0154	0,0449	0,1006	0,01334

min	0,0115	0,0177	0,0388	0,0107
max	0,0154	0,0449	0,1006	0,01334
ARMin(%)	0	-28,62	-12,61	0,56
ARMax(%)	33,91	81,04	126,57	25,37

Tableau 5.8 : Résultats pour 20 éléments jugés (mesure stricte)

Nous remarquons que les meilleures AR ont été obtenues par la formule 3 (3ème itération) et même sur la coupure nxCG[10]. Pour nxCG[50] la formule 3 a permis d'obtenir une AR égale à 126.57%. La même interprétation du phénomène des AR négatives lors du ré-ordonnancement contextuel peut aussi avoir lieu ici.

Le tableau 5.9 présente les résultats obtenus par l'application du ré-ordonnancement par nom de journal en utilisant la mesure stricte avec un nombre d'éléments jugés fixé à 50.

		nxCG[10]	nxCG[25]	nxCG[50]	MAP
	Base_Run	0,0115	0,0248	0,0444	0,01064
Itération 1	**formule_1**	0,0115	0,0232	0,0467	0,01099
	formule_2	0,0115	0,0208	0,0411	0,0106
	formule_3	0,0115	0,0302	0,0513	0,0118
Itération 2	**formule_1**	0,0115	0,0238	0,0515	0,0114
	formule_2	0,0115	0,0312	0,0411	0,0109
	formule_3	0,0115	0,0301	0,0575	0,0127
Itération 3	**formule_1**	0,0115	0,0327	0,0703	0,0119
	formule_2	0,0115	0,0432	0,0458	0,0113
	formule_3	0,0115	0,0318	0,0629	0,0135

min	0,0115	0,0208	0,0411	0,0106
max	0,0115	0,0432	0,0703	0,0135
ARMin(%)	0	-16,12	-7,43	-0,37
ARMax(%)	0	74,19	58,33	26,87

Tableau 5.9 : Résultats pour 50 éléments jugés (mesure stricte)

Nous pouvons constater que la formule 3 a permis d'obtenir la meilleure valeur AR pour le MAP (AR=26.87%) durant la 3ème itération. Les valeurs négatives des AR sont interprétées selon le phénomène déjà expliqué pour le tableau (5.4).

5.3.3. Etude de l'impact des différents facteurs

Dans cette partie nous allons étudier l'impact de quelques facteurs intervenant dans l'expérimentation. L'étude du premier facteur (introduction des éléments non pertinents) est une sorte de comparaison entre les trois formules de calcul de coefficient; alors que, pour les deux autres facteurs, il s'agit d'étudier le comportement des formules en variant des paramètres externes à ces dernières (nombre d'éléments jugés et nombre d'itérations).

5.3.3.1. Introduction des éléments non pertinents dans le calcul de coefficient

Nous pouvons diviser les formules proposées dans le chapitre précédent en deux catégories : formules qui intègrent l'information issue des éléments non pertinents (formules 1 et 3 des deux types de ré-ordonnancement) et les formules (formules 2) qui n'intègrent pas cette information. A travers ces expérimentations nous avons voulu étudier l'impact de l'introduction des éléments non pertinents. Ainsi, nous avons essayé de répondre à la question suivante : "*Est ce qu'il y a des cas pour lesquels nous pouvons dire qu'une des formules s'avère la plus adaptée ?*".

Selon les résultats obtenus (tableaux 5.2 à 5.9), la formule 3 semble la meilleure pour la plupart des requêtes utilisées dans l'expérimentation (sauf quelques unes). La formule 2 a donné les plus mauvais résultats (bien qu'ils soient meilleurs que ceux de l'exécution de base). Les résultats obtenus par la formule 1 sont meilleurs que ceux de la formule 2 mais inférieurs à ceux de la formule 3.

Cependant, il existe des cas pour lesquels ni la formule 2 ni la formule 3 ne peut améliorer les résultats. Un de ces cas est connu sous le nom du phénomène "*query drift* ". Ce phénomène survient lorsque la liste initiale extraite par le système de recherche ne comporte aucun élément pertinent. Dans ce type de cas, l'utilisation de l'information issue des éléments non pertinents

relève d'une importance sûre pour effectuer le ré-ordonnancement. Lors de notre expérimentation nous avons remarqué ce phénomène pour les requêtes : 205, 219, 233 et 241. L'utilisation des formules 2 et 3 sur ces requêtes n'a pas d'effet car le coefficient calculé est égal à 0. Donc, il est conseillé d'utiliser la formule 1 pour remédier à ce phénomène.

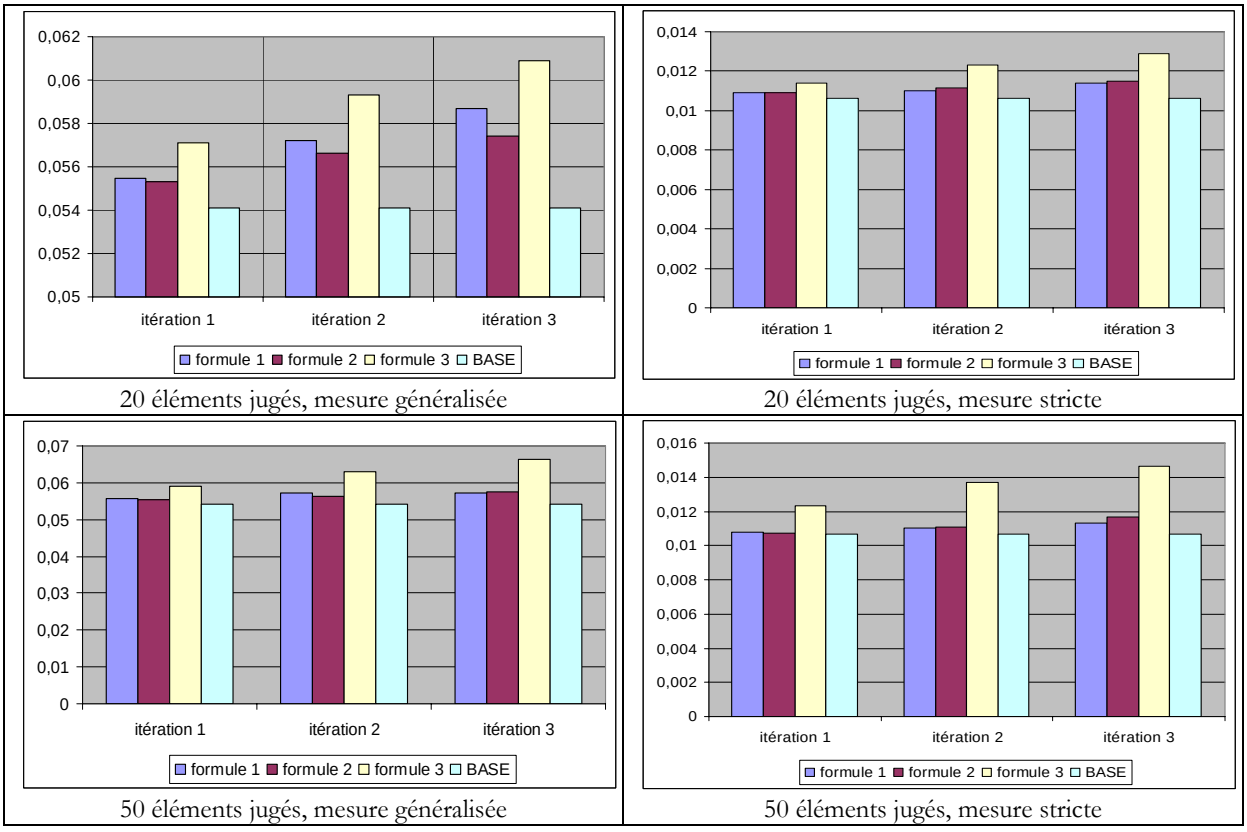
5.3.3.2. Variation du nombre d'éléments jugés

Selon les résultats obtenus (tableaux 5.2 à 5.9), nous pouvons constater que l'utilisation de 50 éléments jugés a permis aux formules d'avoir de meilleures performances que celles en utilisant 20 éléments jugés. Ceci peut être interprété par le fait qu'avec 50 éléments jugés le système aura une meilleure vision, c'est-à-dire plus de données, sur les documents (respectivement journaux), dans le cas du ré-ordonnancement contextuel (respectivement ré-ordonnancement par nom de journal), dans lesquels se trouvent les éléments pertinents. Cette observation est valable pour les deux types de ré-ordonnancement en utilisant les mesures stricte et généralisée sur tous les niveaux de coupures et pour toutes les itérations effectuées. Cependant, dans la pratique le choix de 50 éléments à juger par l'utilisateur n'est pas commode. Ainsi, l'utilisation de 20 éléments s'avère la mieux adaptée.

5.3.3.3. Nombre d'itérations

Le processus de reformulation de requêtes est un processus itératif. Pour cela, nous avons essayé d'étudier le comportement des différentes formules après chaque itération feedback.

La figure 5.3 présente l'évolution des résultats obtenus après chaque itération feedback de l'application du ré-ordonnancement contextuel.



20 éléments jugés, mesure généralisée

20 éléments jugés, mesure stricte

50 éléments jugés, mesure généralisée

50 éléments jugés, mesure stricte

Figure 5.3 : Ré-ordonnancement contextuel : Evolution des résultats après chaque itération feedback (mesure MAP)

D'après les graphes de la figure 5.3 (précédente), nous pouvons constater qu'à chaque nouvelle itération il y a une amélioration des résultats par rapport à l'itération qui la précède. Cette amélioration est valable pour le cas de 20 éléments jugés (respectivement 50 éléments jugés) et aussi pour le cas de la mesure généralisée (respectivement stricte).

La figure 5.4 présente l'évolution des résultats obtenus après chaque itération feedback de l'application du ré-ordonnancement par nom de journal.

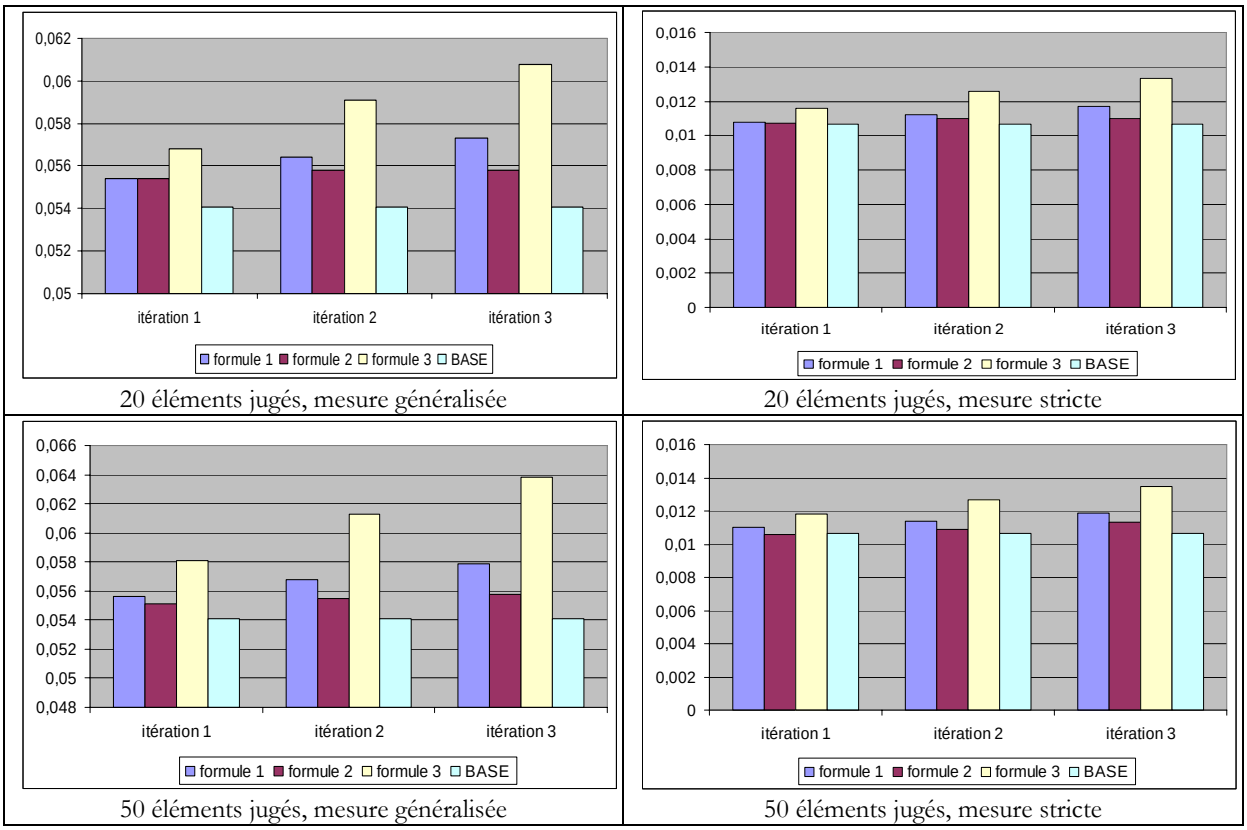


Figure 5.4 : Ré-ordonnancement par nom de journal; Evolution des résultats après chaque itération feedback (mesure MAP)

Nous pouvons constater d'après les graphes de la figure 5.4, que le comportement des formules est le suivant: à chaque nouvelle itération il y a lieu à une amélioration par rapport à l'itération précédente. Comme avec le ré-ordonnancement contextuel cette amélioration est valable aussi bien pour la mesure stricte que généralisée.

5.4. Conclusion

Tout au long de ce chapitre nous avons présenté les résultats des expérimentations effectuées pour évaluer les différentes propositions du chapitre 4. Les évaluations focalisent essentiellement sur les méthodes de ré-ordonnancement. Nous avons décrit en premier lieu l'environnement d'évaluation et les conditions expérimentales qui le caractérisent.

Nous avons organisé cette expérimentation en trois parties:

- Dans la première partie, nous avons évalué les formules du ré-ordonnancement contextuel. Cette évaluation nous a permis de constater que tous les résultats obtenus

après ré-ordonnancement sont meilleurs que ceux de l'exécution de base pour toutes les combinaisons envisagées.

- Dans la deuxième partie, nous avons évalué les formules du ré-ordonnancement par nom de journal. Nous avons remarqué aussi, pour ce type de ré-ordonnancement, une amélioration de la qualité des résultats.
- Enfin, nous avons consacré une troisième partie pour étudier l'impact de quelques facteurs :
    - Pour ce qui concerne l'introduction des éléments non pertinents dans le calcul de coefficient, nous avons constaté que cette information joue un rôle très important dans le processus de ré-ordonnancement car elle permet de défavoriser ces éléments non pertinents, ce qui permet d'améliorer la performance d'un système de recherche. Nous avons aussi constaté qu'il est indispensable d'utiliser des formules qui ont la forme de la formule 1 dans les cas où survient le phénomène "*query drift*";
    - Le deuxième facteur étudié (variation du nombre d'éléments jugés) nous a permis de constater que : "*plus on possède de données sur les éléments pertinents (ou non pertinents) plus nos chances sont meilleures pour améliorer la qualité des résultats*". Mais dans la pratique, on est obligé de choisir un nombre réduit d'éléments pour jugement par l'utilisateur;
    - Le dernier facteur (nombre d'itérations) nous a permis de constater que toutes les formules proposées améliorent les résultats à chaque nouvelle itération.

Même si les résultats obtenus s'avèrent d'une bonne qualité, cela ne peut pas nous garantir à 100% le comportement de notre proposition, car nos expérimentations ont été effectuées sur des données issues du système XFIRM. Or, pour garantir la vérification de ces formules nous devons diversifier les sources de données de test (données renvoyées par plusieurs systèmes).

Conclusion Générale

Synthèse

Notre travail se situe dans le contexte de la *recherche d'information*, plus particulièrement la recherche d'information dans des documents semi structurés de type XML. Actuellement, la recherche d'information a évolué de l'accès à un document ou un ensemble de documents vers l'accès à des informations (des nœuds de documents XML) répondant à un intérêt particulier de l'utilisateur. Dans le cadre de cette nouvelle problématique de nouveaux modèles ont été proposés afin d'exploiter l'information structurelle contenue.

La reformulation de requêtes est une phase importante dans les systèmes de recherche d'information. Elle permet en effet de récrire la requête de l'utilisateur selon les informations retrouvées par la requête initiale. De manière générale, ceci consiste, dans le cas notamment de la réinjection de la pertinence, d'extraire à partir des documents jugés pertinents par l'utilisateur, les mots clés importants puis les rajouter à la requête.

L'objectif nos travaux est de proposer une solution pour adapter ce processus bien connu et bien établi dans les systèmes de recherche d'information plein texte, à la recherche d'information dans les documents XML. Tout d'abord, en tenant compte du mode d'interrogation des documents XML : l'interrogation des ces documents peut être effectuée en n'utilisant que des mots clés (on parlera de requête CO (*Content Only*)) ou des requêtes comportant des mots clés et des contraintes de structure (CAS (*Content And Structure*)). La notion de réinjection de la pertinence doit donc intégrer également à la fois les notions de contenu et de structure.

En RI traditionnelle, l'unité documentaire jugée et donc à partir de laquelle les termes sont extraits, est le document entier. Or, dans le contexte de la recherche d'information structurée, recherche dans les documents XML par exemple, l'unité documentaire peut avoir différentes formes ; elle peut être le document entier ou encore tout élément du document.

La première problématique posée dans le cadre de ce mémoire était : comment effectuer une reformulation de requêtes par réinjection de pertinence dans le contexte de la RI structurée? Pour répondre à cette question, nous avons effectué un état de l'art sur les travaux de recherche menés sur l'utilisation de la technique de réinjection de pertinence dans ce contexte. Cette étude nous a permis de diviser les approches proposées en deux catégories : 1) Approches pour l'expansion de requêtes (*feedback effect*) ; 2) Approches pour le ré-ordonnancement de la liste des résultats (*re-rank effect*). Nous avons adopté la même classification pour nos propositions.

En ce qui concerne l'expansion de requêtes par ajout de nouveaux termes, nous avons pu répondre à la question majeure posée, à savoir : *« Comment extraire les meilleurs termes à partir d'unités d'information jugées pertinentes et non pertinentes par l'utilisateur, sachant que ces unités peuvent avoir des sémantiques différentes (ex : un paragraphe, une section, un titre), et peuvent être imbriquées les unes dans les autres? »*.

En ce qui concerne l'expansion de requêtes par ajout de contraintes structurelles, nous avons effectué une étude statistique sur résultats du système XFIRM **[Sauvagnat, 2005]** et les jugements de pertinence issus de la campagne INEX 2005 **[INEX, 2005]**, afin de répondre à la question suivante : « *est ce que l'ajout des contraintes de structure permet d'améliorer la qualité des résultats ?* ». La même étude nous a permis de confirmer l'existence de types d'éléments (balises) jugés souvent

pertinents et d'éléments jugés souvent non pertinents. A partir de cette étude, nous avons proposé une méthode basée sur la probabilité de pertinence d'un élément (d'une balise) pour sélectionner la contrainte structurelle adéquate à rajouter à la requête.

Dans la deuxième catégorie d'approches, nous avons décrit deux méthodes : la première, dite « *ré-ordonnancement contextuel*», est basée sur le contexte des éléments jugés ; la deuxième méthode, dite « *ré-ordonnancement par nom de journal* », est basée sur l'information structurelle « *Nom de journal* ». Les deux méthodes essayent de profiter de l'information structurelle caractérisant les éléments renvoyés dans la liste initiale retournée.

L'idée du ré-ordonnancement contextuel part du fait que le concepteur d'un document XML suit une certaine unité dans ses idées. En partant de l'hypothèse suivante : « *un élément qui appartient à un document qui contient beaucoup d'éléments jugés par l'utilisateur comme étant pertinents doit être mieux classé qu'un élément se trouvant dans un document contenant peu d'éléments jugés pertinents*». Cette idée a été traduite par le calcul d'un nouveau score de pertinence pour chaque élément retrouvé. Ce poids dépend de l'importance (coefficient) accordée par l'utilisateur à son contexte (document) après jugement.

Le ré-ordonnancement par nom de journal, propre à la campagne INEX, repose sur deux études effectuées dans **[Mihajlovic et al., 2005]** et **[Ramirez et al., 2004]**. Ces études nous ont conduit à l'hypothèse suivante: « *si un élément est jugé comme étant pertinent alors le « Journal » auquel il appartient peut contenir d'autres éléments susceptibles de comporter un contenu similaire* ». Cette information (nom de journal) peut être utilisée pour donner plus de poids aux journaux contenant des éléments jugés pertinents. Pour les deux types de ré-ordonnancement, nous avons envisagé plusieurs variantes pour le calcul de coefficient.

L'évaluation a porté seulement sur les méthodes de ré-ordonnancement car elles ne nécessitent que des résultats renvoyés par un système de recherche d'information dans des documents XML (XFIRM dans notre cas) et des jugements de pertinence issus de la campagne INEX. Afin de valider nos propositions, nous avons effectué une expérimentation. Cette expérimentation est organisée en trois parties: 1) évaluation des formules du ré-ordonnancement contextuel ; 2) évaluation des formules du ré-ordonnancement par nom de journal ; 3) étude de l'impact des facteurs : introduction des éléments non pertinents dans le calcul de coefficient ; variation du nombre d'éléments jugés ; et enfin le nombre d'itérations.

L'évaluation des formules du ré-ordonnancement contextuel nous a permis de constater que tous les résultats obtenus après ré-ordonnancement sont meilleurs que ceux de l'exécution de base pour toutes les combinaisons envisagées. Cette amélioration est aussi constatée pour le ré-ordonnancement par nom de journal. Nous avons constaté que l'introduction des éléments non pertinents dans le calcul de coefficient de pertinence (document/journal) joue un rôle très important dans le processus de ré-ordonnancement car il permet de les défavoriser, ce qui permet d'améliorer la performance d'un système de recherche. La variation du nombre d'éléments jugés nous a permis de constater que : "plus on possède de données sur les éléments pertinents (ou non pertinents) plus nos chances sont meilleures pour améliorer la qualité des résultats", alors que dans la pratique il est plus commode de choisir un nombre réduit d'éléments pour jugement par l'utilisateur. Le nombre d'itérations nous a permis de constater que toutes les formules proposées améliorent les résultats à chaque nouvelle itération.

Les résultats obtenus montre une bonne qualité de performance de nos méthodes (une AR (Amélioration Relative) qui varie entre -17,79% et 118,54% pour la mesure stricte et entre 2,21% et 41.45% pour la mesure généralisée dans le cas du ré-ordonnancement contextuel et entre -16,12% et 126,57% pour la mesure stricte et entre -4,45% et 26,98% pour la mesure

généralisée dans le cas du ré-ordonnancement par nom de journal). Nos résultats sont comparables à ceux obtenus par R. Schenkel et M. Theobald **[Schenkel and Theobald, 2005]**. Il reste cependant recommandable d'utiliser d'autres résultats issus de différents systèmes afin de garantir le comportement des formules proposées.

Perspectives

Comme suite à ce travail, nous pouvons envisager les perspectives suivantes :

- La pondération des termes lors du processus de reformulation de requêtes : dans notre méthode nous avons utilisé la formule proposée par Hlaoua et boughanem **[Hlaoua and Boughanem, 2005]**. D'autres formules peuvent être envisagées.

- La prise en compte de l'information « chemin d'un élément » : cette information peut être importante (selon l'étude effectuée par Schenkel et M. Theobald **[Schenkel and Theobald, 2005]**).

- Formules de calcul de coefficient : d'autres formules (méthodes de ré-ordonnancement) peuvent être proposées afin de prendre en compte les valeurs des deux dimensions (E,S) dans l'étape de jugement.

- L'aspect hétérogénéité : une amélioration peut être envisagée en intégrant une solution de traitement de cet aspect.

Bibliographie

[Aalbersberg, 1992] I. J. Aalbersberg. Incremental relevance feedback. Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pages 11-22. Copenhagen. 1992.

[Adamson and Boreham, 1974] G. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information storage and retrieval, 10 :253–60, 1974.

[Anderson and Pérez-Carballo, 2001] Anderson, J. and Pérez-Carballo, J. The nature of indexing : how humans and machines analyze messages and texts for retrieval : part ii : machine indexing, and the allocation of human versus machine effort. In Information Processing and Management 37, pages 255–277. Tarrytown, NY, USA, Pergamon Press, Inc. 2001.

[Amoussou] Joel Amoussou. Cours : Introduction à XML.efagroup.com.

[Bilodeau] Sylvain Bilodeau. Site de cours sur XQuery. http://www.a525g.com/programmation/xquery.htm.

[Boughanem et al., 2004] M. Boughanem, W. Kraaij, and J.-Y. Nie. Modèles de langue pour la recherche d'information. In Les systèmes de recherche d'informations, pages 163–182. Hermes-Lavoisier. 2004.

[Chang et al., 1971] Y.K. Chang, C. Cirillo and J. Razon. Evaluation of feedback retrieval using modified freezing, residual collection & test and control groups. The SMART retrieval system – experiments in automatic document processing. G. Salton (ed). 1971.

[Clark and Derose, 1999] J. Clark and S. Derose. XML Path Language (XPath), version 1.0. Technical report, World Wide Web Consortium (W3C), W3C Recommendation, http://www.w3c.org/TR/xpath. Novembre 1999.

[Croft and Harper, 1979] W. Croft and D. Harper. Using probabilistic models of information retrieval without relevance information. Journal of Documentation. pages 285-295. 1979.

[Crouch, 2005] Carolyn Crouch. Relevance Feedback at the INEX 2004 Workshop. INEX REPORT, In ACM SIGIR Forum. Vol 39, No 1. June 2005.

[Florescu and Kossmann, 1999] D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. IEEE Data Engineering Bulletin, 22(3) :27–34, 1999.

[Frakes, 1992] W. B. Frakes. Stemming Algorithms, pages 131–160. Frakes W B, Baeza-Yates R (eds) Prentice Hall, New jersey, 1992.

[Fuhr et al., 2002] N. Fuhr, N. Gövert, M. Abolhassani, and K. Grossjohann. Contentoriented XML retrieval with hyrex. In Proceedings of the first INEX Workshop, Dagstuhl, Germany. 2002.

[Fuller et al., 1993] 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.

[Grust, 2002] 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.

[Hanglin, 2004] Hanglin Pan. Relevance Feedback in XML Retrieval. In Proceedings of the ICDE/EDBT. Ph.D. Workshop, Boston, March 2004, pages 193-202. 2004.

[Hanglin et al., 2004] Hanglin Pan, Anja Theobald and Ralf Schenkel. Query Refinement by Relevance Feedback in an XML Retrieval System. Lecture Notes in Computer Science, ISSN: 0302-9743, Volume 3288. Springer-Verlag GmbH. 2004.

[Harman, 1988] D. Harman. Towards interactive query expansion. Proceedings of the Eleventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pages 321-331. Grenoble. 1988.

[Hiemstra, 2005] Djoerd Hiemstra, Vojkan Mihajlovic. The simplest evaluation measures for XML information retrieval that could possibly work. In INEX 2005 Workshop on element retrieval methodology (p6-13), Glasgow, Scotland. 2005.

[Hlaoua and Boughanem, 2005] Lobna Hlaoua and Mohand Boughanem. Towards Contextual and Structural Relevance Feedback in XML Retrieval. 2005.

[Ide, 1971] E Ide. New experiments in relevance feedback. The SMART retrieval system - experiments in automatic document processing. pages 337-354. 1971.

[INEX, 2005] INitiative for the Evaluation of XML Retrieval, disponible sur http://inex.is.informatik.uni-duisburg.de:2005/, 2006.

[INEX, 2006] INitiative for the Evaluation of XML Retrieval, disponible sur http://inex.is.informatik.uni-duisburg.de:2006/, 2006.

[Järvelin and Kekäläinen, 2002] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20(4) :422–446. 2002.

[Kakade and Raghavan, 2005] V. Kakade and P. Raghavan. Encoding XML in vector spaces. In Proceedings of ECIR 2005, Saint Jacques de Compostelle, Espagne. 2005.

[Kazai and Lalmas, 2005] Gabriella Kazai, Mounia Lalmas. INEX 2005 evaluation metric. In INEX 2005 Proceedings (p401-406), Dagstuhl,Allemagne, 2005.

[Kazai, 2003] Gabriella Kazai. Report of the INEX 2003 Metrics working group. In INEX 2003 Proceedings (p184-191), Dagstuhl,Allemagne. 2003.

[Kleinberg, 1999] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5) :604–632. September 1999.

[Lalmas, 2005] Mounia Lalmas. Structure/XML Retrieval. In Proceedings of ESSIR. 2005.

[Lalmas and Piwowarski, 2005] Mounia Lalmas, Benjamin Piwowarski. INEX 2005 Relevance Assessment Guide. In INEX 2005 Proceedings (p40-53), Dagstuhl,Allemagne. 2005.

[Lee et al., 1996] Y. Lee, S. Yoo, and K. Yoon. Index structures for structured documents. In ACM Workshop on XML and IR, Bethesda, pages 91–99. 1996.

[LONELY] The Lonely Planet WorldGuide. http://www.lonelyplanet.com/worldguide/

[Luhn, 1957] Luhn, H. A statistical approach to mechanized encoding and searching of literary information. IBM, 1(4) :309–317. 1957.

[Maron and Kuhns, 1960] Maron, M. and Kuhns, J. On relevance, probabilistic indexing and information retrieval. Journal of the Association for Computing Machinery, 7 :216–244. 1960.

[Mass and Mandelbrod, 2004] Y. Mass and M. Mandelbrod. Relevance Feedback for XML Retrieval. INEX 2004 Workshop Pre-Proceedings, (p154-157). 2004.

[Mihajlovic et al., 2004] Vojkan Mihajlovic, Georgina Ramirez, Arjen P. de Vries, Djoerd Hiemstra. TIJAH at INEX 2004: Modeling Phrases and Relevance Feedback. In INEX 2004 Proceedings, Dagstuhl, Allemagne. 2004.

[Mihajlovic et al., 2005] Vojkan Mihajlovic, Georgina Ramirez, Henk Ernst Blok, Thijs Westerveld, Arjen P. de Vries and Djoerd Hiemstra. TIJAH Scratches INEX 2005: Vague Element Selection, Overlap, Image Search, Relevance Feedback, and Users. In INEX 2005 Proceedings (p54-71), Dagstuhl, Allemagne. 2005.

[Olivier] Olivier Corby. Cours XML.http://www.inria.fr.

[Ponte and Croft, 1998] Ponte, J. and Croft, W. A language modelling approach to information retrieval. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 40–48. 1998.

[Porter, 1980] M. F. Porter. An algorithm for suffix stripping. Program 14. 1980.

[Ramirez et al., 2004] Georgina Ramirez, Thijs Westerveld and Arjen P. de Vries. Structural Features in Content Oriented XML retrieval. In INEX 2004. Dagstuhl,Allemagne. 2004.

[Ramirez et al., 2005] Georgina Ramirez, Thijs Westerveld and Arjen P. de Vries. Structural Features in Content Oriented XML retrieval. In CIKM'05 (ACM 1-59593-140-6/05/0010). Bremen, Germany. 2005.

[Ramirez and P. de Vries] Georgina Ramirez and Arjen P. de Vries. XML and Context Structural Features Relevant to Search Tasks.

[RF_TASK] Définition de la tâche RF. Site web de INEX 2004. http://inex.is.informatik.uni-duisburg.de :2004/

[Rijsbergen, 1979] C. van Rijsbergen. Information retrieval. Butterworths. 1979.

[Robertson, 1977] S. Robertson. The probability ranking principle in IR. Journal of Documentation, 33(4) :294–304. 1977.

[Robertson et al., 1994a] 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., 1994b] S. Robertson, S. Walker, S. Jones, and M. H.-B. andM. Gatford. Okapi at TREC 3. In Proceedings of the 3rd Text REtrieval Conference (TREC-3), pages 109–126. 1994.

[Robertson and Sparck-Jones, 1976] S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. Journal of the American Society of Information Science, pages 129–146. May–June 1976.

[Rocchio, 1971] J. Rocchio. Relevance feedback in information retrieval. The SMART Retrieval System: Experiments in Automatic Document Processing, pages 313–323. Prentice Hall, Englewood Cliffs, New Jersey, USA. 1971.

[Ruthven and Lalmas, 2003] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. Knowledge Engineering Review, 18(1). 2003.

[Salton, 1970] G. Salton. The SMART retrieval system : Experiments in automatic document processing. Prentice Hall. 1970.

[Salton, 1984] G. Salton and M. McGill. Introduction to modern information retrieval. McGraw-Hill Int. Book Co. 1984.

[Salton et al., 1983] Salton, G., Fox, E., and Wu, H. Extended boolean information retrieval. Communications of the ACM, 26(11) :1022–1036. 1983.

[Sauvagnat, 2005] Karen Sauvagnat. Modèle flexible pour la Recherche d'Information dans des corpus de documents semi structurés. Thèse de doctorat, IRIT, Université Paul Sabatier de Toulouse. 2005.

[Sauvagnat et al., 2005] Karen Sauvagnat, Lobna Hlaoua and Mohand Boughanem. XFIRM at INEX 2005 : ad-hoc, heterogeneous and relevance feedback tracks – Preliminary work. In INEX 2005 Proceedings (p72-83), Dagstuhl, Allemagne. 2005.

[Sauvagnat et Boughanem, 2006] Karen Sauvagnat, Mohand Boughanem. Propositions pour la pondération des termes et l'évaluation de la pertinence des éléments en recherche d'information structurée. Actes de CORIA 2006. Lyon – France. 2006.

[Schenkel et al., 2005] R. Schenkel, M. Theobald, and G. Weikum. XXL @ INEX 2003. In INEX 2003. Workshop Proceedings, pages 59–68. 2004.

[Schenkel and Theobald, 2005] Ralf Schenkel and Martin Theobald. Relevance Feedback for Structural Query Expansion. In INEX 2005 Proceedings (p260-273), Dagstuhl, Allemagne. 2005.

[SGML] SGML. http://www.w3.org/markup/sgml/

[Sigurbjörnsson et al., 2005] Börkur Sigurbjörnsson, Andrew Trotman, Shlomo Geva, Mounia Lalmas, Birger Larsen, Saadia Malik. INEX 2005 Guidelines for topic development INEX. In INEX 2005 Proceedings(p375-384), Dagstuhl,Allemagne. 2005.

[Singhal et al., 1996] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. Information Processing and Management, 32(5) :619–633. 1996.

[Shaw et al., 1997] 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.

[Spink, 1997] A. Spink. Study of interactive feedback during mediated information retrieval. Journal of the American Society for Information Science. 48. 5. pp 382-394. 1997.

[St. Laurent, 2000] Simon St. Laurent. Introduction au XML. Collection Micro Pro, OSMAN EYROLLES MULTIMEDIA. ISBN: 2-7464-0094-4. 2000.

[Sturm, 2000] Jake Sturm. Atelier XML. Collection Langages et programmation, Microsoft Press. ISBN: 2-84082-532-5. 2000.

[Tebri, 2004] Hamid TEBRI. Formalisation et spécification d'un système de filtrage incrémental d'information. Thèse de doctorat, IRIT, Université Paul Sabatier de Toulouse. 2004.

[TREC] TREC (Text REtrieval Conference). http ://trec.nist.gov

[Trotman and Lalmas, 2005] Andrew Trotman, Mounia Lalmas. The interpretation of CAS. In INEX 2005 Proceedings (p40-53), Dagstuhl,Allemagne. 2005.

[Trotman and Sigurbjörnsson, 2004] Andrew Trotman and Börkur Sigurbjörnsson. Narrowed extended XPath I (NEXI). In INEX 2003 proceedings (p219–237), Dagstuhl, Allemagne. december 2004.

[Van Zwol et al., 2005] Roelof Van Zwol, Mounia Lalmas, Gabriella Kazai. INEX 2005 Multimedia Track-working group. In INEX 2005 Proceedings (p411-417), Dagstuhl,Allemagne. 2005.

[Walker et al., 1997] 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.

[Wong et al., 1985] S. Wong, W. Ziarko, and P. Wong. Generalized vector space model in information retrieval. In Proceedings of the 8th ACM SIGIR Conference on Research and Development in information retrieval, New-York, USA, pages 18–25. 1985.

[W3C_DOM, 1998] W3C. DOM Level 1 (Document Object Model). Technical report, Wide Web Consortium (W3C), W3C standard. October 1998.

[W3C_QUERY] Groupe de travail « Query » W3C. http://www.w3.org/xml/query.html

[W3C_XML, 1998] W3C. eXtensible Markup Language (XML) 1.0. Technical report, Wide Web Consortium (W3C), Technical report. http://www.w3.org/xml/. february 1998.

[W3C_XPATH, 1999] XML Path Language (XPath) , version 1.0. , World Wide Web Consortium (W3C), W3C Recommendation, version française: http://xmlfr.org/W3C/TR/xpath/. Novembre 1999.

Annexe : Les requêtes de la tâche CO+S

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="202"  query_type="CO+S" ct_no="1" >
<InitialTopicStatement>I'm interested in knowing how ontologies are used to encode knowledge in real world scenarios. I'm writing a report on the use of ontologies. I'm particularly interested in knowing what sort or concepts and relations people use in their ontologies. </InitialTopicStatement>
<title>ontologies case study</title>
<castitle>//article[about(., ontologies)]//sec[about(., ontologies case study)]</castitle>
<description>Case studies in the use of ontologies</description>
<narrative>I'm writing a report on the use of ontologies. I'm interested in knowing how ontologies are used to encode knowledge in real world scenarios.  I'm particularly interested in knowing what sort or concepts and relations people use in their ontologies. I'm not interested in general ontology frameworks or technical details about tools for ontology creation or management. An example relevant result contains a description of the real world phenomena described by the ontology and also lists some of the concepts used and relations between concepts.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="203"  query_type="CO+S" ct_no="5" >
<InitialTopicStatement>Code signing is an approach for authenticating code based on public-key cryptography and digital signatures. The digital signature lets a user of the application code determine which particular key the code was signed with, and further ensures that the code has not been tampered with since it was signed. I am working in a company that authenticates a wide range of web database applications from different software vendors. I am looking for documents or document components that describe the code signing and verification approach. </InitialTopicStatement>
<title>code signing verification</title>
<castitle>//sec[about(., code signing verification)]</castitle>
<description>Find documents or document components, most probably sections, that describe the approach of code signing and verification. </description>
<narrative>I am working in a company that authenticates a wide range of web database applications from different software vendors. My work mainly focuses on the following two activities: checking whether the code that originates from a software vendor is authentic and properly signed, and checking whether the code has been tampered with since it was signed. I am looking for documents or document components that describe the approach of code signing and verification. To be relevant, a document or document component must describe the whole process of code signing and verification, which means ensuring that programs and program components have been created by trusted entities (by validating both the digital signature and the corresponding certificate), and that the programs have been received without tampering (by checking the main integrity of the program). Description of implementations of various approaches to code signing (such as Microsoft's Autheticode and Sun's JAR signing) are also relevant. A document or document component that only describe CRC-type integrity check of received programs will be considered only marginally relevant. Relevant information about this
```

```
topic can probably be found within the section components of the documents in the collection.
</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="204"  query_type="CO+S" ct_no="11" >
<InitialTopicStatement>Dan Moldovan</InitialTopicStatement>
<title>moldovan semantic networks</title>
<castitle>//*[about(.//au, moldovan) and about(., "semantic networks")]</castitle>
<description>Find information by Dan Moldovan on the construction or use of semantic networks.</description>
<narrative>I am building a question-answering system, and want to experiment with knowledge-intensive resources such as semantic networks.  My supervisors told me to be particularly mindful of the work of Dan Moldovan in this area, hence I want to get a good overview of his work in the literature. To be relevant, a retrieved item should be authored by Dan I. Moldovan (i.e., his name occurs in the //fm//au field), and discuss the construction or use of semantic networks.  Elements that are written by other authors are non-relevant, as are elements on other topics, or those that mention semantic networks only in passing.
</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="205"  query_type="CO+S" ct_no="12" >
<InitialTopicStatement>McLuhan</InitialTopicStatement>
<title>marshall mcluhan</title>
<castitle>//bdy//*[about(., "Marshall McLuhan")]</castitle>
<description>Find information about the relevance of Marshall McLuhan's ideas for current digital technologies.</description>
<narrative>I am writing an essay on the influence of new media icon Marshall McLuhan on digital technologies. I'm seeking information describing how McLuhan's views have influenced current digital technologies.  To be relevant, a retrieved item should discuss some aspect of Marshall McLuhan's visionary ideas or famous one-liners in the context of current digital technologies.  Retrieved elements that merely cite some of McLuhan's work are non-relevant, as are elements that discuss ideas not originating from McLuhan. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="206"  query_type="CO+S" ct_no="13" >
<InitialTopicStatement>What kind of new problems raises the miniaturization of microprocessors ?</InitialTopicStatement>
<title>problems physical limits miniaturization microprocessor</title>
<description>What are the problems and physical limits that are encountered by the microprocessor miniaturization process ?</description>
<narrative>I'm writing a report about future of microprocessors, and I wish to understand why the process of reduction of their size is soon expected to reach its limits.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="207"  query_type="CO+S" ct_no="14" >
<InitialTopicStatement> APIs "DOM and SAX"
The Document Object Model (DOM) defines an API for accessing and manipulating XML document as tree structures. The Simple API for XML (SAX) is an event-based API for reading XML documents. I need to read a collection of XML files and store them in a MYSQL relational databases and I wanted to get more information comparing DOM and SAX. </InitialTopicStatement>
<title>DOM and SAX</title>
<castitle>//*[about(., "DOM and SAX")]</castitle>
<description>I am interested to know more about DOM and SAX.</description>
<narrative>Because XML is becoming a larger part of application development everyday,many developers are struggling to understand the various XML constructs and Java APIs that manipulate them.I am interested to know more about DOM and SAX . This will help me for working with XML from within Java programs and application. Extending one of our group's IR engines for retrieval of XML documents is part of my project work. My java code should read and parse XML documents and then store them into MySQL databases. XML elements to be read in pre and post order,API's such DOM or SAX could help. I would like to know more about APIs that allows XML to be dealt with in Java simply and intuitively,without worrying about brackets and syntactical issues. To be relevant, a retrieved document should explain the strengths and weaknesses of both DOM and SAX ,parsing a document with DOM and SAX ,differences such as an event-based push model like in SAX or a tree-based pull model like in DOM and when to use any of them as an API for accessing and manipulating XML document. Some information about APIs in general could be also useful. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="208"  query_type="CO+S" ct_no="21" >
<InitialTopicStatement>I' like to learn more about the history of Artificial Intelligence.</InitialTopicStatement>
<title>"Artificial Intelligence" history</title>
<castitle>//article[about(., "Artificial Intelligence" history)]</castitle>
<description>Find an article dealing with history of Artificial Intelligence.</description>
<narrative>I am interested in history of computing, and especially of artificial intelligence, for my own culture. I do not want to write anything or to become an expert on the domain. I want to have a good idea of the stakes, hopes and first steps of this discipline during the first two or three decades. This is why any element dealing with this subject is relevant, but an entire survey article would be perfect. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="209"  query_type="CO+S" ct_no="23" >
<InitialTopicStatement>mining frequent pattern itemset sequence graph </InitialTopicStatement>
<title>mining frequent pattern itemset sequence graph association</title>
<description>Return all the articles related to how to mine frequent pattern (e.g. graph, itemset, sequence). Note that freqent itemset is usually referred as association rule mining. </description>
<narrative>Mining frequent itemset was a hot topic around ten years ago. We are interested in
```

knowing how that field is moving and writing a survey about it. All the articles related to frequent pattern mining would be relevant. Note that pattern can be referred as itemset, sequence or graph and frequent itermset mining is usually called as association rule mining.
</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="210" query_type="CO+S" ct_no="34" >
<InitialTopicStatement>I'm developing a new lecture for the Master course 'Content Design' and want to discuss the topic "Multimedia document models and authoring". Therefore I want to do a quick background search to collect relevant articles in a reader.</InitialTopicStatement>
<title>+multimedia "document models" "content authoring"</title>
<castitle>//article//(abs|sec)[about(.,+multimedia "document models" "content authoring")]</castitle>
<description>I'm searching for article sections or abstracts that deal with multimedia document models or approaches for multimedia content authoring.</description>
<narrative>I'm developing a new lecture for the Master course 'Content Design' and want to discuss the topic "Multimedia document models and authoring". Therefore I want to do a quick background search to collect relevant articles in a reader. I expect to find information in abstracts or sections of articles. Multimedia content is an essential component of my lecture, thus for fragments to be relevant they should address document models of content authoring approaches for multimedia content. I'm not interested in single media approaches or issues that discuss storing multimedia objects.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="211" query_type="CO+S" ct_no="37" >
<InitialTopicStatement>We are producing small gps units and want to explore new killer applications for mobile devices. </InitialTopicStatement>
<title>applications for mobile devices gps "global positioning system"</title>
<castitle>//article//sec[about(.//p, applications mobile devices gps "global positioning system")]</castitle>
<description>Find descriptions and examples of applications for mobile devices using global positioning systems (gps).</description>
<narrative>We are producing small gps units and want to explore new killer applications for mobile devices. Therefore we are looking for examples and descriptions of applications that use global positioning systems, also referred to as gps, for devices such as mobile phones, pda (personal desktop assistents) and other wireless and mobile devices. We are not interested in technical details of the working of gps-locators recievers and transmitters.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="212" query_type="CO+S" ct_no="49" >
<InitialTopicStatement>I'm writing a paper that uses hidden markov models and I'd like to see how others write the markov model equations, so that my notation may be somewhat consistent with the notation in other literature. I'm searching for components containing equations of hidden markov models and the corresponding text that describes the equation. I expect this topic to be fairly difficult to express in NEXI, yet still an important information

need.</InitialTopicStatement>
<title>HMM "hidden Markov model" equation</title>
<castitle>//*[(about(., HMM equation) OR about(., "hidden Markov model" equation)) AND .//en > 0]</castitle>
<description>Find text containing equations for a hidden Markov model (HMM).</description>
<narrative>I'm writing a paper that uses hidden markov models and I'd like to see how others write the markov model equations, so that my notation may be somewhat consistent with the notation in other literature. I'm searching for components containing equations of hidden Markov models (HMMs) and the corresponding text that describes the equation. </narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="213" query_type="CO+S" ct_no="50" >
<InitialTopicStatement>I want descriptions of the Gibbs sampling algorithm. I am interested in a description of the mathematics and theory behind it. I'm also interested in descriptions of when it is useful, or when another technique might work better.</InitialTopicStatement>
<title>Gibbs sampler</title>
<description>I'm looking for text about the Gibbs sampler.</description>
<narrative>I want descriptions of the Gibbs sampler. I am interested in a description of the mathematics and theory behind it, in particular what assumptions it makes in the approximation and its properties. I'm also interested in descriptions of when it is useful. For it to be highly relevant a description with mathematical formula must be present. Marginally relevant are mentions of the algorithm in passing, such as statements that the Gibbs sampler could be or was used in experiments to estimate a distribution. Fairly relevant text will have a slightly more detailed explanation, perhaps a brief description of the algorithm or some of its properties and how they affected the experiments/approach discussed. I'm a student writing a survey paper of distribution estimation techniques and wish to include the Gibbs sampler. The math and theory behind it is important as I wish to provide intuitions as well as redescribe it in a manner consistent with the notation of the paper I'm writing.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="214" query_type="CO+S" ct_no="51" >
<InitialTopicStatement>We are planning to realize an online course and want to gather information on interactive and adaptive aspects of an online learning system for educational purposes.</InitialTopicStatement>
<title>"adaptive learning" and "interactive learning" in education</title>
<description>Find techniques and approaches used for interactive and adaptive learning in an educational context.</description>
<narrative>We are planning to realize an online course and want to gather information on interactive and adaptive aspects of an online learning system for educational purposes. Relevant information discusses what are the possible techniques and approaches to realize a learning system that an interactive or adaptive component in the context of education. Fragments that discuss interactive or adaptive learning outside the educational context are not relevant. </narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>

```
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="215"  query_type="CO+S" ct_no="52" >
<InitialTopicStatement>I'm looking for any information about the Conference on Information Knowledge Management (CIKM).  Everything from calls for papers to descriptions of the conference after the fact.  I'm not interested references of papers that appeared in the conference, but transcripts of keynote speeches or panels are fine.  I'm also not interested in links to the web page, but submission deadlines are fine.</InitialTopicStatement>
<title>Conference on Information and Knowledge Management CIKM</title>
<description>I'm looking for just about any information about the Conference on Information and Knowledge Management (CIKM).</description>
<narrative>I'm looking for information about the Conference on Information Knowledge Management (CIKM).  Everything from calls for papers to descriptions of the conference after the fact.  I'm not interested references of papers that appeared in the conference, but transcripts of keynote speeches or panels are fine.  I'm also not interested in links to the web page, but submission deadlines are fine, as are vitae of people who served on the program committee in the past.  I'm a historian tasked with archiving details about CIKM that are not readily available in the conference proceedings.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="216"  query_type="CO+S" ct_no="58" >
<InitialTopicStatement>I would like to find more information on multimedia retrieval system architectures.</InitialTopicStatement>
<title>multimedia retrieval system architecture</title>
<castitle>//sec[about(., multimedia retrieval system architecture) or about(.//fig, multimedia retrieval architecture)]</castitle>
<description>I would like to find a document component containing textual description or a figure explaining the architecture of a multimedia retrieval system. </description>
<narrative>I am writing a short survey on multimedia retrieval systems and I would like to perform a broad search on multimedia retrieval system architectures in order to make a distinction between different types of architectures. To be relevant a document component must be about multimedia retrieval and provide information about the complete multimedia retrieval systems architecture or about the main architectural components. The document components with figures depicting different aspects of multimedia retrieval system architectures are considered highly relevant. The document components describing pure communication with multimedia servers are considered irrelevant. I would prefer to get as an answer (short) sections instead of a long documents describing details of a specific multimedia retrieval system architecture.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="217"  query_type="CO+S" ct_no="62" >
<InitialTopicStatement>Web site design is important for business and academic site to provide accessible content. User-centered design is one important approach to developing useful and usable web sites.</InitialTopicStatement>
<title>user-centered design of web sites</title>
<description>Find articles and sections on user-centered design for web sites</description>
<narrative>Our school teaches courses on proper design methods for information systems and web sites, and particularly for search systems on those sites. I would like to find additional
```

materials or examples for students to read in this area. Articles and sections will be considered relevant if they discuss user-centered design specifically for web sites. Components that are only about web design tools are not relevant. The most relevant will be entire articles on user-centered web site design.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="218" query_type="CO+S" ct_no="66" >
<InitialTopicStatement>I am a composer, and I want to know any possible method or way to use computers to help me in my job.</InitialTopicStatement>
<title>computer assisted composing music notes MIDI</title>
<description>Retrieve information about composing music with computer assistance. MIDI is a known method used in composing and note writing.</description>
<narrative>I am a composer, and I want to know any possible method or way to use computers to help me in my job. I?m not only interested in particular programs but also the possibilities computers might offer to a composer. I compose using notes so writing notes with computer would help me. On the other hand I?m using real instruments in composing, so a simple way of creating notes automatically while playing would be nice. I know there is a standard called MIDI, which is related for instance to this.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="219" query_type="CO+S" ct_no="70" >
<InitialTopicStatement>learning object granularity</InitialTopicStatement>
<title>learning object granularity</title>
<castitle>//sec[about(., learning object granularity)]</castitle>
<description>Find documents or document components, most probably sections, that discuss the granularity of learning objects.</description>
<narrative>I am a PhD candidate looking at effective retrieval of digital learning material. I would like to find as much information as possible to inform my research into effective learning object retrieval. The use of learning objects has been proposed as a method to enable the reuse and sharing of digital learning resources. A learning object is a digital resource able to be reused to support learning. The granularity or complexity of a learning object - for example a single image, an interactive exercise about a particular topic, or an entire course - affects the contexts in which the learning object can be reused. The term learning object is relatively new, and a number of other terms, as well as generic descriptions, have been used to describe the same concept. To be relevant, document components should discuss granularity in relation to learning objects, even if they are not named as such, and how granularity affects reuse. Document components about software development and reuse or machine learning are not relevant. Relevant document components are most probably sections.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="220" query_type="CO+S" ct_no="71" >
<InitialTopicStatement>image annotation ontology</InitialTopicStatement>
<title>image annotation ontology</title>
<castitle>//article[about(., image retrieval)]//sec[about(., annotation ontology)]</castitle>
<description>Use of ontologies to assist indexing or querying of annotations in image

retrieval.</description>
<narrative>I am supervising a new PhD student working on automated techniques for image annotation to assist image retrieval. I want to find existing work that specifically addresses how ontologies can be used to help annotate images for subsequent retrieval. To be relevant, a document component should describe or refer to manual or automatic image annotation techniques that specifically use ontologies. Relevant document components most probably represent section components, which are very likely contained by articles about image retrieval.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="221" query_type="CO+S" ct_no="74" >
<InitialTopicStatement>speech recognition software</InitialTopicStatement>
<title>capabilities limitations commercial speech recognition software</title>
<description>Return the names of commercial speech recognition software packages, along with the capabilities and limitations of each.</description>
<narrative>I'm hoping to generate transcripts from audio captured during meetings held through a VoIP tool. I'd like to compile a list of all commercially available speech recognition packages, along with a brief description of their capabilities and any experience (positive or negative) that others have had with them. To be relevant, the name of the software package must be given. Public domain systems are relevant only if they are suitable for commercial use.
</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="222" query_type="CO+S" ct_no="77" >
<InitialTopicStatement>eletronic commerce business</InitialTopicStatement>
<title>eletronic commerce business strategies</title>
<castitle>//article[about(. , bussiness strategies)]//sec[about(. , eletronic commerce e-commerce)]</castitle>
<description>I want to know concepts and some technical information about eletronic commerce. But is relevant articles that describes e-commerce as a business strategy.</description>
<narrative>I'm preparing a seminar about eletronic commerce for initial Information System academics. Therefore is necesssary to consider questions from the business strategies to the e-commerce system arquitecture. I'm not interesting in products and nor in emprise that offer e-commerce solutions. Sections are the most appropriate unit of retrieval, I thought. However the relevance of any articles must be judge from elements that discuss the sense of eletronic commerce. Recent informations will be very interesting.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="223" query_type="CO+S" ct_no="78" >
<InitialTopicStatement>Wireless ATM</InitialTopicStatement>
<title>wireless ATM multimedia</title>
<castitle>//article[about(.//sec, wireless ATM multimedia)]</castitle>
<description>I would like to know about the ATM technology used in the wireless network with applications multimedia.</description>

<narrative>I?m interesting in know how the ATM technology can help to supply the necessary support for wireless network with applications multimedia, for example: video conference and transmission of data in real time as well as the traditional services of access web, email and voice. I need these informations for do a dissertation, therefore concepts would be interesting. I?m not interesting in equipments specifications. I think sections are the best unit of retrieval.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="224" query_type="CO+S" ct_no="79" >
<InitialTopicStatement>I am interested in how to handle incomplete information in database systems. The aim of the search is to survey existing approaches for writing a technical paper related to this problem.
</InitialTopicStatement>
<title>incomplete information database</title>
<castitle>//article[about(.//bb, Lipski)]//*[about(., incomplete information database)]</castitle>
<description>I want to know how to handle incomplete information in database
systems. Relevant items probably cite Lipski.</description>
<narrative>I am interested in how to handle incomplete information in database systems. The aim of the search is to survey existing approaches for writing a technical paper related to this problem. To be relevant, a document/component must discuss the problem of storing, querying, or viewing incomplete information in databases, regardless of the architecture of database systems or what data types they are dealing with. Proper citations are also useful. I am not really interested in other than database technology even if it discusses some kind of incompleteness of data. I think it is useful for the search engine to look for citation to papers of Lipski.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="225" query_type="CO+S" ct_no="80" >
<InitialTopicStatement>I am interested in xml security technology. The aim of the search is to explore technical trends and challenges of this area.</InitialTopicStatement>
<title>xml security</title>
<castitle>//*[about(.//p, xml security)]</castitle>
<description>I want to know technical trends and challenges of xml security. I think these terms can be found in the same paragraph.</description>
<narrative>I am interested in xml security technology. The aim of the search is to explore technical trends and challenges of this area. To be relevant, a document/component must discuss the security consideration, the current status of standardization, or ongoing work in xml-related area, but any non-xml security discussion is irrelevant. Security includes digital signature, encryption, authentication, authorization and so on. I think the terms xml and security can be found in the same paragraph.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="226" query_type="CO+S" ct_no="81" >
<InitialTopicStatement>I am interested in corba technology. The aim of the search is to learn

```
the overview of the technology and its relationship with java. </InitialTopicStatement>
<title>corba java</title>
<castitle>//*[about(.//sec, corba java)]</castitle>
<description>I want to know corba technology, in addition to its relationship with java. I think these terms can be found in the same section.</description>
<narrative>I am a software developer and have to develop a distributed object system for the first time. Before doing that, I want to learn technology about corba. The aim of the search is to understand the overview of the technology and its relationship with java. To be relevant, a document/component must give any helpful comments on corba technology, such as a system architecture and a design guideline. Other distributed object technologies are also relevant if their relationships with java programming are described. I think the terms corba and java can be found in sections.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="227"  query_type="CO+S" ct_no="84" >
<InitialTopicStatement>Ensemble methods have been shown robust in many learning problems, especially when very little training data is available. A machine learning student prepares a survey about application of two well-known ensemble methods, namely Adaboost and/or Bagging, in this article collection.</InitialTopicStatement>
<title>Adaboost Bagging "ensemble learning"</title>
<description>Seek information about two well-known ensemble learning methods: Adaboost and Bagging. </description>
<narrative>We are encounting the learning problem when very little training data is available. Ensemble methods have been shown robust in the similar situation with noisy data. We are interested in two well-known methods of this approach: Adaboost and Bagging. To prepare for a survey article, any piece (except for bibliographic items) which provides hint at any detail level about one of them and their variants is considered relevant. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="228"  query_type="CO+S" ct_no="92" >
<InitialTopicStatement>"IPv6 deployment" "IPv6 support"</InitialTopicStatement>
<title>"IPv6 deployment" "IPv6 support"</title>
<castitle>//article[about(.//abs, IPv6)]//sec[about(., "IPv6 deployment") or about(., "IPv6 support")]</castitle>
<description>support and deployment of IPv6</description>
<narrative>I am interested in the support and deployment of IPv6. To be relevant a document or document component must be about IPv6 deployment or IPv6 support. I am interested in this subject, because I have heard a lot about IPv6, but very little about its deployment. I do not need this information for any perticular task. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="229"  query_type="CO+S" ct_no="96" >
<InitialTopicStatement>Looking for information on latent semantic analysis / latent semantic indexing. The work task is to find detailed information about the technique and its use.  This is for a iterature review for a project involving collecion selection in distributed IR.
```

```
</InitialTopicStatement>
<title>"latent semantic anlysis" "latent semantic indexing"</title>
<castitle>//article[about(.//bdy,"latent semantic analysis" "latent semantic indexing")]</castitle>
<description>I am looking for information in the body of the article about latent semantic analysis or about latent semantic indexing</description>
<narrative>Looking for information on latent semantic analysis / latent semantic indexing. The work task is to find detailed information about the technique and its use. This is for a iterature review for a project involving collecion selection in distributed IR. text in the body and back matter is saught. Bibliography entries are not of interest - we can already get references to article titles and abstracts through other search engines in the library with a more comprehensive collection coverage, including IEEE articles.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="230"  query_type="CO+S" ct_no="97" >
<InitialTopicStatement>I'm looking for applications of differential geometry in brain research. I have recently finished my thesis in a differential geometry related topic, and currently interested in finding a practical use for the subjects I investigated, especially in brain research. I am interested in documents/elements that describe, even in a basic manner, the applications and connection between the subjects, but I am not that interested in references, since I have most of them myself. </InitialTopicStatement>
<title>+brain research +"differential geometry"</title>
<castitle>//article//sec[about(.,brain research "differential geometry")]</castitle>
<description>I want to know about applications of differential geometry in brain research. I'm looking for relevant information within the content of the document.</description>
<narrative>I have just finished my Msc. in mathematics, in the field of differential geometry. My aim is to find possible implementations of my knowladge in current research. I'm mainly interested in applications in brain research. I'm interested in at least a short specification of the nature of implementation (e.g. what is the exact theory used, and to what purpose), hence the relevant elements should be sections, pagargaphs or even abstracts of documents, but in any case, should be part of the content of the document (as opposed to, say, vt, or bib).</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="231"  query_type="CO+S" ct_no="98" >
<InitialTopicStatement>I'm interested in the applications of markov chains in graph theory.
</InitialTopicStatement>
<title>markov chains in graph related algorithms</title>
<castitle>//article//sec[about(.,+"markov chains" +algorithm  +graphs)]</castitle>
<description>Retrieve information about the use of markov chains in graph theory and in graphs-related algorithms.</description>
<narrative>I have just finished my Msc. in mathematics, in the field of stochastic processes. My research was in a subject related to Markov chains. My aim is to find possible implementations of my knowledge in current research. I'm mainly interested in applications in graph theory, that is, algorithms related to graphs that use the theory of markov chains. I'm interested in at
 least a short specification of the nature of implementation (e.g. what is the exact theory used, and to which purpose), hence the relevant elements should be sections, pagargaphs or even abstracts of documents, but in any case, should be part of the content of the document
```

```
(as opposed to, say, vt, or bib).</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="232"  query_type="CO+S" ct_no="101" >
<InitialTopicStatement>I want to survey recent works based on Dempster-Shafer Theory, but only focus on the Database field (i.e., not in AI, IR, etc.). The paper should relate to a database core problem such as query processing, database physical design, etc. The problem should be modeled by Dempster-Shafer Theory, and the solution is shown with expressive experiment results. </InitialTopicStatement>
<title>Dempster Shafer theory Database experiment</title>
<castitle>//article[about(.//abs, Dempster-Shafer theory)]//sec[about(., Dempster Shafer database experiment)] </castitle>
<description>Find references that are about Dempster-Shafer Application, where the application is modeled by Dempster-Shafer theory, and it is a Database related problem (not AI, IR), and the solution is shown with expressive experiment results. </description>
<narrative>The Dempster-Shafer theory, also known as the theory of belief functions, is a generalization of the Bayesian theory of subjective probability. It gives a formal method for uncertain measurement. It has been extensively studied in AI, IR literatures. But our information need is nothing but only in the database filed. Suppose we are looking for Dempster-Shafer applications with Database context and experimental results, in order to prepare a overview. How the authors solve their specified applications is of our interest, such as physical database designing modeled by Dempster-Shafer theory, or resolving proposition conflicts in query plan by using the Dempster-Shafer's rule of combination, etc. </narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="233"  query_type="CO+S" ct_no="103" >
<InitialTopicStatement>I would like to find something about how synthesizers for music production are made, I would consider relevant both documents on traditional synthesizers as well as documents on how they can be emulated by computers</InitialTopicStatement>
<title>Synthesizers for music creation</title>
<castitle>//article[about (.//bdy, synthesizers) and about (.//bdy, music)]</castitle>
<description>I want to find documents describing the development of synthesizers for music creation, including documents describing the history of music synthesizers</description>
<narrative>My main interest in this field is to learn the basics about synthesizers, how they are made and what technologies they use. I am as interested in the old 60's synthesizers as newer ones, including computer programs developed for turning my PC into a synth. I would like this information for an introductory lecture on digitalisation of music. I will not be able to understand the very technical bits of an article, but, still, are interested in such articles if they also provide a lay-man description.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="234"  query_type="CO+S" ct_no="105" >
<InitialTopicStatement>I am planning to organize a workshop on multimedia and I want to collect information about other multimedia events (conferences, workshops, etc.) in order to decide the most appropiate topics and dates for my workshop. </InitialTopicStatement>
```

<title>"call for papers" conference workshop +multimedia</title>
<castitle>//article[about(.//atl,"upcoming events") OR about(.//atl,"call for papers")]//sec[about(., +multimedia conference workshop)]</castitle>
<description>I want information about conferences and workshops in the multimedia field. Relevant items might be in the articles titled "call for papers" or "upcoming events".</description>
<narrative>I am starting to organize a workshop on multimedia and I am collecting information about other multimedia events (mainly conferences and workshops) in order to decide the most appropiate topics and dates for my workshop. Relevant items mention conferences or workshops in a mutlimedia related field with their topics of interest and dates. Components that only mention the name of the conference are somehow relevant because I can look for further information myself. Components mentioning conferences outside the multimedia field or articles about multimedia are not relevant. </narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="235" query_type="CO+S" ct_no="106" >
<InitialTopicStatement>I am searching for information concerning the FBI and CIA monitoring the public, and how it affects people's privacy. Since I am writing a feature article for a newspaper I want a general overview of the area, and I am not concerned with the technical details.</InitialTopicStatement>
<title>"Central Intelligence Agency" "Federal Bureau of Investigation" personal privacy surveillance concerns +Carnivore</title>
<description>I want to know what personal privacy concerns have been raised by the surveillance of the Central Intelligence Agency and he Federal Bureau of Investigation. I cam particularly interested in project Carnivore.</description>
<narrative>I am a journalist writing a feature article about the Carnivore surveillance project. I am interested in how the surveillance of the FBI and CIA has affected people's privacy and what concerned have been raised. I am not interested in the technical details of the project.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="236" query_type="CO+S" ct_no="110" >
<InitialTopicStatement>I want to find information about approaches on machine translation.</InitialTopicStatement>
<title>machine translation approaches -programming</title>
<castitle>//article[about(., machine translation approaches -programming)]</castitle>
<description>I am interested in articles discussing machine translation approaches but not programming languages translation approaches. </description>
<narrative>I am starting a survey on the field of machine translation and I am looking for information about existing approaches. A relevant component discusses one or more approaches on automatic speech or text translation. A relevant component might also be a pointer to relevant literature on the field. Components discussing automatic translations of programming languages, specification languages or ontologies are not relevant. </narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >

<inex_topic topic_id="237" query_type="CO+S" ct_no="116" >
<InitialTopicStatement>Searching for literature on NLP techniques as an Intelligent Information Retrieval approach in the Medical Informatics reasearch arena.</InitialTopicStatement>
<title>"Natural Language Processing" techniques "Artificial Intelligence" "Intelligent Information Retrieval" +"Medical Informatics" </title>
<description>Use of natural language processing techniques and/or artificial intelligence techniques as a Intelligent Information Retrieval approach to discover information and knowledge from Medical Informatics Literature </description>
<narrative>I am only interested in techniques of Natural Language Processing (NLP)
as an Intelligent Information Retrieval approach for the Medical Informatics research arena. I need to search for available literature to feed my research interest on Bioinformatics and help complete some of my research papers. The NLP techniques in Medical Informatics are of important significance as a majority of information stored in Medical literature was in the natural language context. </narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="238" query_type="CO+S" ct_no="123" >
<InitialTopicStatement>The topic should return articles that were published before or during the year 2000 and contain an algorithm for playing chess.</InitialTopicStatement>
<title>neural network algorithm for chess</title>
<castitle>//article[about(.//bdy, "artificial intelligence") and .//yr<=2000]//bdy[about(., chess) and about(., algorithm)]</castitle>
<description>Return artificial-intelligence algorithms, for playing chess, that where published before or during the year 2000.</description>
<narrative>I am writing a program that plays chess and I want the program to use artificial-intelligence algorithms. Therefore, I am interested in all kinds of algorithms that had been proposed for chess. I assume, however, that earlier algorithms are simpler and it will be easier for me to implement them. Hence, I am only interested in articles that were published before or during the year 2000. Relevant articles should include an algorithm for playing chess using artificial-intelligence.</narrative>
</inex_topic>

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="239" query_type="CO+S" ct_no="124" >
<InitialTopicStatement>The topic is a request for an article, published in either the year 2000 or the year 2001, that deals with quantum computation and is not about quantum mechanics.</InitialTopicStatement>
<title>quantum computation</title>
<castitle>//article[about(.//bdy//sec, quantum computation) and (.//yr=2000 or .//yr=2001) and about(.//(atl|abs|kwd), - mechanics)] </castitle>
<description>Look for an article, published in either the year 2000 or the year 2001, that deals with quantum computation and is not about quantum mechanics.</description>
<narrative>I am writing a paper about quantum computation, and I remember that I read a very good article about quantum computation in either 2000 or 2001. My initial search for the article returned many articles that deal with quantum mechanics, but hardly reffering to quantum computation. I wish to find the article that I read in the past, or similar articles. Relevant articles should deal with quantum computation. They may mention quantum mechanics, however,

```
quantum mechanics should not be their main topic.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="240"  query_type="CO+S" ct_no="127" >
<InitialTopicStatement>Find sections regarding software quality in the articles that deal with quality measurement and control. As a hint, we will possibly look for words such as quality, control and measure under keyword or abstract tags.</InitialTopicStatement>
<title>Software quality control and measurement</title>
<castitle>//article[about(.//(abs|kwd),quality control measure)]//sec[about(.//p,software quality)]</castitle>
<description>Find sections regarding software quality in articles that deal with quality measurement and control. As a hint, we will possibly look for words such as quality, control and measure under keyword or abstract tags.</description>
<narrative>A candidate for a quality-manager job is getting prepared for a job interview and she is interested in getting a basic idea about product management.She seeks for all sections about software quality in the context of how to control and measure it.Relevant sections should have at least one paragraph referring to the concept of software quality.</narrative>
</inex_topic>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd" >
<inex_topic topic_id="241"  query_type="CO+S" ct_no="138" >
<InitialTopicStatement>Find articles about LDAP and single sign on procedures.</InitialTopicStatement>
<title>Single sign on + LDAP</title>
<description>Find articles about LDAP and single sign on procedures. </description>
<narrative>In my daily work I need to sign on to a range of different systems both locally and remote. On many of them I have different userids and different passwords or PINs. With more and more systems coming into reach via the Internet using them daily it is highly annoying to have to verify your identity again and again when using different application in one session. In addition it is demanding to maintain all these ids and passwords and to keep them secure. I have previously come across LDAP and other single sign on procedures. I wish to learn more about them so that I can assess the potentials for creating a single sign on procedure for our local network (with both unix, linux, pc and mac platforms). Therefore single sign on is the most important part of the query. I am also looking for articles that describe the state of the situation as to user authentication methods. Articles dealing with the battle between proprietary and open standards would be especially useful. </narrative>
</inex_topic>
```