REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
• X •ΘΛ•ΣX [://:V •X[•Λ[•O
FACULTE DU GENIE ELECTRIQUE
ET INFORMATIQUE

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en informatique

« Option : Systèmes Informatiques »

Thème

Implémentation d'une méthode d'exploitation des liens dans les documents XML

Réalisé par :

M^elle Lynda MESTOUR
M^r Slimane BERKANI

Proposé et Dirigé par :

M^me S.FELLAG

Année universitaire 2011/2012

Sommaire

Introduction générale

1. Contexte du travail 1

2. Problématique 1

3. Contribution 2

4. Organisation du mémoire 2

Chapitre I : Généralités sur la Recherche d'Information

I.1 Introduction 4
I.2 Le processus de Recherche d'Information 4
I.2.1 Concepts de base de la RI 5
I.2.1.1 Le document 5
I.2.1.2 La requête 6
I.2.1.3 Notion de pertinence 6
I.2.2 Principales phases du processus de RI 6
I.2.2.1 L'indexation 7
I.2.2.2 La fonction d'appariement requête-document 9
I.2.2.3 Reformulation de la requête 9
I.3 Modèles de Recherche d'Information 10
I.3.1 Le modèle booléen 11
I.3.2 Le modèle vectoriel 11
I.4 Paramètres d'évaluation d'un SRI 13
I.4.1 Rappel et précision 14
I.4.2 Précision à X 16
I.5 Conclusion 17

Chapitre II : Recherche d'information Structurée

II.1 Introduction 18
II.2 Présentation de XML 18
II.2.1 Structure d'un document XML 19
II.2.1.1 Le Prologue 19

II.2.1.2 L'arbre d'éléments 20
II.2.1.3 Les commentaire 23
II.3 Déclaration de type de document (DTD) 23
II.3.1 La DTD interne 23
II.3.2 La DTD externe 23
II.3.3 Exemple d'un document avec sa DTD associée 24
II.4 Analyseur (Parser) XML 24
II.4.1 Qu'appelle-t-on un Analyseur 24
II.4.2 DOM (Document Object Model) 25
II.4.3 SAX (Simple API for XML) 26
II.5 Problématiques de la RI structurée 26
II.6 Approches de RI dans les documents XML 27
II.6.1 Les approches orientées données 27
II.6.2 Les approches orientées documents 28
II.6.3 Interrogation d'un document XML 28
II.7 La compagne d'évaluation INEX 29
II.7.1 La collection de test 29
II.7.2 Les requêtes 29
II.7.3 La recherche Ah-doc 30
II.7.4 Mesure d'évaluation 31
II.8 Conclusion 33

Chapitre III : Généralités sur les liens

III.1 Introduction 34
III.2 Définition des liens Hypertextes 34
III.3 Types de liens hypertextes 34
III.4 Comportement d'un lien 35
III.5 Les liens XML 35
III.5.1 XLink 35
III.5.1.1 Principales caractéristiques 35
III.5.1.2 types de liens 36
III.5.1.3 Comportement d'un lien XLink 38
III.5.2 XPointer 39
III.5.2.1 Syntaxe 40
III.5.2.2 Localisation 40
III.5.2.3 Les différents **types de localisation** 41
III.6 Conclusion 42

Chapitre IV : Algorithmes d'exploitation des liens

IV.1 Introduction 43
IV.2 Exploitation des liens dans la RI 43
IV.2.1 Page Rank 43
IV.2.1.1 Formule de Page Rank 44
IV.2.2 HITS 45
IV.2.3 L'algorithme Salsa 46
IV.2.4 L'activation propagée 47
IV.3 Exploitation des liens dans la RIS 48
IV.3.1 XRank 48
IV.3.1.1 ElemRank 48
IV.3.2 DocRank 49
IV.4 Autres approches 49
IV.5 Notre approche 50
IV.6 Conclusion 50

Chapitre V : Notre approche pour l'exploitation des liens

V.1 Introduction 51
V.2 Les principales étapes 51
V.2.1 Indexation 51
V.2.1.1 Indexation des titres 51
V.2.1.2 Indexation des liens 51
V.2.1.3 Exemple d'indexation 52
V.3 Présentation de la formule 53
V.4 Algorithme de recherche 55
V.5 Exemple d'application de l'approche 56
V.5.1 Processus de propagation de score 59
V.6 Conclusion 61

Chapitre VI : Implémentation et expérimentation

VI.1 Introduction 62
VI.2 Description de l'environnement de travail 62
VI.2.1 Sous Linux 62
VI.2.1 Sous Windows 62
VI.3 Architecture du modèle de liens 62
VI.4 Utilisation des liens pour le réordonnancement des résultats (Reranking) 63
VI.5 Implémentation de l'approche 64
VI.5.1 Notre travail par étape 66
VI.6 Expérimentation 66
VI.6.1 La collection INEX 2006 66
VI.6.2 Protocole d'évaluation 67
VI.6.2.1 Les principales tâches d'évaluation 67
VI.6.2.2 Les mesure d'évaluation 67
VI.6.2.3 Requête et paramètres utilisée 70
VI.6.2.4 Outil d'évaluation 71
VI.6.3 Résultats de l'évaluation 71
VI.6.3.1 Résultats de l'évaluation en calculant le nxCG 71
VI.6.3.2 Résultats de l'évaluation en utilisant EvalJ 72

VI.7 Conclusion 73

Conclusion et perspectives

Conclusion générale 74
Perspectives 75
Annexe 76
Bibliographie 93

LISTE DES FIGURES

Figure 1.1 : Processus en U de recherche d'information..... 5

Figure 1.2 : Bruit et Silence..... 15

Figure 1.3 : Courbe Rappel-Précision 16

Figure 2.1 : L'arbre éléments d'un document XML..... 21

Figure 2.2 : Exemple d'arbre DOM. 26

Figure 2.3 Exemple d'une requête INEX 2007. 30

Figure 3.1: Exemple d'un XLink simple..... 36

Figure 3.2: Exemple de liens XLink étendus..... 37

Figure 3.3 : Exemple de liens étendus en ligne 37

Figure 3.4 : Exemple de liens étendus hors ligne 38

Figure 3.5: Exemple de localisation dans un document XML 40

Figure 4.1: Exemple des pages pivots (Hubs) et autorités 45

Figure 4.2: Graphe biparti de Salsa..... 47

Figure 5.1 : Exemple d'un document XML a indexé..... 52

Figure 5.2 : Document 1.xml..... 57

Figure 5.3 : Arbre de référence entre documents..... 58

Figure 6.1 : Architecture du système d'exploitation de liens 63

Figure 6.2 : Exemple de réordonnancement des documents pour la collection INEX 2006 requête 289..... 64

RÉSUMÉ

Dans le cadre de la recherche d'information structuré on s'est intéressé à l'exploitation des liens dans la recherche d'information structuré ; pour non seulement permettre un accès aux documents non retournés dans une première recherche classique sans prise en compte de liens mais aussi de réaliser un ré-ordonnancement des documents retournés dans cette phase suivant leurs valeur de pertinence.

Dans ce mémoire nous présentons un modèle de recherche basé sur une vision qui consiste à exploiter deux informations considérées comme étant des facteurs importants pour la description du contenu informationnel d'un document à savoir le titre du document et le texte ancre des liens, l'information contenue dans ces derniers est utilisée pour calculer les valeurs de pertinence de chaque document, ces valeurs sont propagées vers les documents voisin suivant plusieurs niveaux, ainsi et grâce à ces valeurs un ré-ordonnancement est effectué après chaque requête utilisateur. Enfin notre modèle est évalué dans le cadre de la compagne d'évaluation INEX 2006.

Mots clés: recherche d'information, recherche d'information structurée, XML, prise en compte des liens, texte ancre du lien, balise titre, propagation de pertinence, INEX.

ABSTRACT

Within the context of information retrieval in structured documents we were interested in the exploitation of links in research of structured information, for not only allowing one access to the documents not turned over in a first traditional research without taking into account of links, but also to make a regrouping of the documents turned over in this phase according to their value of relevance.

In this thesis we present a model of research based on a vision which consists in exploiting two information considered as significant factor for the description of the informational contents of a document such as the title of the document and the text anchors of links, the information contained in the latter is used to calculate the values of relevance of each document, these values are propagated towards the neighbor documents according to several levels, thus and thanks to these values a regrouping is made after each request user. Finally our model is evaluated within the partner of evaluation INEX 2006.

Key-words: information retrieval, information retrieval in structured documents, XML, link analysis, link anchor text, title tag, relevance propagation, INEX.

1. Contexte du travail

Les systèmes de recherche d'information (SRI) sont conçus, à l'origine, pour répondre aux besoins d'automatiser la gestion de la documentation. Avec l'avènement d'Internet, le volume des documents et le nombre de personnes à gérer se sont accrus de manière vertigineuse ; Les SRI sont alors confrontés à un défi dû à la disparité et à la quantité des types de documents à gérer autant qu'à la multiplicité des demandes des utilisateurs.

La recherche d'informations sur le web est basée sur les liens hypertextes. En effet, ces derniers permettent de lier les pages web entre elles, ce qui octroie aux utilisateurs, la possibilité de cliquer sur ces liens pour naviguer d'une page à une autre. Des techniques d'analyse de liens ont été développées, pour améliorer les performances de la recherche d'information sur le web, en calculant une valeur de pertinence d'un document, en fonction non pas de son contenu seul, mais également en fonction de son voisinage (documents reliés par des liens hypertextes). En effet, la plupart de ces techniques, tiennent compte de la notion de popularité d'une page: *"une page référencée par plusieurs pages est une bonne page"*. Les deux algorithmes les plus populaires, qui ont exploité cette notion sont PageRank (Brin et al., 1998), qui a fait la notoriété du moteur de recherche Google, et HITS (Kleinberg, 1999) de Jon kleinberg.

La communauté (RIS) ou RI dans les documents structurés, plus précisément XML, s'est intéressée elle aussi à l'exploitation des liens dans ses dits documents. Rappelons toutefois que, dans la RIS, le granule d'information restitué suite à une requête donnée n'est plus le document entier, mais une partie de celui-ci en l'occurrence *l'élément*.

C'est dans ce cadre que se situe notre travail, qui consiste à implémenter une méthode d'exploitation de liens pour la recherche dans les documents XML.

2. Problématique

Avec l'essor du web et avec l'explosion des ressources d'information disponibles et leur hétérogénéité, la RI a été propulsée au premier plan. Plusieurs moteurs de recherche sont ainsi apparus permettant l'accès à l'information à grande échelle, et des systèmes de recherche d'information (SRI) ont été crées. L'objectif d'un SRI est donc d'une part de permettre aux utilisateurs d'exprimer facilement leurs besoins en information au moyen des requêtes, et d'autre part de leur fournir les documents potentiellement pertinents par rapport aux besoins en information transmis au système.

Cette explosion du web a néanmoins fait que la RI se retrouve confrontée à un problème: collections gigantesques, diversifiées et surabondance de l'information. Ainsi, la tâche principale de la RI qui est de retrouver l'information pertinente à une requête devient difficile à accomplir.

Dans le but d'améliorer la qualité des SRI, plusieurs facteurs peuvent être pris en compte, on cite par exemple les liens de citation et de référence. La problématique qui se pose alors est:

« *Comment exploiter l'information des liens pour améliorer la réponse apportée à l'utilisateur?* ».

Plusieurs travaux ont été entrepris dans ce sens et ont montré leur intérêt. Ces travaux exploitent pour la plupart les liens entrants et sortants pour réordonner les résultats obtenus suite à une requête. Dans ce mémoire, nous implémentons une nouvelle approche pour répondre à cette problématique, qui consiste à considérer en plus des liens entrants et sortants, le texte ancre des liens et le titre des documents.

3. Contribution

Notre contribution dans le cadre de la RI structurée se décline en trois principaux points :

1. Réalisation d'un état de l'art sur la recherche d'information classique et structurée, on y présente notamment les travaux relatifs à la prise en compte des liens dans la recherche d'information;

2. Implémentation d'une nouvelle méthode pour la prise en compte des liens dans la recherche d'information qui tient compte, en plus des liens entrants et sortants, du texte ancre des liens, de la balise titre des documents XML, plus précisément les éléments, et la propagation des termes.

3. Évaluation de ce modèle sur la collection de test INEX 2006 en faisant une comparaison des niveaux de précision entre les résultats initiaux retournés par le système Maxplank et notre modèle qui prend en compte les liens.

4. Organisation du mémoire

Ce mémoire s'articule autour de trois parties : la recherche d'information, l'exploitation des liens et enfin la proposition d'une méthode pour l'exploitation des liens dans les documents XML.

Dans le **chapitre I**, nous introduisons la recherche d'information et le processus de recherche d'information. On présente par la suite les différents modèles de la RI et les principaux paramètres d'évaluation d'un SRI.

Dans le **chapitre II,** Nous débutons par la présentation des documents XML, ainsi que de quelques technologies se rapportant à XML (les autres sont traitées en annexe A). Ensuite, nous identifions les problématiques qu'engendre la prise en compte de la structure dans le processus de RI et les différentes solutions proposées. Nous abordons enfin, les aspects relatifs à l'évaluation des systèmes de recherche d'information XML où il est question de présenter la compagne d'évaluation INEX.

La seconde partie de ce mémoire est consacrée à l'exploitation des liens dans les documents XML.

Dans le **chapitre III**, nous présentons la notion de liens de façon générale, les liens dans les documents XML ainsi que les notions Xlink et Xpointer.

Le **chapitre IV** est consacré à la description des principaux algorithmes d'exploitation de liens dans la recherche d'information structurée.

La troisième partie inclut deux chapitres. Dans le **chapitre V,** nous présentons notre modèle pour la prise en compte de liens dans la recherche d'information. On décrit la procédure d'indexation et par r la suite notre formule qui tient compte des titres des documents, des liens et du texte ancre des liens.

Le **chapitre VI** clôt ce mémoire où, on présente l'implémentation et les résultats de l'évaluation du modèle étudié sur la collection de test INEX 2006.

I.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, en se basant sur le profil utilisateur, 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 a pour but de présenter les principaux concepts de la RI, les principaux modèles de recherche ainsi que les approches d'évaluation des SRI. Nous commençons tout d'abord par la description des étapes d'un processus de RI. Nous passons ensuite en revue les différents modèles de recherche. Enfin, nous présentons les plus importants critères d'évaluation d'un SRI.

I.2 Le processus de Recherche d'Information :

Le but d'un SRI est de mettre en correspondance des informations disponibles d'une part, et les requêtes (besoins) de l'utilisateur d'autre part. Ce processus cherche alors à mettre en relation des besoins utilisateurs et des informations, afin que le SRI, retourne à l'utilisateur le maximum de documents pertinents par rapport à son besoin en information et le minimum de documents non-pertinents (un document est pertinent s'il satisfait l'utilisateur).

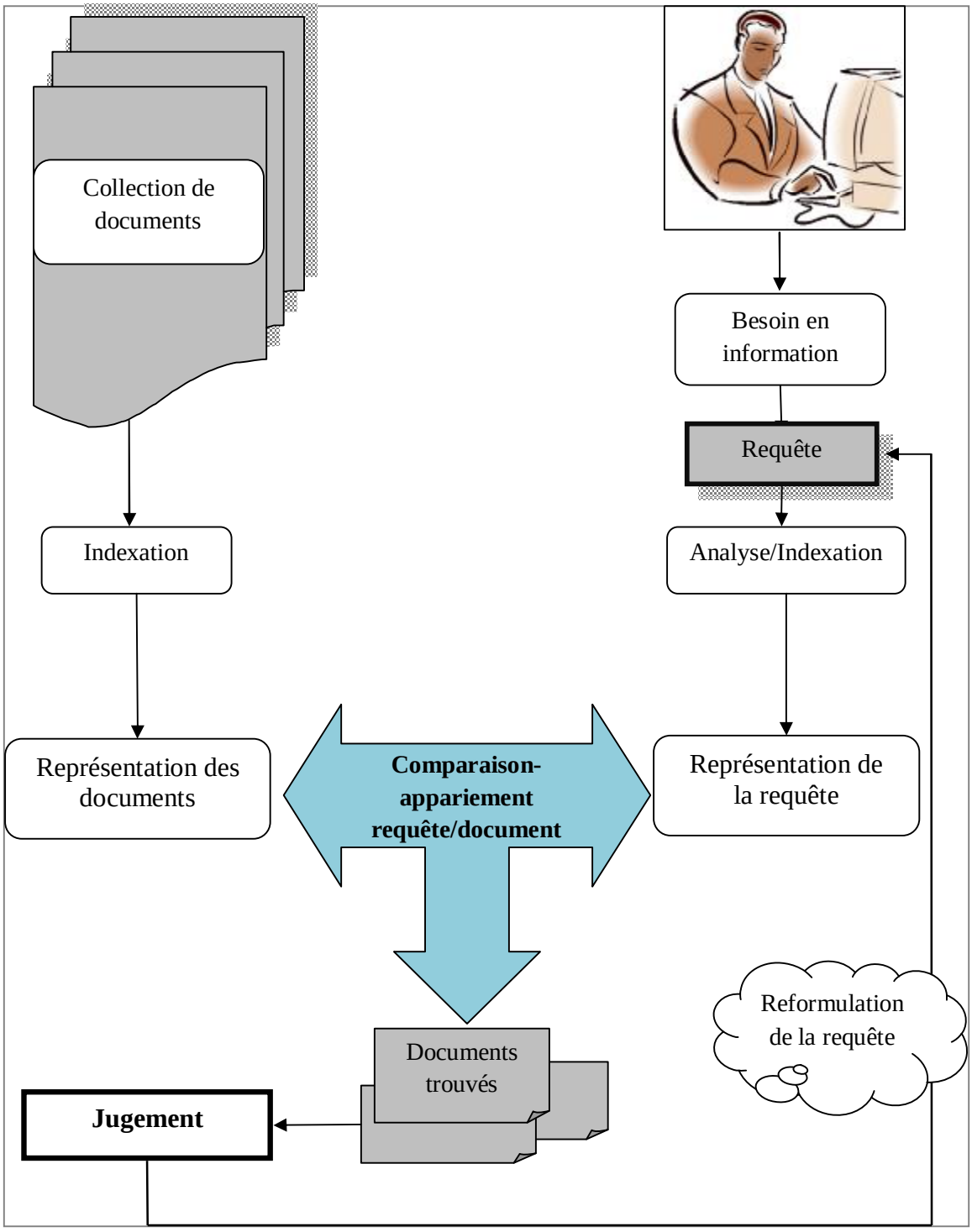


Figure 1.1 : Processus en U de recherche d'information. [BADR, 2007]

I.2.1 Concepts de base de la RI :

I.2.1.1 Le document :

Le document constitue l'information élémentaire d'une collection documentaire. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une

partie d'un document. Dans la suite de ce rapport, nous utiliserons indifféremment les termes document ou information pour designer un granule documentaire.

I.2.1.2 La requête :

Une requête (en anglais query) correspond à l'interrogation d'une base, ici les documents, pour en récupérer une certaine partie des données. D'autre part, le besoin de l'utilisateur est l'expression mentale de ce qu'il recherche. Ce besoin est alors représenté au travers d'une requête qui sera traitée par le SRI. Il s'agit en général d'un ensemble de mots-clés exprimés en :

- Le langage booléen: la requête est un ensemble de mots clés (termes de recherche) séparés par des opérateurs logiques (non, ou, et);
- Le langage naturel: la requête est exprimée avec un ensemble de mots clés dans un langage libre ;
- Le langage graphique: la requête est construite à partir d'une liste de mots clés connus. Pour ce faire, une vue d'ensemble représentant le contenu, notamment sémantique, des documents est offerte à l'utilisateur pour l'assister à formuler sa requête.

I.2.1.3 Notion de pertinence :

Plusieurs définitions ont été associées à la pertinence, et d'après **[Nie, 2004]** elle représente la correspondance entre un document et une requête, une mesure d'informativité du document à la requête et un degré de relation (chevauchement, relativité, …) entre le document et la requête.

La notion de pertinence est une notion subjective, car elle dépend étroitement de l'utilisateur. Et on distingue deux types:

- **La pertinence système :** le système répond à la requête formulée par l'utilisateur, par un ensemble de documents trouvés qu'il juge pertinents dans la base documentaire qu'il possède.
- **La pertinence utilisateur** : Le résultat est jugé par un utilisateur ; ainsi ce résultat peut être pertinent pour le système mais pas pour l'utilisateur.

I.2.2 Principales phases du processus de RI :

Le processus de Recherche d'information, appelé « Processus en U de Recherche d'Information » se compose des ces principales fonctions:

- 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.

I.2.2.1 L'indexation :

L'indexation est une étape très importante dans le processus de RI. Elle consiste à déterminer et extraire les termes représentatifs du contenu d'un document ou d'une requête, qui couvrent au mieux leur contenu sémantique. La qualité de la recherche dépend en grande partie de la qualité de l'indexation.

Le résultat de l'indexation constitue, ce que l'on nomme le descripteur du document ou de requête. Ce dernier est souvent une liste de termes significatifs pour l'unité textuelle correspondante, généralement assortis de poids représentant leurs degrés de représentativité du contenu sémantique de l'unité qu'ils décrivent.

Les descripteurs des documents (mots, groupe de mots) sont rangés dans une structure appelée dictionnaire constituant le langage d'indexation.

L'indexation peut être manuelle, automatique ou semi-automatique :

a) Indexation manuelle :

L'indexation manuelle est réalisée par un expert documentaliste qui après analyse des documents, et utilisant son savoir faire, propose des termes représentants le contenu des documents considérés. L'indexation manuelle est fondée sur le jugement d'un être humain, et est cependant trop dépendante de l'état de connaissances des indexeurs, ce qui induit à une subjectivité des résultats. Son application est très coûteuse en temps et inadaptée à des collections de taille importante. Elle est cependant, utilisée notamment dans le monde de la documentation, comme dans les bibliothèques, et sert en général à la classification des ouvrages par thèmes. Dans ce cadre, un ensemble prédéfini de mots-clés et de catégories est défini précisément, pour faciliter l'indexation et la formulation des requêtes (vocabulaire contrôlé[1]).

b) Indexation automatique :

L'indexation automatique est la technique d'indexation la plus utilisée lors d'un processus de RI. Elle offre l'avantage d'être totalement automatisée et donc plus adaptée pour les bases documentaires de grandes tailles.

Le SRI choisit les termes les plus représentatifs dans chaque document et leur associe des poids. Ce traitement est réalisé en plusieurs étapes: l'analyse lexicale, l'élimination des mots vides, la lemmatisation et la pondération.

[1] Dans le cas d'une indexation contrôlée, une expertise préalable sur le domaine d'application considéré, établit un vocabulaire exhaustif représenté dans une structure dénommée le thésaurus.

Ø L'analyse lexicale :

La première étape de l'indexation automatique consiste à extraire du document un certain nombre d'unités lexicales (termes). Une unité lexicale est définie comme étant une suite de caractères délimitée par des séparateurs (blancs, virgules …).

Ø L'élimination des mots vides

Cette étape permet de réduire la taille de l'index, et de ce fait, le temps de réponse du système. Elle consiste à éliminer les mots non significatifs, comme les articles, les prépositions et les conjonctions. Ce sont des mots qui ne sont pas discriminatoires lors d'une recherche, car ils ne traitent pas le sujet du document et sont présents en grand nombre dans les documents de la collection.
Deux techniques sont principalement utilisées pour l'élimination des mots vides:

- L'utilisation d'une liste prédéfinie de mots vides, aussi appelée anti-dictionnaire ou stoplist **[Fox, 1992]** ;
- L'utilisation de mesures statistiques sur la collection de documents pour déterminer les mots les plus fréquents ou les mots rares, qui sont « probablement » des mots vides.

Ø La lemmatisation

La lemmatisation consiste à extraire la forme canonique (racine) de chacun des termes issus des étapes précédentes. L'objectif étant d'éliminer les variations morphologiques des mots (nombre, genre …).
L'algorithme de Porter **[Porter, 1997]** est l'algorithme le plus utilisé pour la lemmatisation des mots de la langue anglaise. En français, il n'existe pas d'algorithmes fiables pour la lemmatisation vu, par exemple, la complexité des formes prises par les verbes du troisième groupe (le verbe être peut prendre les formes: est, sommes, fûmes …). De ce fait, on a souvent recours à un dictionnaire.

Ø La pondération

La pondération permet d'associer à chaque terme *ti* un poids représentant son importance dans un document *dj*. En général, ce poids est mesuré sur la base de considérations statistiques en utilisant les deux mesures suivantes:

- La fréquence locale (*tf*): représente la fréquence du terme *ti* dans le document *dj*. La fréquence d'un terme représente le nombre d'occurrences de ce terme dans un document. Il existe plusieurs variantes pour calculer *tf*, notamment par l'application du logarithme sur la fréquence du terme afin d'atténuer les écarts entre les fréquences (de l'ensemble des termes).

- La fréquence globale (*idf*): elle vise à donner une importance plus élevée aux termes qui apparaissent peu dans la collection de documents. Ainsi, un terme considéré comme rare dans la collection de documents aura un pouvoir discriminant plus élevé qu'un terme plus répandu **[Salton, 1975]** Pour ce faire, on associe à chaque terme une mesure égale au logarithme de l'inverse de la proportion des documents qui contiennent le terme :

$$idf(t_i) = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

Avec :

ti : le terme dont on cherche la fréquence;
|{dj : ti Є dj}|: le nombre de documents où le terme ti apparait;
|D| : le nombre total de documents dans la collection;
Idf (ti): la fréquence globale du terme ti.

Ces deux mesures (*tf* et *idf*) sont combinées en *tf* × *idf* pour fournir le poids du terme *ti.* Ce produit représente une bonne approximation du poids du terme, particulièrement si les documents de la collection sont de taille (longueur) homogène. Cependant, si la collection contient des documents très hétérogènes en termes de longueur, les documents les plus longs se voient favorisés (car plus un document est long, plus un terme a de chances d'apparaitre). Pour pallier cet inconvénient, Singhal **[Singhal, 1996]** et Robertson **[Robertson, 1997]** proposent d'intégrer la taille des documents a la formule de pondération : on parle de facteur de normalisation.

c) L'indexation semi-automatique

L'indexation semi-automatique est une alternative qui profite des deux techniques d'indexation précédentes. La liste des descripteurs est construite d'abord de façon automatique, ensuite un expert documentaliste rajoute, manuellement, les termes qu'il juge représentatifs pour chaque document.

I.2.2.2 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 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.

I.2.2.3 Reformulation de la requête

En plus des deux processus de base décrits précédemment, un SRI peut intégrer un processus de reformulation de la requête. Ce processus a pour but d'améliorer la performance du système en offrant un mécanisme de raffinement de la requête utilisateur. Les techniques utilisées durant ce processus se classent en deux catégories **[Boubekeur, 2008]** : *les méthodes locales* et *les méthodes globales*.

- **Les méthodes locales :**

Elles s'appuient sur la technique dite de réinjection de pertinence (*relevance feedback*). En se basant sur la requête initiale, une liste de documents sera présentée à l'utilisateur, qui va ensuite choisir les documents qu'il juge pertinents. La requête initiale sera ensuite enrichie avec les termes des documents ainsi choisis.

- **Les méthodes globales**

La requête est réajustée en se basant sur des ressources externes telles que les bases lexicales, les thésaurus et les ontologies. Ainsi, des termes, non nécessairement présents dans la requête initiale, sont suggérés à l'utilisateur pour raffiner sa requête.

I.3 Modèles de Recherche d'Information:

Le modèle de RI a pour rôle de déterminer le comportement clé d'un Système de Recherche d'Information. Il fournit donc un cadre théorique pour la modélisation de la mesure de pertinence.

Dans cette partie nous citons les principaux modèles de la Recherche d'Information qui ont été décrits avant dans de nombreux travaux sur la Recherche d'Information.**[Robertson,1994]** :

- Les modèles booléens basés sur la théorie des ensembles dont on distingue : Le modèle booléen (booleen model), le modèle booléen étendu (extended boolean model) et le modèle basé sur les ensembles flous (fuzzy set model). Dans ces modèles, les termes de la requête sont séparés par des operateurs logiques (OR, AND, NOT). On peut alors effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associes a chaque terme.

- Les modèles algébriques (vectoriels) comme : Le modèle vectoriel (vector model), le modèle vectoriel generalise (generalized vector model), Latent Semantic Index model (LSI) et le modèle connexionniste. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel.

- Les modèles probabilistes : Le modèle probabiliste général, le modèle de réseau de document ou d'inférence (Document Network), et le modèle de langage. Ces modèles reposent 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 la suite, nous détaillons les trois principaux modèles exploités en RI à savoir : le modèle booléen, le modèle vectoriel et le modèle probabiliste.

I.3.1 Le modèle booléen:

Le modèle booléen a été le premier à être utilise dans le monde de la RI. C'est le plus simple des modèles de RI, et permet de faire une recherche très restrictive et obtenir, pour un utilisateur expérimenté, une information exacte et spécifique **[Salton, 1971]**. Il est basé sur l'algèbre de Boole et considère ainsi une requête comme une expression logique. Il considère aussi que les termes de l'index sont soit présents soit absents d'un document. En conséquence, les poids des termes dans l'index sont binaires c'est-a-dire wij= {0,1}.

Dans ce modèle :

Un document *d* est représenté comme une conjonction logique des termes non pondérés.

Exemple: d =t1 $\wedge$ t2 $\wedge$ t3 $\wedge$ ... $\wedge$ tn[2].

Une requête *q* est composée de termes liés par 3 connecteurs logiques ET, OU, NON.

Exemple : q = (t1 $\wedge$ t2) $\vee$ (t3 $\wedge \neg$t4) ou *q* : la requête, *ti*: terme d'indexation.

La similarité entre un document et une requête est définie par :

$$RSV(q, d) = \begin{cases} 1 & Si\ d\ appartient\ à\ l'ensemble\ décrit\ par\ la\ requête \\ 0 & Sinon \end{cases}$$

Ainsi, le modèle booléen affirme que chaque document est soit pertinent soit non-pertinent, donc répond exactement a la requête qui a été formulée. Il n'y a pas de notion de réponse partielle aux conditions de la requête.

D'autre part, il est souvent très difficile pour l'utilisateur d'exprimer son besoin en information avec des expressions booléennes ce qui ne permet pas d'utiliser au mieux les caractéristiques de ce modèle.

Enfin, tous les termes dans un document ou dans une requête sont pondérés de la même façon simple (0 ou 1), donc tous les termes ont la même importance dans le document. Cela ne correspond pas a ce qu'on souhaite avoir en RI.

I.3.2 Le modèle vectoriel :

Le modèle vectoriel introduit par Salton **[Salton, 1971]**, repose sur les bases mathématiques des espaces vectoriels. Dans ce modèle, les documents et les requêtes sont représentés dans un espace vectoriel engendré par l'ensemble des termes d'indexation $t_1,t_2,t_3,\ldots t_T$. Où N est le nombre total de termes issus de l'indexation de la collection des documents.

Chaque document est représenté par un vecteur : $D_j = (d_{1j}, d_{2j}, \ldots\ldots, d_{ij}, \ldots\ldots\ldots, d_{Tj})$.

Chaque requête est représentée par un vecteur : $Q = (q_1,q_2,\ldots\ldots\ldots\ldots,q_i,\ldots\ldots\ldots,q_T)$.

[2] On note $\wedge, \vee, \neg$ pour designer ET, OU, NON logique respectivement.

Avec :

d_{ij} : Poids du terme t_i dans le document D_j.

q_i : Poids du terme t_i dans la requête Q.

Les termes de poids nul représentent les termes absents dans un document alors que les poids positifs représentent les termes assignés.

La fonction de calcul du coefficient de similarité entre chaque document, représenté par le vecteur *Dj*(d_{1j} , d_{2j} , ……,d_{ij},……..,d_{Tj}) , et la requête, représentée par le vecteur *Q* (q_1,q_2,……,q_i,……...,q_T) est appelée (***Retrieval Status Value)*** ou ***RSV***.

Ce coefficient de similarité est calculé sur la base d'une fonction qui mesure la colinéarité des vecteurs documents et requête. On peut citer notamment les fonctions suivantes :

- **Produit scalaire :**

$$RSV(Q, D_j) = \sum_{t=1}^{T} q_i * d_{ij}$$

- **Mesure de Jaccard :**

$$RSV(Q, D_j) = \frac{\sum_{t=1}^{T} q_i * d_j}{\sum_{t=1}^{T} q_i^2 + \sum_{t=1}^{T} d_{ij}^2 - \sum_{t=1}^{T} q_i * d_{ij}}$$

- **Mesure de cosinus :**

$$RSV(Q, D_j) = \frac{\sum_{t=1}^{T} q_i * d_j}{(\sum_{t=1}^{T} q_i^2)^{1/2} * (\sum_{t=1}^{T} d_{ij}^2)^{1/2}}$$

Le modèle vectoriel permet de pallier à l'un des inconvénients majeurs du modèle booléen, en permettant de trier les documents répondant à une requête. Les documents dans le modèle vectoriel, 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 donc pertinent pour l'utilisateur.

Un des principaux défauts de l'approche vectorielle est, qu'elle considère tous les termes comme étant indépendants, et ne tient donc pas compte des liens entre ceux-ci (exemple : voiture, automobile)

I.3.3 Le modèle probabiliste :

Le modèle probabiliste a été proposé par Robertson **[Robertson]** et Maron **[Maron, 1960]**, il utilise un modèle mathématique fondé sur la théorie de la probabilité conditionnelle (appelé aussi modèle de la théorie de pertinence). Lors du processus d'indexation deux probabilités conditionnelles sont utilisées :

- **P ($t/_{pert}$)** : La probabilité pour que le terme « ***t*** » apparaisse dans un document donné sachant que ce document est pertinent pour la requête.

- **P ($t/_{NonPert}$)** : La probabilité pour que le terme « ***t*** **»** apparaisse dans un document donné sachant que ce document n'est pas pertinent pour la requête.

En supposant que la distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents, et que les variables « document pertinent » et « document non pertinent» sont indépendantes, la fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document **P (Pert/D).**

- P (Pert/D) = $\frac{P(D|Pert)*P(Pert)}{P(D)}$
- P (NonPert/D) = $\frac{P(D|NonPert)*P(NonPert)}{P(D)}$

Avec P(D)= P (D/Pert)*P(Pert) +P (D/NonPert)*P(NonPert).

Où : P (D/Pert) est la probabilité d'observer D sachant qu'il est pertinent.
P (Pert) est la probabilité à priori qu'un document soit pertinent.

Le coefficient de similarité requête document (*RSV*) peut être calculé par différentes formules. Robertson et Spark-Jones proposent la formule suivante :

- RSV = $\sum log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)}$

Avec :

N: nombre total de documents de la base,
n: nombre de documents contenant le terme,
R: nombre de documents connus comme étant pertinents,
r: nombre de documents connus comme étant pertinents et contenants le terme.

L'ajout de 0.5 à tous les membres s'explique par la nécessité d'écarter tous les cas limites qui entraîneraient des valeurs nulles de ces membres.

Le modèle probabiliste est fonctionnel et a montré une certaine efficacité mais néanmoins il présente quelques inconvénients comme la complexité des modèles de calcul de probabilité par rapport aux autres modèles.

I.4 Paramètres d'évaluation d'un SRI :

La démarche de validation en RI se base sur l'évaluation expérimentale des performances du modèle ou du système proposé. L'évaluation des performances d'un modèle de RI, permet de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et de fournir des éléments de comparaison entre modèles.

Cette évaluation peut porter sur plusieurs critères : le temps de réponse, la pertinence, la qualité et la présentation des résultats, etc. Le critère le plus important est celui qui mesure la

capacité du système à satisfaire le besoin en information de l'utilisateur, c'est à dire la pertinence. Deux facteurs permettent d'évaluer ce critère. Le premier est le **taux de rappel** et le deuxième est **le taux de précision**.

I.4.1 Rappel et précision :

- **Rappel :**

Le rappel est le rapport de documents pertinents restitués par le système sur l'ensemble des documents pertinents contenus dans la base documentaire. Elle mesure la capacité du système à retrouver tous les documents pertinents répondant à une requête.

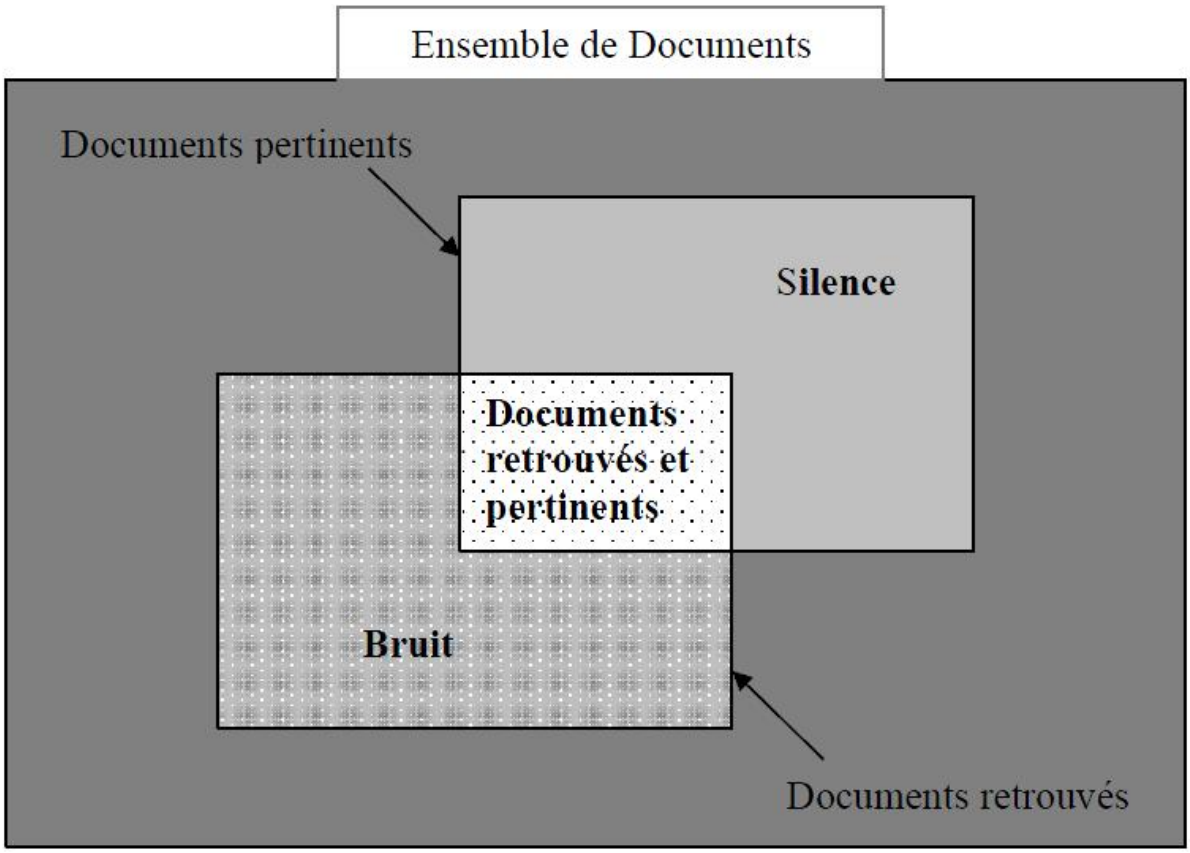
$$\text{Rappel} = \frac{\text{nombre de documents pertinents trouvés}}{\text{nombre de documents pertinents}}$$

- **Précision :**

La précision est le rapport de documents pertinents trouvés sur l'ensemble des documents restitués par le système. Elle mesure la capacité du système à rejeter tous les documents non pertinents à une requête donnée.

$$\text{Précision} = \frac{\text{nombre de documents pertinents trouvés}}{\text{nombre de documents trouvés}}$$

Le rappel et la précision permettent aussi de définir le silence documentaire et le bruit documentaire qui représentent respectivement le nombre des documents pertinents non retrouvés et le nombre de documents non pertinents retrouvés. La figure 1.2 schématise ces deux variables dans l'ensemble des documents.



.

Figure 1.2 : Bruit et Silence **[Soule, 2007]**

Idéalement, on voudrait qu'un système donne de bons taux de précision et de rappel en même temps. Les deux métriques ne sont pas indépendantes. Il y a une forte relation entre elles: *quand l'une augmente, l'autre diminue*.

Le comportement d'un système peut varier en faveur de précision ou en faveur de rappel (au détriment de l'autre métrique). Ainsi, pour un système, on a une courbe de précision-rappel qui a en général l'aspect suivant:

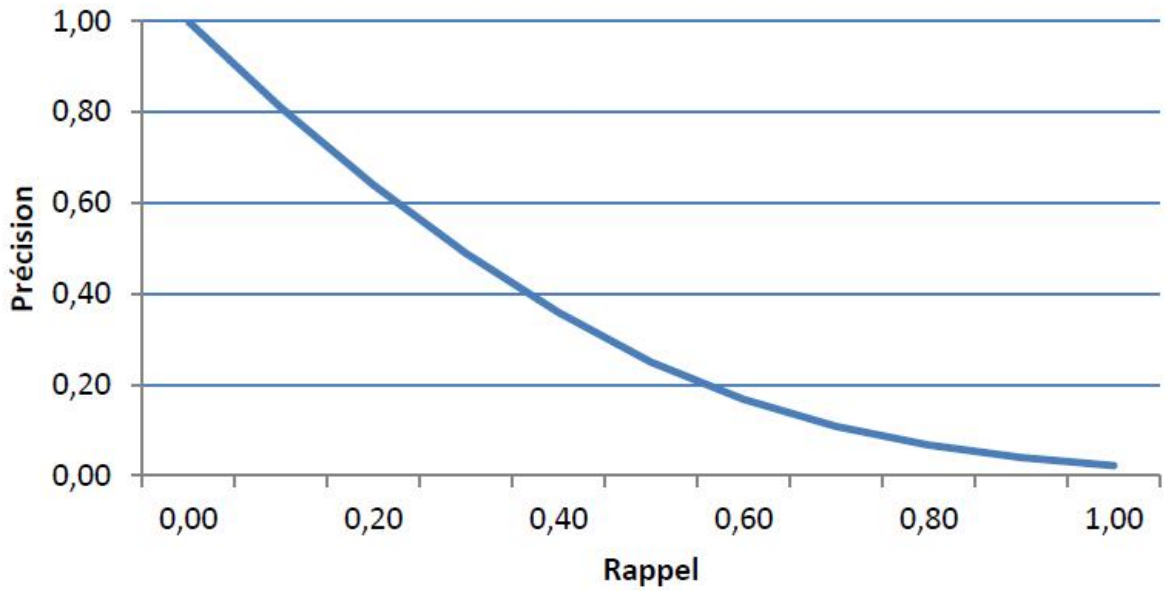


Figure 1.3 : Courbe Rappel-Précision

I.4.2 Précision à X :

Pour faciliter les calculs de précision et de rappel, on calcule en général la précision à x. Elle représente la proportion de documents pertinents présents dans les *x* premiers documents retrouvés, en supposant ces documents triés selon leur valeur de pertinence (*RSV*). Ainsi, on calcule typiquement les précisions à *5, 10, 15* pour un système, et ces mesures sont notées alors *P5, P10, P15*, etc. La *précision exacte* découle de ces mesures. C'est la précision à *x* où *x* est le nombre total de documents pertinents dans la collection pour la requête.

La *précision moyenne* est aussi calculée pour chaque requête. C'est la moyenne des précisions pour chaque document pertinent. La précision d'un document pertinent est la précision à *x*, *x* étant la position de ce document dans la liste triée des documents retrouvés. Si un document pertinent n'est pas retrouvé, sa précision est nulle. Enfin, les précisions moyennes pour l'ensemble des requêtes, qui sont donc les moyennes des précisions (moyennes, exactes, . . .) par requête, permettent d'obtenir une mesure de la performance globale du système.

I.5 Conclusion:

Ce premier chapitre a porté essentiellement sur l'étude des SRI de manière générale, et des modèles de recherche et de représentation d'information de manière particulière. Chacun des modèles, contribue en partie à la résolution des problèmes inhérents à la RI : perception du besoin en information, représentation du sens véhiculé par les documents, formalisation de la pertinence etc…

La finalité de chaque SRI est de satisfaire les besoins des utilisateurs, pour qui la préoccupation majeure est la récupération des informations satisfaisants leurs besoins en information de manière rapide et efficace. Cependant, ces SRI traitent les documents comme une unité indivisible et n'exploitent de ces derniers que le contenu sémantique (le texte).

Actuellement, de nouveaux formats de structuration de l'information comme XML sont en train de s'imposer, permettant ainsi, de représenter conjointement l'information textuelle et les contraintes structurelles (balises) d'un document. La RI plein texte, évolue vers une RI structurée, prenant donc en compte la structure et le contenu des documents.
Ce qui représente un nouveau défi pour la RI. C'est dans ce contexte, que se situe notre travail, qui consiste à restituer à l'utilisateur non pas un document intégral suite à une formulation de sa requête, mais un fragment de document structuré répondant ainsi, de manière ciblée et efficace à sa requête. Nous décrivons dans le prochain chapitre, les différentes approches issues de la RI pour permettre, l'utilisation et l'exploitation de cette information supplémentaire qu'est la structure.

II.1 Introduction :

L'objectif principal d'un SRI classique est de retrouver les documents dont le contenu est conforme à une requête donnée. Dans cette optique, les documents sont représentés par un ensemble de mots-clés décrivant leur contenu. La structure du document n'est pas prise en considération ni au niveau de la requête, ni au niveau de la réponse pour retourner les parties pertinentes : la réponse à une requête reste le document tout entier. Aujourd'hui, l'utilisation de l'information apportée par la structure devient une nécessité dans le domaine d'accès à l'information. Deux communautés se sont engagées à proposer des solutions pour la recherche d'information dans des documents structurés : la communauté de la RI qui s'intéresse à la structure présentée dans les documents pour apporter des réponses plus précises à une requête d'un utilisateur, et la communauté des bases de données qui s'intéresse à la structure comme un moyen plus souple pour stocker des données (Pour plus de détail voir **[Fellag, 2006]**).

L'objectif principal sur lequel se focalisent ces deux communautés est un type de document qui est de plus en plus répandu sur Internet : le langage XML (eXtensible Markup Langage) est utilisé aujourd'hui comme format de données structurées sur le Web **[Neil, 2002]**, il impose au SRI de retrouver des unités d'information qui ne sont pas nécessairement le document entier.

Dans ce chapitre on va présenter en premier lieu le langage XML, en décrivant notamment la structure d'un document XML, les parseur DOM et SAX. Ensuite, on s'intéresse à la problématique de la recherche d'information dans un document structuré. On présente, en particulier les approches de la RI structurée. Pour finir, on présente les techniques d'évaluation des SRIS (système de recherche d'information structurée) à travers la présentation de la compagne d'évaluation INEX.

II.2 Présentation de XML:

XML (e**X**tensible **M**arkup **L**anguage) ou langage à balises extensible est apparu comme nouveau standard pour la représentation et l'échange de données sur Internet. Son développement a commencé en 1996 avec le XML Working Group du W3C[1], qui en développe la recommandation en Février 1998[2].

XML est un langage permettant de séparer le contenu des documents des instructions de présentations. Il permet aussi de représenter et d'assurer l'échange de documents *semi-structurés*. Les documents XML sont dits *semi-structurés* car, ils possèdent une structure qui n'est pas imposée par une norme, un standard ou une recommandation, mais une structure que le créateur peut déterminer lui-même au moment de la conception du document.

[1] W3C: **W**orld **W**ide **W**eb **C**onsortium. Pour en savoir plus consulter le site : http://www.w3.org
[2] http://www.w3.org/TR/1998/REC-xml-19980210.

II.2.1 Structure d'un document XML:

Tout document XML se compose **[Allorge, 2000]**:

- d'un *Prologue*, dont la présence est facultative mais conseillée. Il contiendra un certain nombre de déclarations.
- d'un *arbre d'éléments*. Il forme le contenu du document.
- de *commentaires* et *d'instructions de traitement*, dont la présence est facultative. Ils pourront, moyennant certaines restrictions, apparaître aussi bien dans le prologue que dans l'arbre d'éléments.

II.2.1.1 Le Prologue :

Il peut contenir une déclaration XML, des instructions de traitement et une déclaration de type de document.

a. Déclaration XML:

Syntaxe : **<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>**

On distingue trois informations fournies dans cette déclaration :

1-Version : version du XML utilisée dans le document, 1.0 en ce qui nous concerne (la dernière version du langage, 1.1, date de février 2004 mais ne change rien quant à ses bases) ;

2-Encoding : le jeu de codage de caractères utilisé. Le jeu de caractères habituel pour le français est le ISO[3]-8859-1. Il a tendance à être remplacé par l'ISO-8859-15 en attendant la généralisation de l'Unicode. Par défaut, l'attribut encoding a la valeur UFT -8. Cela permet à l'ordinateur de « savoir » quel caractère il doit afficher en réponse aux combinaisons de 1 et de 0 que contient le fichier sur le disque dur ;

3-Standalone : dépendance du document par rapport à une déclaration de type de document. Si standalone a la valeur, « yes », le processeur de l'application n'attend aucune déclaration de type de document extérieure au document. Sinon, le processeur attend une référence de déclaration de type de document. La valeur par défaut est « yes ».

Cette déclaration est facultative, mais il est préférable de l'utiliser. Dans ce cas les attributs version, encoding et standalone doivent être placés dans cet ordre.

b. Instructions de traitement:

Une instruction de traitement est une instruction interprétée par l'application servant à traiter le document XML. Elle ne fait pas totalement partie du document. Les instructions de

[3] International Organization for Standardization.

traitement qui servent le plus souvent sont la déclaration XML ainsi que la déclaration de feuille de style. Exemple d'instruction de traitement :

<?xml-stylesheet type="text/xsl" href="biblio.xsl"?>

Dans cet exemple, l'application est xml-stylesheet, le processeur de feuille de style de l'application traitant le document XML. Deux feuilles de style différentes peuvent être utilisées, les XSL (propres au XML) ainsi que les CSS (feuilles de style apparues avec le HTML). L'attribut type indique de quel type de fichier il s'agit (text/css pour les feuilles de style CSS, par exemple) et l'attribut href indique l'URL du fichier. Cette instruction de traitement est notamment utilisée par les navigateurs Internet pour la mise en forme du document.

c. Déclaration de type de document (DTD) :

La DTD Document Type Definition ou Définition de type de document permet d'établir les règles de définition de la structure d'un document XML, les éléments à inclure, leur type et les valeurs par défaut à leur attribuer. Elle peut être déclarée au sein du document ou sous forme d'un document externe.

Il n'est pas obligatoire pour un document XML de se conformer à une DTD. Cependant, il doit être bien formé, c'est-à-dire qu'il doit respecter les règles générales de syntaxe d'XML. Lorsqu'une DTD est associée à un document XML, on dit qu'il est valide**.** (Détaillée en section II-3).

II.2.1.2 L'arbre d'éléments :

a. Introduction :

Un document XML peut se représenter sous la forme d'une arborescence d'éléments. Cette arborescence comporte une racine (unique), des branches et des feuilles. Prenons l'exemple suivant :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="Voiture.css"?>
<!DOCTYPE Voiture SYSTEM "Voiture.dtd" >
<Voiture marque="Renault" modèle="Safrane">
  <Carosserie couleur="rouge">
    <Capot>Un peu cabossé</Capot>
  </Carosserie>
  <Moteur>
    <Cylindres />
    <Allumage>Défectueux</Allumage>
  </Moteur>
  <Transmission type="automatique" nb_vitesses="5">
    <Boîte />
    <TrainAV />
    <TrainAR />
  </Transmission>
</Voiture>
```

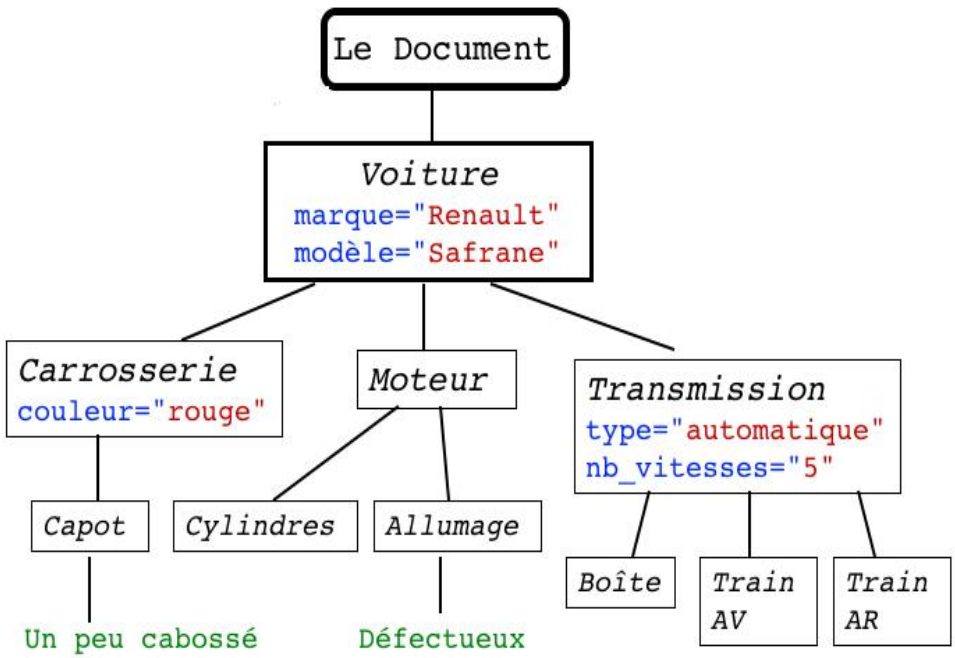


Figure 2.1 : L'arbre éléments d'un document XML

b. Élément racine :

L'élément-racine (en anglais : document élément) est, comme son nom l'indique, la base du document XML. Il est unique et englobe tous les autres éléments. Il s'ouvre juste après le prologue, et se ferme à la toute fin du document. Dans l'exemple ci-dessus, l'élément racine est Voiture.

c. Les éléments :

Les éléments forment la structure même du document : ce sont les branches et les feuilles de l'arborescence. Ils peuvent contenir du texte, ou bien d'autres éléments, qui sont alors appelés « éléments enfants », l'élément contenant étant quant à lui appelé logiquement « élément parent ».

Exemple d'élément contenant du texte :

```
<Allumage>Défectueux</Allumage>
```

Exemple d'élément contenant d'autres éléments :

```
<Moteur>
  <Cylindres />
  <Allumage>Défectueux</Allumage>
</Moteur>
```

d. Les attributs :

Tous les éléments peuvent contenir un ou plusieurs attributs. Chaque élément ne peut contenir qu'une fois le même attribut. Un attribut est composé d'un nom et d'une valeur. Il ne peut être présent que dans la balise ouvrante de l'élément (par exemple, on n'a pas le droit d'écrire (</livre lang= "en">).

Exemple d'utilisation d'un élément avec attribut :

```
<instrument type="vent">trompette</instrument>
```

Exemple d'utilisation d'un élément vide avec attributs :

```
<img src="ours.png" alt="Gros ours" width="56" height="100" />
```

II.2.1.3 Les commentaire :

Des commentaires peuvent être insérés dans les documents. Ils sont encadrés pas les marques de début *<!--et de fin -->* de commentaire.

Syntaxe :

```
<!--commentaire -->
```

Le corps du commentaire peut contenir n'importe quel caractère à l'exception de la chaîne « --». On ne peut donc pas inclure un commentaire dans un autre.

II.3 Déclaration de type de document (DTD) :

Le DTD ou Document Type Declaration ou encore Document Type Definition est l'ensemble des règles et des propriétés que doit suivre le document XML. Ces règles définissent généralement le nom et le contenu de chaque balise et le contexte dans lequel elles doivent exister. Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML.

II.3.1 La DTD interne :

On peut inclure sa propre DTD au code source du fichier XML. On parlera alors d'une DTD interne. L'entête du fichier XML suit la syntaxe suivante :

```
<?xml version="1.0" standalone="yes"?>
<!DOCYPE élément-racine [déclaration des éléments]>
```

• standalone="yes" indique que votre fichier est indépendant (DTD interne).

Exemple :

```
<?xml version="1.0" standalone="yes" ?
<!DOCTYPE carte [<!ELEMENT carte (#PCDATA)>]>
<carte>As de pique</carte>
```

II.3.2 La DTD externe :

On peut utiliser une DTD externe au code source du fichier XML. On parlera alors d'une DTD externe. L'entête du fichier XML suit la syntaxe suivante :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE racine SYSTEM "fichier.dtd">
```

• SYSTEM indique que votre fichier DTD est situé dans un répertoire local.

Exemple :

Fichier jeu.xml

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE carte SYSTEM "carte.dtd">
<carte>As de pique</carte>
```

• Dans un même document XML, il est possible d'utiliser ensemble une DTD interne et une DTD externe. Dans ce cas, la déclaration DOCTYPE prend la forme suivante :

```
<!DOCTYPE nœudracine SYSTEM "URI DTD externe" [
   instructions DTD interne
]>
```

II.3.3 Exemple d'un document avec sa DTD associée :

```
< !--Exemple de document XML-->
<personne>
<nom>Doe</nom>
<prenom>John</prenom>
<fonction>PDG</fonction>
</personne>
```

```
< ? xml version="1.0" ?>
< !--DTD pour personne.xml-->
< !ELEMENT personne(nom, prenom,fonction)>
< !ELEMENT nom (# PCDATA)>
< !ELEMENT prenom (# PCDATA)>
< !ELEMENT fonction (# PCDATA)>
```

II.4 Analyseur (Parser) XML:

XML permet de définir la structure d'un document, ce qui permet d'une part de pouvoir définir séparément la présentation de ce document, d'autre part d'être capable de récupérer les données présentes dans le document pour les utiliser.

Toutefois la récupération des données encapsulées dans le document nécessite un outil appelé analyseur syntaxique (en anglais parser), permettant de parcourir le document et d'en extraire les informations qu'il contient.

II.4.1 Qu'appelle-t-on un Analyseur ?

L'analyseur syntaxique (généralement francisé en parseur) est un outil logiciel permettant de parcourir un document et d'en extraire des informations.

Dans le cas de XML (on parle alors de parseur XML), l'analyseur permet de créer une structure hiérarchique contenant les données contenues dans le document XML.

On distingue deux types de parseurs XML :

• les parseurs validants (validating) permettant de vérifier qu'un document XML est conforme à sa DTD

• les parseurs non validants (non-validating) se contentant de vérifier que le document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base)

Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document. On distingue actuellement deux types d'approches :

• Les API (Application Programming Interface ou interface de programmation d'application) utilisant une approche hiérarchique : les analyseurs utilisant cette technique construisent une structure hiérarchique contenant des objets représentant les éléments du document, et dont les méthodes permettent d'accéder aux propriétés. La principale API utilisant cette approche est DOM (Document Object Model)

• Les API basées sur un mode événementiel : permettent de réagir à des événements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application utilisant cette API. SAX (Simple API for XML est la principale interface utilisant l'aspect événementiel

Ainsi, on tend à associer l'approche hiérarchique avec DOM et l'approche événementielle avec SAX.

II.4.2 DOM (Document Object Model) :

Le DOM (Document Object Model ou modèle objet de document) est une API permettant de représenter un document XML sous forme d'arbre d'objets reliés entre eux.

Chaque objet représente une entité (document, élément, attribut, texte,…) d'un document XML.

DOM permet une navigation aisée dans un document XML mais nécessite le chargement complet en mémoire de sa structure arborescente.

Le modèle DOM est une recommandation du W3C depuis octobre 1998.

Exemple de document XML et l'arbre DOM qui lui est associé :

```
<?Xml Version= "1.0" Encoding= "Iso-8859-1" ?>
<Cinema>
<Nom> Epée De Bois </Nom>
<Adresse>100, Rue Mouffetard</Adresse>
<Metro>Censier-Daubenton </Metro>
</Cinema>
```

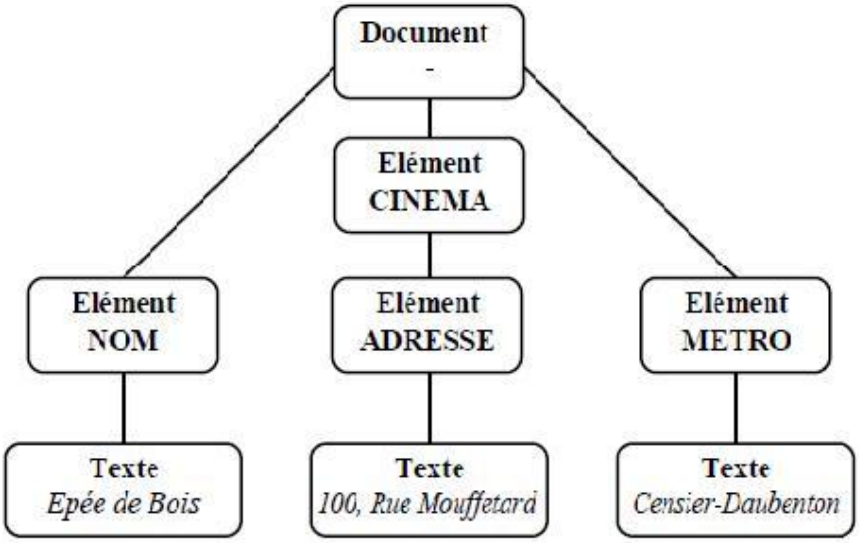


Figure 2.2 : Exemple d'arbre DOM.

II.4.3 SAX (Simple API for XML):

SAX (*Simple API for XML*) est une API (*Application Programming Interface*) développée par les membres du groupe de discussion *XML-dev* (*XML-dev mailing list*). Elle a été proposée comme une alternative à DOM, afin de proposer une solution moins gourmande en mémoire pour l'analyse des documents XML. Originellement spécifique à Java, SAX s'est par la suite généralisé à d'autres langages de programmation.

SAX se base sur un modèle évènementiel pour l'analyse d'un document XML. Cela signifie que, lors de l'analyse du document par le parseur, des évènements (signalant le début ou la fin du document, l'ouverture ou la fermeture d'un élément, du contenu texte …) sont générés et envoyés à l'application. Celle-ci peut alors soit les prendre en considération soit les ignorer. Par conséquent, l'application n'a pas de représentation globale du document comme c'est le cas avec DOM, mais a, en revanche, une empreinte mémoire plus faible et permet d'analyser un document plus rapidement surtout si celui-ci est d'une taille importante.

II.5 Problématiques de la RI structurée :

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

Ø 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: 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 ?

Ø La deuxième problématique est celle de l'interrogation des corpus de documents 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 structurés (textuelle et structurelle).

Ø 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.

II.6 Approches de RI dans les documents XML :

Pour répondre aux problématiques citées précédemment, et permettre la recherche d'information dans des documents structurés, plusieurs approches ont été proposées, et se classent en deux principales catégories: *les approches orientées données* proposées par la communauté des bases de données (BD), et *les approches orientées documents* prises en charge par la communauté de la recherche d'information (RI). Ces approches sont décrites dans ce qui suit.

II.6.1 Les approches orientées données :

Les approches orientées données (*data-centric*) considèrent le document XML comme un ensemble de données typées et relativement homogènes. Elles s'intéressent davantage à la structure du document en considérant le document XML plus comme un support de stockage que comme un document textuel.

Ces approches utilisent les bases de données pour stocker toute l'information textuelle et structurelle des documents. C'est pourquoi la plupart des langages de requêtes qui y sont utilisés, sont des langages proches de la syntaxe de *SQL*. Ils permettent de faire des requêtes avec des contraintes très précises sur la structure, mais n'offrent guère de souplesse au niveau des contraintes sur le contenu textuel qui sont généralement traitées de manière booléenne

(présent ou absent). Dés lors, il n'est plus possible de trier les résultats restitués à l'utilisateur suite à une requête.

Un exemple d'une requête portant sur la structure pour laquelle il convient d'utiliser ce type d'approches est la suivante « *trouver tout les étudiants âgés de 20 ans* ». Celle-ci ne requiert aucun triage des résultats, et un appariement exact de l'âge des étudiants avec l'âge souhaité suffit pour répondre au besoin en informations de l'utilisateur.

II.6.2 Les approches orientées documents

Les approches orientées documents (*document-centric*) considèrent les documents semi-structurés comme une collection de documents textes comportant des balises et des relations entre ces balises.

Ces approches se basent, pour la plupart, sur les modèles de la RI classique qui sont étendus pour intégrer la structure dans le processus de recherche d'information. Elles utilisent des langages de requêtes plus simples que ceux proposés dans les approches orientées données. En outre, l'appariement entre une requête et un document se fait, comme dans la RI classique, grâce au calcul d'un degré de pertinence. Ainsi, une plus grande souplesse est offerte à l'utilisateur car il n'est pas nécessaire qu'il y ait une correspondance exacte entre la requête et les résultats, et, de plus, ceux-ci peuvent être triés.

II.6.3 Interrogation d'un document XML :

L'interrogation des documents XML diffère de celle des documents plats du au fait que le document XML possède une structure.

Pour que l'utilisateur puisse exprimer son besoin en information, deux types de requêtes sont utilisés :

- **Les requêtes orientées contenu CO (Content Only)** : ce sont des requêtes composées de simples mots clés, ils sont utilisées quand l'utilisateur n'a pas une idée précise de ce qu'il recherche ,comme celles utilisées dans les moteurs de recherche traditionnels. Exemple de requête CO *"algorithme génétique"*

- **Les requêtes orientées contenu et structure CAS (Content And Structure):**

L'utilisateur peut préciser son besoin, en ajoutant des conditions de structure sur l'information qu'il désire avoir, ils sont utilisées quand l'utilisateur a au moins une connaissance partielle sur la collection qu'il interroge. Exemple : ***"article/section/paragraphe/"base de données"***

II.7 La compagne d'évaluation INEX:

INEX (**IN**itiative for the **E**valuation of **X**ML Retrieval) est actuellement la seule campagne d'évaluation des différents SRIs pour des documents XML. Organisée chaque année depuis 2002. Le but principal d'INEX est de promouvoir l'évaluation de la recherche sur des documents XML en fournissant une collection de test, des procédures d'évaluation et un forum pour permettre aux différentes organisations participantes de comparer leurs résultats. La collection de test consiste en un ensemble de documents XML, requêtes et jugements de pertinence. Le langage de requêtes utilise dans INEX est NEXI (**N**arrowed **E**xtended **X**path **I**).

INEX est composé de plusieurs tâches tels que la tâche ad-hoc, la tâche multimédia, la tâche "relevance feedback", ou encore la tâche hétérogène.

II.7.1 La collection de test :

De 2002 a 2005, INEX a utilisé une collection composée d'articles provenant de revues de la " IEEE computer Society ", d'une taille totale avoisinant les 700 Mo, et sauvegardé sous le format XML. L'ensemble de documents contient 8 millions d'éléments. En 2005, une nouvelle collection a été ajoutée pour la tâche multimédia qui est la collection "LonelyPlanet".

A partir de 2006 et jusqu'a 2008, la collection " Wikipedia " a été utilisée dans la plupart des tâches. Cette collection de 6 Go, est composée de 659.388 documents d'une profondeur (nombre de niveaux) moyenne de 6.72. Le nombre moyen de nœuds XML par document est 161,35. Cette collection est également utilisée dans la tâche multimédia, elle contient environ 246.730 images.

D'autres collections sont aussi fournies par la campagne d'évaluation pour évaluer d'autres taches telles que la collection mmwikipedia pour une sous- tâche de la tâche multimédia, ou encore les collections fournies pour la tâche hétérogène.

Durant l'année 2009, une extension de la collection Wikipedia est fournie : elle est composée de 2.666.190 articles de Wikipedia annotés et elle a une taille de 50.7GB. Cette collection est utilisée dans la tâche ad-hoc ainsi que dans d'autres tâches.

II.7.2 Les requêtes:

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

- **Les requêtes CO (Content Only) :**

Ce sont des requêtes en langage naturel formulées avec de simples mots clés. Les SRI, en réponse à ces requêtes, doivent retrouver des éléments XML de granularité variée (du plus petit élément au document entier).

- **Les requêtes CAS (Content And Structure) :**

Ces requêtes permettent d'exprimer des contraintes sur la structure des documents en plus des contraintes sur le contenu.

Comme le montre la figure 2.3, chaque requête est caractérisée par un ensemble de champs: un champ 'title' donne la forme CO (ensemble de mots clés) de la requête, éventuellement un champ 'castitle' pour la forme CAS (structurée) de la requête. Un champ 'description' qui donne une description de la requête en langage naturelle, un champ 'narrative' qui donne le détail de la requête et les intentions de l'auteur de la requête.

```
<inex_topic topic_id="414" ct_no="3">
  <title>hip hop beat</title>
  <castitle>//*[about(., hip hop beat)]</castitle>
  <description>what is a hip hop beat?</description>
  <narrative>
    To solve an argument with a friend about hip hop music and beats, I
    want to learn all there is to know about hip hop beats. I want to know
    what is meant by hip hop beats, what is considered a hip hop beat,
    what distinguishes a hip hop beat from other beats, when it was
    introduced and by whom. I consider elements relevant if they
    specifically mention beats or rythm. Any element mentioning hip hop
    music or style but doesn't discuss abything about beats or rythm is
    considered not relevant. Also, elements discussing beats and rythm,
    but not hip hop music in particular, are considered not relevant.
  </narrative>
</inex_topic>
```

Figure 2.3 Exemple d'une requête INEX 2007.

II.7.3 La recherche Ad-hoc :

La recherche ad-hoc est la tâche principale d'INEX. Elle consiste en une simulation de l'utilisation d'une bibliothèque composée de documents XML, et interrogée par des requêtes utilisateurs portant à la fois, sur le contenu, et sur la structure. Les participants peuvent participer à trois sous-tâches distinctes :

La tâche **CO**, qui consiste à répondre avec des éléments ou des documents XML aux requêtes CO,

La tâche **SCAS**, qui consiste à répondre avec des éléments ou documents XML aux requêtes CAS de manière exacte.

La tâche **VCAS**, relative aux requêtes CAS, pour lesquelles les participants peuvent répondre de manière vague, c'est à dire avec des éléments ou documents qui satisfont globalement les requêtes.

II.7.4 Mesure d'évaluation :

Plusieurs mesures d'évaluation ont été définies selon les tâches et les années. L'ensemble de ces mesures tournent autour de deux notions :

- **l'exhaustivité :** elle mesure si tous les aspects de la requête ont été abordés dans le document.

- **la spécificité :** elle détermine si tout le contenu du document est concentré sur le thème de la requête.

En 2002, une première échelle de pertinence à deux dimensions a été proposée, basée sur le degré de pertinence et la couverture des éléments. Depuis 2003, ces deux dimensions ont été remplacées par la spécificité et l'exhaustivité. Pour chacune une échelle de 4 niveaux a été définie: pas exhaustif (resp. pas spécifique), marginalement exhaustif (resp. marginalement spécifique), assez exhaustif (resp. assez spécifique) et tres exhaustif (resp. très spécifique).

En 2005 et 2006, l'exhaustivité est mesurée selon une échelle a 4 niveaux: (e=2 exhaustivité élevée; e=1 exhaustivité moyenne; e=0 pas d'exhaustivité; e=? élément trop petit) et la spécificité quant a elle est mesurée dans un intervalle continu [0,1] ou s=1 représente un élément totalement spécifique.

Jusqu'au 2004, l'évaluation de pertinence des différents systèmes proposés par les participants utilise des méthodes basées sur les mesures de rappel et précision en tenant compte de la structure des documents XML et de la possible imbrication des résultats. Dans les campagnes INEX 2005 et 2006, d'autres mesures ont été définies pour permettre une évaluation plus appropriée des performances des systèmes de recherche en RI structurée **[Xavier]** : le gain cumule (xCG) et l'effort précision (ep).

Ø La mesure xCG cumule les scores de pertinence des éléments de la liste des résultats. Etant donnée une liste triée d'éléments dans laquelle les identifiants des éléments sont remplacés par leurs scores de pertinence, le gain cumulé au rang i, noté xCG[i], est calculé comme la somme des pertinences jusqu'à ce rang :

$$xCG(i) = \sum_{j=1}^{i} xG(j)$$

Où :

- *i* et *j* sont des rangs
- xG(j) est le score du document de rang *j*

Pour chaque requête on calcule un vecteur de gain idéal xCI à partir de la base de rappel, en cumulant les scores de pertinence des éléments triés par ordre décroissant. Le xCG peut alors être comparé au gain idéal. Le xCG normalisé (nxCG est obtenu par) :

$$nxCG(i) = \frac{xCG(i)}{xCI(i)}$$

Pour un rang i donné, le gain cumulé nxCG[i] reflete le gain relatif que l'utilisateur accumule jusqu'à ce rang, comparé à ce qu'il aurait pu atteindre si le système avait produit une liste triée optimale.

Ø L'effort précision (ep) est calculé comme suit :

$$ep(r) = \frac{e_{ideal}}{e_{system}}$$

Où :

- *e* est le rang auquel le gain *r i.*

Depuis 2007, les mesures officielles sont basées sur l'interpolation de Rappel/Précision sur 101 niveaux.

II.8 Conclusion :

Plusieurs collections de documents XML commencent à se constituer, développant dans le même temps des langages et des systèmes pour chercher l'information dans ces collections.

L'aspect structurel des documents XML offre un double avantage, celui de permettre à l'utilisateur de spécifier précisément ce qu'il cherche, et celui de permettre au système de retourner des documents ou parties de documents les plus spécifiques et les plus exhaustifs. En effet, les documents XML permettent aux utilisateurs de formuler des requêtes non seulement sur leur contenu, mais aussi d'utiliser leur structure pour des recherches plus précises.

Dans ce chapitre nous avons présenté la RI structurée, notamment dans des documents XML. Nous avons vu que la prise en compte de la structure dans le processus de recherche d'information engendrait de nouvelles problématiques au niveau de la granularité de l'information à retourner à l'utilisateur, au niveau l'expression du besoin en informations de l'utilisateur et au niveau de l'indexation des documents. Ensuite, nous nous sommes intéressés aux approches de recherche d'information pour répondre aux problématiques citées ci-dessus. Pour finir, nous avons présenté la compagne d'évaluation INEX, qui est la compagne de référence pour l'évaluation des systèmes de RI structurée. Dans le prochain chapitre nous allons voir des généralités sur les liens ainsi que les liens XML XLink et XPointer.

III .1 Introduction :

Parmi les technologies modernes de diffusion de l'information, la révolution de l'information est celle de l'hypertexte et du Web, qui permettent l'interconnexion des données, des ordinateurs et, finalement, des personnes. De même que pour la structure des documents, l'avènement de XML, comme le langage de description de liens XLink, tend à la description de plus en plus fine de la structure du Web, avec un typage des liens et des mécanismes d'adressage plus complets.

Dans ce chapitre nous allons donner une définition des liens et ses types ensuite nous présenterons les technologies XLink et XPointer.

III.2 Définition des liens Hypertextes :

Un hypertexte est une représentation non-linéaire d'une information textuelle sous la forme d'un graphe de nœuds connectés par des liens. La consultation d'un hypertexte nécessite une phase interactive de navigation.

Un nœud est une unité d'information, c'est-a-dire un fragment de texte (chapitre, section, etc.), ou un document entier. Il peut contenir un paragraphe, une page, ou même une image, un son ou une vidéo dans le cas d'hypermédia.

Un lien hypertexte définit une connexion entre deux nœuds (Unités d'informations) de l'hypertexte, le nœud source et le nœud destination du lien. Il permet à l'utilisateur de gérer un document ou un ensemble de documents de manière non linéaire. L'intérêt est de pouvoir naviguer dans un espace d'information en choisissant de suivre les associations que l'utilisateur juge pertinentes au moment de sa lecture.

III.3 Types de liens hypertextes :

Il existe trois types:

- **Liens internes :**
  Un lien est dit interne lorsqu'il pointe vers une page de la même machine.

- **Liens externes :**
  Un lien est dit externe lorsqu'il pointe vers une page se trouvant sur une autre machine.

- **Les signets :**
  Les signets sont utilisés dans le but de pointer un nœud de la page actuelle, en marquant un endroit précis de la page afin de s'y rendre grâce à un lien hypertexte contenant dans son URL le signet précédé d'un #.

III.4 Comportement d'un lien :

Le comportement d'un lien définit ce qui se produit lors du suivi d'un lien vers une ressource distante, le comportement d'un lien est décrit par les attributs ***show*** et ***actuate***.

L'attribut show peut prendre les valeurs suivantes :

- **new** : la ressource distante sera traitée dans un contexte n'affectant pas la ressource ayant activé le lien (ex: ouverture d'une nouvelle fenêtre).
- **replac**e : la ressource distante remplacera la ressource ayant activé le lien.
- **embed** : la ressource distante s'imbriquera dans la ressource ayant activé le lien.
- **other** : l'application doit chercher ailleurs dans le lien le comportement à adopter.
- **none** : pas de contrainte sur le comportement à adopter.

L'attribut actuate peut prendre les valeurs suivantes :

- **onLoad** : la ressource distante doit être chargée en même temps que la ressource locale.
- **onRequest** : la ressource distante ne sera chargée que sur demande explicite de l'utilisateur.
- **other** : l'application doit chercher ailleurs dans le lien le comportement à adopter.
- **none** : pas de contrainte sur le comportement à adopter.

III.5 Les liens XML :

Les liens dans les documents XML sont gérés par XLink combiné à XPointer, dans cette partie on va présenter ces deux concepts.

III.5.1 XLink :

XLink[1] pour « *XML Linking language* », est une recommandation du W3C depuis 2001 qui permet de créer des liens entre des documents XML ou des parties de documents XML grâce à *XPointer*[2]. Un lien XLink peut être ajouté à n'importe quel élément d'un document XML. Pour cela, un certain nombre d'attributs contenus dans l'espace de nom[3], sont mis à disposition des développeurs. Parmi ces attributs, on trouve notamment l'attribut « *type* » qui définit le type du lien, et l'attribut « *href* » qui définit l'URL[4] où pointe le lien.

III.5.1.1 Principales caractéristiques :

- XLink est capable de spécifier des liens pouvant relier plus de deux ressources, bidirectionnels, multidirectionnels et externes aux documents liés.
- XLink peut préciser le comportement des applications par rapport aux liens, notamment ouvrir un lien dans une nouvelle fenêtre.

[1] http://www.w3.org/TR/xlink11/
[2] http://www.w3.org/TR/xptr/
[3] http://www.w3.org/1999/xlink
[4] URL pour « *Uniform Resource Locator* », défini dans la RFC 3986: http://tools.ietf.org/html/rfc3986

- XLink offre un moyen de définir des liens dans un document à part, séparant le contenu des documents de la "navigation".

III.5.1.2 types de liens :

Deux types de liens sont supportés par XLink :

- Les liens simples
- Les liens étendus

a. Les liens simples :

Permettent de créer des liens simple reliant deux documents, où le document source est le document où est déclaré le lien, et le document destination est identifié par l'URL contenue dans l'attribut « *href* ».

Exemple

```
<cours>
<titre>cours de Java</titre>
<partieDeCours xlink:type="simple"
xlink:href="ferment/1er_semestre/java.html" xlink:show="embed"
xlink:role="initiation" />
<partieDeCours xlink:type="simple"
xlink:href="richome/2eme_semestre/java.html" xlink:show="embed"
xlink:role="perfectionnement" />
<documentation xlink:type="simple"
xlink:href="http://www.sun.com/javadoc.xml" xlink:show="embed"
xlink:role="documentation" />
</cours>
```

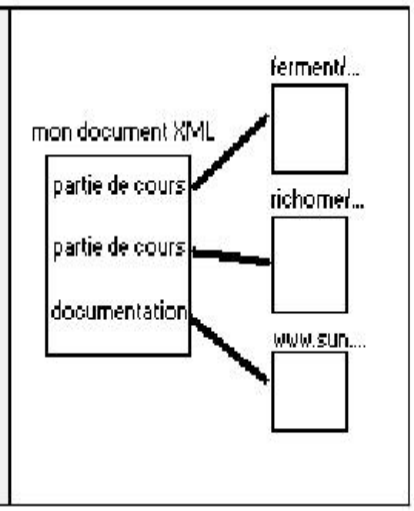


Figure 3.1: Exemple d'un XLink simple

Dans cet exemple, tous les liens sont de type simple. L'attribut show="embed" signifie que le contenu pointé est intégré dans la page source du lien. Ainsi, le document final (visualisé) est constitué de plusieurs blocs.

b. Les liens étendus:

Un lien étendu est un lien qui associe un nombre arbitraire de ressources. Les ressources participantes peuvent être représentées par toute combinaison de ressources distantes et locales, un lien étendu peut être en ligne (au moins une ressource locale) ou hors ligne. Il peut être unidirectionnel, bidirectionnels ou multidirectionnel.

Exemple

```
<COURS_HTML xlink:type="extended">
<CHAPITRE xlink:type="locator" xlink:role="chapitre1"
xlink:href="livre_java/chapitre1.xml" />
<CHAPITRE xlink:type="locator" xlink:role="chapitre2"
xlink:href="livre_java/chapitre2.xml" />
<CHAPITRE xlink:type="locator" xlink:role="chapitre3"
xlink:href="livre_java/chapitre3.xml" />
<CHAPITRE xlink:type="locator" xlink:role="chapitre4"
xlink:href="livre_java/chapitre4.xml" />
<LIEN xlink:type="arc" xlink:from="chapitre1"
xlink:to="chapitre2"/>
<LIEN xlink:type="arc" xlink:from="chapitre2"
xlink:to="chapitre3"/>
<LIEN xlink:type="arc" xlink:from="chapitre3"
xlink:to="chapitre4"/>
</COURS_HTML>
```

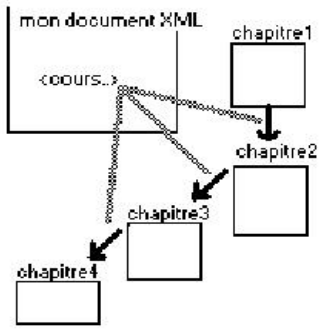


Figure 3.2: Exemple de liens XLink étendus.

Dans l'exemple ci-dessus, le type extended définit un lien étendu pouvant contenir des éléments XML ayant des attributs de type resource (source de lien), locator (pointeur sur un élément ou un document distant prenant part au lien) ou arc (renseigne sur le sens des relations entre élément "source" et élément "locator", ou entre éléments "locator" seulement).

Ø Les liens en ligne :

Un lien étendu est dit en ligne lorsqu'il contient au moins une ressource locale

```
<VERSIONS xlink:type="extended"
          xlink:title="Versions étrangères">
  <DOCUMENT xlink:type="resource"
            xlink:role="source">
    tutoriel de Java </DOCUMENT>
  <ORIGINAL xlink:type="locator" xlink:role="US"
     xlink:href="http://www.sun.com/javadoc.xml"
  <TRADUCTION xlink:type="locator" xlink:role="FR"
     xlink:href="http://www.pausejava.org/doc_java.xml"
  <TRADUCTION xlink:type="locator" xlink:role="RU"
     xlink:href="http://www.u-picardie.ru/bkj.xml"
     xlink:title="Version russe"/>
  <LIEN xlink:type="arc"
     xlink:title="version originale en anglais"
     xlink:from="source" xlink:to="US"/>
  <LIEN xlink:type="arc" xlink:title="en français"
     xlink:from="source" xlink:to="FR"/>
  <LIEN xlink:type="arc" xlink:title="en russe"
     xlink:from="source" xlink:to="RU"/>
</VERSIONS> est assez complet pour apprendre
```

Figure 3.3 : Exemple de liens étendus en ligne

Type = arc renseigne sur le sens des relations entre élément "source" et élément "locator", ou entre éléments "locator" seulement.

from = « valeur_de_role » et « to = valeur_de_role » précisent les éléments de départ et d'arrivée de l'arc.

Ø Les liens hors ligne :

Les liens hors ligne sont des liens étendus ne possédant aucune ressource locale au lien, leurs intérêts se situent dans leurs capacités à lier plusieurs ressources sans qu'ils aient à le savoir.

La gestion des liens donc peut alors se faire sans modifier les ressources, ce qui est utile lorsque :

- Ø Le nombre de ressources est élevé.
- Ø Les ressources sont en lecture seule ou inaccessible.
- Ø Il est onéreux de modifier les ressources.

```
<COURS_HTML xlink:type="extended">
   <CHAPITRE xlink:type="locator" xlink:role="chapitre1"
     xlink:href="livre_java/chapitre1.xml" />
   <CHAPITRE xlink:type="locator" xlink:role="chapitre2"
     xlink:href="livre_java/chapitre2.xml" />
   <CHAPITRE xlink:type="locator" xlink:role="chapitre3"
     xlink:href="livre_java/chapitre3.xml" />
   <CHAPITRE xlink:type="locator" xlink:role="chapitre4"
     xlink:href="livre_java/chapitre4.xml" />
   <LIEN xlink:type="arc" xlink:from="chapitre1"
     xlink:to="chapitre2"/>
   <LIEN xlink:type="arc" xlink:from="chapitre2"
     xlink:to="chapitre3"/>
   <LIEN xlink:type="arc" xlink:from="chapitre3"
     xlink:to="chapitre4"/>
</COURS_HTML>
```

Figure 3.4 : Exemple de liens étendus hors ligne

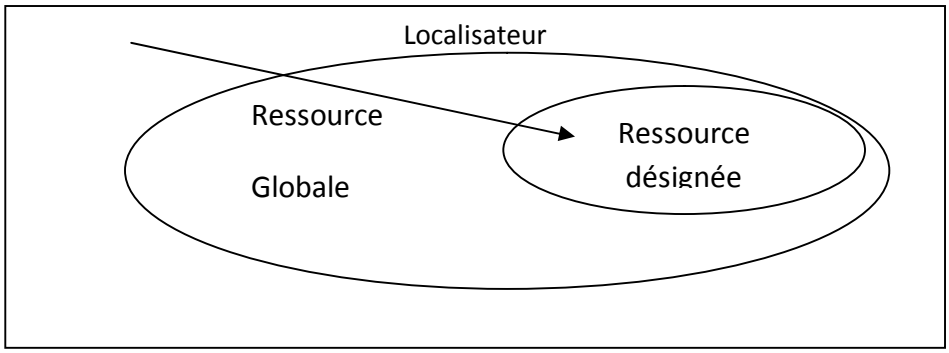
Les liens hors ligne (out-of-line) étant dans un autre fichier, ils permettent de distinguer les documents décrivant le contenu, de la spécification des relations entre les différents éléments du contenu. Autrement dit, de distinguer le contenu et la navigation dans ce contenu. La navigation peut être modifiée sans toucher aux documents du contenu.

III.5.1.3 Comportement d'un lien XLink :

Un lien XLink est composé des éléments suivants :

- Ø **simple**: lien simple;
- Ø **extended**: lien étendu;
- Ø **resource**: ressource locale au lien;

- **arc**: sens de liaison entre deux ressources du lien;
- **title**: élément directement lisible du lien;
- **none**: élément non interprété.
- **locator**: ressource distante du lien, il désigne une ressource qui dans le cas général est une **sous-ressource** d'une ressource complète.



Chaque élément possède des attributs :

- **type :** un attribut caractérisant l'élément ;
- **href :** un attribut de localisation ;
- **role :** permet de définir une typologie de liens (par exemple, un lien est de type note de bas de page, de type référence bibliographique, de type CV d'un auteur, etc.) ;
- **title** permet de titrer un lien (ce titre pourra être utilisé par un système de navigation).
- **label** permet de désigner une ressource;
- **from** et **to** permettent de donner à un arc sa direction.

III.5.2 XPointer

XPointer permet de référencer avec une granularité plus fine des parties de documents. XPointer est un mécanisme d'adressage de structure interne d'un document XML, qui utilise lui-même le langage XPath[5] (XML Path Language).

Aprés 3 ans de travail, XPointer est devenu une recommandation officielle de WC3 **[Grosso, 2003]**.

Elle se compose de 3 recommandations : XPointer Framework (la base), XPointer element scheme (adressage des éléments), et Xpointer xmlns scheme (interprétation des espaces de nommage dans les pointeurs).

- XPointer Framework décrit les types de média internet auxquels les recommandations XPointer proposées s'appliquent, ainsi que la syntaxe du langage XPointer.

[5] Langage permettant de localiser avec précision une partie donnée d'un document XML (Voir Annexe A).

- XPointer element scheme décrit comment, conjointement au XPointer Framework, il convient d'utiliser XPointer pour adresser des éléments XML dans une application.
- Xpointer xmlns scheme décrit le nom de domaine XML utilisé pour les pointeurs XML, y compris dans les préfixes et les noms qualifiés.

III.5.2.1 Syntaxe:

La syntaxe de base est la suivante : «en*xp+». Soit d'abord, éventuellement, des définitions d'espaces de nom («en»). Les espaces de noms sont décrits de la manière suivante : « xmlns(pref=URI)» où «pref» est le préfixe qui sera utilisé et «URI» décrit l'espace de noms. Puis une suite de références («xp»), la première dans l'ordre de déclaration retournant un résultat non vide est alors conservée. Les différents «xp» possibles sont :

- «xpointer(chemins_xpath_étendu)» ;
- «element(position_arbre)» : c'est une définition d'une séquence d'enfants à la place d'un chemin XPath qui est soit à partir de la racine ("/"), soit à partir d'un ID. Ensuite, les enfants sont notés par niveaux, puis, éventuellement, par positionnement dans le nœud texte (point ou région).

III.5.2.2 Localisation:

XPointer permet la localisation de parties XML (emplacements, lieux...) indépendamment des structures XML. Comme exemple d'utilisation, il peut servir à la localisation d'annotations après sélection (sur-lignage) par un utilisateur. Cela passe par la numérotation des nœuds et de points entre les nœuds et dans les textes. Par contre, il n'y a pas de localisation dans les espaces de noms et les attributs, car ils ne sont pas ordonnés.

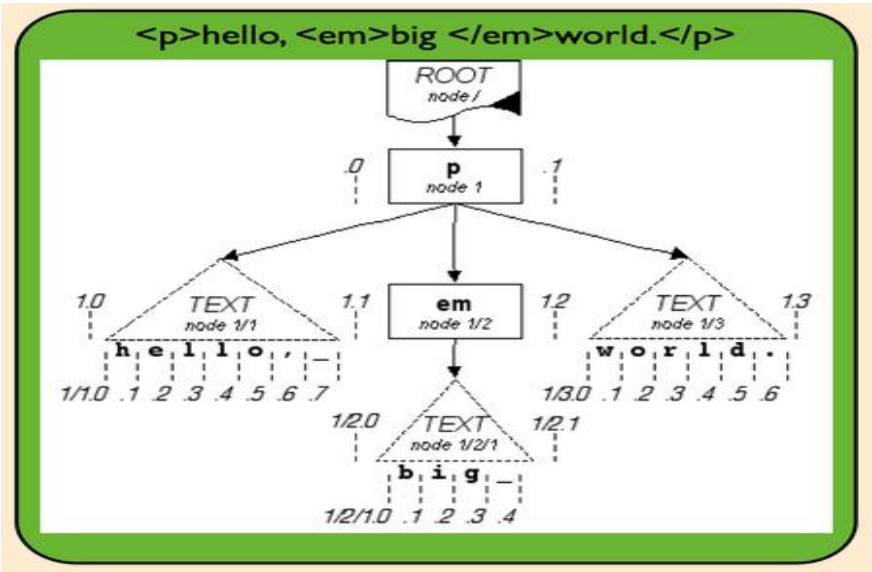


Figure 3.5: Exemple de localisation dans un document XML. **[Web 1].**

III.5.2.3 Les différents types de localisation :

a. Localisation absolue :

- Ø Accès a quelques parties importantes d'un document XML ;
- Ø Adressage de l'élément racine d'un document, d'un élément utilisant un mot clé, et d'un élément vec un identifiant donné.

- Ø Les mots clés sont : **root**(), **origin**(), **html** (*nom*), **id**(*nom*)

- Ø **id(nom) :** spécifie l'élément ayant un attribut ID de valeur nom ;

- Ø **root():** spécifie la racine du document actuel (valeur par défaut) et pointe sur le document tout entier **;**

- Ø **origin() :** pointe sur l'élément de lien lui-même et fournit fréquemment un emplacement absolu pour les étiquettes d'emplacements relatifs qui le suivent.

b. Localisation relative :

Ø Parcourir l'arbre du document :

Définir la direction à suivre par rapport à une localisation source qui peut être soit le terme de localisation absolu, soit le terme de localisation relatif précédent.

Il existe 12 termes relatifs, permettant par exemple de se diriger vers les éléments enfants, descendants, parents ou ancêtres de l'élément servant de localisation source.

Mot clé	***Effet***
Child	Sélectionne les éléments enfants de la source d'emplacement
Descendant	Sélectionne les éléments apparaissant dans le contenu de la source d'emplacement
Ancestor	Sélectionne les éléments dans lesquels le contenu de la source d'emplacement est trouvé (éléments parents)
Preceding	Sélectionne les éléments apparaissant avant la source d'emplacement

c. Localisation des parties :

Localiser toute une partie d'un document XML située entre deux emplacements dans le document appelés « points ». Pour localiser une région, on utilise l'opérateur « to » encadré par les points de début et de fin. Le deuxième point est calculé avec le premier comme référence.

d. Localisation de chaines :

localiser un texte dans un document XML : Fonction **« string-range() »** : elle permet de rechercher des chaînes dans un document et de renvoyer un ensemble de positions où elles apparaissent.

III.6 Conclusion :

Nous avons vu dans ce chapitre les concepts généraux relatifs aux liens, en commençant par les liens simples jusqu'aux liens XML à savoir les liens simples et étendus avec Xlink combiné à XPointer qui est un outil très puissant pour spécifier précisément les liens entre les documents. Les liens dans les documents structurés offrent plus de possibilités que ceux des documents non-structurés, donc pour en bénéficier ces derniers doivent être exploités, ainsi dans ce cadre des travaux ont été réalisés pour les exploiter, chose que nous allons voire en détail dans le chapitre qui suit sur l'exploitation des liens dans les documents XML.

IV.1 Introduction :

Dans le web, il est souvent possible de passer automatiquement d'un document à un autre à travers un simple clic sur un texte bien spécifique, ce dernier est ce qu'on appelle un lien, le premier document est appelé document consulté, et le deuxième document lié.

L'idée de base d'utilisation des liens est la suivante : un lien d'une page A vers une page B peut être visualisé comme une approbation de B par A, et en tant que jugement positif par A du contenu de B. En se basant sur cette idée, plusieurs algorithmes ont été proposés dans la RI et surtout sur le Web **[Carriere, 1997] [Kleinberg, 1998] [Mendelzon, 2000] [Najork, 2007]**. Ces travaux ont montré que l'exploitation des liens est très efficace pour l'amélioration des résultats. En effet, l'exploration des liens est utilisée généralement pour améliorer les performances de recherche en classant les pages les plus pertinentes à une requête utilisateur aux premiers rangs des résultats (Reranking) ou bien encore pour ramener d'autres pages pertinentes qui ne peuvent êtres retournées par une recherche classique.

Parmi les techniques les plus connues, nous citons PageRank et HITS. La popularité de ces deux algorithmes et le succès phénoménal du moteur de recherche Google, qui utilise PageRank, ont engendré un grand nombre d'algorithmes de recherche qui exploitent l'analyse des liens.

Nous décrivons dans ce chapitre les principaux algorithmes utilisés dans la recherche d'information classique et la recherche d'information structurée.

IV.2 Exploitation des liens dans la RI :

Les modèles de cette catégorie se basent sur les liens document-document. Ce type de liens est le plus répandu sur le web. Dans cette section, nous présentons les modèles les plus connus

.

IV.2.1 Page Rank :

Quelques moteurs de recherche, dont le plus connu est Google, ont pris le pari d'utiliser un autre mode de classement des résultats. Les pages Web sont ordonnées selon leur notoriété (popularité). Ce principe veut qu'une page web qui est la cible d'un grand nombre de liens est probablement non seulement une page validée (page parcourue par un très grand nombre de lecteurs, qui ont jugé bon de la citer en référence) mais aussi une page détenant un contenu utile à un grande nombre d'utilisateur.

Le page Rank **[Brin, 1998]** est la méthode de classement qui a fait la spécificité du moteur de recherche Google. L'approche du Page Rank repose sur la notion de propagation de popularité.

Le principe consiste à évaluer l'importance d'une page en fonction de chacune des pages pointant vers elle. La propagation met en avant les pages qui jouent un rôle particulier dans le

graphe des liens, avec l'hypothèse suivante : *"une page est importante quand elle est beaucoup citée ou citée par une page très importante"*.

IV.2.1.1 Formule de Page Rank :

La mesure de Page Rank (PR) est une distribution de probabilité sur les pages. Elle mesure en effet la probabilité PR, pour un utilisateur navigant au hasard, d'atteindre une page donnée. Elle repose sur un concept très simple : un lien émis par une page P vers une page B est assimilé à un vote de A pour B. Plus une page reçoit de votes, plus cette page est considérée comme importante. Le Page Rank (PR) se calcule de la façon suivante:

$$\mathbf{PR(P) = (1-d) + d\ [(PR(P_1)/C(P_1) + \ldots. + (PR(P_m)/C(P_m))]}$$

Avec :

PR(P) = Page Rank d'une page P.
$C(P_j)$ = nombre de liens sortant de la page P_j.
d = probabilité de suivre effectivement les liens pour atteindre la page A.
1-d = la probabilité d'atteindre la page A sans suivre de liens.

La valeur du Page Rank (score) d'une page P, est calculée par un algorithme itératif. Initialement, toutes les pages sont équiprobables, leur valeur de Page Rank est alors égale à 1/N avec N est le nombre de documents de la collection. Un cycle d'itération propage les probabilités sur les liens. L'algorithme s'arrête théoriquement lorsqu'une nouvelle itération ne produit plus de modifications dans le graphe des valeurs de PR.

L'avantage du PageRank est que le calcul des valeurs de popularité des documents n'est fait qu'une seule fois après l'indexation. Ainsi, on évite de refaire le calcul pour chaque requête au niveau de l'interrogation. Cependant, ceci signifie qu'un document populaire, le sera pour toutes les requêtes. Or, un document peut être populaire et servir de référence pour une requête portant sur un domaine particulier, et ne pas l'être pour une autre requête. Un autre problème lié à l'utilisation du PageRank se pose lorsque la collection de documents comporte des liens circulaires. Par lien circulaire, on entend tout lien qui pointe vers un document qui, lui-même, contient un lien qui pointe vers le document qui contient le premier lien. Ce problème fait que le PageRank des documents concernés (contenant des liens circulaires) augmente à chaque itération, sans converger vers une valeur fixe. Une solution à ce problème a été proposée grâce à l'utilisation du principe du surfer aléatoire **[Chebolu, 2008]** (« *Random Walk* »). D'autres inconvénients peuvent aussi apparaitre lorsque le PageRank est utilisé pour le web, notamment les problèmes liés au spamming qui consiste à créer des pages fictives pointant vers une page particulière pour augmenter son PageRank.

Vu le succès que connait PageRank, particulièrement pour le web, plusieurs auteurs ont proposé des approches similaires. On citera notamment le PageRank modulaire **[Jeh, 2003]** (« *Modular PageRank* »), le Page Rank calculé par blocs **[Kamvar, 2003]** (« *BlockRank* ») et le PageRank sensible à la thématique **[Haveliwala, 2002]** (« *Topic sensitive PageRank* »).

IV.2.2 HITS :

HITS (**H**ypertext **I**nduced **T**opic **S**earch), qui est un algorithme basé sur la "cocitation", a été proposé par Jon Kleinberg **[Kleinberg, 1998]** dans le cadre d'exploitation des liens pour le réordonnancement des résultats. Selon Kleinberg, les pages peuvent être divisées en deux catégories :

- **Les annuaires :** (hubs) qui sont des pages contenant peu d'informations pertinentes, mais beaucoup d'hyperliens ;

- **Les autorités :** qui sont des pages contenant peu de liens, mais beaucoup d'informations pertinentes.

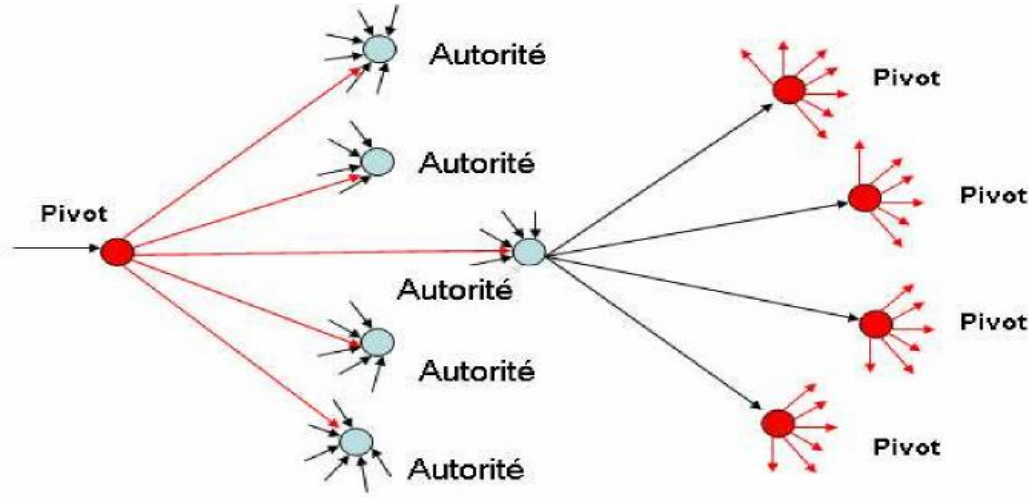


Figure 4.1: Exemple des pages pivots (Hubs) et autorités [Chibane, 2008]

Cet algorithme consiste à construire une collection C composée des pages Web de qualité concernant une requête donnée **q**. Il analyse par la suite la structure des liens introduite par cette collection afin de trouver les pages autorités et les pages hubs pour la requête **q**. La collection C est présentée par :

- Un ensemble S composé de N premières pages obtenues grâce à un moteur de recherche basé mot-clé.
- Un ensemble de base B contenant l'ensemble S en lui ajoutant les pages pointées par une page de S et les pages qui pointent vers une page de S.

Initialement, l'algorithme HITS associe à chaque page un poids autorité **xp** et un poids hub **yp** initialisés à 1. Les poids sont ensuite mis à jour de la façon suivante :

- pour chaque page p, la valeur de son poids autorité est mise à jour par la somme des poids hub de toutes les pages q pointant vers p.
- le poids hub d'un document est mis à jour par la somme des poids autorités des pages vers lesquelles ce document pointe.

Les opérations de mise à jour sont répétées jusqu'à la convergence des valeurs des poids hub et autorité (avoir une valeur fixe). A chaque itération, une normalisation est effectuée sur les poids de sorte que la somme des poids hubs (également autorités) de toutes les pages est égale à 1. Les hubs et les autorités son calculés de la façon suivante :

$$Y_p = X_{q1} + X_{q2} + \ldots\ldots\ldots + X_{qn} \qquad X_p = Y_{p1} + Y_{p2} + \ldots\ldots\ldots + Y_{pn}$$

L'un des avantages de l'algorithme de HITS est le double classement. En pratique, le résultat de HITS est composé de deux listes ordonnées : une liste de bonnes pages autorités et une autre de bonnes pages pivots qui seront renvoyées à l'utilisateur. L'utilisateur a l'embarras du choix entre les deux listes et il peut être intéressé par une liste au détriment de l'autre selon la recherche demandée.

Un des avantages de HITS est qu'il permet le classement des documents selon deux critères: le degré de pivots et le degré d'autorité qui est généralement retenu pour le classement final des documents. Côté inconvénients, HITS nécessite un temps de calcul supplémentaire au niveau de l'interrogation, car HITS dépend de la requête et les poids de pivots et d'autorité ne sont calculés qu'après la requête. Aussi, tout comme l'algorithme de PageRank, HITS est vulnérable aux techniques de spamming de liens lorsqu'il est utilisé pour le web.

Plusieurs travaux visant à améliorer les performances de HITS ont été proposés. On citera notamment, l'algorithme statistique PHITS proposé par Cohn et Chang **[Cohn, 2000]** afin de déterminer les hubs et les autorités. Ce modèle entièrement probabiliste a l'avantage de ramener plus d'informations que le modèle utilisé par l'algorithme HITS et l'algorithme TOPHITS **[Kolda, 2006]** similaire à HITS mais qui exploite en plus le texte de lien. Il calcule les scores hubs et les autorités ainsi que les scores des termes qui sont utilisés dans le texte des liens entre eux. Les termes qui ont les scores les plus élevés sont les termes les plus descriptifs d'une page.

IV.2.3 L'algorithme Salsa :

SALSA (**S**tochastic **A**lgorithm for **L**ink **S**tructure **A**nalysis) **[Lempel, 2001]** est un autre algorithme itératif combinant les techniques de HITS et Page Rank. Il applique le principe de surfer aléatoire de Page Rank sur le graphe biparti (HITS) en alternant entre le coté hub (pivot) et le coté autorité. Les pages les plus visitées correspondent alors aux solutions recherchées. Intuitivement, les pages qui sont considérées comme autorités pour une requête donnée sont liées depuis beaucoup de pages dans le sous-graphe étudié. Ainsi, en parcourant les liens du graphe, la probabilité de visiter une page autorité est élevée. Cette idée de détermination de l'intérêt d'une page se rapproche de celle utilisée dans le calcul du Page Rank. L'algorithme SALSA peut être considéré comme une variante de l'algorithme HITS. Dans l'algorithme HITS, les hubs diffusent leurs poids aux autorités et chaque autorité fait la sommation des poids des hubs qui lui pointent. Cependant, dans l'algorithme SALSA, chaque hub divise son poids de façon égale entre les autorités qu'il pointe. De même, chaque autorité divise son poids de façon égale entre les pages hubs qui lui pointent.

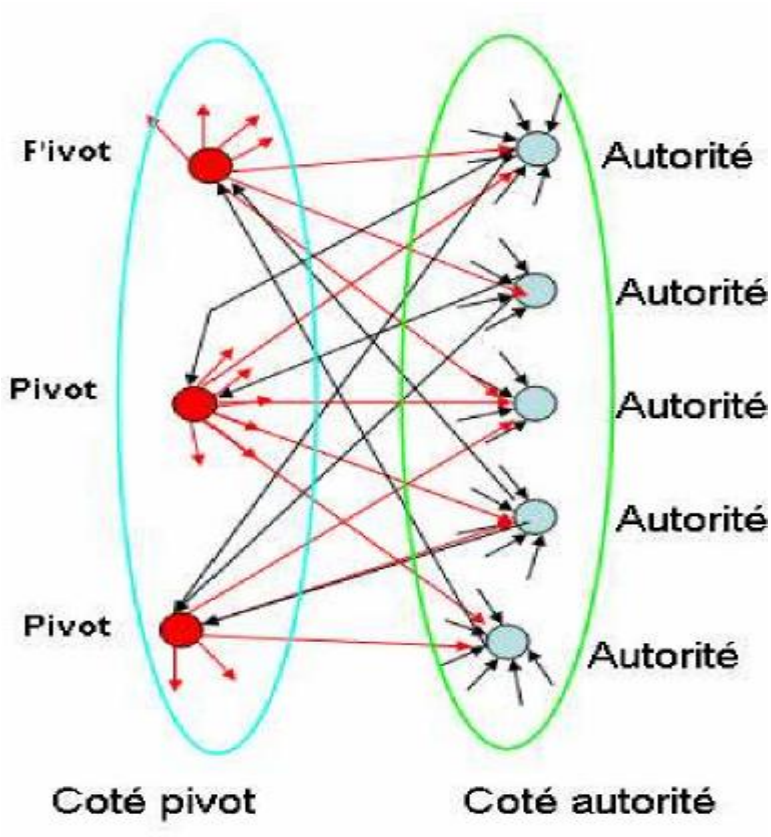


Figure 4.2: Graphe biparti de Salsa [Chibane, 2008]

IV.2.4 L'activation propagée :

Le principe de cette approche **[Crestani, 2000] [Savoy, 2000]** consiste à propager des valeurs de similarité de documents par rapport à une requête avec l'hypothèse suivante : "un document référencé par un grand nombre de documents pertinents est un bon document". La typologie du graphe des liens entre les pages est prise en compte au moment du calcul de la valeur de pertinence (Relevance Status Value RSV) des documents. Elle est alors fonction de la valeur de pertinence du document D_i par rapport à la requête Q, mais elle va aussi dépendre des valeurs de pertinence des documents liés au document D_i par rapport à la requête. Ainsi la valeur de pertinence finale notée RSV_{fin} se calcule comme suit :

$$RSV_{fin}(D_i, Q) = RSV_0(D_i, Q) + \lambda \sum_{i=1}^{k} RSV(Dj, Q)$$

Avec :

RSV_{fin} : La valeur finale de pertinence par rapport à la requête Q.
Di : un document.
Dj : Les documents liés à Di.
K : Le nombre de documents liés au document Di.
RSV_0 : La pertinence initiale du document Di.
λ : une constante destinée à atténuer la propagation.

La première étape consiste à calculer la similarité entre une requête donnée et les documents de la collection. Cette valeur est notée $RSV_0(D_i,Q)$. La deuxième étape consiste à propager cette valeur dans le réseau de documents à travers un certain nombre de cycles en utilisant des facteurs de propagations. A chaque cycle, la pertinence d'un document change en fonction de la pertinence des documents voisins.

IV.3 Exploitation des liens dans la RIS:

Les modèles de cette section s'intéressent aux liens élément-élément dans les documents XML. Ce type de liens est ajouté grâce à XLink.

Contrairement aux modèles de la catégorie précédente, la popularité de deux éléments dans un même document XML peut être différente. De plus, ce n'est pas une liste de documents qui est réordonnée mais une liste d'éléments.

Dans ce qui suit nous présentons les algorithmes XRANK et DocRank :

IV.3.1 XRank :

XRank **[Guo, 2003]** est un algorithme qui prend en compte les liens XML pour le réordonnancement de la liste des résultats. Dans leur méthode, le score d'un élément est en fonction de trois scores relatifs aux ensembles de relation CE (liens hiérarchiques entre les nœuds (lien ancêtre vers descendant)), HE (liens XLink entre les nœuds) et CE−1 (le même ensemble CE sauf que le sens des liens est inversé (lien descendant vers ancêtre)).

IV.3.1.1 ElemRank :

ElemRank d'un élément x du document XML représente l'importance de cet élément dans le document. Celui-ci est calculé en utilisant la structure d'hyperliens hiérarchiques du document. Sur le plan conceptuel, l'ElemRank similaire au PageRank sauf que l'ElemRank est défini à la granularité de l'élément non à la granularité du document entier.

La formule utilisée pour calculer l'ElemRank est la suivante :

$$e(x) = d_1 \sum_{(u,x)\epsilon HE} \frac{e(u)}{Nh(u)} + d_2 \sum_{(u,x)\epsilon CE} \frac{e(u)}{Nh(u)} + d_3 \sum_{(u,x)\epsilon CE} \frac{e(u)}{Nh(u)} + \frac{1-d1-d2-d3}{Nd*Nd(x)}$$

où :

e (i) : ElemRank de l'élément i .

N_h (u) : nombre d'hyperliens sortants de u.

N_d (x) : nombre de d'éléments dans le document contenant l'élément x.

N_d : nombre totale de documents.

IV.3.2 DocRank :

DocRank **[Mattaoui, 2009]** représente une extension du PageRank adaptée aux collections de documents XML. Dans cette approche chaque élément se voit attribuer un score en se basant sur le score de popularité du document qui le contient. Ce score de popularité est calculé pendant l'indexation en suivant la formule :

$$\text{DocR (D)} = \frac{1-\text{d}}{\text{colldoc}} + \text{d} \sum_{(i,D)\in links} \text{DocR(i)}$$

Avec :

DocR (D) : Le score de popularité du document D ;

d : Un facteur d'amortissement;

colldoc : Le nombre de documents dans la collection;

links (i,d) : Représente un lien du document i vers le document D ;

A la fin de l'indexation chaque documents aura son score de popularité, ainsi suite à chaque requête le score des éléments est calcule de la manière suivante :

$$\text{DocR } (E_i) = \alpha * \text{ScoreInit } (E_i) + (1-\alpha) * \text{DocR } (D)$$

Avec :

DocR (E_i) : Le score de l'élément E_i
D : Le document qui contient l'élément E_i

DocR (D) : Le score de popularité du document D ;

α : Un paramètre qui permet de définir le degré de contribution des différents scores dans le score final;

ScoreInit : Lle score initialement affecté par le système de recherche à l'élément

IV.4 Autres approches :

D'autres travaux **[Wang, 2008]** et **[Kamps, 2008]** utilisent les liens XML pour le réordonnancement des résultats renvoyés selon deux degrés :

- local indegree : représente le nombre de liens de la collection entrants à un article.
- global indegree : représente le nombre de liens entrants à un article à partir des documents renvoyés comme résultats à une requête donnée.

L'algorithme HITS est utilisé par d'autres utilisateurs pour le filtrage de documents, en affectant un rang à chaque document dans le corpus filtré.

IV.5 Notre approche :

La méthode que nous utilisons est proposée par Mme Fellag **[Fellag, 2012]**. Elle propose de transmettre une fraction de score de pertinence de chacune des pages extraites du Web vers ses voisines (les pages liées directement). Un modèle de recherche est utilisé pour obtenir une liste triée de pages Web, et en se basant sur cette liste, les scores de pertinence des documents seront propagés selon les hyperliens sans tenir compte de leur orientation entrants ou sortants, tout en exploitant l'information portée sur le titre et le texte ancre des liens.

Cette méthode sera détaillée et illustrée avec un exemple applicatif dans le chapitre qui suit.

IV.6 Conclusion :

Dans ce chapitre, nous avons défini les principes généraux qui régissent les modèles de propagation en recherche d'information et répertorié les algorithmes les plus classiques. De plus Nous avons soulevé certains problèmes majeurs de chaque approche étudiée. En effet, plusieurs travaux ont été menés sur l'utilisation des liens dans la recherche d'information sur le Web, et actuellement de nombreuses approches sont en cours d'expérimentation pour avoir un gain significatif par rapport aux méthodes de recherche reposant seulement sur le contenu. Dans ce qui suit nous présentons un modèle de propagation de score de pertinence.

V.1 Introduction :

Dans le chapitre précédent, nous avons vu que la plupart des modèles existants pour la prise en compte des liens dans la RI se limitent à l'exploitation du nombre de liens (HITS et SALSA) et à la propagation des scores des documents (Activation propagée). Nous étudions un nouveau modèle de RI structurée qui exploite, le texte ancre des liens, la balise titre des documents et la propagation des termes pour améliorer la qualité de réponse du système.

Dans ce chapitre nous présentons en détail notre modèle en indiquant en premier lieu les dispositions à prendre au niveau de l'indexation, puis nous détaillons notre contribution pour la prise en compte des liens dans les documents XML, enfin nous terminons par un exemple illustratif montrant le fonctionnement de notre approche.

V.2 Les principales étapes :

V.2.1 Indexation

L'indexation est une phase très importante pour pouvoir mettre en œuvre notre approche, ainsi pour chaque document il est nécessaire d'indexer son titre, les liens qu'il contient ce qui nous permettra à chaque instant et pour chaque document de savoir la liste des documents qu'il référence et la liste des documents qui l'ont référencés et le texte ancre de chaque lien. Toutes ces informations vont nous permettre de faire le calcul des fonctions de correspondance.

Dans ce qui suit, nous présentons l'indexation des éléments indexés.

V.2.1.1 Indexation des titres

L'indexation des titres est réalisée en même temps que l'indexation de l'information structurelle, pour indexer ces derniers on passe par les étapes suivante : l'analyse lexicale, l'élimination des mots vides et la troncature des termes a 7.

V.2.1.2 Indexation des liens

L'indexation des liens consiste à extraire depuis chaque document l'ensemble de liens qu'il référence et le texte ancre de ces derniers.

Le texte ancre des liens consiste à attacher pour chaque lien, l'ensemble des termes qu'il contient et ces termes sont indexés de la même manière que le texte de la balise titre.

V.2.1.3 Exemple d'indexation

```
- <article>
    <name id="12">Anarchism</name>
    <conversionwarning>0</conversionwarning>
  - <body>
      <template name="anarchism"> </template>
      <emph3> Anarchism </emph3>
      is a generic term describing various
      <collectionlink xlink:type="simple" xlink:href="23040.xml"> political philosophies </collectionlink>
      and
      <collectionlink xlink:type="simple" xlink:href="234984.xml"> social movement </collectionlink>
      s that advocate the elimination of all forms of
      <collectionlink xlink:type="simple" xlink:href="12229.xml"> government </collectionlink>
```

Figure 5.1 : Exemple d'un document XML a indexé.

L'indexation de la balise titre et des liens de la figure ci-dessus nous donnent les 2 tables suivantes :

Id_doc	Titre_doc
12	anarchi

Figure 5.2 : Indexation du titre.

Doc_source	Doc_destination	Terme
12	23040	Polotic philoso
12	234984	Social movemen
12	12229	governm

Figure 5.3 : Indexation des liens.

V.3 Présentation de la formule:

L'exploitation des liens dans notre approche est réalisée lors de la phase de recherche, tout en rappelant que la granule de l'information n'est pas le document entier mais seulement une partie de ce dernier ; c.-à-d. l'élément. La recherche des éléments réponses se fait en suivant deux phases :

- la première phase consiste à retrouver les éléments réponses à une requête ***q*** donnée sans prise en compte des liens en interrogeant le corpus de document. Suite à l'interrogation de ce dernier à travers la requête ***q***, un ensemble ***E*** d'éléments réponses est retourné dans lequel chaque élément ***e*** est représenté par un score de pertinence appelé **scoreelt**. Les éléments réponses de l'ensemble *E* sont obtenus sans prise en compte des liens.

- la seconde phase consiste à exploiter les liens pour non seulement retrouver des éléments inaccessibles lors de la première phase mais aussi pour réordonnancer les éléments restitués lors de cette phase. Cette phase est réalisée selon l'intuition suivante **[Fellag, 2012]** : "*si des termes de la requête sont présents dans le lien, le document référencé se rapporte forcément à la requête, et si de plus les termes de la requête se retrouve dans son titre alors ce document ne peut être que pertinent pour la requête et de ce fait ces éléments le sont aussi ".*

Pour une bonne mise en œuvre de cette méthode deux scores sont définies : **scoreLink** qui mesure le contenu informationnel du lien et **scoretitre** qui mesure l'information portée par la balise titre du document référencé, le calcul de ces deux scores se réalise par les deux fonctions décrites ci-dessous :

Ø Le score du titre « *SCORE* $_{titre}$ » :

Le titre du document est considéré comme un élément important car il résume le contenu informationnel du document, ainsi si le titre du document contient des termes de la requête alors ce dernier ne peut être que pertinent pour la requête, et donc son score est calculé comme suit :

$$\mathbf{score_{titre}} = \frac{\sum_{i=1}^{n} \mathbf{tf_{qi}}}{\sum_{j=1}^{m} \mathbf{tf_{titrej}}} \qquad (1)$$

avec

- tf_q : fréquence du terme de la requête figurant dans le titre ;
- tf_{titre} : fréquence du terme dans le titre du document référencé ;
- n : nombre de termes de la requête figurant dans le titre ;
- m : nombre de termes total du titre et $n <= m$.

Ø Le score du lien « *SCORE* $_{link}$ » :

Les liens entrants vers un document sont exploités selon l'intuition **[Fellag, 2012]**: "*le concepteur d'un document se rapportant à un sujet donné ne fait référence à un autre document que s'il le considère comme pertinent sur ce même sujet".* Et en se basant aussi sur l'intuition : « *si des termes de la requête sont présents dans le lien, le document référencé se rapporte forcément à la requête* » un score link est calculé pour chaque lien entrant vers le document considéré en appliquant la formule suivante :

$$\mathbf{score_{link}} = \frac{\sum_{i=1}^{n} \mathbf{tf_{qi}}}{\sum_{j=1}^{m} \mathbf{tf_{linkj}}} \qquad (2)$$

avec

- Ø tf_q : fréquence du terme de la requête figurant dans le lien ;
- Ø tf_{link} : fréquence du terme dans le lien ;
- Ø n : nombre de termes de la requête figurant dans le lien ;
- Ø m : nombre de termes total du lien et $n <= m$.

Ainsi un document A peut juger un autre document B comme étant pertinent et cela en propageant son score de pertinence à ce dernier en se basant sur la formule [3] suivante :

$$\mathbf{Score_{doc}} = a\mathbf{Score_{titre}} + \sum_{i=1}^{k} g_i \mathbf{score_{refi}} \qquad (3)$$

Avec :

- Ø ***SCORE*** $_{doc}$ est le score calculé pour chaque document cible ;
- Ø Le paramètre a mesure l'importance accordée à l'information portée par la balise titre et prend ses valeurs dans l'intervalle [0,1]. Sa valeur finale sera fixée par expérimentation.
- Ø ***SCORE*** $_{titre}$ **:** mesure l'information portée par la balise titre du document référencé, celui-ci est défini ci-dessus;
- Ø g_i est le facteur de pondération de score de pertinence propagé du document source vers le document cible et mesure le contenu informationnel du lien ainsi :

$$g = \mathbf{score_{link}} + l$$

- Ø ***SCORE*** $_{link}$ **:** le score des liens entrant vers le document cible, celui-ci est défini ci-dessus.
- Ø Le paramètre l est égale a la formule ci-desous si $score_{link} \in [0,1[$ sinon l est egale a 0

l =1/(Nombre de liens sortants du doc initial+ε) ;

- La valeur du paramètre ε sera fixée par expérimentation.

- ***Score*** $_{refi}$: score de la *source* qui a référencé le document, c'est le score de l'élément si c'est la première référence c'est le score du document par la suite.

- *k* : le nombre de liens entrants vers le document.

Le résultat obtenu à l'issue de cette étape est un ensemble *D* de documents ordonnés par ordre décroissant des scores.

Comme rappelé précédemment, l'unité d'information à restituer dans la RIS étant l'élément il est impératif d'identifier à partir de *D* les éléments pertinents, comme chaque document du corpus est représenté par un ensemble d'éléments, l'identification des éléments devient aisée. Néanmoins, le score sera calculé comme suit :

$$\mathbf{Score}_{\mathbf{elt}} = b\mathbf{Score}_{\mathbf{eltinitial}} + \mathbf{(1}-b\mathbf{)Score}_{\mathbf{doc}} \qquad (4)$$

Avec :

- **Score** $_{elt}$ est le nouveau score de l'élément;
- **Score**$_{eltinitial}$ est le score initial de l'élément calculé dans la première phase de recherche.
- ***SCORE*** $_{doc}$ est le score du document calculé dans la formule (3).

- Le paramètre b prend sa valeur dans l'intervalle [0,1]. Sa valeur finale sera fixée par expérimentation.

V.4 Algorithme de recherche :

Données : *C* = {*d* / *d*= ensemble d'élément *e*}, *e*={(*t1*,*p1*), …… ,(*tn*,*pn*)/ *n* :nombre de termes représentants l'élément *e*}, *q* = {(*t1*,*pq1*), …… ,(*tn*,*pqm*)}/ *m* : nombre de termes de la requête}
Résultats : Ensemble *E'* d'éléments réponses

1. Pour chaque élément *e* d'un document *d* du corpus *C*
   1.1 calculer le *scoreelt* de *e* avec la requête *q*
   1.2 sauvegarder dans *E* par ordre décroissant du *scoreelt*
2. Pour chaque *e* de l'ensemble *E*
   2.1 retrouver tous les liens sortants vers les documents de *C*
   2.2 pour chaque document *cible*
      2.2.1 calculer *scoretitre* avec la formule *(1)*

2.2.2 calculer *scorelink* de tous les liens *entrants* vers ce dernier avec la formule *(2)*
2.2.3 calculer *scoredoc* avec la formule *(3)*
2.2.4 sauvegarder chaque document dans *D* par ordre décroissant des scores.
2.2.5 reprendre à partir de 2.1

3. Pour chaque document *d* de *D*

3.1 identifier les éléments *e*
3.2 Pour chaque élément *e*
3.2.1 calculer *scoreelt* avec la formule *(4)*
3.2.2 sauvegarder dans *E'* par ordre décroissant des scores.

V.5 Exemple d'application de l'approche :

Avant de montrer la mise en œuvre de notre approche on considère un ensemble de 10 premiers documents retournés par un SRI classique classés par ordre décroissant de valeur de score et qui sont montrés dans la table suivante :

Rang du document	Numéro du Document	Valeur RSV
1	1.xml	1.55
2	8.xml	1.53
3	5.xml	1.401
4	13.xml	1.034
6	12.xml	0.90
7	4.xml	0.855
8	9.xml	0.3124
9	2.xml	0.31004
10	20.xml	0.29

La liste des documents retournés dans une première recherche

Pour mieux expliquer la méthode, on considère le premier document retourné **1.xml** dont la valeur RSV est 1.55 qui nous envoi vers deux documents **2.xml** et **3.xml**, nous allons montrer la variation des valeurs RSV des documents retournées et ce, en appliquant la formule indiqué ci-dessus pour un nombre de niveaux égal à 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
<name id="1">exemple</name>
```

Figure 5.4 : Document 1.xml

Dans l'exemple que nous avons considéré nous allons lister pour chaque document (2.xml et 3.xml) la liste des documents qui les ont référencés, la liste des documents qu'ils ont référencés et le nombre de liens sortants. En partant de ces informations on peut former le graphe suivant :

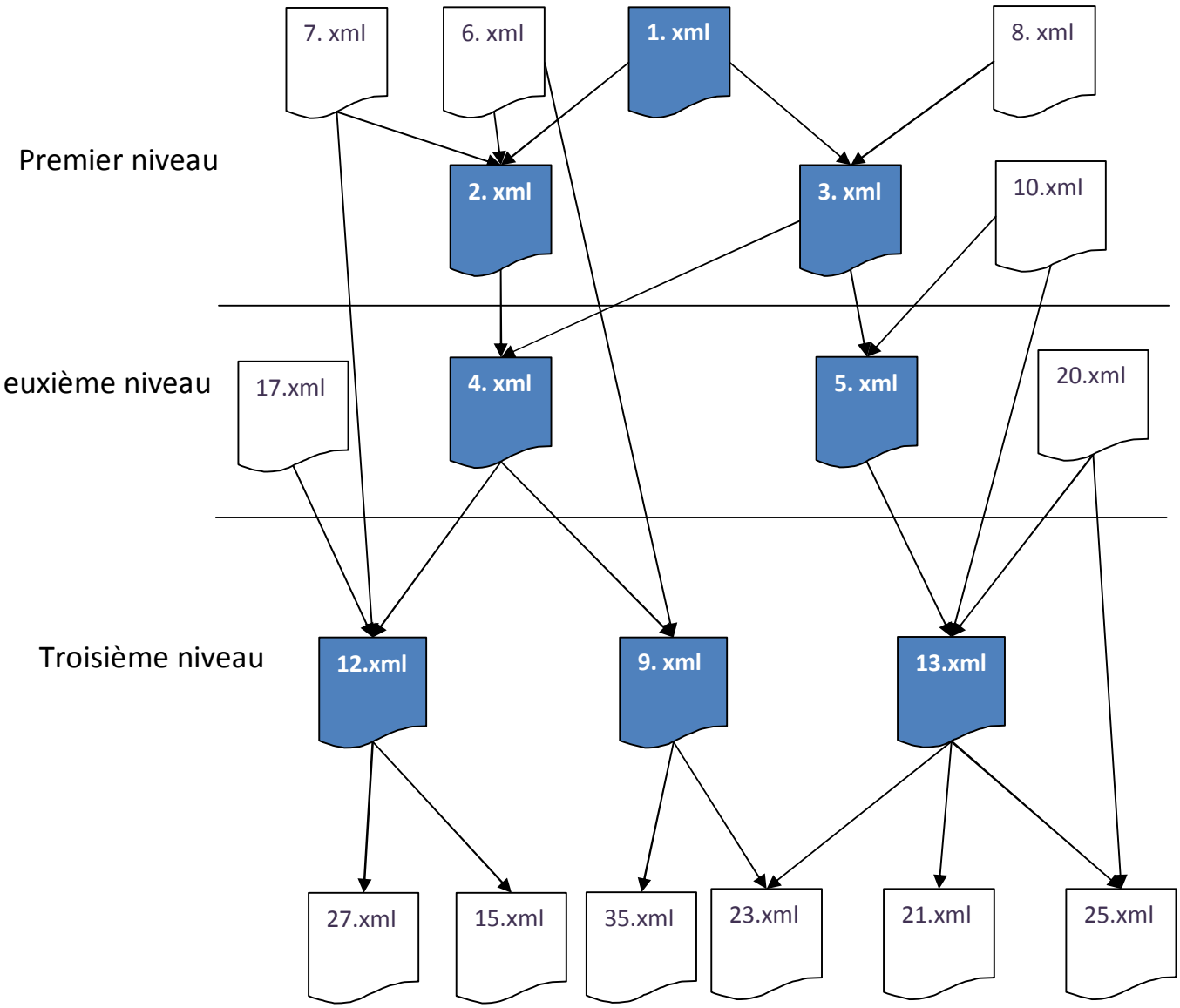


Figure 5.6 : Arbre de référence entre documents

A partir de cet arbre on peut construire une table récapitulative contenant (pour chaque document référencé) les informations nécessaire pour appliquer notre formule. Ainsi l'arbre de notre exemple nous donne la table suivante : (le document, la listes des documents qui ont référencés ce documents et leurs score avec Si étant le score du document numéro i, la liste des documents qu'il a référencé, et sa valeur RSV « Si » calculé suivant la formule .

Document	Les documents qui l'ont référencé / son score		Les documents qui a référencé :	Nombre de liens sortants	Valeur RSV
2. xml	1. xml	S1	4. xml	1	S2
	6. xml	S6			
	7. xml	S7			
3. xml	1. xml	S1	4. xml	2	S3
	8. xml	S8	5.xml		
4. xml	2. xml	S2	12. xml	2	S4
	3. xml	S3	9. xml		
5. xml	3. xml	S3	13. xml	1	S5
	10. xml	S10			
12. xml	7. xml	S7	27.xml	2	S12
	17. xml	S17			
	4. xml	S4	15.xml		
9. xml	6. xml	S6	35.xml	2	S9
	4. xml	S4	23.xml		
13. xml	10. xml	S10	25. xml	3	S13
	5. xml	S5	21. xml		
	15. xml	S15	23. xml		

Figure 5.7: tableau récapitulatif sur l'arbre des documents.

V.5.1 Processus de propagation de score :

Notre formule ne s'applique pas uniquement sur les documents 1.xml et 2.xml, car pour chaque ensemble de documents retourné on cherche les documents référencés ainsi on réapplique la formule pour avoir de nouveaux scores, ce processus est répété pour un nombre x de fois jusqu'à ce que les scores obtenus soient non significatif (le score de la prochaine itération est le même que celui de l'itération précédente). Pour l'exemple qu'on a pris on

considère trois niveaux : (avec Ti le score du titre du document i, et Li le score du lien sortant du document i suivant une requête donnée)

- **Premier niveau :**

A partir du document 1.xml on constate qu'il référence deux documents : 2.xml et 3.xml ainsi pour chacun des deux on doit chercher la liste des documents qui l'ont référencé ainsi :

Score (2.xml) = α T2 + [(L1 * ½ *S1) + (L6 * ½ *S6) + (L7 * ½ *S7)]

Score (3.xml) = α T3 + [(L1 * ½ *S1) + (L8 * ½ *S8)]

- **Deuxième niveau :**

Etant le résultat de la première operation est (2.xml et 3.xml) on cherche les documents référencé par ces derniers et on calcule le score de chacun, ainsi on a :

Score (4.xml) = α T4 + [(L2 * ½ *S2) + (L3 * ½ *S3)]

Score (5.xml) = α T3 + [(L3 * ½ *S3) + (L10 * ½ *S10)]

- **Troisième niveau :**

De même pour la troisième on a :

Score (12.xml) = α T12 + [(L7 * ½ *S7) + (L17 * ½ *S17) + (L4 * ½ *S4)]

Score (9.xml) = α T9 + [(L6 * ½ *S6) + (L4 * ½ *S4)]

Score (13.xml) = α T13 + [L10 * ½ *S10) + (L5 * ½ *S5) + (L20 * ½ *S20)]

Le paramètre *a* mesure l'importance accordée à l'information portée par la balise titre et prend ses valeurs dans l'intervalle [0,1]. Une série de test et d'expérimentation va nous permettre de fixer sa valeur finale.

Ce processus est appliqué pour chaque document retourné à savoir (8, 5, 13, 12, 4, 9, 2, 20) et le résultat final est un ensemble de documents correspondant à l'ensemble initial mais avec un ordre différents et des valeurs de scores différentes comme on peut constater l'apparition de nouveau document non retournés au préalable (ex : 3.xml), ainsi le résultat final aura la forme suivante :

Rang du document	Numéro du Document	Valeur RSV
1	5.xml	1.056
2	1.xml	0.953
3	8.xml	0.810
4	3.xml	0.600
6	12.xml	0.590
7	4.xml	0.440
8	9.xml	0.400
9	12.xml	0.399
10	2.xml	0.345

La liste des résultats après exploitation des liens

Après l'application de la formule de nouveaux scores sont obtenus, les valeurs de ces scores permettent de changer l'ordre de classement ainsi le réordonnancement est réalisé.

V.6 Conclusion :

Nous avons présenté dans ce chapitre une approche permettant l'exploitation des liens dans la recherche d'information semi structuré, comme nous avons détaillé la façon de mettre en jeu le contenu informationnel des balises à savoir le titre, le lien et les termes de ces liens ainsi l'intégration de ces informations pour le calcul des différents scores.

Dans ce qui suit nous allons implémenter ce modèle de recherche pour faire des expérimentations et évaluations sur la collection de test INEX 2006.

VI.1 Introduction :

Nous avons étayé dans les chapitres précédents un état de l'art des travaux pour la recherche d'information structurée. Afin de pouvoir retrouver l'information pertinente au sein des documents XML, nous avons également présenté la notion de liens ainsi que les différents travaux consacrés à leur exploitation, nous avons par la suite implémenté une méthode pour exploiter ces liens, en utilisant les résultats retournés par le système de recherche Maxplank[1] **[Martin, année],** ce que nous allons détailler dans le présent chapitre.

VI.2 Description de l'environnement de travail :

Au cours du développement de cette application, nous avons eu affaire à plusieurs composantes technologiques et API :

VI.2.1 Sous Linux :

Pour réaliser des tests et des évaluations préliminaires nous avons exploité le système de recherche XFIRM[2] pour des petites collections de documents, ce système fonctionne en ligne de commande via un terminal et nécessite un environnement de travail Linux et une base de donnée oracle dans laquelle sont stockés les index. Les résultats de notre modèle auraient pu être comparés avec ceux obtenus avec XFIRM, mais vu la taille immense de la collection INEX 2006 qui aurait pris beaucoup de temps pour l'indexer et exécuter les différentes requêtes nous avons utilisé des résultats qui étaient disponibles qui sont ceux de SRI Maxplanck. Comme nous avons utilisé un IDE Eclipse sous linux pour exécuter nos différents programmes. Ce système nous a permis également de mettre en œuvre l'outil d'évaluation EvalJ[3] (voir section IV.6.2.4).

VI.2.2 Sous Windows :

La grande partie de notre travail a été effectué sous linux, mais la dernière partie qui consistait à exploiter les résultats retournés par le système de recherche Maxplanck a été réalisé sous Windows, ainsi différents outils ont été mis en jeu à savoir un environnement de développement Eclipse et une base de données Postgres dans laquelle sont stockés les résultats d'exécution de chaque requête pour chaque système, la liste des topics et différents index.

VI.3 Architecture du modèle de liens :

L'idée principale est d'exploiter le contenu du lien et le titre de document référencé, et ce via la propagation de score vers ce dernier en utilisant une formule de propagation de score. Le processus de recherche est illustré dans le schéma suivant :

[1] http://www.mpi-inf.mpg.de/departments/d5/
[2] Voir Annexe B
[3] http://sourceforge.net/projects/evalj/

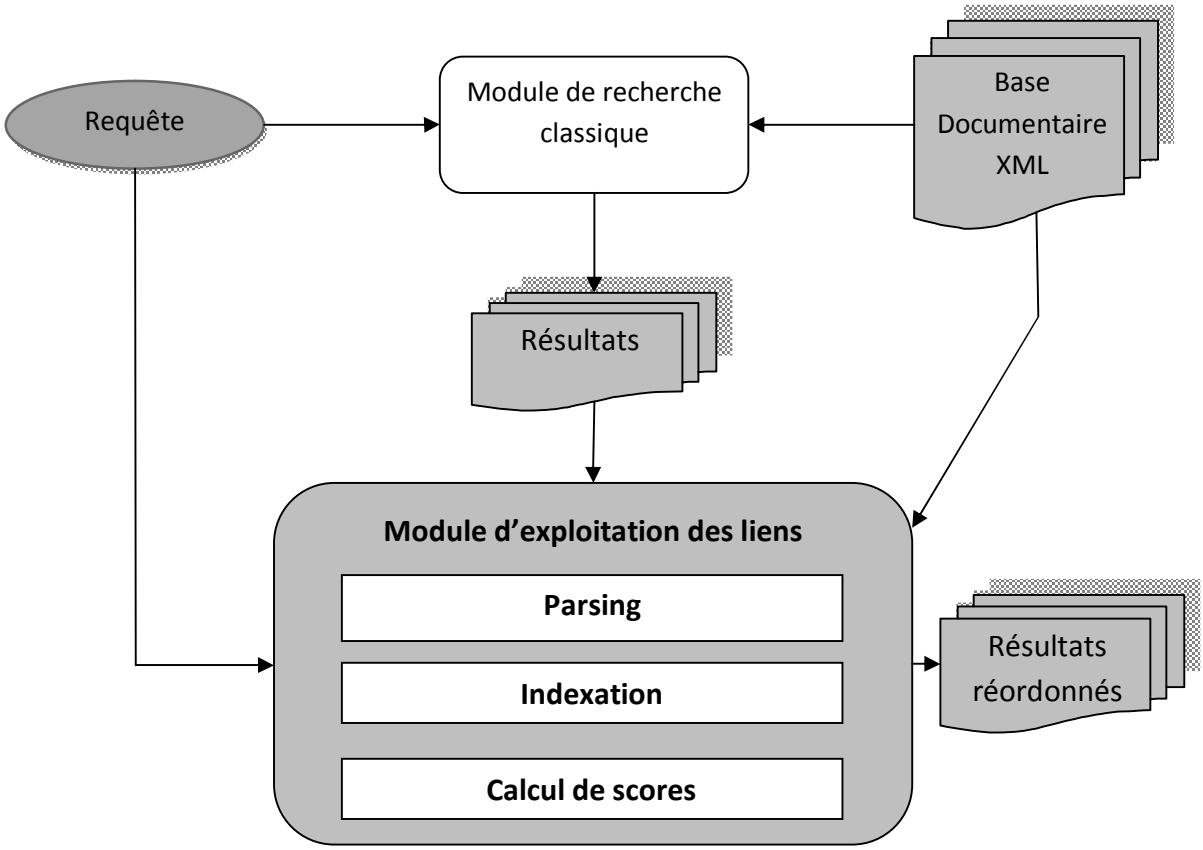


Figure 6.1 : Architecture du système d'exploitation de liens.

Cette architecture exploite les résultats retournés par un système de recherche sans prise en compte de liens à une requête donnée qu'il reçoit en entrée et retourne en final une liste de résultats réordonné après exploitation des liens.

VI.4 Utilisation des liens pour le réordonnancement des résultats (Reranking) :

Pour réaliser un réordonnancement des premiers résultats, nous avons utilisé dans nos expérimentations, les scores initiaux pour le réordonnancement final des éléments. Les expérimentations présentées dans cette section ont pour but d'évaluer l'impact des liens pour le réordonnancement des résultats, la figure ci-dessous montre les résultats de la requêtes 289 (emperor "Napoleon I" Polish) avant et après exploitation des liens

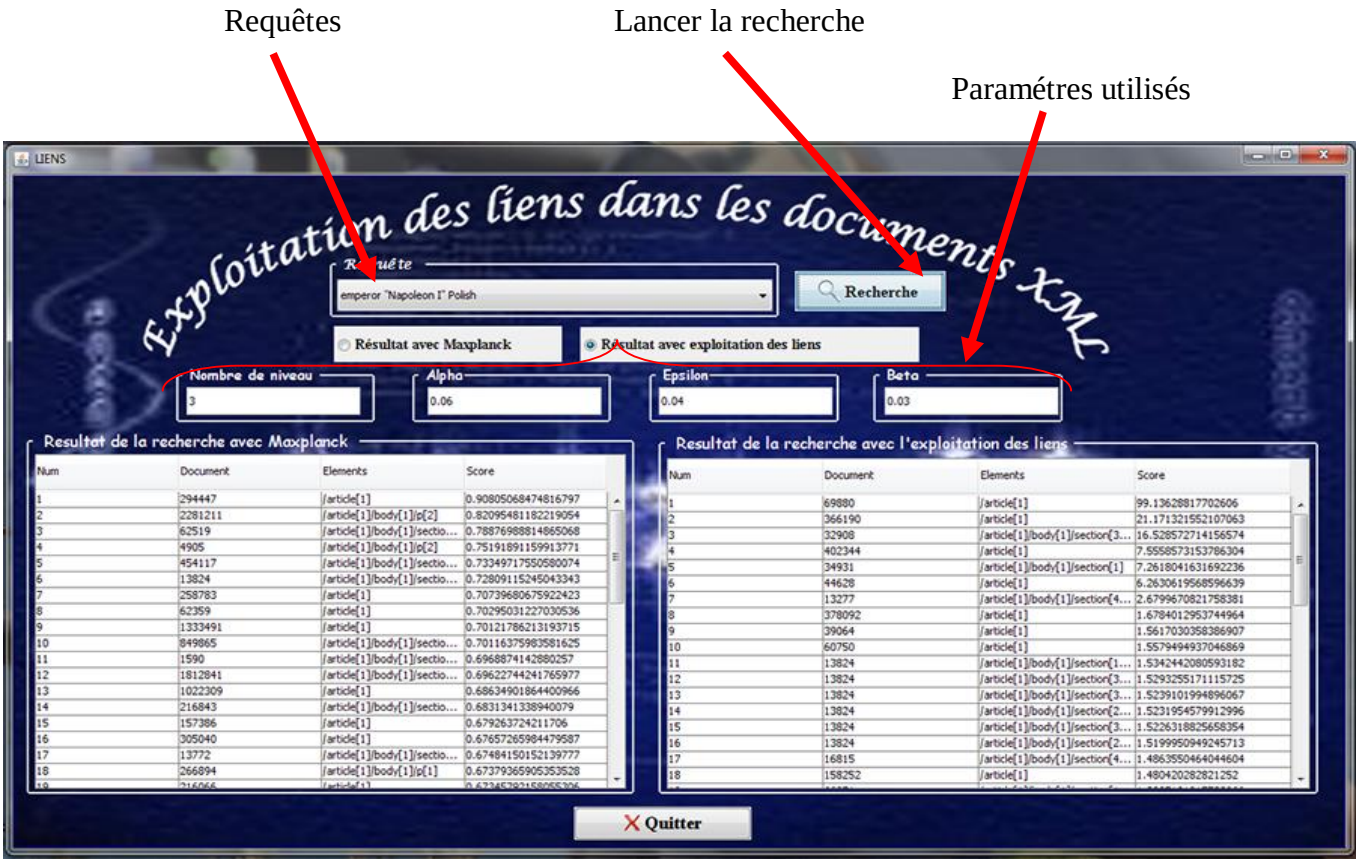


Figure 6.2 : Exemple de réordonnancement des documents pour la collection INEX 2006 requête 289.

La figure ci-dessus montre un résultat d'un réordonnancemet réalisé après l'application des différentes formules. Les résultats sont classés par ordre décroissant des valeurs des scores, ainsi certains éléments sont remontés dans le classement et d'autres on perdu de place dans le classement à l'image du document 13824 qui s'est classé 6^{ieme} dans la première recherche sans exploitation de liens et 11^{ieme} avec la deuxième recherche en exploitant les liens.

VI.5 Implémentation de l'approche :

Pour permettre une bonne mise en œuvre de notre modèle nous avons implémenté un ensemble de classes nous permettant au préalable d'indexer la collection ainsi que d'autres classes pour implémenter les différentes formules ainsi ces deux groupes de classes sont répartie selon deux niveaux :

Ø Au niveau de l'indexation :

Dans la phase de l'indexation, un ensemble de classes est mis en jeu, ces dernières nous permettent d'extraire à partir de chaque document les informations nécessaire à la mise en œuvre de notre modèle à savoir le titre et le texte ancre des liens, ces informations sont stockées dans une base de données.

Les classes utilisées sont les suivantes :

- La classe *MySaxParser* : qui est le parseur SAX qui permet d'extraire le titre des documents et le texte ancre des liens.
- La classe *TestMotVide* : qui permet d'extraire les mots, d'éliminer les mots vides en utilisant une StopList et de faire la troncature.
- La classe *Sauvegarde* : cette dernière classe permet de sauvegarder dans la base de données les différentes informations telles que le numéro du document, les mots indexés… etc

Les tables de base de donnée qu'on a crée au niveau de l'indexation sont :

- La table document : contient trois colonnes, numéro du document, titre du document (indexé), et le score du document qui est initialisé à 0.
- La table lien : contient les informations sur les liens de chaque document à savoir, numéro du document source du lien, le numéro du document cible et le texte ancre du lien (indexé).

Ø Au niveau de l'appariement :

Les classes utilisées au niveau de l'appariement consistent à exploiter l'information contenue dans les liens que nous avons préalablement indexés pour implémenter les différentes formules pour avoir un nouveau score pour chaque élément afin de réaliser l'ordonnancement.

Les classes crées à ce niveau sont :

- La classe *FenetrePrincipale* : qui permet de choisir la requête et les différents paramètres décrits dans le chapitre précédent et récupérer les résultats du système Maxplanck en réponse à cette requête.
- La classe *CalculeScore* : permet d'effectuer le calcul de score sur les éléments résultats en appliquant les formules décrites dans le chapitre précédent.
- La classe *CalculeScoreElt* : permet de calculer le score final de l'élément en appliquant la formule (4^{ieme} formule) décrite dans le chapitre précédent et insérer les résultats dans la base de données.

Les tables de base de données créée à ce niveau sont :

- La table *resultat* : contient les informations sur les résultats des requêtes obtenues en utilisant le système Maxplanck.
- La table *resulta_elt* : contient les informations sur les résultats des requêtes obtenues en utilisant notre méthode.
- La table *resultat_Inex* : contient les jugements de pertinence de la compagne d'évaluation INEX 2006 utilisés pour l'évaluer de nos résultats.
- La classe *topic* : contient le numéro et le titre de la topic.

VI.5.1 Notre travail par étape :

Pour permettre une bonne mise en œuvre de notre modèle de recherche, nous avons répartie notre travail en plusieurs phases, dans chacune d'elle un ensemble d'outils et de programmes est utilisé pour réaliser différentes tâches :

Ø Première phase : L'indexation est la première étape dans le processus de recherche, ainsi pour permettre l'application de nos formules deux éléments sont indexés : Le titre et les textes ancres des liens (voir les sections V.2.1.1 et V.2.1.2).

Ø Deuxième phase : La deuxième étape consiste à exploiter les éléments indexés pour appliquer les formules décrite dans la section V.3. Ces formules sont programmées et testées sur des petites collections de données et les résultats sont comparés avec ceux obtenus avec le system XFIRM.

Ø Troisième phase : La troisième partie consiste à réaliser des expérimentations sur la collection de test INEX 2006. Un ensemble de requêtes issu de la compagne d'évaluation INEX 2006 est exécuté tout en exploitant un ensemble initial de résultats obtenus par le SRIS Maxplanck.

Ø Quatrième phase : La quatrième partie consiste à faire une évaluation de notre modèle, ainsi deux mesures d'évaluation recommandées pour la compagne 2006 sont utilisées à savoir la précision interpolée calculée grâce à l'outil d'évaluation EvalJ et la mesure nxCG, ces deux mesures nous ont permis de réaliser une comparaison entre notre modèle et le système Maxplanck avec les jugements de pertinence d'INEX 2006 que nous avons présenté sous forme de graphes.

VI.6 Expérimentation :

VI.6.1 La collection INEX 2006:

Dans la compagne INEX 2006 la collection de test utilisée comporte plus de 650000 documents extraits des différents articles de l'encyclopédie libre Wikipédia, elle offre un ensemble de 126 requête pour réaliser l'ensemble des évaluations, les caractéristiques principales sont récapitulée dans la table suivante :

Taille de la collection	**4.6 GO**
Nombre de documents	**659388**
Nombre de liens	**16737300**
Nombre de Topics	**126**

Table 6.1 Caractéristique de la collection INEX 2006

VI.6.2 Protocole d'évaluation:

VI.6.2.1 Les principales tâches d'évaluation:

- La Tâche Ad-hoc : consiste à simuler l'utilisation d'une bibliothèque, cette bibliothèque contient un ensemble de documents sur lesquels les systèmes exécutent les requêtes utilisateurs.
- La tâche CO : Le but de cette tâche est de répondre à des requêtes utilisateur de type CO par des éléments/documents XML. Aucune indication sur la granularité de l'information de l'information à renvoyer n'est indiquée, la requête CO ne contient que des mots clés.
- La tâche CO+S : Cette tâche permet à un utilisateur d'améliorer ses résultats en ajoutant des contraintes structurelles pour sa requête.
- La tâche co.focussed : La tâche co.focussed évalue la capacité du système à retrouver le meilleur élément de réponse dans un chemin donné.

VI.6.2.2 Les mesure d'évaluation:

La définition des mesures d'évaluation dans le cadre de la recherche d'information dans les documents XML est un problème de recherche ouvert. Il fait partie des parties des problèmes traités dans le cadre des compagnes d'évaluation. Les chercheurs ont proposé plusieurs mesures. Pour chaque compagne (chaque année) des mesures officielles sont retenues pour l'évaluation. Les mesures dépendent aussi de la tâche considérée.

Les mesures d'évaluation utilisées dans INEX 2006 sont la précision moyenne interpolée (AiP[x]) et la moyenne des précisions moyennes interpolées (MAip[x]). Les mesures d'évaluation dans INEX 2006 sont basées sur les mesures xCG (le gain cumulé), ep (l'effort précision) et sur la quantité du texte souligné retournée. Les évaluateurs soulignent pour chaque requête les phrases représentant l'information pertinente dans chaque document. Le

texte retourné par un système est comparé au nombre et à la localisation du texte identifié comme pertinent pour la requête (souligné par les évaluateurs).

Formellement, soit *pr* la partie du document (où élément XML) rangée au rang *r* dans la liste des résultats *Lq*, retournée par un système de recherche pour un requête ; et soit *rsize(pr)* la longueur en nombre de caractères du texte souligné (pertinent) contenu par *pr* . Soit *size(pr)* la longueur totale du texte contenu par *pr* , et soit *Trel(q)* la quantité totale (nombre de caractères) du texte souligné (pertinent) pour la requête *q*. *Trel(q)* est calculé en considérant tout les documents ; i-e la somme des longueurs du texte souligné dans tous les documents pour la requête *q*.

La précision à un rang r est définie par :

$$P[r] = \frac{\sum_{i=1}^{r} rsize(pi)}{\sum_{i=1}^{r} size(pi)}$$

Le rappel à un rang r est définie par :

$$R[r] = \frac{\sum_{i=1}^{r} rsize(pi)}{Trel(p)}$$

La précision interpolée iP[x] est calculée comme suit :

$$iP[x] = \begin{cases} \max_{i \le r \le |lq|}(P[r] \wedge R[r] \geq x) \; si \; x \; \leq R[|Lq|] \\ 0 \; si \; x > \; R[|Lq|] \end{cases}$$

Où R[|Lq|] est le rappel pour tous les documents restitués. Par exemple, iP [0.01] calcule la précision interpolée à 1% du rappel pour une requête donnée.

En plus d'utiliser la précision interpolée à un niveau de rappel, un score de performance globale, basé sur la mesure précision moyenne interpolée AiP est utilisé. Pour une requête, AiP est calculé par la moyenne des précisions interpolée calculées à 101 niveaux de rappel standards.

$$AiP = \frac{1}{101} \sum_{x=0.00,0.01,\ldots,1.00} iP[x]$$

La performance en considérant un ensemble de requêtes est mesurée par le calcul de la moyenne des valeurs des AiP obtenue pour chaque requête.

En supposant qu'on a n requêtes, la moyenne des précisions moyennes interpolées est donnée par :

$$MAiP = \frac{1}{n}\sum_{t} AiP[t]$$

Ø La mesure nXCG :

Afin d'évaluer notre système, nous avons créée des classes permettant de calculer la valeur nxCG décrite dans le chapitre 2 (section II.7.4), comme suit :

$$nxCG(i) = \frac{xCG(i)}{xCI(i)}$$

Le principe consiste à calculer la valeur nxCG pour 4 rangs, ainsi *i* aura les valeurs (10, 20, 30, 50) qui représentent respectivement le nombre des premiers résultats considérés. Cette valeur est calculée pour les deux systèmes (notre modèle, Maxplanck) pour permettre de faire une comparaison.

xCG représente la somme des scores des *i* premiers résultats obtenu par le système considéré à savoir Maxplanck et notre modèle.

xCI représente la somme des *i* premiers scores affectés aux jugements de pertinence de INEX 2006.

VI.6.2.3 Requêtes et paramètres utilisés:

Ø Requêtes

Les requêtes que nous avons utilisé sont de type CO et sont issues de la compagne d'évaluation INEX 2006, Les requêtes utilisées sont les suivantes :

Numéro de la requête	Titre de la requête
289	emperor "Napoleon I" Polish
290	"genetic algorithm"
292	Italian Flemish painting Renaissance -French -German
294	user interface design usability guidelines
300	Airbus A308 ordered
304	allergy treatment
308	wedding traditions and customs
314	"food additive toxin carcinogen "E number""
315	spider hunting insect
318	"the atlantic ocean islands and the slave trade"

Table 6.2 Requêtes de la compagne INEX 2006 testées.

Ø Paramètres :

Nous n'avons pas utilisés plusieurs variations des valeurs des paramètres α, β, ε, pour les tests puisque la collection est gigantesque et que chaque exécution prend beaucoup de temps, nous avons donc fixé ces valeurs comme suit:

- Ø $\alpha=0.06$
- Ø $\beta=0.03$
- Ø $\varepsilon= 0.04$
- Ø Nombres de niveau = 3

Les résultats de recherche de maxplanxk que nous avons considéres sont ceux obtenus lors de la compagne d'Inex 2006 pour la tâche focused.

VI.6.2.4 Outil d'évaluation:

Pour permettre l'application des différentes mesures d'évaluation recommandées pour la collection de test 2006 nous avons utilisé EvalJ [**Piwowarski, 2005]** qui est une application entièrement écrite en java et exploitable sous Windows ou bien sous Linux via les lignes de commandes ou à partir d'un IDE (ex : Eclipse). Evalj est paramétrée grâce à un fichier de configuration contenant les différentes informations nécessaires pour réaliser l'évaluation (Type de tâches, les métriques, les différents chemins….etc.). Pour lancer l'évaluation il faut exécuter la commande suivante à partir de terminal sous linux :

```
java -Dorg.xml.sax.driver=gnu.xml.aelfred2.XmlReader -jar jars/EvalJ.jar [-e] [-q] [-jfree] –config configfile.prop
```

Les paramètres :

`-q :` Pour sauvegarder les résultats d'évaluation de chaque requêtes.

`-jfree :` Pour réaliser les évaluations sous forme de graphes.

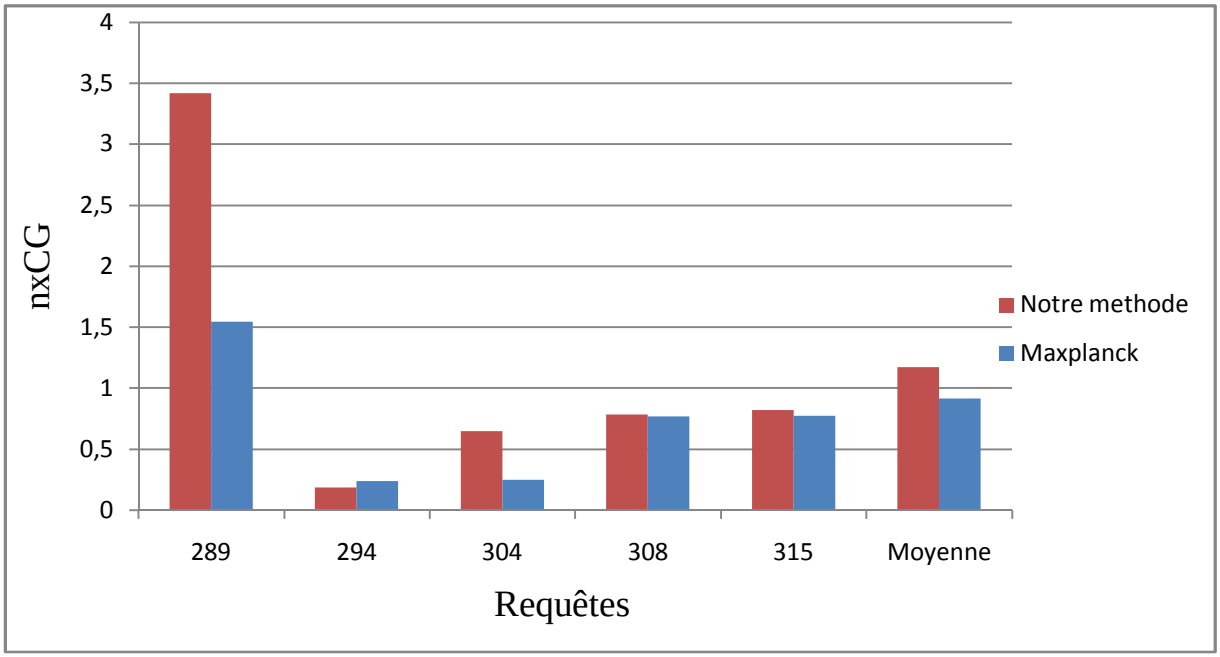
`configfile.prop :` fichier contenant les paramètres de configuration.

VI.6.3 Résultats de l'évaluation:

Dans ce qui suit nous présentons les résultats des expérimentations obtenues sur le modèle étudié et le système Maxplanck pour la collection de test INEX 2006 en utilisant l'outil EvalJ et en calculant le gain cumulé normalisé (nxCG[i]).

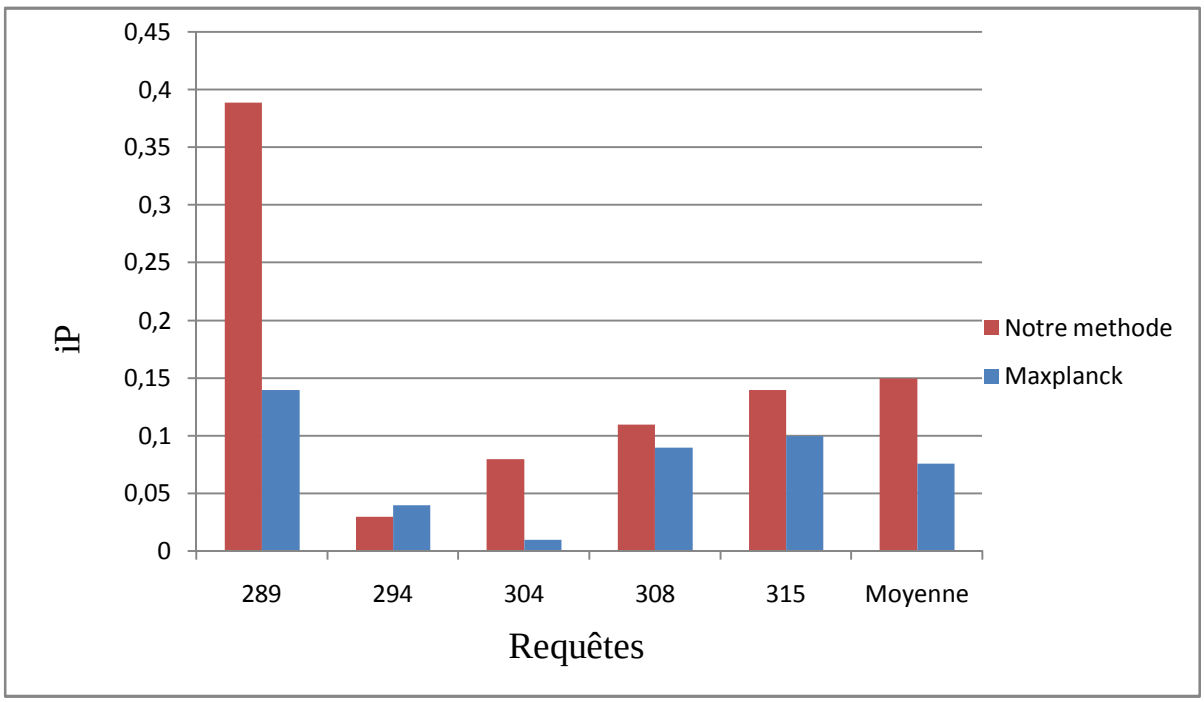
VI.6.3.1 Résultats de l'évaluation en calculant le nxCG:

Le graphe suivant montre les résultats obtenus en utilisant les mesures d'évaluation nxCG



VI.6.3.2 Résultats de l'évaluation en utilisant EvalJ:

En appliquant les différentes formules d'évaluation grâce à l'outil d'évaluation EvalJ sur les résultats du modèle que nous avons implémenté et celle du SRIS Maxplanck pour quelques requêtes issue de la compagne INEX 2006 nous avons réalisé une comparaison entre les deux systèmes, cette comparaison est illustrée par le graphe suivant :



Les deux graphes montrent une nette amélioration au niveau de la précision de notre modèle qui exploite les liens par rapport au modèle Maxplanck qui ne prend pas en charge les liens pour chaque requête exécutée. Nous pouvons conclure que l'exploitation des liens a apporté un gain significatif au niveau de la précision et que la prise en compte du titre des documents, du texte ancre des liens et la propagation des scores améliorent de manière significative les résultats de la recherche, leur prise en compte s'avère donc importante pour réaliser un système de recherche fiable répondant aux besoins et attentes des utilisateurs.

VI.7 Conclusion :

Nous avons présenté dans ce chapitre l'implémentation et l'évaluation de notre approche sur la collection INEX 2006 en comparaison avec un système sans prise en compte de liens qui est Maxplanck, ces expérimentations nous ont permis de situer notre modèle devant un système qui a participé dans la compagne d'évaluation.

Conclusion générale

Dans ce mémoire de mastère, nous avons abordé le problème lié à l'exploitation des liens dans la recherche d'information dans des fichiers structurés (XML).

En premier lieu nous avons donné un état de l'art sur la recherche d'information de façon générale, ensuite on s'est focalisé sur la recherche d'information structurée.

Nous avons détaillé par la suite les concepts généraux relatifs aux liens dans les documents XML, puis nous avons listé les différents algorithmes exploitant ces liens pour enfin donner une description globale du modèle étudié qui est proposé par Mme Fellag.

Et comme l'objectif de notre projet est de faire une étude comparative entre un modèle de recherche classique et un modèle de recherche avec prise en compte de liens nous avons fait une évaluation du modèle avec la collection de test INEX 2006.

Le modèle implémenté exploite le contenu informationnel portée sur le titre et le texte ancre de liens, deux paramètres important pour décrire le contenu général du document courant et les documents référencés, ces deux paramètres sont utilisés dans les fonctions de correspondance pour le calcul des scores. Initialement notre algorithme commence par exploiter un ensemble d'éléments retourné par un SRI classique, cette phase nous a permi d'extraire, pour chaque élément, l'ensemble de liens contenus dans ce dernier, en seconde partie un calcul de score pour chaque documents référencé dans la liste des résultats est utilisé en fonction du nombre de documents qui l'ont référencé et leurs valeur de pertinences, ainsi au bout de K niveaux nous avons obtenu un ensemble de documents ayant chacun son propre score, ces derniers seront combinés avec les scores initiaux des éléments pour réaliser un ré-ordonnancement finale des résultats.

Une première évaluation de notre approche consistait à fixer les paramètres (α, β, ε) nous permettant d'avoir des résultats optimaux.

Cette approche a montré une certaine amélioration au niveau de précision des résultats par rapport un SRI classique sans prise en compte de liens.

Perspectives :

Suite à notre travail quelques perspectives sont envisagées :

- La première perspective consiste à intégrer la sémantique dans le calcul des scores des textes ancres des liens et les titres des documents.
- La deuxième perspective consiste à améliorer les temps d'exécution des requêtes pour ainsi réduire l'impact du nombre immense des liens contenus dans la collection dans le calcul des scores.
- La troisième perspective consiste à maximiser le nombre de documents/éléments considérés dans la première recherche pour arriver à considérer tout les résultats.
- Faire varier les valeurs des paramètres (α, β, ε) pour optimiser les résultats.

Annexe A

Les Technologies de la famille XML

Le succès de XML et sa vulgarisation en tant que technologie phare dans divers domaines d'application, ne sont pas seulement dus à sa simplicité, sa flexibilité et sa portabilité mais aussi aux nombreuses technologies construites autour de lui comme le montre la figure A.1. Dans cette annexe, nous présentons certaines de ces technologies.

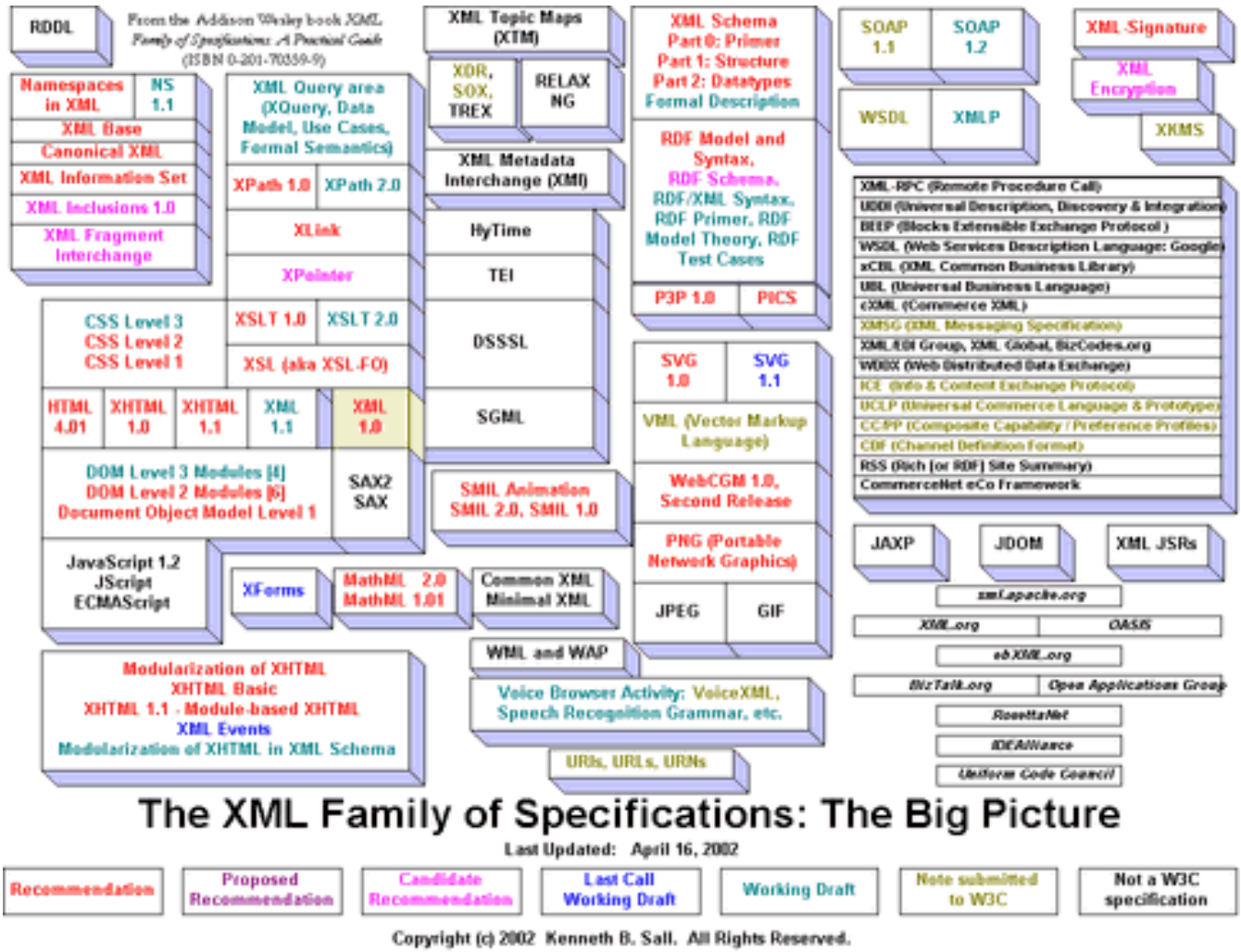


Figure A.1 : Les technologies de la famille XML **[Sall, 2002]**

A.2. Les Espaces de Noms (Namespaces)

Les espaces de nom[1] XML sont spécifiés dans une recommandation du W3C. Ils permettent de distinguer de manière unique des éléments et des attributs portant le même nom lorsqu'ils proviennent d'applications XML différentes.

Les espaces de noms sont associés aux éléments grâce à l'attribut « *xmlns* » (*XML NameSpace*) qui aura pour valeur une URI (*Uniform Resource Identifier*) qui identifiera de manière unique l'espace de noms. Cet attribut peut également avoir la forme suivante : « *xmlns:préfixe* ».

Exemple :

Un document qui contient les informations générales d'une entreprise :

```
<entreprise>
        <nom>  Paradise Soft </nom>
          <adresse>
          ...
        </adresse>
</entreprise>
```

Un document qui contient les informations sur le personnel de l'entreprise :

```
<personne>
    <Identification>
        <nom>Doe</nom>
        <prenom>Johnl</preno
    </Identification>
    <fonction>PDG</fonction>
</personne>
```

La balise *nom* propose des caractéristiques différentes selon qu'elle soit utilisé dans le document entreprise ou le document personne. Si l'on désire créer un document unique décrivant l'entreprise et ses employés, un moyen permettant d'identifier les balises est nécessaire, c'est l'objectif des espaces de noms.

Le document XML unique décrivant l'organisation de l'entreprise est alors :

[1] http://www.w3/org/TR/Rec-xml-names.

```
organisation   xmlns:entreprise='http://www.entreprise.org'
                xmlns:pers ='http://www.personne.org'>

   <entreprise:nom>Paradise Soft</entreprise:nom>
   <entreprise:adresse>.... </entreprise:adresse>
   <pers:personne>
    <pers:Identification>
      <pers:nom>Doe</pers:nom>
      <pers:prenom>john</pers:prenom>
    </pers:Identification>
    <pers:fonction>PDG</pers:fonction>
  <pers:personne>

</organisation>
```

A.3 XML Schema

XML Schema[2] est une recommandation du W3C depuis 2001 qui a pour but de remplacer les DTD (Document Type Definition) existantes. Comme nous l'avons vu dans le chapitre II, la DTD d'un document XML contient des informations de structure et de typage des données du document XML. XML Schema présente de nombreuses améliorations par rapport aux DTD, notamment une plus grande flexibilité et un typage plus important des données (string, décimal, boolean,…).

Exemple :

Soit le fragment de DTD suivant :

< !ATTlST livre prix CDATA #IMPLIED>

Cette déclaration signifie que l'attribut prix peut prendre n'importe quelle chaîne de caractère comme valeur. Les codes suivants seraient valides :

<livre prix = 'rien'>, <livre prix = '22'>, <livre prix = '.'>

En utilisant XML schema, on peut à la différence, définir la valeur de prix comme devant être une valeur décimal comme suit : *<attribute name = « prix » type= « decimal »/>* ainsi seules les valeurs décimales pour l'attribut prix seront acceptées.

[2] http://www.w3.org/XML/Schema

A.4 XSL (Extensible Stylesheet Language)

Le XSL[3] pour ***eXtensible Stylesheet Language*** ou « *langage extensible de feuilles de style* » est une recommandation du W3C datant de novembre 1999. C'est donc un standard dans le domaine de la publication sur le Web. Le XSL est en quelque sorte le langage de feuille de style du XML. Un fichier de feuilles de style reprend des données XML et produit la présentation ou l'affichage de ce contenu XML selon les souhaits du créateur de la page.

Il existe en fait deux langages sous l'ombre d'XSL :

Ø Le XSL**T** (***Extensible Style Scheet Language Transformation***) qui est un langage de transformation de documents XML en d'autres formats ; en Html par exemple.

Ø XSL-FO (*XSL **F**ormatting **O**bjects*) qui gère la pagination d'un document, les notes de bas de pages, les marges, le positionnement des objets sur la page, les polices de caractères, l'affichage de tableaux, …etc.

A.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.

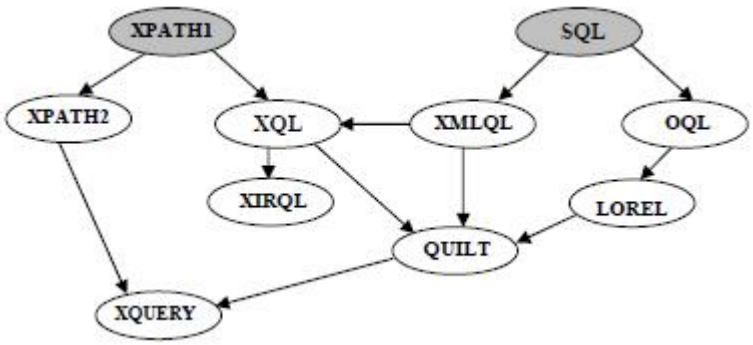


Figure A.2 : langages d'interrogation XML

A.5.1 XPATH (XML Path Language)

XPATH est un langage permettant d'adresser des parties ou des sous arbres d'un document XML, afin de pouvoir en extraire avec précision les informations requises. XPath 1.0[4] est

[3] http://www.w3.org/TR/xsl

[4] http://www.w3.org/TR/xpath/, http://www.w3.org/TR/xpath20/

devenu une recommandation du W3C depuis le 16 novembre 1999. L'adressage des documents est réalisé en spécifiant *des chemins de localisation.* Le chemin de localisation de XPath est constitué d'une séquence d'étapes de localisation, séparés par le caractère / (similaire au chemin de répertoires du DOS). Un chemin de localisation spécifie une traversée de l'arbre du document depuis un nœud de départ vers un ensemble de nœuds cibles.

Un chemin est :

Soit ***Absolu :*** commençant à la racine : */niveau1/…/niveauN*
Soit ***relatif :*** commençant à un nœud courant : *niveau1/…/niveauN*

La syntaxe des chemins de localisation fait fréquemment référence à des *axes*, ces derniers peuvent être utilisés dans des expressions Xpath en respectant la syntaxe suivante :

Nom axe :: sélecteur[predicat]

Le tableau suivant illustre quelques noms d'axes utilisés par Xpath :

Nom d'axe	Description
Child	enfant du nœud contextuel.
Self	nœud contextuel lui-même.
Parent	nœud parent.
Ancestor	tous les ancêtres entre le nœud racine et le nœud parent.
Descendant	tous les descendants du nœud contextuel

Tableau A.1: description des noms d'axes utilisés par Xpath

Le sélecteur est composé du nom de nœud sélectionné (élément ou @attribut)

Le Prédicat est une expression booléenne mise entre crochets permettant de sélectionner un ou plusieurs nœuds en fonction du critère de sélection utilisé.

[Fonction(noeud) = valeur]

Xpath utilise un certain nombre de fonctions dans ses expressions qui servent à renvoyer des nœuds, des ensembles de nœuds, de travailler avec des chaînes, des nombres ou des booléens.

Parmi les fonctions de Xpath on peut citer :

Fonction	*Description*
Name ()	permet de renvoyer le nom d'un nœud.
Text ()	renvoie de contenu PCDATA d'un nœud.
Position ()	permet d'obtenir la position d'un nœud dans un ensemble de nœuds, selon l'ordre du document.
Contains ()	indique si une chaîne contient une autre chaîne.
Boolean ()	évalue une expression Xpath selon qu'elle soit true ou false.
Sum ()	permet d'additionner les valeurs numériques d'un ensemble de nœuds.

Tableau A.2: description des fonctions utilisées par Xpath

A.5.2 XQuery (eXtensible Query)

XQuery[5] (pour *XML Query Langage*) est une recommandation du W3C, qui définit un langage de requête permettant non seulement d'extraire des informations d'un document XML, ou d'une collection de documents XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML.

Côté syntaxe, XQuery offre deux manières pour formuler les requêtes. Une syntaxe « *naturelle* » non XML, dite aussi *FLWOR*, dont le nom vient des cinq clauses principales qui la composent (*for*, *let*, *where*, *order by* et *return*). Et la syntaxe *XQueryX* dans laquelle une requête est un document XML. Généralement.

Exemple :

Soit à *« trouver le metteur en scène du film Titanic »,* la requête sera :

```
xquery version "1.0";
<FILMS>
{
for $b in (/FILMS/FILM)
where $b/@titre = 'titanic'
return
 <met_sc>{ $b/MES }</meet_sc>
}
</FILMS>
```

A.5 Les vocabulaires métier

Autour d'XML, il existe aussi un certain nombre de vocabulaires métier (pour lesquels la DTD est fixée) proposés par des groupes de travail spécialisés.Parmi eux on peut citer :

- **XHTML** : langage permettant de reconcevoir les documents HTML sous une forme XML stricte.

- **MathML** (***Mathematical Markup Language***) : langage de notation mathématique sur le web

- **CML** (***Chemical Markup Language***), pour la publication Internet des formules chimiques, de molécules, des équations.

- **VML** (***Vector Markup Language***) : langage de balisage d'information graphique vectorielle.

[5] http://www.w3.org/TR/xquery/

- **PGML** (***P**recision **G**raphics **M**arkup **L**anguage*), qui décrit des structures de données graphiques complexes avec les primitives du langage Postscript. Il permet la conversion de documents aux formats *ps* et *pdf* en XML.

- **SMIL** (***S**ynchronized **M**ultimedia **I**ntegration **L**anguage*), pour la création multimédia. Il spécifie comment et quand des éléments multimédia peuvent apparaître dans une page web.

- **WML** (*Wireless **M**arkup **L**anguage*) : langage de balisage pour l'internet mobile.

Annexe B

Le Systéme de recherche XFIRM

B.1 Introduction :

Le modèle XFIRM (XML Flexible Information Retrieval Model) est un modèle de RI orienté pertinence, basé sur une méthode de propagation de la pertinence. Il repose sur un modèle de représentation des documents permettant de conserver à la fois toute l'information structurelle et toute l'information textuelle des documents. Il a été développé par ***Karen Pinel-Sauvagnat*** de l'université de Paul Sabatier de Toulouse en 2005 dans le cadre de la thèse de doctorat.

Le code source du « cœur » du moteur de recherche XFIRM (xfirm Core) est entièrement écrit en java et compilé avec le JDK 1.6 fonctionnant dans un environnement de travail Linux.

Dans ce qui suit nous allons voire les différents répertoires nécessaires à l'installation, la configuration et la mise en œuvre du moteur de recherche, les différents packages qui composent le cœur du moteur et le modèle de stockage dans la base de données ORACLE ainsi les classes de lancement de l'indexation et de recherche.

B.2 Les répertoires du cœur XFIRM :

B.2.1 : Le répertoire Config :

Le répertoire **config** contient les fichiers nécessaires à la configuration du moteur, ainsi dans le sous-répertoire « connect » on trouve le fichier OracleConfig qui est un fichier texte contenant les paramètres de configuration pour établir une connexion à la base de données Oracle comme on trouve dans le sous-répertoire « dict » des fichiers permettant d'établir une équivalence entre balise.

```
jdbc:oracle:thin:@localhost:1521:database_name
username
password
```

Figure B.1 : Exemple d'un fichier connexion

B.2.2 : Le répertoire Doc :

Le répertoire **Doc** contient la documentation relative au cœur du moteur qui consiste en un ensemble de page Html généré au moment de compilation en exécutant le script compile.sh dans le répertoire script.

B.2.3 : Le répertoire Examples :

Le répertoire **examples** contient des exemples de shells de lancement pour l'indexation ou la recherche, comme il contient des fichiers de requêtes (.co et .cos) pour les requêtes de type (co et cos) respectivement ; ces fichiers se trouvent dans le sous-répertoire « queries » et des fichiers de résultats (co**i**.res2 et cos**i**.res2) qui sont les résultats d'une requête numéro i et de type (co et cos) respectivement ; ces fichiers se trouvent dans le sous-répertoire (results).

B.2.4 : Le répertoire proceduresPlsql :

Le répertoire **proceduresPlsql** contient un ensempble de procédures PL/SQL, ces dernières sont exécutées en ligne de commande sous oracle et permettent de préfixer les tables avec le nom de la collection.

```
SQL>@Nom_de_la_procédure
```

Figure B.2 :Exemple de compilation d'une procédure PL/SQL

Ces procédures sont les suivantes :

- **InsertionCollection.sql:** permet d'insérer les éléments de la table COLLECTION
- **InsertionDocument.sql** : permet d'insérer les éléments de la table DOCUMENT
- **InsertionImages.sql** : permet d'insérer les éléments de la table IMAGES
- **InsertionPath.sql** : permet d'insérer les éléments sauf les blobs chemin et nœuds_fils_textuels de la table PATH
- **InsertionTermTemp.sql** : permet d'insérer les éléments de la table temporaire TERMTEMP
- **InsertionTag.sql** : permet d'insérer les éléments de la table TAG
- **InsertionLink.sql** : permet d'insérer les éléments de la table LINKS
- **InsertionAttributType.sql** : permet d'insérer les éléments de la table ATTRIBUTSTYPE
- **InsertionAttributs.sql** : permet d'insérer les éléments de la table ATTRIBUTS

B.2.5 : Le répertoire script :

Le répertoire **script** contient trois Shell : **compile.sh** permettant la compilation de toutes les classes et génération de la documentation, **restaurationBase.sh** qui permet de restaurer une base de données et **sauvegardeBase.sh** qui permet de sauvegarder une base de données.

B.2.6: Le répertoire utils :

Ce répertoire contient des exécutable javacc pour compilation des grammaires (fichiers *.jj). Ces grammaires au nombre de trois sont :

- ***StandarTokenizer.jj*** : Définit les unités d'indexation c-à-d les termes.
- ***QueryParser.jj* :** Définit la grammaire des requêtes orientées contenu.
- ***structureQueryparser.jj* :** Définit la grammaire des requêtes orientées contenu et structure.

La compilation de ces grammaires génère des fichier.java ainsi la compilation des trois grammaires citées en haut génère sept fichiers.java qui sont (*CharStream.java*, *Token.java*, *TokenMgrError.java*, *StandardTokenizer.java*, *StandardTokenizerConstants.java*, *StandardTokenizerTokenManager.java*, et *ParseException.java*. Ces fichiers doivent ensuite être déplacés dans le répertoire qui contient la grammaire.

B.2.7 : Le répertoire xfirm :

Le répertoire *xfirm* contient les packages suivant :

- **analysis** : ce package contient toutes les classes qui vont permettre de parcourir les documents, et d'extraire les unités d'indexation ainsi que les attributs associés. Il est principalement utilisé lors de l'indexation.

- **analysis.standard** : ce package propose une implémentation standard du package analysis. La grammaire *StandardTokenizer.jj* définit ce qu'est une unité d'indexation.

- **document** : ce package contient la définition des principaux objets composant un document XML (*Document*, *Node*, …). Il est principalement utilisé lors de l'indexation

- **index** : les objets de ce package sont en fait des éléments qui vont aller dans les index (c'est-à-dire les tables de la base de données Oracle). Les instanciations de la classe IndexWriter permettent en outre de construire ces index.

- **inex** : Ce package contient les exécutables spécifiques à INEX.

- **queryParser** : ce package contient la grammaire des requêtes orientées contenu (fichier *QueryParser.jj*).

- **search** : ce package fournit des classes permettant de traiter des requêtes orientées contenu (composées de simples mots-clés).

- **store** : les classes de ce package permettent de créer ou lire les index (*BaseWriter* et *BaseReader*), grâce à une liaison directe avec la base de données Oracle.

- **structureQueryParser** : ce package contient la grammaire des requêtes orientées contenu et structure (fichier *StructureQueryparser.jj*).

Ø **structureSearch** : Ce package fournit des classes permettant de traiter des requêtes orientées contenu et structure (mots-clés+ conditions de structure).

Ø **test** : Ce package contient des classes de test (TestIndexer) pour l'indexation et (TestQuery) pour l'interrogation.

Ø **tree** : Ce package permet de charger les documents indexés en mémoire et de reconstruire leur structure arborescente. Il est principalement utilisé pendant la recherche.

Ø **util** : Ce package contient des classes "outils" qui peuvent servir dans les autres packages.

B.3 La base Oracle :

Les index sont stockés sous une base de données Oracle dont nous voici la liste des différentes tables :

Ø **La table Document :** contient les informations sur les documents de la collection :

Champs	Type	Clé
doc_id	NUMBER(38)	primary key
document	nvarchar2(300)	index
term_nb	NUMBER(38)	
deb	NUMBER(38)	
fin	NUMBER(38)	
nbr_niv	NUMBER(38)	
kuid	NUMBER(38)	

Ø **La table Tag :** contient des informations sur les balises de la collection

Champs	Type	Clé
tag_id	NUMBER(38)	primary key
tag	NVARCHAR2(100)	index

Ø **La table Path :**

Champs	Type	Clé
path_id	NUMBER(38)	primary key
doc_id	NUMBER(38)	foreign key references Document(doc_id), index
tag_id	NUMBER(38)	foreign key references Tag(tag_id)
ordre	NUMBER(38)	
nbrDirectFils	NUMBER(38)	
path_uid	NVARCHAR2(150)	index
nbrTermUniq	NUMBER(38)	
nbrTermsTot	NUMBER(38)	
nbrFilsText	NUMBER(38)	

Ø **La table Termes :** contient des informations sur les termes de la collection

Champs	Type	Clé
term_id	NUMBER(38),	primary key
term	NVARCHAR2(150),	ndex
freqColl	NUMBER(38),	
nbrDoc	NUMBER(38),	
nbrNoeuds	NUMBER(38),	
poids	blob	

Ø **La table AttributsType :** contient des informations sur les noms d'attributs présents dans la collection

Champs	Type	Clé
type_att_id	NUMBER(38)	primary key
att_name	NVARCHAR2(100)	index

Ø **La table Attributs :** permet de récupérer la valeur des attributs

Champs	Type	Clé
node_id	NUMBER(38)	foreign key references Path(path_id), primary Key
type_att_id	NUMBER(38)	foreign key references Path(path_id), primary Key
att_val	NVARCHAR2(100)	

Ø **La table Dict :** permet d'établir des équivalences de balises dans la collection

Champs	Type	Clé
tag_id	NUMBER(38)	primary key, foreign key references Tag(tag_id)
list_tag_id	blob	

Ø **La table Collection:** contient des informations générales sur la collection

Champs	Type	Clé
nbrDoc	NUMBER(38)	
nbrNods	NUMBER(38)	
nbrNodText	NUMBER(38)	

- **La table Images :** contient des informations sur les images de la collection

Champs	Type	Clé
nodeIm_id	NUMBER(38)	references Path(path_id)
trmIm	NVARCHAR2(100)	index

- **La table Links :** ***contient*** des informations sur les liens de la collection

Champs	Type	Clé
path_id	NUMBER(38)	references Path(path_id)
doc_name	NVARCHAR2(100)	

- **La table NodeToTerm :** permet de connaître les termes présents dans un nœud donné

Champs	Type	Clé
node_id	NUMBER(38)	primary key
nbrTrmUq	NUMBER(38)	
nbrTrmTot	NUMBER(38)	
listTerm	blob	

- **La table termTemp :** table temporaire utilisée afin de remplir la table Termes. Elle est supprimée à la fin de l'indexation

Champs	Type	Clé
term_id	NUMBER(38)	
term	NVARCHAR2(100)	
doc_id	NUMBER(38)	
node_id	NUMBER(38)	
int	NUMBER(38)	

B.3.1 Création des tables :

Avant tout lancement de l'indexation d'une collection, il faut Créer les tables servant d'index dans la base de données. La classe ***createTables*** permet de supprimer les anciennes tables de la base de données si elles existent et de récréer des nouvelles tables propres à la collection. Ces tables seront préfixées par nomCollection (I3E par exemple). Ainsi nous voici un exemple de lancement de main *createTables.*

```
>java xfirm/test/createTables config/connect/OracleConfig.txt I3E
```

B.4 Lancement de l'indexation :

La classe *TestIndexer.java* du répertoire *xfirm/test/* nous permet de lancer l'indexation, ainsi dans le terminal il faut taper la commande suivante :

```
java  xfirm/test/TestIndexer  p1 p2 p3 p4 p5 p6 p7
```

La ligne est composée du répertoire contenant la classe chargée de faire l'indexation (**`xfirm/test/TestIndexer`**) et d'un ensemble de paramètres (p1……p7) qui sont :

- p1: chemin de la collection à indexer
- p2 : fichier contenant les paramètres de connexion à la base
- p3 : nom de la collection utilisé pour la création des tables dans la base Oracle (ex :3E)
- p4 : le chemin de fichier dictionnaire qui est normalement localisé sous config/dict/
- p5 : ATTO pour faire l'indexation des attributs, ATTN pour ne pas faire
- p6 : NTO pour faire l'indexation NodeToTerm, NTN pour ne pas faire. Cette indexation est nécessaire si on veut faire de la reformulation de requêtes.
- p7 : -n ou –u : -n pour créer une nouvelle base et –u pour mettre a jour une base qui existe déjà.

B.5 Lancement de la recherche :

La classe *TestQuery.java* du répertoire *xfirm/test/* nous permet d'exécuter les requêtes utilisateurs, ainsi pour lancer la recherche des résultats on doit exécuter la commande suivante dans le terminal :

```
> java xfirm/test/TestQuery p1 p2 p3 p4 p5 p6 p7 [ p8 p9 p10 | p8 p9]
```

La ligne est composée du répertoire contenant la classe chargée de faire l'indexation (**`xfirm/test/TestQuery`**) et d'un ensemble de paramètres (p1……p10) qui sont :

- p1: fichier contenant les paramètres de connexion à la base
- p2 : requête, entre "" (CO ou COS)
- p3 : nom de la collection utilisé pour la création des tables dans la base Oracle
- p4 : paramètre max :
  - dans le cas de requête CO, nombre maximum de documents utilisés pour récolter les nœuds feuilles (250 ou 500);
  - dans le cas de requête COS, nombre maximum d'éléments de structure à rechercher par sous-requête
- p5 : formule de calcul du poids des nœuds feuilles:

- ü 1=tf.idf ;
- ü 2= tf.ief;
- ü 3=tf.idf.ief

Ø p6 : type de propagation utilisée
- ü 1 : propagation avec overlap ;
- ü 2: propagation sans overlap

Ø p7: type des requêtes
- ü CO
- ü COS

Si le type des requêtes est CO

Ø p8 : valeur du paramètre alpha (entre 0 et 1)
Ø p9 : valeur du paramètre seuil (nombre de termes minimum dans les resultats)
Ø p10 : valeur du paramètre pivot (entre 0 et 1)

Si le type des requêtes est COS

Ø p8 : valeur du paramètre alpha (entre 0 et 1)
Ø p9 : valeur du paramètre strict :
- ü 1 : indique que les conditions de structure sur les éléments support de la requête doivent être traitées de manière stricte (c'est-à-dire qu'un élément cible n'est pas renvoyé si les conditions sur les éléments supports ancêtres ou descendants ne sont pas respectées)
- ü 0 : indique que les conditions de structure sur les éléments support de la requête doivent être traitées de manière vague

B.6 Recherche dans le cadre de la compagne d'évaluation INEX :

B.6.1 Lancement de l'exécution du run :

La classe *TestInexRun.java* du répertoire *xfirm/inex/TestInexRun* nous permet de lancer la recherche dans le cadre de compagne d'évaluation INEX, ainsi pour lancer la recherche des résultats on doit exécuter la commande suivante dans le terminal :

```
>java xfirm/inex/TestInexRun p1 p2 p3 p4 p4 p5 p6 p7 p8
```

La ligne est composée du répertoire contenant la classe chargée de faire l'indexation (**`xfirm/test/TestInexRun`**) et d'un ensemble de paramètres (p1……p8) qui sont :

Ø p1 : fichier contenant les paramètres de connexion à la base
Ø p2 : nom du fichier contenant les requêtes (la premier colonne du fichier indique le numéro de la requête, séparé de la requête par une virgule) par exemple :

```
134, problems integration +XML technologies +"relational databases" solutions
```

- p3 : chemin jusqu'aux fichiers résultats + premières lettres du fichier résultat
- p4 : nom de la collection utilisé pour la création des tables dans la base Oracle

- p5 : paramètre max :
  - dans le cas de requête CO, nombre maximum de documents utilisés pour récolter les nœuds feuilles (250 ou 500);
  - dans le cas de requête COS, nombre maximum d'éléments de structure à rechercher par sous-requête
- p6 : formule de calcul du poids des nœuds feuilles:
  - 1=tf.idf ;
  - 2= tf.ief;
  - 3=tf.idf.ief
- p7 : type de propagation utilisée
  - 1 : propagation avec overlap ;
  - 2: propagation sans overlap
- p8: type des requêtes
  - CO
  - COS

Si le type des requêtes est CO

- p9 : valeur du paramètre alpha (entre 0 et 1)
- p10 : valeur du paramètre seuil (nombre de termes minimum dans les resultats)
- p11 : valeur du paramètre pivot (entre 0 et 1)

Si le type des requêtes est COS

- p9 : valeur du paramètre alpha (entre 0 et 1)
- p10 : valeur du paramètre strict :

  - 1 : indique que les conditions de structure sur les éléments support de la requête doivent être traitées de manière stricte (c'est-à-dire qu'un élément cible n'est pas renvoyé si les conditions sur les éléments supports ancêtres ou descendants ne sont pas respectées).
  - 0 : indique que les conditions de structure sur les éléments support de la requête doivent être traitées de manière vague.

B.6.2 Lancement de formatage du run :

La classe *TestInexFormat*.java du répertoire */inex/TestInexFormat* nous permet de formater les résultats retournée lors de l'exécution du run, ainsi pour lancer le formatage des résultats on doit exécuter la commande suivante dans le terminal :

```
> xfirm/inex/TestInexFormat p1 p2 p3 p4 p5 p6 p7
```

La ligne est composée du répertoire contenant la classe chargée de faire l'indexation (**`xfirm/test/TestInexFormat`**) et d'un ensemble de paramètres (p1……p7) qui sont :

- p1 : type de la requête: CO / CAS / COS / MM / MMFF
- p2 : année de la champagne: 2003 / 2004 / 2005 / 2006 / 2007 / 2008
- p3 : chemin jusqu'aux fichiers résultats + premières lettres des fichiers résultat
- p4 : nom de la collection utilisé pour la création des tables dans la base Oracle
- p5 : fichier contenant les paramètres de connexion à la base
- p6 : rsv : 1 si affichage du rsv, 0 si non
- p7 : rank : 1 si affichage du rsv, 0 si non

[Allorge, 2000] Stéphane Allorge, *" XML"*, département ASI, 2000.

[BADR, 2007] Georges BADR. "Recherche d'Information sur le Web: Impact de la structure des documents sur la pertinence des résultats". Maitrise, IRIT, Université Paul Sabatier & Institut National Polytechnique de Toulouse. 2007.

[Boubekeur, 2008] Boubekeur-Amirouche F. Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets. Thèse de doctorat. Toulouse: Université Paul Sabatier; 2008.

[Brin, 1998] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems. 1998:107-117.

[Carriere, 1997] J. Carriere and R. Kazman. Webquery : Searching and visualizing the web through connectivity. In Proceedings of the Sixth International Conference on the World Wide Web, Santa Clara CA, 1997.

[Chebolu, 2008] Chebolu P, Melsted P. PageRank and the random surfer model. In: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2008. p. 1010-1018.

[Chibane, 2008] : Idir CHIBANE. "Impact des liens hypertextes sur la précision en recherche d'information.". Thèse de doctorat. L'UNIVERSITÉ PARIS XI ORSAY. 2008.

[Cohn, 2000] Cohn D, Chang H. Learning to Probabilistically Identify Authoritative Documents. In: Proceedings of the Seventeenth International Conference on Machine Learning. San Francisco, CA: Morgan Kaufmann Publishers Inc.; 2000. p. 167-174.

[Crestani, 2000] Crestani F, Lee PL. Searching the Web by constrained spreading activation. Information Processing and Management. 2000:585-605.

[Fellag, 2012] Fellag-Berchiche S. Propagation de pertinence et exploitation du texte ancre des liens et de la balise titre pour améliorer la recherche dans les documents XML. Technical Report. Tizi-Ouzou: Université Mouloud Mammeri de Tizi-Ouzou; 2012.

[Fellag, 2006] Samia FELLAG."Recherche d'information dans des documents semi_structurés XML". Thèse de Magister. Université Mouloud Mammeri de Tizi-Ouzou.2006.

[Fox, 1992] Fox C. Lexical analysis and stoplists. In: Information retrieval. Upper Saddle River, NJ: Prentice-Hall; 1992.

[Grosso, 2003] P. Grosso, E. Maler, J. Marsh, and N. Walsh. XML Pointer Language (XPointer). Technical report,World Wide Web Consortium (W3C),W3C Recommendation, march 2003.

[Guo, 2003] Guo L, Shao F, Botev C, Shanmugasundaram J. Xrank : Ranked search over XML documents. In: SIGMOD'03. New York, NY: ACM; 2003. p. 16-27.

[Haveliwala, 2002] Haveliwala TH. Topic-sensitive PageRank. In: Proceedings of the 11th international conference on World Wide Web. New York, NY: ACM; 2002. p. 517-526.

[Jeh, 2003] Jeh G, Widom J. Scaling personalized web search. In: Proceedings of the 12th international conference on World Wide Web. New York, NY: ACM; 2003. p. 271-279.

[Kamps, 2008] J. Kamps and M. Koolen. The importance of link evidence in wikipedia.In Lecture Notes in Computer Science, pages 270–282, Heidelberg, 2008.

[Kamvar, 2003] Kamvar SD, Haveliwala TH, Manning CD, Golub GH. Exploiting the Block Structure of the Web for Computing PageRank. Technical Report. Stanford; 2003.

[Kleinberg, 1998] J. Kleinberg. Authoritative sources in a hyperlinked environment. InProc. 9th Annual ACM-SIAM Symposium Discrete Algorithms, pages 668–677, 1998.

[Kolda, 2006] Kolda T, Bader B. The tophits model for higher-order web link. In: Workshop on Link Analysis, Counterterrorism and Security. Vol 7. 2006. p. 26-29.

[Lempel, 2001] Lempel R, Moran S. SALSA: the stochastic approach for link-structure analysis. ACM Transactions on Information Systems. 2001:131-160.

[Maron, 1960] Maron ME, Kuhns JL. On Relevance, Probabilistic Indexing and Information Retrieval. Journal of the ACM. 1960:216-244.

[Martin] Martin Theobald, Andreas Broschart, Ralf Schenkel, Silvana Solomon, and Gerhard Weikum. TopX – AdHoc and Feedback Tasks. Max-Planck-Institut fur Informatik Saarbrucken, Germany.

[Mattaoui, 2009] Mattaoui M, Mezghiche M. Prise en compte des liens pour améliorer la recherche d'information structurée. In: CORIA 2009. Alger 2009. p. 363-372.

[Mendelzon, 2000] A. O. Mendelzon and D. Rafiei. What do the neighbors think ? computing web page reputations. In IEEE Data Engineering Bulleting, September 2000.

[Najork, 2007] M. Najork, H. Zaragoza, and M. Taylor. Hits on the web : How does it compare ? In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007.

[Neil, 2002] Neil Bradley. The XML Companion. Addison Wesley, 3rd edition, 2002.

[Nie, 2004] Jian-Yun Nie, cours hiver 2004; Module recherche d'information ; université Montreal. Départment D'informatique et de recherche opérationelle (I.R.O) Hiver 2004. (*http://www.iro.umontreal.ca/~nie/IFT6255*).

[Porter, 1997] Porter MF. An algorithm for suffix stripping. In: Readings in information retrieval. San Francisco, CA: Morgan Kaufmann Publishers Inc.. p. 313-316. 1997.

[Piwowarski, 2005] B. Piwowarski. Precision Recall with User Modelling in EvalJ. Center for Web Research Universidad de Chile Santiago, Chile. 2005

[Robertson, 1997] Robertson SE, Walker S. On relevance weights with little relevance information. SIGIR :16–24. 1997.

[Robertson] Robertson SE. The probability ranking principle in IR. Journal of Documentation. 294-304.

[Sall, 2002] K. B. Sall. XML family of specifications.

[Salton, 1970] Salton .G . The SMART retrieval system : Experiments in automatic document processing. Prentice Hall. 1970.

[Salton, 1971] Salton .G. A comparison between manual and automatic indexing methods. Journal of American Documentation, 20(1): pages 61–71, 1971.

[Salton, 1975] Salton G, Wong A, Yang CS. A vector space model for automatic indexing. Communications of the ACM :613–620.1975.

[Salton, 1984] Salton .G and M. McGill. Introduction to modern information retrieval.

McGraw-Hill Int. Book Co. 1984.

[Savoy, 2000] Savoy J, Rasolofo Y. Report on the TREC-9 experiment: Link-based retrieval and distributed collections. In: Proceedings of TREC-9. Gaithersburg, MD: NIST; 2000. p. 579-588.

[Singhal, 1996] Singhal A, Buckley C, Mitra M. Pivoted document length normalization. SIGIR.:21–29. 1996.

[Soule, 2007] Soule-Dupuy C., Recherche et exploration d'information, Objectifs de la RI, Cours Master 2IH, 2006/2007.

[Web 1] http://miage.univnantes.fr/miage/D2X1/chapitre_xpath/section_xpointer.htm# introduction (Xpointer).

[Wang, 2008] C. Wang, L. Zhang, and H.-J. Zhang. Scalable markov model-based image annotation. In CIVR'08 : Proceedings of the 2008 international conference on Content-based image and video retrieval, pages 113–118, New York, NY, USA, 2008.

[Xavier] Xavier Tannier. Indexation et Recherche d'Information " Recherche d'information semistructurée ". Université PARIS-SUD 11.