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

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire de Master

***Option** : Systèmes Informatiques*

Thème :

Propagation de pertinence et exploitation du texte ancre des liens et de la balise titre pour améliorer la recherche dans les documents XML

Proposé et dirigé par :

M^me^ Samia FELLAG

Réalisé par :

M^elle^ HABRACHE Lynda

2011/2012

Sommaire

Chapitre2 : La recherche d'information structurée

Chapitre3 : Exploitation des liens

Liste des figures

La recherche d'information (RI) était déjà présente dans la vie quotidienne avant l'avènement de l'informatique et les technologies de communication, elle était réalisée par l'utilisation des moyens empiriques (répertoires ou annuaires d'index manuels) ainsi que des annotations et résumés pour accéder aux documents contenant l'information recherchée.

L'élaboration de systèmes automatisés pour gérer l'explosion du volume d'informations disponibles produites par des sources d'informations distribuées est devenue dans un tel contexte une nécessité.

Depuis son apparition, le WEB n'a cessé de progresser pour devenir une grande source d'informations immédiatement disponibles. Toutefois, l'accès à l'information est devenu de plus en plus difficile car l'internaute est perdu dans la masse disponible.
En effet, les moteurs de recherche sur les WEB et les systèmes de recherche d'information (SRI) sont une aide inestimable pour rechercher une information.

L'objectif principal des Systèmes de Recherche d'Information (SRI) est de répondre au besoin en information des utilisateurs. Les utilisateurs interrogent, à travers une requête, une base de documents et les SRI leur renvoient une liste de documents susceptibles de répondre à leur besoin.
De nouveaux standards de représentation de l'information telle que le XML sont créés pour améliorer la qualité de réponse des SRI, ainsi Les systèmes de recherche d'information doivent tirer profit de cette nouvelle notion qui est la structure.
Cette structure peut alors servir à traiter l'information textuelle avec une autre granularité que le document tout entier.
L'exploitation des liens est l'un des problèmes confrontés dans la RIS. Plusieurs études concernant l'exploitation des liens, en particulier sur le Web ont été effectuées. Comme résultats de ces études, plusieurs algorithmes se basant sur le nombre de liens entrants et sortants tels que PageRank , HITS et XRank en RIS ont été proposés et ont montré leur intérêt.
L'objectif de ce travail consiste en l'implémentation d'une approche qui exploite les liens entrants dans les documents XML, en prenant en compte le contenu de la description du lien ainsi que le titre du document lié, en attribuant un score à chacune de ces représentations, et monter aussi que ces paramètres vont permettre d'améliorer les résultats d'une recherche classique.

La problématique à laquelle tente de répondre ce présent mémoire est :
« Comment exploiter l'information des liens et le titre pour améliorer la réponse apportée à l'utilisateur? ».
Pour ce faire, nous l'avons organisé en 5 chapitres :
Dans le premier chapitre, nous présentons les notions de base de la RI et les modèles de RI les plus connus, le deuxième chapitre est consacré à la présentation du XML et recherche d'information dans les documents structurés.

Nous enchainerons dans le troisième chapitre avec une présentation des différents travaux effectués sur l'exploitation des liens dans le web et les documents structurés.

Par la suite, dans le quatrième chapitre nous présentons notre approche pour la prise en compte des liens dans la recherche d'information et un exemple applicatif de cette méthode pour bien illustrer son fonctionnement.

Le dernier chapitre est consacré à l'implémentation et l'évaluation de la méthode ainsi que les différents outils utilisés pour son développement.

Nous terminons ce mémoire par une conclusion générale qui comporte une synthèse de l'ensemble de notre travail, d'éventuelles perspectives et une annexe sur XML.

I. introduction

La recherche d'information (RI) traite de la représentation, du stockage, de l'organisation et de l'accès à l'information. Cette discipline est apparue dans un contexte où les progrès des technologies de l'information ont changé la perception de l'accès à l'information. Avec l'avènement du web, l'expansion de l'informatique à tous les domaines de la vie courante, a eu pour conséquence directe, l'accessibilité par un large public d'utilisateurs, autre que des documentalistes spécialisés, à des masses d'information volumineuses et hétérogènes.

Les efforts continus des chercheurs en RI ont permis jusqu'à présent d'améliorer sans cesse les performances et la qualité des services d'accès à l'information de manière délibérée à travers les Systèmes de Recherche d'Informations.

II. définition de la RI

D'après [L'AFNOR, 1979] [Urfist, 2004] « la recherche d'information (RI) est un ensemble de méthodes et de procédures ayant pour objet d'extraire d'un ensemble de documents, les informations voulues. Dans un sens plus large, la RI est toute opération (ou ensemble d'opérations) ayant pour objet la recherche, la collecte et l'exploitation d'information en réponse à une question sur un sujet précis »

III. Les Systèmes de Recherche d'Informations

III.1. Définition d'un SRI

Un SRI (Système de Recherche d'Informations**)** est un ensemble de programmes informatiques qui a pour but de sélectionner des informations pertinentes à partir d'une base de documents volumineuse, répondant à des besoins utilisateurs, exprimés sous forme de requêtes.

Ils sont apparus en vue d'automatiser la gestion des documents et de renvoyer à l'utilisateur l'information qu'il recherche.

Cette définition fait apparaître trois notions clés : document, requête et pertinence.

- **Document :** un document peut être un texte, une page web, une image, une vidéo…..etc. c'est toute source d'information qui constitue une réponse à une requête utilisateur.

- **Requête :** La requête constitue l'expression du besoin en information de l'utilisateur, généralement sous forme d'un ensemble de mots clés. Plusieurs types de langage d'interrogation ont été proposés afin de bien représenter ces besoins, ainsi une requête peut être décrite soit :

En langage naturel : comme dans les systèmes SMART (Salton's Magical Automatic Retriever of Text) [Salton, 1971], [Robertson & al.1998]

Par exemple: << donnez moi toute les agences de voyage existante en France >>

En langage booléen: cas des systèmes DIALOG [Bourne & Anderson **,** 1979]

Par exemple, pour les maladies génétiques :<<maladie AND génétique>>

En langage graphique: à partir d'une interface graphique, cas des systèmes de projet NEURODOC [Lelu & François, 1992].

- **Pertinence :** La pertinence est une notion très importante qui détermine le degré de correspondance entre le document et la requête.

Beaucoup de travaux de recherche s'accordent sur la difficulté de la définition de la pertinence. Plusieurs définitions lui sont associées parmi lesquels [Nie, 2004] :

- la correspondance entre un document et une requête, une mesure d'informativité du document à la requête;
- un degré de relation (chevauchement, relativité, …) entre le document et la requête...

On distingue deux niveaux de pertinence :

Une pertinence système: où le document jugé pertinent par le SRI à l'aide d'une fonction de pertinence appliquée à une requête [Cleverdon CW, 1970].

Une pertinence utilisateur: c'est l'utilisateur qui juge le document par rapport à son besoin en information exprimé dans la requête [Harter S ,1992].

III.2. Le processus de Recherche d'Information

Le processus de Recherche d'Information a pour but de mettre en correspondance les représentations des informations contenues dans un fond documentaire d'une part avec celle des besoins de l'utilisateur d'autre part, ou par d'autres termes, de correspondre au mieux la pertinence système avec la pertinence utilisateur Ce processus fait ressortir trois mécanismes de base :

1. l'indexation des documents disponibles et des requêtes utilisateurs
2. l'appariement de requête-document pour une comparaison.
3. la reformulation de requêtes pour réécrire autrement la requête utilisateur.

Pour résumer ces mécanismes, nous pouvons les représenter schématiquement par ce que l'on appelle communément le processus « en U », tel que dans cette figure :

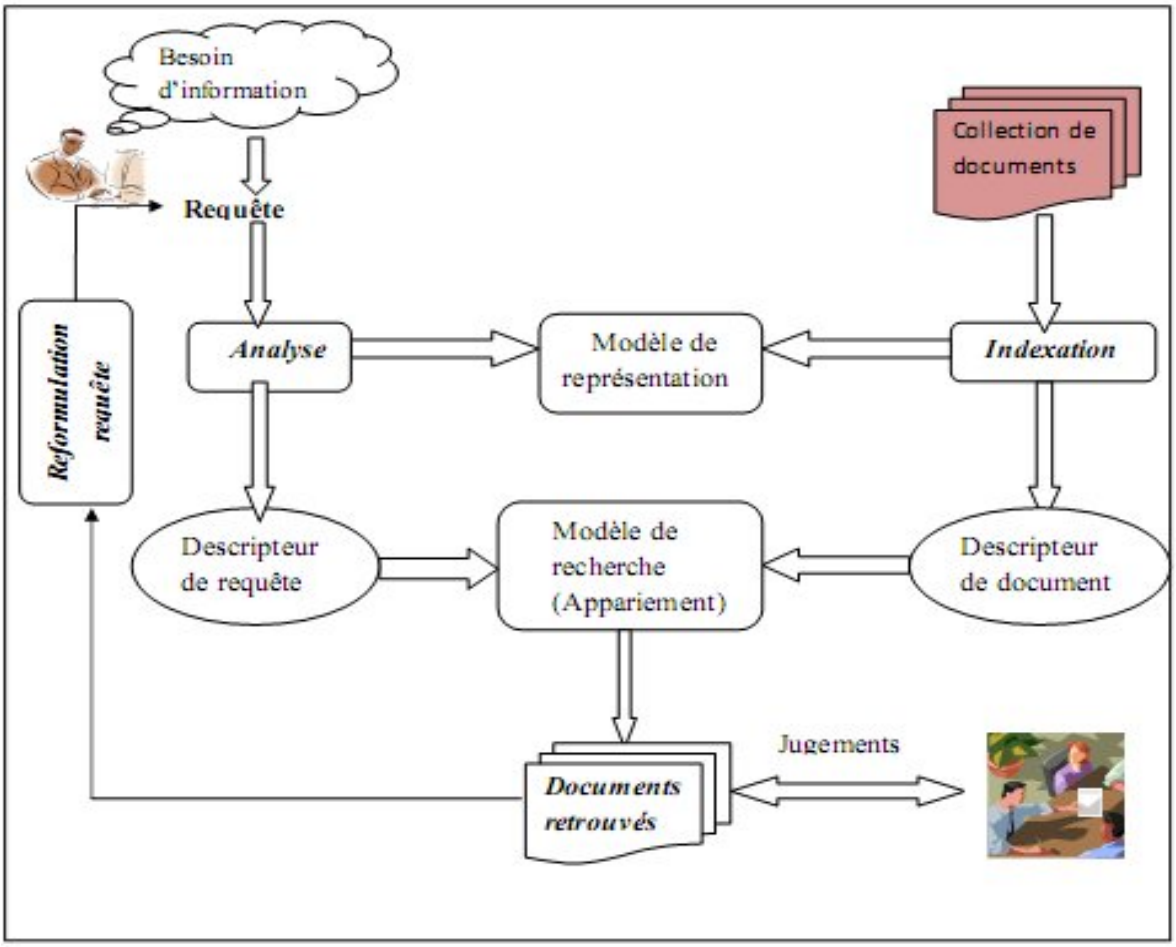


Figure 1 : processus U en recherche d'information [Fellag.S, 2006]

Modèle de représentation

C'est un système référentiel constituant d'un ensemble de règles et notations permettant d'extraire d'un document ou d'une requête une représentation structurée qui couvre au mieux son contenu sémantique. Ce processus est appelé **l'indexation.**

Le résultat de l'indexation constitue *le descripteur de document* qui est composé d'un ensemble de termes significatifs (mots simples, mots composés ou groupe de mots).

Modèle de recherche

C'est la fonction principale d'un SRI. C'est à ce niveau que se mesure l'**appariement** d'un document vis-à-vis d'une requête, à chaque réception d'une requête, le système crée une représentation similaire à celle des documents puis il calcule le degré de correspondance entre la représentation de chaque document et celle de la requête. Ce degré peut être binaire ou multi value.

Les jugements des utilisateurs sont pris en considération pour améliorer la représentation des requêtes, ce qu'on appelle la reformulation de la requête.

Reformulation de requête

La reformulation de requête consiste à optimiser la fonction de pertinence qui a pour but de rapprocher la pertinence utilisateur de la pertinence système.

Elle permet de construire une nouvelle requête en se basant sur des informations extraites des documents ou des connaissances disponibles dans des ressources spécifiques.

IV. le processus d'indexation

L'indexation recouvre un ensemble de techniques visant à transformer les documents (ou requêtes) en substituts ou descripteurs capables de représenter leur contenu

[Salton & McGill, 1983]. Ces descripteurs forment le langage d'indexation représenté selon une structure souvent basé sur un ensemble de mots clés ou groupes de mots représentant le contenu textuel du document.

Dès lors, l'indexation consiste à détecter les termes les plus représentatifs du contenu du document. Différents modes d'indexation existent en RI :

L'indexation manuelle : faite par un expert humain.

L'indexation automatique : chaque document est analysé à l'aide d'un processus entièrement automatisé.

L'indexation semi-automatique : L'indexation semi-automatique [Maniez & al, 1991] [Balpe & al, 1995], appelée aussi indexation supervisée, est une combinaison des deux approches d'indexation précédentes. Dans ce cas, les indexeurs utilisent un vocabulaire contrôlé sous forme de thésaurus ou de base terminologique. Le choix final des termes d'indexation à partir du vocabulaire fourni, est laissé ainsi à l'indexeur humain (généralement spécialiste du domaine).

IV.1.L'indexation manuelle

Lors de l'indexation manuelle, chaque document est analysé par un spécialiste du domaine ou un documentaliste qui se charge de caractériser, selon ses connaissances propres, le contenu sémantique d'un document. Cette approche présente plusieurs inconvénients :

1-Elle nécessite assez de temps pour sa réalisation, en plus, des termes différents peuvent être présentés par deux documentalistes pour représenter un même document.

2- elle est pratiquement inapplicable aux corpus de textes volumineux.…

Cependant, elle a l'avantage d'être plus précise dans les résultats [Ren & al, 1999], car les spécialistes d'un domaine choisissent de meilleurs termes pour indexer les documents.

IV.2.Indexation automatique

En indexation automatique [Luhn, 1957] [Salton & al, 1968], c'est un processus complètement automatisé qui se charge d'extraire les termes caractéristiques du document. L'intérêt d'une telle approche réside dans sa capacité à traiter les textes nettement plus rapidement que l'approche précédente, et de ce fait, elle est particulièrement adaptée aux corpus volumineux.

L'indexation automatique regroupe un ensemble de traitements automatisés sur un document. On distingue : l'extraction automatique des mots des documents, l'élimination des mots vides, la lemmatisation (radicalisation ou normalisation), le repérage de groupes de mots et la pondération des mots.

IV.2.1. L'extraction automatique des mots des documents

C'est un processus qui permet de convertir le texte d'un document en un ensemble d'unités lexicales, ce qui permet de distinguer les espaces entre les mots, les chiffres les ponctuations,...etc.

IV.2.2. l'élimination des mots vides

Un des problèmes majeurs de l'indexation consiste à extraire les termes significatifs et à éviter les mots vides (pronoms personnels, prépositions,...).

Les mots vides peuvent aussi être des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc). On distingue deux techniques pour éliminer les mots vides :

• **Utilisation d'un anti-dictionnaire (Stoplist)** : les mots vides sont rangés auparavant dans une liste qui est consultée au moment de l'indexation: si le mot figure dans la liste il sera éliminé de texte sinon il sera lui aussi indexé.

• **Prise en considération de l'occurrence d'apparition de terme:** la fréquence d'apparition des termes dans le document peut aider à déterminer ceux qui sont représentatifs pour celui-ci., d'ou on pourra choisir les mots représentatifs selon leur fréquence d'apparition dans le document.

IV.2.3. La lemmatisation

La lemmatisation est la fonction qui associe à chaque mot sa forme canonique (radicale).

Elle consiste à éliminer les différentes variations morphologiques d'un mot en extrayant le radical du mot (lemme ou racine).

Etant donné qu'un mot peut avoir plusieurs formes dans un texte, mais leur sens reste le même ou très similaire par exemple les mots : représentation, représenter, représentable, Il n'est pas utile d'indexer tous ces mots alors qu'on peut les représenter en un seul mot qui a le même concept : représent.

Frakes et Baeza-yates [Frakes & Baeza-yates, 1992] présente cinq types stratégiques de lemmatisation et parmi on cite :

- la table de consultation (dictionnaire)

- l'´elimination des affixes (on peut par exemple citer l'algorithme de Porter [Porter, 1980])

- La troncature.

IV.2.4. repérage de groupes de mots

Consiste à essayer d'indexer un terme complexe composé par exemple de deux mots qu'on ne peut pas séparer sinon le sens sera perdu [Fagan, 1987] [Salton, 1988], comme le terme "système d'information", si on essaye de le représenter par le mot "système" et "information" il perd son sens.

IV.2.5. la pondération des termes

Cette étape est généralement basée sur des formules de pondération qui affecte à chaque terme un degré d'importance (une valeur de discrimination), qui indique l'importance du terme dans la caractérisation d'un document. Cette mesure est souvent calculée en se basant sur des propriétés et interprétations statistiques.

Il existe un grand nombre de formules de pondération dont la plus connue est :

Tf*Idf qui est basée sur deux facteurs [Robertson & al., 1997] [Singhal & al., 1997]

[Sparck, 1979] fréquence de terme (TF) et fréquence inverse de document (IDF), définis dans ce qui suit :

1. fréquence de terme (TF) : La fréquence du terme (term frequency) est le nombre d'occurrences de ce terme dans le document considéré. L'idée sous-jacente est que plus un terme est fréquent dans ce document, plus il est important dans la description de celui-ci.

- *quelques fonctions de calcul de TF :*

- $tf = n_{oc}$

- $tf = 1$ si le terme existe dans le document, **0** sinon.

- $tf = (n_{oc}/tf_{max})$….

Avec :

$\mathbf{n_{oc}}$: Nombre d'occurrences du terme dans le document.

$\mathbf{tf_{max}}$: Nombre maximum d'occurrences d'un terme dans le document.

2. fréquence inverse de document (IDF) : La fréquence inverse de document (inverse document frequency) est une mesure de l'importance du terme dans l'ensemble du corpus.

- quelques fonctions de calcul de IDF *:*

$$=$$

$$= \quad -$$

$$= \quad +$$

Avec :

N : Nombre total de documents dans le corpus (la collection).

$\mathbf{n_i}$: Nombre total de documents contenant le terme i.

La mesure $\boldsymbol{tf * idf}$ donne une bonne approximation de l'importance du terme dans le document, particulièrement dans les corpus de documents de taille homogène.

V. les modèles de recherche d'information

Le modèle de recherche d'information repose sur les représentations de la requête et des documents issues de l'indexation. Il permet de leur donner une interprétation afin de déterminer les documents qui sont similaires à la requête.

Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence.

On peut distinguer trois grandes classes de modèles :

- Les modèles booléens
- Les modèles vectoriels
- Les modèles probabilistes

V.1. modèle booléen

Ils sont basés sur la théorie des ensembles [Salton, 1971b] et l'algèbre de Boole, ce sont les plus simples et les premiers à avoir été mis en place.

V.1.1. modèle booléen de base

Un document est représenté par un ensemble de termes d= ($t_1 \wedge t_2 \wedge t_3 ... \wedge t_n$). La requête est représentée par un ensemble de mots clés reliés par des opérateurs booléens (AND, OR et NOT). L'appariement requête-document est strict et se base sur des opérations ensemblistes selon les règles suivantes :

- RSV (t_i, d) =1 si $t_i \in$ d, 0 sinon
- RSV (t_i AND t_j, d) =1 si $(t_i \in d) \wedge (t_j \in d)$, 0 sinon
- RSV (t_i OR t_j, d) =1 si $(t_i \in d) \vee (t_j \in d)$, 0 sinon
- RSV (NOT t_i, d) =1 si $t_i \notin$ d, 0 sinon

On peut remarquer les inconvénients suivants:

- La sélection d'un document est basée sur une décision binaire.
- Formulation de la requête difficile pas toujours évidente pour beaucoup d'utilisateurs. Problème de collections volumineuses : le nombre de documents retournés peut être considérable.

V.1.2. modèle booléen étendu

Le modèle booléen étendu a été introduit par Salton [Salton & al, 1983]. C'est une extension du modèle précédent qui vise à tenir compte d'une pondération des termes dans le corpus. Cela permet de pallier les problèmes du modèle de base en ordonnant les documents retrouvés par le SRI et en intégrant des poids d'indexation dans l'expression de la requête et document. Ceci a pour conséquence, la sélection de documents sur la base d'un appariement rapproché et non exact.

Considérons un ensemble de termes $t_1, t_2 . . ., t_N$, et soit d_{ij} le poids du terme t_i dans le document Dj = ($d_{1j}, . . ., d_{Nj}$), avec $1 \leq i \leq N$ et $0 \leq d_{ij} \leq 1$. La similarité entre le document Dj et une requête q_k décrite sous une forme conjonctive ou disjonctive est donnée comme suit :

Pour l'operateur **OR :**

$$RSV(dj, qk) = \frac{\Sigma}{\Sigma}$$

Pour l'operateur **AND :**

$$RSV(dj, qk) = - \frac{\Sigma}{\Sigma}$$

Où P est une constante $0 \leq P \leq \infty$, et q_{ik} le poids du terme t_i dans la requête q_k.

Lorsque p=1, il n'y a aucune distinction entre les deux connecteurs OR et AND.

L'avantage principal de ce modèle est de pouvoir classer les documents suivant leur pertinence. Mais les requêtes restent complexes et difficilement formulées par l'utilisateur.

V.2. Le Modèle Vectoriel

Les modèles vectoriels [Salton, 70] sont des modèles algébriques. Les documents et requêtes sont représentés par des vecteurs de poids dans un espace vectoriel composé de tous les termes d'indexation. La pertinence d'un document vis à vis d'une requête est définie par des mesures de distances entre vecteurs.

Formellement, un document d_i est représenté par un vecteur de dimension n et représenté comme suit :

$$D_j = \begin{bmatrix} d_{1j} \\ d_{2j} \\ \\ d_{Nj} \end{bmatrix} \quad Q_k = \begin{bmatrix} q_{1k} \\ q_{2k} \\ \\ q_{Nk} \end{bmatrix}$$

Où :

d_{ij} et q_{ik} correspondent respectivement aux poids du terme t_i dans le document D_j et dans la requête Q_k.

Le modèle vectoriel estime le degré de pertinence entre un document et la requête par un degré de corrélation entre leurs vecteurs associés. Cette corrélation peut être spécifiée par le calcul de similarité entre vecteurs.

Diverses mesures de similarité ont été proposées dans ce modèle qui sont :

Inner Product (produit interne)

$$, \quad =$$

Coefficient de Dice

$$, \quad = \frac{* \Sigma}{\Sigma \quad +}$$

Mesure du cosinus

$$, \quad = \frac{\Sigma}{\Sigma \quad / \quad * \quad \Sigma \quad /}$$

Mesure de Jaccard

$$, \quad = \frac{\Sigma}{\Sigma \quad + \Sigma \quad - \Sigma}$$

Les avantages d'un tel modèle sont nombreux :

- La pondération des termes augmente les performances du système.
- Le modèle permet de renvoyer des documents qui répondent approximativement à la requête.
- la fonction d'appariement permet de trier les documents selon leur degré de similarité avec la requête.

Ainsi, l'inconvénient majeur est que la représentation vectorielle suppose l'indépendance entre termes.

V.3. Modèle Probabiliste

Le modèle probabiliste est basé sur la théorie des probabilités, et estime des probabilités de pertinence d'un document en fonction de la requête. Et parmi les modèles on trouve :

V.3.1. Le modèle probabiliste de base

Le premier modèle probabiliste a été proposé par Maron et Kuhn [Maron and Kuhns, 1960] au début des années 1960. Ils proposent de modéliser le processus de sélection des documents dans un SRI en se basant sur la théorie de probabilités. Le principe de base de ce modèle consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête. Pour cela deux formules de probabilité conditionnelles sont utilisées:

P (w_{ij} / Pert) : Probabilité que le terme t_i occure dans le document Dj sachant que ce dernier est pertinent pour la requête.

P (W_{ij}/ NonPert) : Probabilité que le terme t_i occure dans le document Dj sachant que ce dernier n'est pas pertinent pour la requête

Si on suppose l'indépendance des variables documents « pertinents » et « non pertinents », la Fonction de recherche sera comme suit :

$$(\quad / \quad) = \frac{(\quad / \quad)}{(\quad / \quad)}$$

Le modèle de probabiliste a l'avantage sur le modèle vectoriel de prendre en compte la dépendance entre les termes dans le calcul de la pertinence.

V.3.2. Le modèle de langue

Les systèmes de recherche d'information utilisant les modèles de langues suivent une approche différente des autres modèles.

En effet, les autres modèles, cherchent à retrouver les documents pertinents en le comparant à la requête.

Le modèle de langue part de l'observation que l'utilisateur crée la requête à partir d'une représentation hypothétique qu'il se fait du document recherché. La requête est donc générée à partir des documents voulus.

Le but de ce modèle est donc de générer des requêtes à partir des documents, et les comparer avec la requête de l'utilisateur. La pertinence d'un document est donc estimée en calculant la probabilité que la requête utilisateur soit inférée par celui-ci. En supposant les termes indépendants, on a [Ponte et Croft, 1998] :

$$| = | * - |$$
$$\notin$$

Où :

- Q_k : requête exprimée par une suite de termes : $q_{1k}, q_{2k}, \ldots q_{nk}$
- Tous les termes d'indexation sont considérés dans l'estimation de cette probabilité (ceux présents dans la requête et ceux absents de la requête)

[Hiemstra, 2002] propose une interpolation entre le modèle de document ($P(q_{ik}|D_j)$) et le modèle de contexte du document (la collection)($P(q_{ik})$) comme suit :

$$| = (-\lambda) ( ) + \lambda |$$

Avec :

( ) : Probabilité d'occurrence du terme t_{ik} dans la collection considérée.

| : Probabilité pour que le terme t_{ik} soit dans le document pertinent Dj.

λ_i : paramètre à estimer, ($0 \leq \lambda_i \leq 1$) probabilité pour que le terme soit important.

VI. paramètres d'évaluation d'un SRI

Afin de satisfaire les besoins en information de l'utilisateur, l'évaluation des performances des systèmes de recherche est indispensable. L'évaluation permet de caractériser le modèle et de fournir des éléments de comparaison entre différents modèles.

L'évaluation des systèmes peut être entamée selon deux aspects : l'efficience et l'efficacité.

- *L'aspect efficience* dépend de l'évaluation cognitive de l'utilisateur, tels que la facilité d'utilisation du système, rapidité d'accès, temps de réponse à une requête, présentation des résultats, etc.
- *L'aspect efficacité* concerne la capacité du système à sélectionner le maximum de documents pertinents et un minimum de documents non pertinents.

On s'intéresse dans ce qui suit à présenter l'aspect efficacité qui est souvent mesuré par deux paramètres Rappel et Précision, [Kent & al, 55].

VI.1. les mesure de rappel et précision

- **Le rappel :**

Le rappel mesure la proportion de documents pertinents renvoyés par le système relativement à l'ensemble des documents pertinents contenus dans la base documentaire. Il est exprimé sous la forme suivante :

Soit :

R : les documents pertinents dans la collection.

Ra : les documents pertinents renvoyés par le système.

A : les documents retournés par un système.

=

- **La précision**

La précision mesure la proportion de documents pertinents relativement à l'ensemble des documents renvoyée par le système. Elle est exprimée sous la forme suivante :

é =

Pour un système idéal, le taux de précision est égal à celui de rappel. Le système parfait trouverait seulement les documents pertinents, avec une précision et un rappel de 100%. Ce qui n'est pas le cas en pratique, ou les mesures de rappel et précision évoluent inversement.

VII. Conclusion

Dans ce chapitre nous avons présenté les concepts de base de la recherche d'information à savoir le processus de Recherche d'Information et ses différentes étapes, ainsi que les différents modèles de la recherche et enfin les paramètres d'évaluation d'un SRI.

Les SRI que nous avons décrit fonctionnent généralement sur des documents multi-formats (structurés, non structurés, balises ou non balises), cependant ils exploitent uniquement le contenu sémantique des documents (texte).

Actuellement, de nouveaux formats de structuration sont proposés pour permettre la Combinaison de la recherche textuelle et la recherche structurelle à travers XML que nous allons détailler dans le chapitre suivant.

I. Introduction

La nature des sources d'information évolue, et les documents numériques traditionnels "plats" ne contenant que du texte s'enrichissent d'information structurelle et multimédia. Cette évolution est accélérée par l'expansion du Web, et les documents semi-structurés de type HTML(HyperText Markup Language) et XML (eXtensible Markup Language) tendent à former la majorité des documents numériques mis a disposition des utilisateurs.

Les SRI traditionnels accèdent uniquement au contenu de ces documents sans prendre en considération la coexistence de l'information structurelle et de l'information de contenu. La prise en compte de la dimension structurelle devrait permettre de mieux répondre aux différents besoins des utilisateurs.

II. Qu'est ce qu'XML

XML (eXtensible Markup Langage ou langage de balisage extensible), conçu pour la représentation et l'échange de données sur Internet par la description hiérarchique du document, il permet aussi de représenter et d'échanger les documents semi-structurés tels que les documents XML. Ces derniers sont dites semi-structurés car ils possèdent une structure qui n'est pas prédéfinie, mais une structure que le programmeur peut déterminer lui-même au moment de la conception.

XML est aussi un **métalangage**, c'est-à-dire, un langage pour écrire d'autres langages. Il est déjà à la base de toute une série de nouveaux langages comme le XHTML, le MathML.

Autour des spécifications d'XML, de nombreuses technologies, proposées par le W3C (**W**orld **W**ide **W**eb **C**onsortium), sont apparues. Le schéma ci-dessous, extrait de

[Vieillard, 2000] présente les plus importantes.

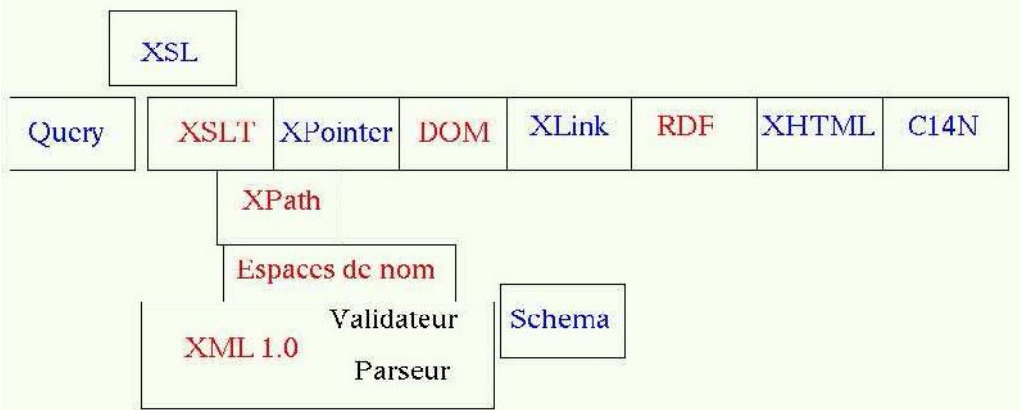


Figure 2 : Les technologies de la famille XML

III .Structure d'un document XML

Un document XML est une structure arborescente dont chaque nœud contient les données. Il ya en général *un prologue :* facultatif mais fortement conseillé. Ensuite l'arbre lui même avec le contenu du document(les données). Dans certains cas, des *commentaires* et des *instructions de traitement* peuvent apparaitre dans le prologue ou le contenu.

III.1.Le Prologue

C'est les premières lignes qui forment le document XML, consiste en : la déclaration XML, les instructions de traitements et éventuellement la déclaration d'une DTD.

III.1.1. Déclaration XML

C'est tout le code de l'entête il fait partie des instructions de traitements.

Exemple de déclaration XML :

Syntaxe :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

On distingue trois informations fournies dans cette déclaration :

- **version** : version du **XML** utilisée dans le document.
- **encoding** : le jeu de codage de caractères utilisé. Le jeu de caractères habituel pour le français est le ISO-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 UTF-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.
- **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 no.

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. Si elle est utilisée, elle doit être placée en toute première ligne du document

III.1.2. Instruction du 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 traitement qui servent le plus souvent sont la déclaration XML ainsi que la déclaration de feuille de style.

III.1.3. Déclaration de Type de Document (DTD) :

Cette déclaration, lorsqu'elle est présente, permet de définir la structure du document. Elle peut être de deux types, externe ou interne. Exemple de déclaration de type de document :

Syntaxe :

```
<!DOCTYPE biblio SYSTEM "biblio.dtd">
```

Ce type de déclaration est celui d'une déclaration de type de document externe).Elle définit l'ensemble des éléments utilisables dans le document, y compris l'élément-racine (ici biblio) ainsi que l'emplacement où se trouve le fichier biblio.dtd dans lequel se trouve définie la structure du document.

Bien que facultative, il est souvent très intéressant de posséder une DTD, en particulier externe, simplement pour vérifier la validité du document XML.

L'autre type de document permettant de définir la structure d'un fichier, le schéma XML (voir Annexe).

III.2.Les Commentaires

En XML, les commentaires se déclarent de la même façon qu'en HTML. Ils commencent donc par **<!--** et se terminent par **-->**. Ils peuvent être placés à n'importe quel endroit tant qu'ils se trouvent à l'extérieur d'une autre balise.

Exemples de commentaires valides :

```
< !--ceci est correct -->

<elt >< !--ceci est correct aussi -->
```

III.3.L'arbre d'éléments

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. Voici un exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <biblio>
        <livre>
                    <!-- Élément enfant titre -->
                    <titre>Les Misérables</titre>
                    <auteur>Victor Hugo</auteur>
                    <nb_tomes>3</nb_tomes>
         </livre>
         <livre>
                     <titre>L'Assommoir</titre>
```

III.3.1. Elément Racine :

L'élément-racine (en anglais : document element) 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 biblio.

III.3.2. 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 :

```
<titre>Les Misérables</titre>
```

Exemple d'élément contenant d'autres éléments :

```
<livre>
    <titre> L'Assommoir</titre>
    <auteur> Émile Zola</auteur>
    <couverture couleur="rouge" />
```

D'autres éléments sont vides : ils ne contiennent pas d'élément-enfant. Exemple d'élément vide :

```
<couverture couleur="rouge" />
```

III.3.3. 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>
```

IV. Les Parseurs XML

XML est uniquement un langage de structuration et de représentation de données. Il ne comporte pas d'instructions de contrôle et ne permet donc pas d'exploiter directement les données. Pour traiter ces données, il faut disposer d'un *analyseur*. Un *analyseur* (ou *parser* en anglais), permet de récupérer dans une structure XML, des balises**,** leur contenu**,** leurs attributs et de les rendre accessibles.

XML dispose de deux types de parseur :

- le parser **SAX** (***S****imple* ***A****PI for* ***X****ML*), orienté événement.

- le parser **DOM** (***D**ocument **O**bject **M**odel*) orienté hiérarchie.

IV.1. SAX (*Simple API for XML*)

SAX (Simple API for XML) est une API basée sur un modèle événementiel, qui transforme un document XML en un flux d'évènements déclenchés par la lecture d'éléments syntaxiques XML (balise ouvrante, balise fermante, etc ...).

SAX a comme avantage, grâce à son fonctionnement, de ne lire le code que par petites portions, ce qui lui évite de le charger en mémoire intégralement,

SAX utilise des méthodes qui permettent la gestion des événements (*startDocument()*,

endDocument(), *startElement(..)*, *endElement(..)* ...). C'est grâce à cela que SAX génère un événement à chaque fois qu'une partie d'un document précis est rencontrée (début ou fin de

document, début d'un élément...).

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

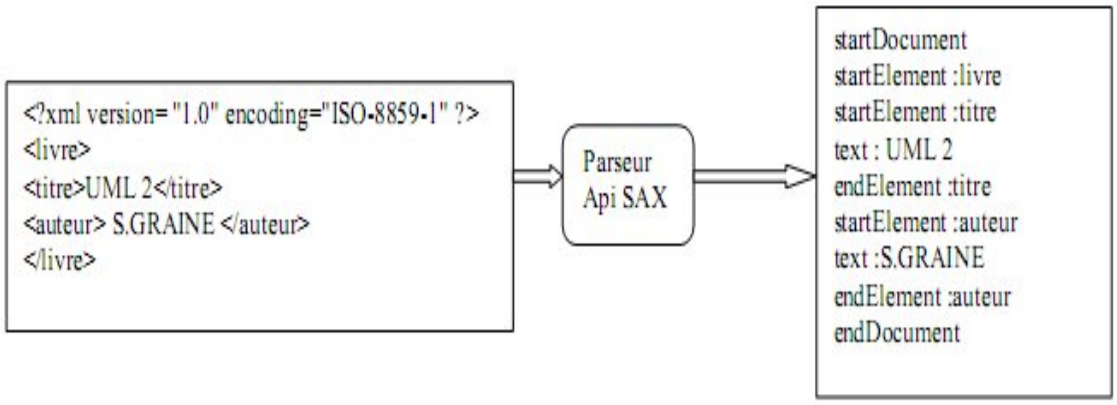


Figure 3: Exemple d'arbre SAX

IV.2. DOM (*Document Object Model*)

DOM (Document Object Model) est une API permettant d'accéder au contenu d'un document XML sous la forme d'une structure arborescente. Le document XML, après avoir été totalement chargé en

mémoire, est accessible au travers d'un ensemble d'objets correspondant aux différents types de nœuds qui s'y trouvent [Hunter&al, 2001], et exposant les méthodes permettant de parcourir l'arbre, de façon hiérarchique ou transversale.

Il existe des implémentations de DOM dans pratiquement tous les langages interprétés ou compilés existant pouvant lire des documents XML.

Exemple de document XML et l'arbre DOM qui lui est associé [Amann, 2003]

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

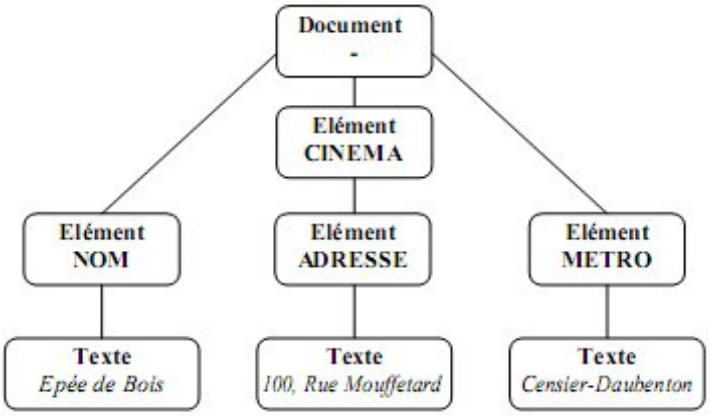


Figure 4 : Exemple d'arbre DOM

V. Les liens dans les documents XML

Deux types de liens sont à considérer dans les documents XML, les liens intra documentaires et les liens extra documentaires. Les liens intra documentaires sont représentés par XPointer [Grosso et al., 2003], et les liens extra documentaires sont référencés par XLink.

V.1. XPointer (XML Pointer)

XPointer fournit un moyen simple d'adresser la structure interne (une partie) d'un document XML. Pour cela, il utilise le langage XPath qui décrit la localisation des nœuds et extrait des valeurs de l'arbre du document XML.

Un pointer est composé d'une référence absolue suivie ou non d'une série de renvois relatifs.

- ***Les références absolues :***

Root () : la racine de document.

Origine () : le document en cours. Cette référence doit être suivie d'autres références.

Id (valeur) :l'élément dont l'attribut de type Id a la valeur indiquée.

- ***Les références relatives*** **:** la référence relative suit une référence absolue et elle en est séparée par un point (.). elle est une expression XPath.

Xpointer offre des extensions à Xpath, on distingue :

-Viser non seulement des nœuds entiers mais également des parties de nœuds dans un document XML.

- Compléter les URI *(URI#XPOINTER)*, pour localiser des adresses et étendre ainsi la puissance et la flexibilité des liens.

-Utiliser les fonctions : *string-range()*, *range()* et *range-to()* ; *here()* et *origin()* ; *startpoint()* et *end-point().*

Start-point **:** considère un ensemble de localisation comme paramètre, renvoie et un ensemble de localisation contenant les points de départ de tous les emplacements trouvés dans l'ensemble de

localisation d'entrée. Par analogie la fonction ***end-point()*** fournit les points de terminaison utilisés de la même manière avec la même structure d'arguments.

Range () et Range-to() *:* elle prennent un ensemble de localisation comme argument et renvoient un ensemble de localisation comme résultat.

La fonction ***here()*** **et** ***origin()*** sont utilisées pour retourner l'élément situé aux points ou le pointeur Xpointer est localisé (*here())* ou à l'endroit ou la traversée a commencé (*origin()*).

Voici un exemple d'un document avec XPointer :

```
<?xml version="1.0" encoding="utf-8" ?> <article>
<titre id="2305">Physique nucléaire</titre>
<body>: le
<collectionlink
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="simple"
        xlink:href="6147.xml">noyau
        atomique</collectionlink>
<section>
<titre>Cohésion du noyau</titre>
<p>
A l'intérieur du noyau, les
<collectionlink
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xlink:type="simple"
        xlink:href="6718.xml#xpointer(id('intéraction
        magnétique'))"…
</p> …
</section>
```

V.2. XLink (XML Link)

XLink permet de définir des liens entre différents documents ou des parties de documents, grâce à un mécanisme plus puissant que celui de l'HTML.

XLink, qui spécifie des constructions syntaxiques que l'on peut insérer dans des documents XML pour décrire des liens entre objets. Un lien, ici, est une relation explicite entre deux objets de données ou portions d'objets.

XLink utilise une syntaxe XML pour créer des structures capables de décrire les liens unidirectionnels simples qui associent deux ressources dont l'une est locale et l'autre est distante. Pour cela Deux types de liens Xlink sont définis : les liens **simple** (*simple link*) et les liens **étendus** (*extended link*).Comme pour un lien hypertexte HTML, le *lien simple* permet de mettre en relation, de façon unidirectionnelle, une source et une destination.

Le *lien étendu* permet de mettre en relation (*arc*) un nombre arbitraire de ressources qui peuvent être locales (*resource*)(à l'endroit où est défini le lien) ou distantes (*locator*).

Le type *arc* définit la direction que doit suivre le lien, le type *resource* utilisé dans un élément de type étendu pour créer des ressources locales et le type *locator* utilisé dans un élément de type étendu pour indiquer les ressources distant.

Exemple de lien Xlink étendu :

```
<film xlink: href=”affiches.xml” xlink:type=”extended”>
   <titre xlink:type=”arc”
      xlink:from=”film:titre”
      xlink : to=”film:affiche”
      xlink:show=”replace”
      xlink:actuate= "onRequest”
      xlink:role= "film:obtenirImageAffiche”
```

Tels que :

-*href* appelé *attribut localisateur*, indique l'URI (universal ressource identifier) utilisé pour extraire une ressource.

-*type* indique le type d'élément Xlink crée, ce qui est obligatoire pour les liens.

-*from* et *to* indiquent les directions des liens.

-*schow* précise la façon dont la ressource doit être affichée une fois extraite, elle peut remplacer (*replace*) celle existante, s'ajouter à l'existante (*embed*) ou simplement être vue ailleurs, dans une nouvelle fenêtre (*new*)

- *actuate* : permet de définir si le lien doit être activé au moment du chargement (*onLoad*) ou sur demande (*onRequest*).

-*title* spécifie la ressource dans un format lisible par l'homme, utilisé pour la création d'info bulle.

VI. Recherche d'information dans les documents XML

L'émergence de format XML comme format standard pour la représentation des documents où la structure de ces derniers est apparente , a permis à la communauté de recherche d'information de commencer à s'intéresser à l'utilisation de cette nouvelle source d'information. Le balisage de documents XML permet de structurer les documents sous formes d'éléments imbriqués les uns dans les autres. Le but de la RI dans tels documents structurés (on parle plutôt de la RI structurée) est alors d'utiliser cette structure afin de renvoyer à l'utilisateur des éléments se focalisant sur son besoin c.-à-d. des éléments de granularité appropriée.

VI.1.Problématiques liées à la RI structurée

Pour les documents textes "plats", le contenu textuel des documents est traité afin de trouver et de pondérer les termes les plus représentatifs des documents. Dans le cas des documents semi-structurés, la dimension structurelle s'ajoute au contenu, et les questions suivantes se posent alors :

- que doit-on indexer de la structure des documents ?

- Comment relier cette structure au contenu du document ?

- Par rapport à quel élément doit-on pondérer les termes d'indexation ?

Donc, la problématique dans le cadre de l'indexation se situe essentiellement au niveau de la structure du document.

Au niveau du processus d'interrogation, Il s'agit de permettre à l'utilisateur d'exprimer des besoins diversifiés (contenu et/ou structure), et ce de manière simple. La dernière problématique concerne les modèles de recherche et de tri des unités d'information.

VI.2. Indexation des documents XML

L'indexation est l'un des processus piliers dans la Recherche d'Information Structuré. Dans le cas des documents textes, elle consiste à traiter la collection des documents disponibles afin de trouver et de pondérer les termes les plus représentatifs pour ces documents.

Mais dans le cas des documents semi-structurés tels que les documents XML. La dimension structurelle s'ajoute à l'information textuelle des documents.

Une technique d'indexation des documents XML doit répondre aux critères suivants :

- Permettre la restauration des documents, décomposés dans la structure de stockage.

- Permettre le traitement d'expressions de chemin dans les documents XML.

-Assurer une rapidité de navigation dans les documents XML.

-Permettre la recherche par mots clés.

VI.3.La Pondération des termes

La pondération des termes doit être vue sous un nouvel angle. Alors qu'en RI

traditionnelle, le poids d'un terme cherche à rendre compte de son importance de manière locale au sein du document et de manière globale au sein de la collection, en RIS s'ajoute l'importance du terme au niveau de l'élément qui le contient. L'apparition des termes ne

suivent pas forcément une loi de Zipf. Dans les documents XML le nombre de répétitions des termes peut être réduit dans les documents et l'utilisation d'idf (Inverse Document Frequency) n'est pas forcément appropriée. L'utilisation d'ief (Inverse Element Frequency) a été proposée pour généraliser la notion d'idf en considérant les nœuds d'un certain type plutôt que le document.

L'importance des termes dans des documents XML peuvent aussi être évaluée en prenant en compte les paramètres suivants : la fréquence du terme au sein de l'élément, la fréquence du terme au sein de document, la longueur de l'élément et la longueur moyenne des éléments de la collection.

VI.4.L'interrogation des documents XML

L'interrogation des documents XML diffère de celle des documents plats et ce est 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 : "recherche d'information".

- **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 de requête CAS : "ec :// article["recherche d'information"]"

VI.5.La pertinence

En RI classique l'unité d'information est le document et les jugements ont une valeur qui

peut varier entre 1 (le document est pertinent) et 0 (le document est non pertinent).

L'utilisation d'une telle échelle pour la RIS est problématique : lorsqu'une section d'un document contient un paragraphe pertinent, quelle valeur de pertinence donner à la section ? Il n'est pas souhaitable de lui donner la valeur 1 puisque la section contient beaucoup d'information non pertinente ; et non pas le 0 puisque la section contient de l'information pertinente. Si une valeur intermédiaire est choisie, il sera alors impossible de distinguer une section qui contient un paragraphe pertinent avec une section moyennement pertinente (mais qui ne contient pas de paragraphe pertinent).

La pertinence est composée de deux dimensions : la première mesure l'exhaustivité et la seconde la spécificité du l'élément pour une question donnée.

Exhaustivité : L'exhaustivité ne tient compte que de la présence ou de l'absence de l'information recherchée dans un élément, même si cette information n'apparaît que dans une toute petite partie d'élément.

Spécificité : La spécificité est totalement liée à l'évaluation de documents structurés. Cette mesure s'intéresse au degré avec lequel l'élément traite de toute l'information recherchée si l'élément contient l'information recherchée, ou d'une partie de cette information si l'élément en contient une partie.

VII. La Campagne d'Evaluation INEX

INEX (Initiative for the Evaluation of XML Retrieval) est à ce jour la seule campagne d'évaluation des différents SRI pour la recherche d'information sur des documents XML.

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

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

VII.1. La Collection de Test

La collection INEX est composée d'articles scientifiques provenant de la IEEE Computer Society, balisés au format XML. La collection, d'environ 500Mo, contient plus de 12000 articles, publiés de 1995 à 2002, et provenant de 18 magazines ou revues différents.

Les articles sont généralement composés d'une en-tête (<fm>), d'un corps (<body>) et d'annexes (<bm>). Chacun de ces éléments se rédécompose :

Par exemple, le corps est composé de section <sec> elles-mêmes composées de paragraphes <p> et les annexes sont composées de référence bibliographiques <bibl> et éventuellement de curriculum vitae <vt>.

Un article moyen est composé d'environ 1500 éléments, et la profondeur moyenne des documents est de 6.9. Au total, la collection contient 8 millions de nœuds et 192 balises différentes.

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.

VII.2.Requêtes

Les requêtes (ou Topics) sont crées par les différents participants et doivent être représentatives des demandes de l'utilisateur moyen sur la collection. Les topics se divisent en deux catégories principales : les CO (Content Only) et Les CAS (Content And Structure).

Pour chaque Topic, différents champs permettent d'expliciter le besoin de l'auteur : le champ Title donne la définition formelle de la requête, le champ Key-words contient un ensemble de mots-clés qui ont permis l'exploration du corpus avant la formulation définitive de la requête, et les champs Description et Narrative, explicités en langage naturel, indiquent les intentions de l'auteur.

La formulation des requêtes est étroitement liée à la tâche de recherche associée.

VII.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.

VII.4. les jugements de pertinence

L'évaluation de la pertinence des SRI passe par une première phase de validation des documents renvoyés par les SRI. Chaque élément/document est jugé à la main (par les participants) pour chaque requête, en utilisant le système de jugement en ligne [Piwowarski &Lalmas, 04]
En 2002, une première échelle de pertinence à deux dimensions a été proposée, basée sur le degré de pertinence et la couverture des éléments.

- ✓ **La pertinence**

Elle tient compte de la présence ou l'absence de l'information recherchée. Un élément est considéré comme pertinent même si l'information recherchée se trouve dans une petite partie de cet élément.
Quatre niveaux de pertinence sont distingués :

- **Non pertinent (Irrelevant) :** l'élément ne contient aucune information concernant le sujet de la requête.

- **Peu pertinent (Marginnaly relevant)** : l'élément contient une partie marginale du sujet de la requête.

- **Assez pertinent (Fairly relevant) :** l'élément contient une grande partie du sujet de la requête.

- **Très pertinent (Highly relevant) :** l'élément contient la réponse à la requête.

- ✓ **La couverture**

Elle s'intéresse au degré avec lequel l'élément traite toute l'information recherchée, elle est spécifique à l'évaluation des documents structurés. Quatre niveaux de couverture sont distingués :

- Pas de couverture (No coverage) : l'élément retourné ne contient pas de passage pertinent.

- **Trop large (Too large) :** l'élément retourné contient une proportion importante d'information non pertinente.

- **Trop petit (Too small) :** l'élément retourné couvre la majorité du sujet de la requête, mais il est trop petit pour être considéré comme une unité pertinente.

- **Exact (Exact coverage) :** l'élément retourné constitue exactement la bonne réponse.

Depuis la campagne d'évaluation 2003, les dimensions de pertinence et de couverture ont été remplacées par les dimensions d'exhaustivité et spécificité.

- ✓ **L'exhaustivité**

Décrit jusqu'à quel point l'élément discute du sujet de la requête. Quatre niveaux apparaissent

- **Non exhaustif:** l'élément ne traite pas du tout le sujet de la requête.

- **Faiblement exhaustif:** l'élément traite quelques aspects du sujet de la requête.

- **Moyennement exhaustif:** l'élément traite plusieurs aspects de la requête.

- **Totalement exhaustif:** l"élément traite exhaustivement (tout ou la majorité) le sujet de la requête.

- ✓ **La spécificité**

Décrit à quel point l'élément se focalise sur le sujet de la requête. Quatre niveaux sont également distingués :

- Non spécifique: le sujet de la requête n'est pas un thème de l'élément.

- Faiblement spécifique: le sujet de la requête est un thème mineur de l'élément.

- Moyennement spécifique: le sujet de la requête est un thème majeur de l'élément, celui-ci peut contenir quelques informations non pertinentes.

- Totalement spécifique: le sujet de la requête est le seul thème de l'élément.

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, 2007] : le gain cumule (xCG) et l'effort précision (ep).

- 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 :

$$( ) = \qquad (j)$$

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) :

$$( ) = \frac{(\mathrm{i})}{(\mathrm{i})}$$

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.

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

$$( ) = \frac{\text{e ideal}}{}$$

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.

VIII. Conclusion

L'aspect structurel des documents XML offre un double avantage, celui de permettre au utilisateur de spécifier exactement ce qu'il recherche, et celui de permettre au système de retourner des documents ou parties des documents les plus spécifiques et les plus exhaustifs.

Dans ce chapitre nous avons présenté quelque notions de bases d'XML, ainsi que la recherche dans les documents XML et les différents problèmes liés à la recherche et enfin la compagne d'évaluation INEX.

I. Introduction

Devant les limites des moteurs de recherche actuels, différentes méthodes ont été développées afin d'utiliser les liens dans le processus de recherche d'information.

Ces travaux ont montré que l'exploitation des liens est très efficace pour l'amélioration et la performance des résultats de la recherche.

Dans ce chapitre nous présentons l'exploitation des liens sur le web notamment l'algorithme Page Rank qui est utilisé par Google, HITS, Savoy…etc et l'exploitation des liens dans la recherche d'information structurée (RIS) en citant l'algorithme XRank.

II. Notion des liens

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-à-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 définit une connexion entre deux nœuds 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 (par opposition au livre qui se lit de manière 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.Exploitation des liens dans la recherche d'information non-structuré

Plusieurs algorithmes exploitant les liens ont été élaborées notamment dans le cadre du Web. Et les plus connues sont : les algorithmes PageRank et HITS.

III.1. l' algorithme PageRank

Cet algorithme [Brin & al, 1998] est basé sur la notion de propagation de popularité. Le principe est d'évaluer l'importance d'une page en fonction de chaque page pointant vers elle.

La propagation met en avant les pages qui jouent un rôle particulier dans le réseau des liens, avec l'hypothèse :

"*Une page référencée par un grand nombre de pages est une bonne page*".

Cette mesure est une distribution de probabilité sur les pages. Elle mesure en fait la probabilité PR pour un internaute navigant au hasard, d'atteindre une page donnée P. Cette probabilité est d'autant plus forte que le nombre de pages P1 à Pm qui réfèrent P est important.

PR est donc fonction de la somme des probabilités des pages qui référencent P.

Il faut aussi tenir compte du fait que les pages qui référencent P ont d'autres liens sortant vers d'autres pages que P. Il faut donc diviser cette probabilité par le nombre C(Pi) de liens sortant des pages Pi qui référencent P. Finalement. Finalement, il faut tenir compte du fait qu'un internaute peut arriver sur une page quelconque sans suivre de liens.

La formule proposée [Brin et al., 1998] tient alors compte de la probabilité d de suivre effectivement les liens.

La probabilité d'arriver à la page P sans suivre de liens est donc de (1-d). La formule de calcul du PageRank est alors la suivante :

$$PR(P) = (1\text{-}d) + d\ [(PR(P1)/C(P1) + ... + (PR(Pm)/C(Pm))]$$

Avec :

PR(P) **:** Page Rank de P.

C(Pi) **:** le nombre de liens sortants de la page Pi, i=1...m

d **:** un facteur compris entre 0 et 1, fixé en général à 0,85. Il représente la probabilité de suivre effectivement des liens pour atteindre la page P.

(1-d) **:** représente la probabilité d'atteindre la page P sans suivre de liens.

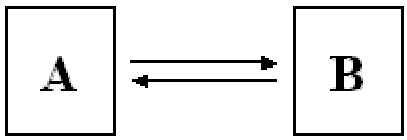
Le Page Rank de la page P se calcule à partir du Page Rank de toutes les pages Pi.

Calcul du Page Rank :

Exemple 1, site à 2 pages (liens internes)

Considérons l'exemple très simple d'un site contenant seulement 2 pages : une page A et une page B.

Dans la page A il y a un lien vers la page B.
Dans la page B il y a un lien vers la page A.
Prenons comme hypothèse de départ que chaque PR(B) = 0.



PR(A) = (1 - 0,85) + 0,85 (PR(B)/C(B)) or PR(B) = 0 et C(B) = 1
PR(A) = (1 - 0,85) + 0,85 * 0
PR(A) = 0,15

nous avons un PR(A) = 0,15 nous allons l'utiliser pour le calcul de PR(B)

PR(B) = (1 - 0,85) + 0,85 (PR(A)/C(A)) or PR(A) = 0,15 et C(A) = 1
PR(B) = (1 - 0,85) + 0,85 * 0,15/1
PR(B) = 0,2775

Maintenant que nous avons de vraies valeurs de PR calculées, recommençons le calcul (itération).

PR(A) = (1 - 0,85) + 0,85(PR(B)/C(B)) or PR(B) = 0,2775 et C(B) = 1
PR(A) = (1 - 0,85) + 0,85 * 0,2775
PR(A) = 0,385875

PR(B) = (1 - 0,85) + 0,85(PR(A)/C(A)) or PR(A) = 0,385875 et C(A) = 1
PR(B) = (1 - 0,85) + 0,85 * 0,385875
PR(B) = 0,47799375

Recommençons le calcul avec les nouvelles valeurs.

PR(A) = (1 - 0,85) + 0,85(PR(B)/C(B)) or PR(B) = 0,47799375 et C(B) = 1
PR(A) = (1 - 0,85) + 0,85 * 0,47799375
PR(A) = 0,5562946875

PR(B) = (1 - 0,85) + 0,85(PR(A)/C(A)) or PR(A) = 0,5562946875 et C(A) = 1

PR(B) = (1 - 0,85) + 0,85 * 0,5562946875

PR(B) = 0,622850484

Regardons comment évoluent les PR(A) et PR(B).

PR(A) = 0,15 puis **0,385875** puis **0,5562946875.**

PR(B) = 0,2775 puis **0,47799375** puis **0,622850484.**

Nous remarquons que les valeurs augmentent avec le nombre d'itérations. En fait, au bout d'un certain nombre d'itérations on tend vers les valeurs :

PR(A) = 1

PR(B) = 1

Si nous changeons la valeur de départ par exemple 60. Recommençons les calculs.

PR(A) = (1 - 0,85) + 0,85(PR(B)/C(B)) or PR(B) = 60 et C(B) = 1

PR(A) = (1 - 0,85) + 0,85 * 60

PR(A) = 51,15

PR(B) = (1 - 0,85) + 0,85(PR(A)/C(A)) or PR(A) = 51,15 et C(A) = 1

PR(B) = (1 - 0,85) + 0,85 * 51,15

PR(B) = 43,6275

Encore une fois

PR(A) = (1 - 0,85) + 0,85(PR(B)/C(B)) or PR(B) = 43,6275 et C(B) = 1

PR(A) = (1 - 0,85) + 0,85 * 43,6275

PR(A) = 37,233375

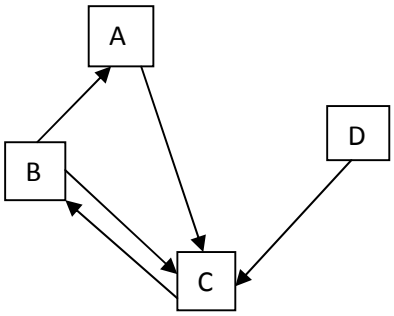
PR(B) = (1 - 0,85) + 0,85(PR(A)/C(A)) or PR(A) = 37,233375 et C(A) = 1

PR(B) = (1 - 0,85) + 0,85 * 37,233375

PR(B) = 31,79836875

Contrairement à la première remarque les valeurs diminuent à chaque itération, on en conclue que la moyenne des PageRank de tous les documents converge toujours vers 1.

Exemple2 :



Dans cet exemple, nous avons un site comprenant quatre pages, dont une ne recevant aucun lien (la page D). Le PR de cette page sera donc de 0.15, grâce au premier terme de la formule du PageRank (1 - d).

Après plusieurs itérations nous arriverons aux résultats suivants :

PR(A)=1.49, PR(B)=0.78, PR(C)=**1.58**, PR(D)=0.15

Le PageRank de la page C est le plus élevé vu que c'est la page qui contient plus de liens entrant. La page D par contre a le moins bon score, car elle n'a aucun lien entrant.

III.2. l'algorithme HITS

HITS (Hypertext Induced Topic Search), qui a été proposé par Jon Kleinberg
[Kleinberg, 1999] est un algorithme qui consiste à calculer les annuaires (*Hub*) et l'autorité (*Authority*) d'un document et ce pour classer les documents résultats par rapport à une requête.

–**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.

L'hypothèse est : "*Un document qui pointe vers beaucoup de bonnes autorités est un bon Hub, et un document pointé par beaucoup de bons Hubs est une bonne autorités* " [Kleinberg, 1999].

Dans la *Figure 5* nous présentons un exemple de documents *Hubs* et de documents *Autorités*.

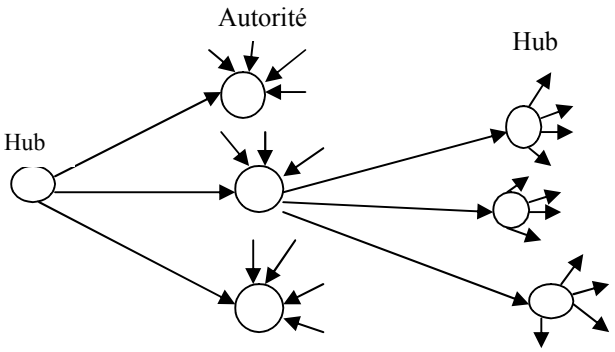


Figure 5: Exemple de page Hub et de page Autorité

Voici comment fonctionne l'algorithme HITS. Soit la requête d'un utilisateur q.

On fait d'abord une recherche classique et on construit une collection C de pages plus pertinentes. À partir de l'ensemble des pages trouvées C, on construit un plus grand ensemble S, en exploitant les liens, S contient :
- les pages qui contiennent des liens vers C.
- les pages qu'on trouve à partir d'un lien sur une page se trouvant dans C.

Une fois que C et S sont trouvés, on peut calculer la mesure « autorité » Xp ainsi que la mesure « hub » Yp pour chaque page p∈ S. La mesure « autorité » quantifie la qualité de la page en tant que page qui reçoit des liens, alors que la mesure « hub » quantifie le statut de la page en tant que page de liens.

On fixe d'abord Xp =1 et Yp =1 pour tout p ∈ S . On note q→p la condition « *q* contient un lien pointant vers *p* ». On effectue alors les opérations suivantes en séquence, les répétant

autant de fois que nécessaire, jusqu'à ce que les valeurs Xp et Yp convergent vers des valeurs *stables*.

$$x_p = \sum_{q \mapsto p} y_q \qquad y_p = \sum_{p \mapsto q} x$$

Nous aurons deux listes de documents : une classée en fonction des *Hubs* et l'autre classée en fonction des *Autorités*. Finalement, en utilisant une procédure de normalisation, la liste finale des documents classés sera construite.

Inconvénients de HITS

– les valeurs des poids peuvent être affectées facilement en ajoutant un lien supplémentaire. Un petit changement de lien peut causer donc un grand changement de classification des pages.

–le sous-ensemble initial contient plusieurs pages non pertinentes qui sont ajouté à l'ensemble final.

– L'algorithme HITS est inefficace au temps de la requête. En effet, la construction de la collection de base et le calcul des valeurs des poids consomment beaucoup de temps.

III.3. L'activation propagée

L'activation propagée [Savoy et Rasolofo, 2000] est une technique utilisée pour le calcul des scores des pages. Cette approche consiste à transmettre une fraction de score (notée λ) 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 des documents seront propagés selon les hyperliens sans tenir compte de leur orientation (entrants ou sortants).

Le nouveau score sera calculé selon l'équation suivante :

$$'(D_i) = RSV(D_i) + \lambda \sum \quad RSV\ (Dj)$$

Avec :

RSV′ (Di) : indique le nouveau score de la page Di.

RSV (Di) : Le score de la page Di au départ, basé sur la première liste.

RSV (Dj) : Le score du document Dj lors de l'itération.

λ **:** une constance destinée à atténuer la propagation.

Dj : sont les documents liés à Di.

Cette propagation peut être limitée aux r meilleurs voisins(les mieux classés par le moteur de recherche).

IV. Exploitation des liens dans la recherche d'information structurée

Peu de travaux ont été proposés pour l'exploitation des liens en recherche d'information dans des documents XML.

IV.1. XRank (Ranked Keyword Search over XML)

XRank [Lin et al ,2003] est l'un des premiers travaux qui propose une méthode permettant la prise en compte des liens XML pour le réordonnancement de la liste des résultats.

IV.1.1.L'architecture du système XRank

Les composants du système XRANK sont montrés dans la figure suivante :

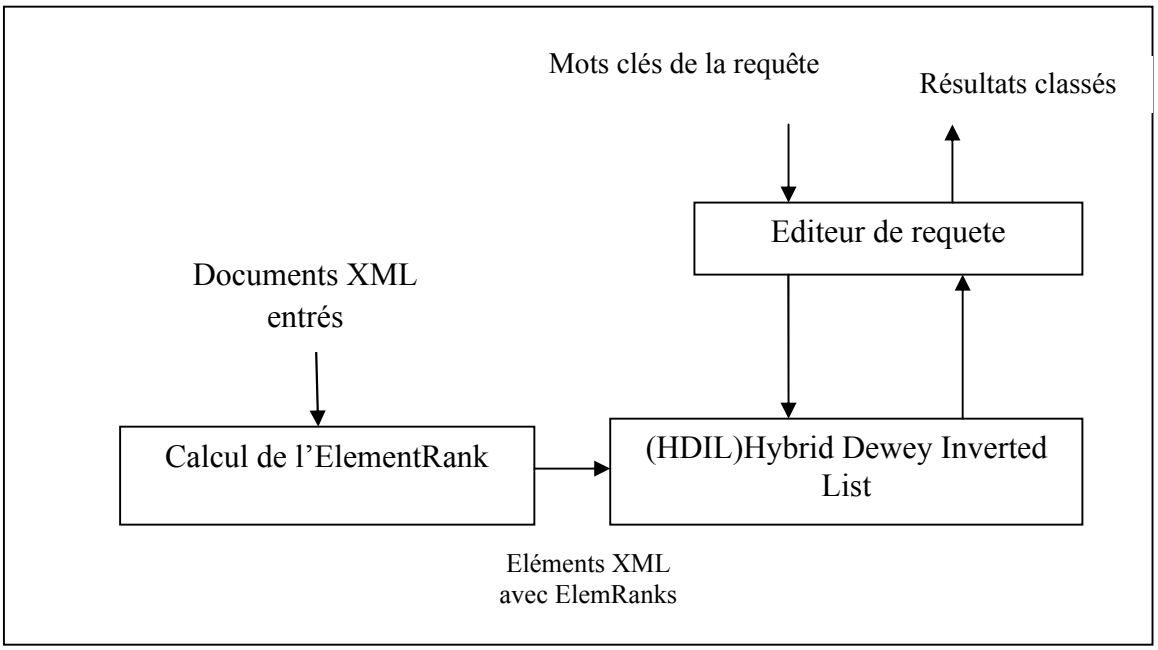


Figure 6 : L'architecture XRank [Lin et al ,2003]

Le module de calcul d'ElemRank calcule *l'ElemRank* de chaque Élément de document XML. *Les ElemRanks* sont alors combinés avec des informations ancêtres pour produire une structure d'index appelé HDIL (Hybrid Dewey Inverted List).

L'éditeur de requête évalue les requêtes en utilisant HDIL, et retourne les résultats classés.

IV.1.2. Calcul des ElemRanks

ElemRank est une mesure de l'importance objective d'un élément XML, et est calculée en fonction de la structure des liens hypertextes des documents XML. L'ElemRank est similaire au PageRank de Google sauf que l'ElemRank est défini à la granularité du document entier.

Le score d'un élément (ElemRank) est calculé en fonction de trois scores relatifs aux ensembles CE, HE, CE^{-1} , avec :

CE **:** Les liens hiérarchiques entre nœuds (lien ancêtre vers descendant).

HE **:** liens XLink entre les nœuds.

CE^{-1} **:** le même ensemble CE sauf que le sens des liens est inversé (lien descendant vers ancêtre).

Pour calculer l'ElemRank on utilise la formule suivante :

$$e(v) = d_1 \sum_{(\ ,\ )} \frac{(\ )}{(\ )} + d_2 \sum_{(\ ,\ )} \frac{(\ )}{(\ )} + d_3 \sum_{(\ ,\ )} \frac{(\ )}{(\ )} + \frac{}{*\ (\ )}$$

Ou :

e(i) : ElemRank de l'élément i.

$N_h(u)$: nombre d'hyperliens sortant de u

$N_d(v)$: nombre d'élément dans le document contenant l'élément v.

N_d : nombre totale de document.

Le XML a été conçus pour gérer les nouvelles caractéristiques des documents structurés XML, un de ses avantages est qu'il peut être utilisé pour interroger un ensemble de document structuré et non structuré

IV.2. DocRank

DocRank [Mattaoui M, 2009] est une autre adaptation de PageRank aux collections de documents XML. Les auteurs proposent d'assigner

un score à un élément en se basant sur le score de popularité du document qui le contient.

Le score de popularité du document est calculé au moment de l'indexation de la manière suivante:

$$\mathrm{DocRank}(D) = \frac{1-d}{\quad} + d \qquad \mathrm{DocRank}(i)$$

, ∈

Où:

- DocRank(D): est le score de popularité (DocRank) du document ;
- d : est un facteur d'amortissement;
- : est le nombre de documents dans la collection.
- links : représente l'ensemble des paires de liens (i, j) internes à la collection tel que le document i contient un lien vers le document j.

Une fois le DocRank de tous les documents de la collection calculé, le score d'un élément est calculé au moment de la requête avec la formule suivante:

$$\mathrm{NDocRank}(E_i) = \mathrm{ScoreInitial}(E_i) + (1- \ )\mathrm{DocRank}(D)$$

Où:

- NDocRank(E_i): est le score de l'élément ;
- D : est le document qui contient l'élément E_i ($E_i \in D$);
- DocRank(D) : est le score de popularité du document D;
- : est un paramètre qui permet de définir le degré de contribution des différents scores dans le score final;
- ScoreInitial(E_i) : représente le score initialement affecté par le système de recherche à l'élément E_i.

Les expérimentations effectuées dans [Mattaoui M, 2009] ont montré des améliorations de la qualité des résultats suite à l'utilisation de DocRank.

IV.3. Autres algorithmes

Benny K. applique l'algorithme HITS sur les résultats retournés, afin de filtrer les résultats retournés à l'utilisateur.

D'autres travaux qui ont été fait par Khairun N. F. et al [Khairun et al, 2008] , Jaap K. et Marijin K. [Jaap et al, 2008] utilisent les liens XML pour le réordonnancement des résultats renvoyés selon deux degrés : "local indegree" et "global indegree".

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.

V. Notre approche

Ce mémoire illustre une méthode mise au point par Mme Fellag [Fellag, 2012] basée sur la propagation de score qui consiste à 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. Cette méthode sera détaillée dans le chapitre suivant.

VI. Conclusion

De nombreux travaux ont été menés sur la problématique de la recherche d'information. Il s'agit en effet d'un problème crucial dans une époque ou la quantité d'information accessible en permanence devient extrêmement importante, mais dans le cas de documents structurés qui deviennent de plus en plus nombreux sur le web, peu de travaux exploitant les liens ont été réalisés, bien que l'analyse des liens a montré son intérêt dans la recherche d'information.

I. Introduction

Après avoir présenté un état de l'art de l'exploitation des liens dans les documents non-structurés et semi-structurés, nous entamons la description de notre méthode qui est basée sur la propagation de score et la prise en compte de l'information portée par le texte ancre des liens et la balise titre des documents pour améliorer la qualité des documents retournés.

II. Présentation de notre approche

Notre approche est une méthode mise au point par Fellag. S, qui consiste à transmettre une fraction de score de pertinence de chaque document retournée par une recherche classique vers ses voisins (les documents liés directement), puis de réordonner les éléments trouvés en prenant en compte cette fois les liens entre documents afin de retourner les éléments pertinents non retournés par la première recherche.

- **Exploitation des liens**

L'exploitation des liens dans notre approche est réalisée lors de la phase de recherche. Rappelons qu'en RIS, le granule d'information considéré n'est plus le document entier mais une partie de celui-ci : *l'élément*.

La recherche d'éléments réponses se fait en deux phases :

- la première phase consiste à retrouver les éléments réponses à la requête *q* donnée sans prise en compte des liens en interrogeant le corpus de document,

- 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.

Nous nous intéressons à la deuxième phase et montrons ainsi que la propagation de scores des documents pertinents retrouvés dans la phase initiale de recherche, ainsi que l'exploitation de l'information portée aussi bien par le lien que par le titre est importante pour une restitution pertinente des résultats.

L'intuition suivie par l'auteur est : "*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*

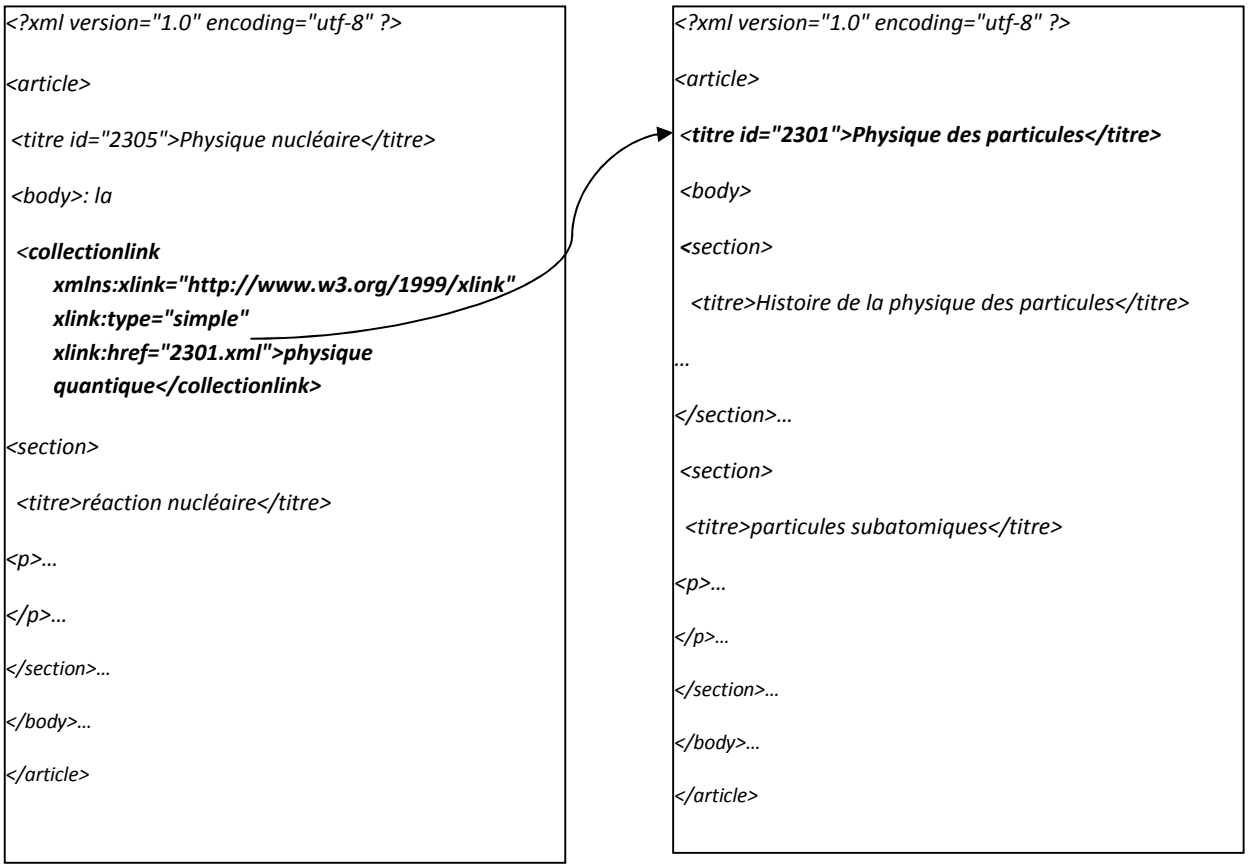
titre alors ce document ne peut être que pertinent pour la requête et de ce fait ces éléments le sont aussi ".

On peut éventuellement penser que si les termes de la requête figure dans la balise titre des documents alors ces derniers ont été forcément restitués lors de la première phase de recherche puisqu'ils sont pertinents, mais il ne s'agit pas uniquement de retrouver des documents pertinents mais aussi et surtout de réordonner les documents restitués lors de cette première phase de recherche en exploitant les liens qu'ils référencent.

- **Les liens utilisés**

Les liens XML qu'on exploite dans notre approche sont les liens extra documentaires, référencés par XLINK. Un exemple représentant ce type de lien est présenté ci-dessous :

L'exemple suivant Représente un lien Xlink entre deux extraits de documents XML, 2305.xml et 2301.xml



- Le document *2305.xml* (source) référence le document *2301.xml* (cible) avec un lien représenté par la balise *collectionlink*, qui a pour ancre *« physique quantique »*
- La balise titre du document *2301.xml* (cible) est composée de deux termes *« physiques de particules »*.

Ce sont ces deux paramètres, lien du document source et titre du document cible, en plus de la propagation de score du document source, que nous proposons d'exploiter dans notre approche, qui est étayée dans la section suivante.

III. Détails de la méthode

Avant de calculer le score de chaque document, son titre et ses textes ancres des liens ainsi que la requête seront indexés (voir le chapitre suivant).

Deux scores sont définis dans cette méthode : score $_{titre}$ et score $_{link}$

III.1. Le score du titre : Score $_{titre}$

Le titre du document est un élément caractéristique du contenu du document, il mesure l'information portée par la balise titre du document référencé.

Son score est calculé comme suit :

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

Ou :

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 $n <= m$

III.2. Le score du lien : Score $_{link}$

Mesure le contenu informationnel du lien, il est calculé par la formule suivante :

$$\mathbf{score}_{\mathbf{link}} = \frac{\sum_{i=1}^{n} \mathbf{tf}_{\mathbf{qi}}}{\sum_{j=1}^{m} \mathbf{tf}_{\mathbf{linkj}}}$$

tf_q : fréquence du terme de la requête figurant dans le lien

tf_{link} : fréquence du terme dans le lien

n : nombre de termes de la requête figurant dans le lien

m : nombre de termes total du lien et $n <= m$

Les liens exploités dans notre approche sont les liens *entrants* vers un document, l'intuition suivie : "*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*". De ce fait un document *i* pertinent qui référence un autre document *j* atteste que ce dernier est pertinent.

Dans ce cas nous avons utilisé le $score_{link}$ comme facteur de pondération du score du document source à propagé.

III.3. Le score du document :$Score_{doc}$

La formule qui permet de calculer le score du document est la suivante:

$$Score_{doc} = \alpha Score_{titre} + \sum_{i=1}^{k} \gamma_i score_{refi}$$

Avec :

$\mathbf{Score}_{\mathbf{refi}}$: le 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.

$\boldsymbol{k}$: le nombre de liens entrants vers le document.

$\boldsymbol{\gamma_i}$ **:** 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 :

γ prend ses valeurs dans l'intervalle]0,1].

$$\boldsymbol{\gamma = \mathbf{score}_{\mathbf{link}} + \lambda}$$

$$\lambda = \begin{cases} \neq 0 \quad \text{si} \quad \text{score}_{\text{link}} \in [0,1[\\ 0 \quad \text{sinon} \end{cases}$$

Le paramètre λ prend ses valeurs dans l'intervalle]0,0.1[

Le paramètre α 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.

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 :

$$\text{Score}_{\text{elt}} = \beta \text{Score}_{\text{eltinitial}} + (1-\beta)\text{Score}_{\text{doc}}$$

Le $\text{score}_{\text{eltinitial}}$ est le score initial de l'élément calculé dans la première phase de recherche.

Dans ce qui suit nous proposons un exemple pour mieux comprendre l'approche.

IV. Exemple illustratif

Nous montrons le fonctionnement de notre approche au travers d'un exemple illustratif, comme suit :

Soit *q* une requête contenant deux termes pondérés à 1 chacun, *q = {t1,1), (t2 ,1)}* et soit à considérer les liens référencés par l'élément *e1*.

La balise titre et le texte ancre de liens considérés contiennent au maximum deux termes de fréquence 1 chacun. Les différents calculs de scores effectués sur les documents référencés, sont représentés dans le tableau suivant:

Le titre T=ϕ (respectivement link L=ϕ*)* signifie que le titre (respectivement le link) ne contient aucun terme de la requête.

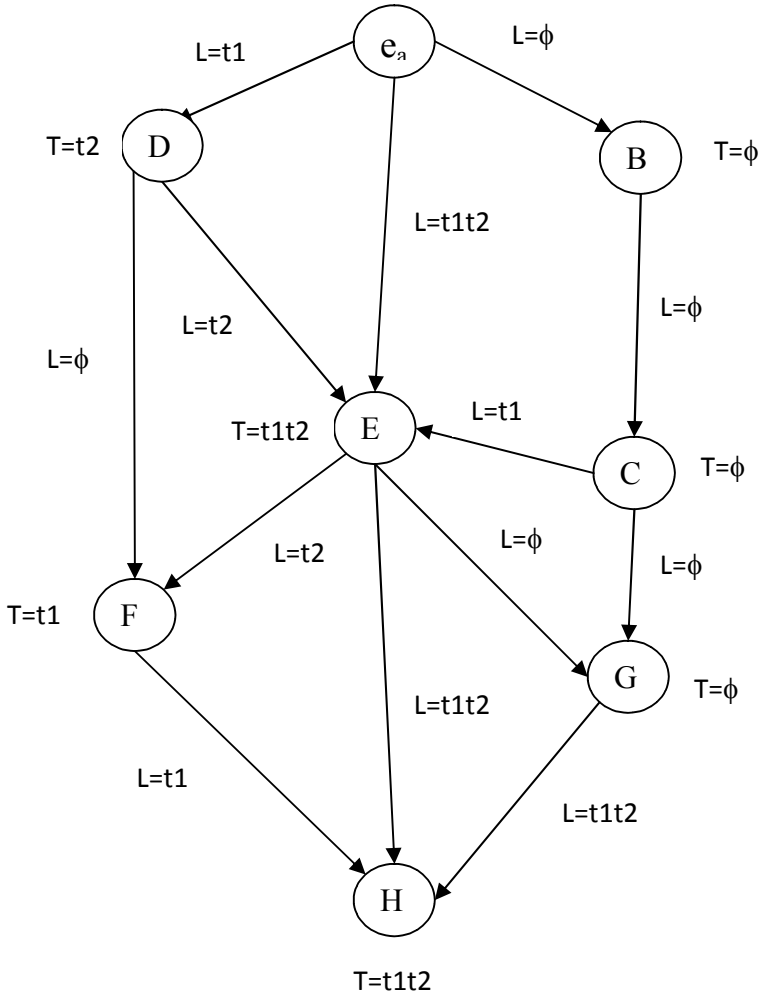


Figure7: Graphe représentant les liens entrants et sortants de documents XML liés.

Soit score (e_a)=0.7

Score (D) = α (0.5) + [(0.5 + λ) Score (e_a)]

Score (E) = α (1) + [(1 + λ) Score (e_a) + (0.5 + λ) Score (D)]

Score (F) = α (0.5) + [(0.5 + λ) Score (E) + (0+ λ) Score (D)]

Score (H) = α (1) + [(1 + λ) Score (E) + (0.5+ λ) Score (F)]

Score (G) = α (0) + [(0 + λ) Score (E) + (0+ λ) Score (C)]

Score (B) = α (0) + [(0 + λ) Score (G)]

Score (C) = α (0) + [(0 + λ) Score (B)]

Si on donnera des valeurs pour les paramètres α et λ on obtiendra :

Document	α=0.6 et λ=0.02	Les liens entrants
H	2.8530	F, E, G
E	1.6452	e_a , D, C
F	1.1687	D, E
e_a	0.7000	
D	0.6640	e_a
G	0.3290	E, C
B	0.0006	e_a
C	0.00003	B

Les valeurs de α et λ seront fixés après plusieurs expérimentations.

Nous constatons d'après le tableau que l'information portée par le lien et le titre très importante à savoir que B et C sont mal classés car ces deux la n'ayant aucune information concernant les termes de la requête aussi bien dans le link que dans le titre.

Les documents E et H se voient classés en tête de liste car leurs titres et leurs liens sont porteur d'information pertinente.

V.Conclusion

Dans ce chapitre nous avons proposé une méthode de RI pour la prise en compte des liens qui intègre le titre des documents, les liens entrants, ainsi que le texte ancre des liens pour le calcul des scores.

Dans le chapitre suivant nous nous intéressons à l'implémentation et à la validation de notre modèle à travers les expérimentations menées sur la collection de tests INEX 2006.

I. Introduction

Au cours des chapitres précédents nous avons décrit les différentes méthodes utilisées pour l'exploitation des liens afin de retourner l'information pertinente à l'utilisateur, comme nous avons présenté notre approche pour la prise en compte des liens dans les documents structurés.

Dans ce chapitre nous allons implémenter cette méthode, définir l'architecture générale de son système et ses différents modules ainsi que les différents outils utilisés. Nous décrivons également la collection de tests INEX 2006, et enfin, nous présentons les résultats de nos expérimentations sur cette collection.

II. Système d'exploitation des liens

II.1. l'architecture du Système

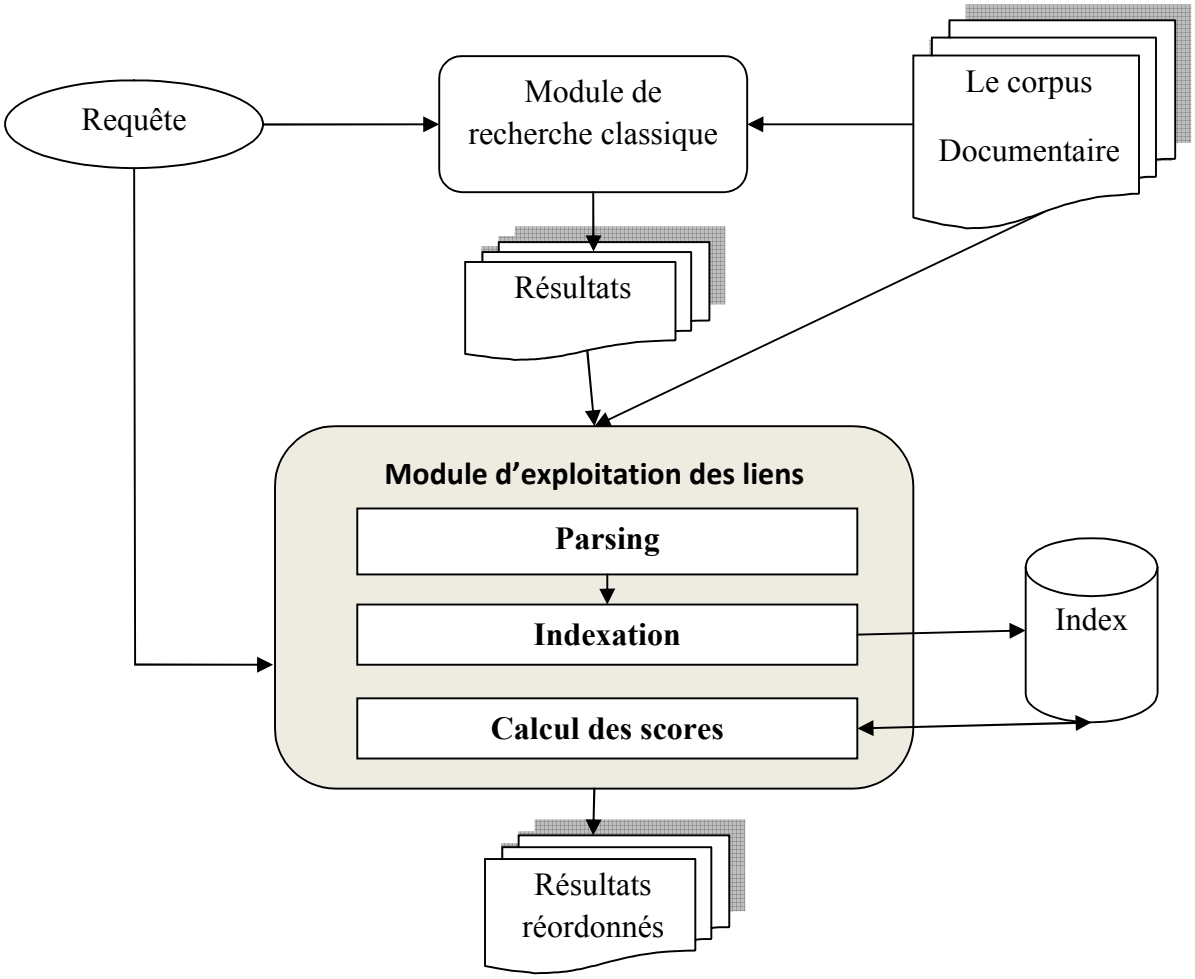


Figure8 : Architecture du système d'exploitation des liens

II.2. Le corpus documentaire

Ensemble de documents XML contenant des liens XLink simples.

Exemple d'un document du corpus : 624.xml

```
<?xml version="1.0" encoding="utf-8"?>
<article>
<name id="624">Alaska</name>
<conversionwarning>0</conversionwarning>
<body>:
<emph2>This article is about the U.S. state; for other meanings, see
<collectionlink  xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="2534014.xml">History of Alaska  </collectionlink>.
</emph2>
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="276733.xml">FrankMurkowski</collectionlink>
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="87469.xml">Juneau</collectionlink>
```

Le titre de chaque document est dans la balise <name>, dans notre exemple « Alaska » c'est le titre, les liens sont dans les balises <collectionlink> qui contiennent l'attribut « xlink:href » qui indique le document référencé, et la description du lien est le texte qui se trouve entre la balise ouvrante et fermente de <collectionlink>

II.3. Requêtes

Les requêtes seront utilisées dans le module de recherche classique puis réutilisées dans notre module pour calculer les scores. Et pour cela nous avons utilisé les requêtes CO (*Content Only*)

Exemple de requête utilisée :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE inex_topic SYSTEM "topic.dtd">

<inex_topic topic_id="289"  ct_no="2">

<title>emperor "Napoleon I" Polish</title>

<castitle>//*[about(., emperor "Napoleon I" Polish)]</castitle>

<description>I want to know everything about the emperor Napoleon I and Polish
people.</description>

<ontopic_keywords>"duchy of Warsaw", "Marie Laczynska", "countess Malewski",
"Prince Poniatowski", Eylau, Russia</ontopic_keywords>

</inex_topic>
```

II.4. module de recherche classique

Dans notre application, nous avons utilisé les résultats obtenus par le système Maxplanck qui est un système de l'université Maxplanck [Martin T & al, 2007] et qui a participé à INEX.

II.5. Résultats

Ensemble de documents XML triés par ordre décroissant des scores, retournés par la recherche classique.

II.6. module d'exploitation des liens

II.6.1. le parsing

Le parsing permet de parcourir un document XML à la recherche d'une information bien précise. Pour analyser la collection nous avons utilisé le parseur SAX pour sa nature événementielle *(début document, fin document, début élément, fin élément)*.

Notre but d'utilisé SAX est de récupérer :

- L'id et le titre des documents.
- Le numéro et de document lié et le texte ancre des liens de ce dernier.

II.6.2.l'indexation

Pour mettre en œuvre notre approche il est nécessaire d'indexer au préalable les titres des documents et texte ancre des liens ainsi que la requête, afin de garder uniquement les termes significatifs. Et pour cela nous avons utilisé une indexation à trois étapes :

- l'analyse lexicale : dans laquelle on a supprimé les espaces, la ponctuation, la casse, et la mise en page.
- Elimination des mots vides : on utilise une liste de mots vides nommé stop-list (antidictionnaire).
- Troncature à 7 de chaque terme : si le mot est supérieur à 7 prendre les 7 premiers caractères sinon prendre tout le mot.

Exemple d'indexation :

```
<?xml version="1.0" encoding="UTF-8"?>

<article>

<name id="309">An American in Paris</name>

<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="6104.xml">

symphonic

</collectionlink>

<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="31882.xml">

American

</collectionlink>

</article>
```

Après le parsing avec le parseur SAX, ce dernier nous retourne :

L'identifiant	**Le titre**	**Texte ancre**
309	An American in Paris	symphonic
		American

Après l'élimination des mots vides et la troncature à 7 on aura :

L'identifiant	Le titre	Texte ancre
309	America Paris	symphon
		America

II.6.3. le module de sauvegarde

Après l'indexation de tous les documents, nous avons crée une base de données pour sauvegarder le titre de chaque document, les liens entrants vers ce document ainsi que les liens sortants, les textes ancres et les scores des documents après exploitation des liens.

Nous avons également sauvegardé les requêtes et les résultats retournés par la première recherche. Les tables utilisées sont :

- La table document : contient trois colonnes, numéro du document, titre du document indexé, et le score du document.
- La table liens : contient les informations sur les liens de chaque document à savoir, numéro du document source du lien, le numéro du document destination et le texte ancre du lien indexé.
- La table resultat : contient les résultats des requêtes obtenus en utilisant le system Maxplanck (avant l'exploitation des liens).
- La table resultats_elt : contient les informations sur les résultats des requêtes obtenus en après l'exploitation des liens.
- La table topic : contient le numéro et le titre de la requête.

II.6.4. calcul des scores

Le calcule des scores s'effectue en consultant a chaque fois la base de donnée pour récupérer les titres et les textes ancre des documents.

Les formules présentées précédemment sont utilisées pour trouver, pour chaque document, un nouveau score de pertinence qui sera utilisé pour reclasser les documents avant de les restituer à l'utilisateur. Dans ce qui suit nous résumons l'algorithme de recherche basé sur les fonctions que nous avons présentée précédemment.

Données : *C* = {*d* / *d*= ensemble d'élément *e*} et *q* est la requête de l'utilisateur

***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* en utilisant un modèle de RI sans prise en compte des liens

1.2. Sauvegarder dans *E* par ordre décroissant du *scoreelt*

Cette étape est faite par le système Maxplanck, ce qui nous concerne c'est E.

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* jusqu'à 3 niveaux :

2.2.1. calculer son $score_{titre}$.

2.2.2. calculer $score_{link}$ de tous les liens *entrants* vers ce dernier.

2.2.3 calculer son $score_{doc}$

2.2.4 sauvegarder chaque document dans *D* par ordre décroissant des scores.

2.3.5 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 $score_{elt}$

3.2.2 sauvegarder dans *E'* par ordre décroissant des scores.

II.6.5. Résultats réordonnés

Ensemble d'éléments retournés par le module d'exploitation des liens, ordonné par ordre décroissant des scores.

Pour effectuer le reordonnancement nous avons conçu et réalisé une application dans laquelle toutes les formules présentées précédemment sont implémentés, cette application permet de fixer les différents paramètres utilisés à savoir (α, λ, β), le nombre de niveaux utilisés et la requête à exécuter.

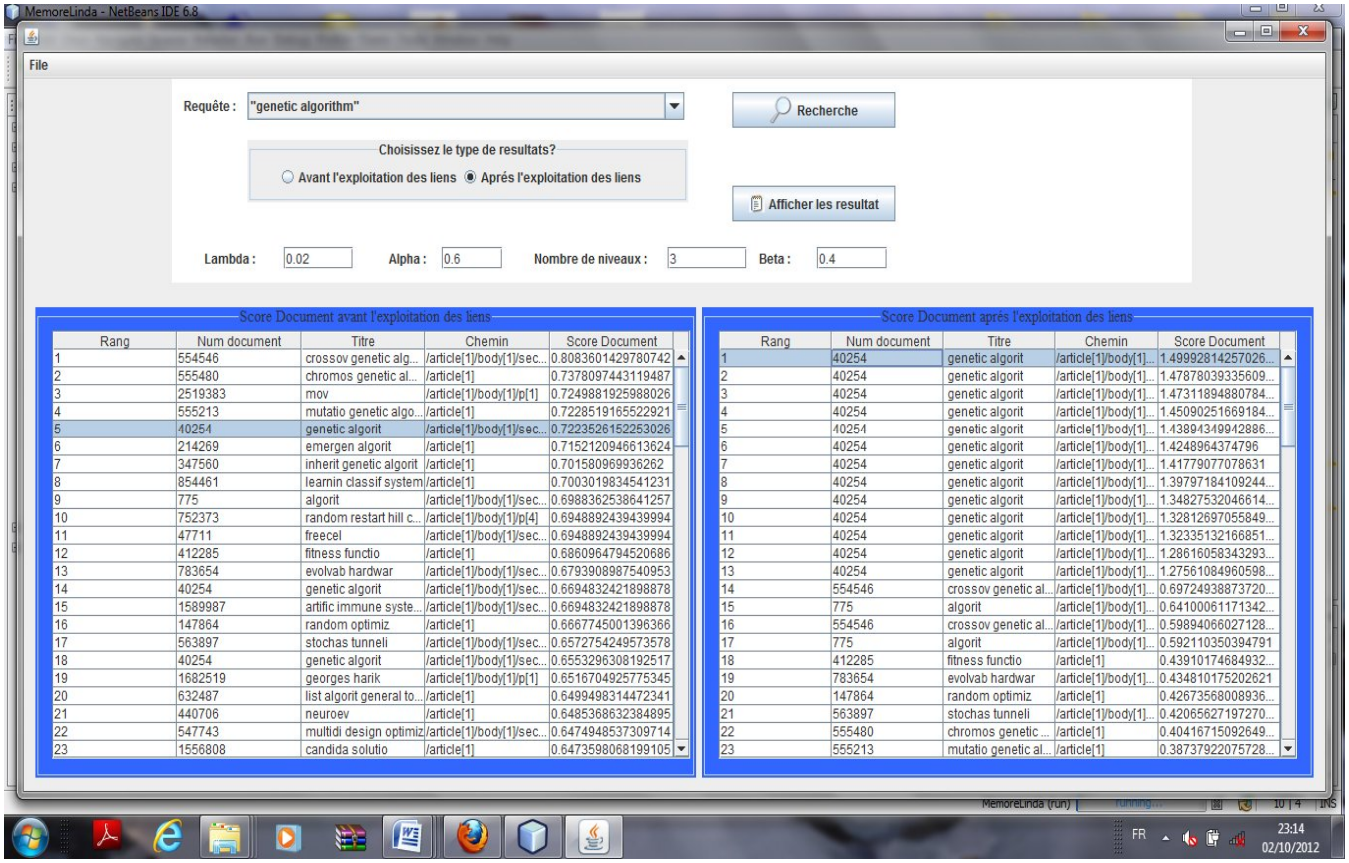


Figure9 : L'application de réordonnancement

Cette figure montre le nouvel ordonnancement des éléments retournés par le module classique.

Suite aux résultats obtenus après l'exploitation des liens, On remarque que plusieurs documents ont changé leurs positions par exemple le document 40254 qui était en 5ème position dans la recherche classique est remonté à la 1ére position dans les résultats de l'exploitation des liens et cela grâce à son titre qui contient les mêmes termes de la requête.

III. outils de développement

Pour réaliser notre application nous avons utilisé :

- ✓ Un langage de programmation qui est JAVA ;
- ✓ Un environnement de développement qui est NetBeans ;
- ✓ un système de gestion de base de données relationnelle et objet postegreSQL ;
- ✓ Le pilote JDBC

III.1. Le langage de programmation JAVA

Nous avons choisit pour l'implémentation de notre système d'exploitation des liens le langage de programmation JAVA, il a été mis au point par Sun Microsystems en 1991. C'est un langage de programmation à usage général, il est multi plate forme grâce à sa machine virtuelle appelée Java Virtual Machine (JVM) évolué et orienté objet de très haut niveau capable de s'exécuter sur n'importe quelle machine, parmi les caractéristiques les plus intéressantes de Java, on peut citer :

- ✓ Un langage orienté objet ;
- ✓ Indépendant vis-à-vis de la plate forme ;
- ✓ Avoir la capacité d'exécuter du code source extérieur de façon sécurisée permettant de compiler le code Java, c'est-à-dire de produire un exécutable capable de fonctionner hors de l'environnent Java ;
- ✓ La programmation peut se faire pour des exemples simples avec le compilateur Javac, mais pour avoir plus de confort il est préférable d'utiliser un environnement de développement intégré ou IDE, celui que nous avons utilisé est l'environnement NetBeans ;
- ✓ Doté d'un standard de bibliothèque de classes très riches comprenant la gestion des interfaces graphiques (fenêtres, boites de dialogues, contrôles, menus, graphismes), la programmation multithreads (multitâches), la gestion des exceptions, les accès aux fichiers et au réseau…. L'utilisation de ces bibliothèques facilite grandement la tache du programme lors de la construction d'applications complexes.

III.2. Présentation de NetBeans:

NetBeans est un environnement de développement intégré (EDI) pour java, placé en open source par Sun Microsoft en juin 2000 sous licence de CDDL (commun developpment and distribution licence). En plus de JAVA, NetBeans permet de supporter différents autres langages, comme Python, C, C++, XML, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi langages, éditeur graphique et de pages Web, …etc.).

Il est disponible sous Windows, Linux, Solaris,…etc.

L'installation de l'IDE NetBeans nécessite l'installation de la JDK (Java Developpement Kit) le Kit de développement Java compatible avec la version d'IDE.

Pour concevoir notre système nous avons utilisé la version NetBeans 6.8 et sont interface principale est donné dans la figure suivante :

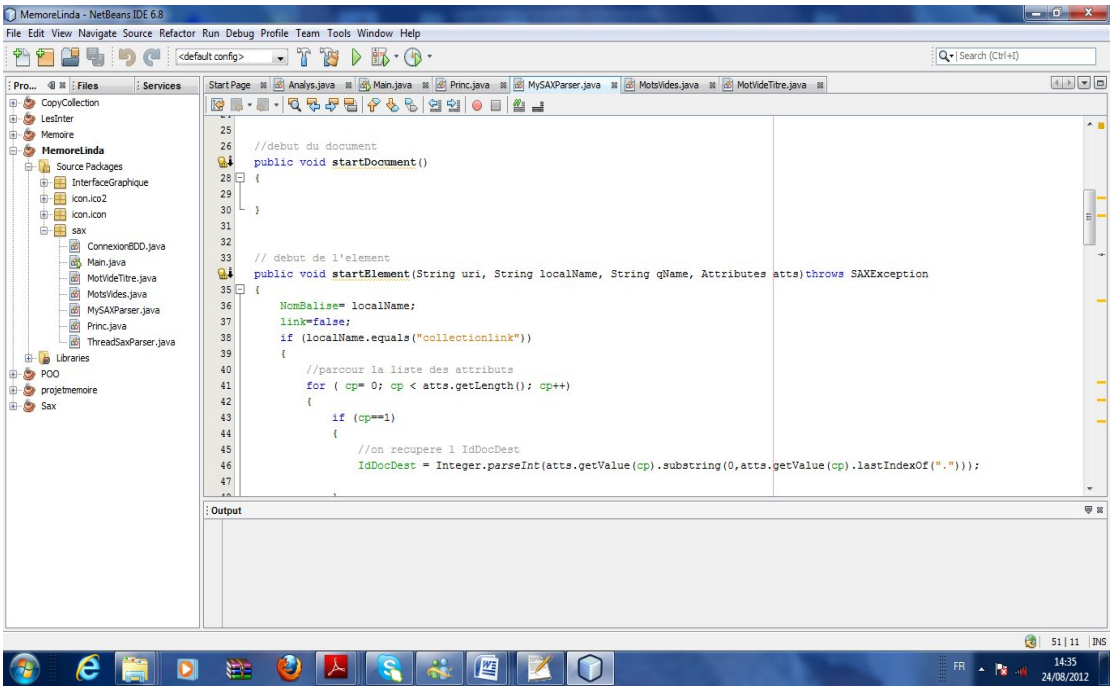


Figure 10 : l'interface principale de l'environnement NetBeans.

III.3. postegreSQL

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible sur Internet, il a été développé à l'université de Californie au département des sciences informatiques de Berkeley.

POSTGRES est à l'origine de nombreux concepts qui ne seront rendus disponibles au sein de systèmes de gestion de bases de données commerciales que bien plus tard.

postegreSQL est un descendant OpenSource du code original de Berkeley. Il supporte une grande partie du standard SQL tout en offrant de nombreuses fonctionnalités modernes :

- requête complexe.
- clés étrangère.
- Triggers.
- Intégrité des transactions

De plus postegreSQL est extensible par l'utilisateur de plusieurs façon. En ajoutant, exemple :

- De nouveau type de données ;
- De nouvelles fonctions ;
- De nouveaux operateurs....

III.4. Le pilote JDBC

La technologie JDBC (Java Data Base Connectivity) est une API(Application Programing Interface) fournit avec Java, définie par Sun microsystème afin de permettre l'accès aux bases de données, c'est-à-dire que JDBC constitue un ensemble de classes permettant de développer des applications capable de se connecter à des serveurs de bases de données(SGBD).

L'API a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe indépendamment du SGBD.

Il y a quatre(04) classes importantes, chacune correspond à une étape de l'accès aux données :

- ✓ **Driver Manager :** charger et configurer le driver de la base de données.
- ✓ **Connexion :** réalise la connexion et l'authentification à la base de données.
- ✓ **Statement :** contient la requête SQL et permet de la transmettre à la base de données.
- ✓ **ResultSet :** permet de parcourir les informations retournées par la base de données dans le cas de sélection de données.

IV. Expérimentations

Les expérimentations que nous avons menées ont été réalisées sur la collection INEX 2006.

IV.1.La collection INEX 2006

La collection de tests INEX 2006 a été construite à partir d'articles de l'encyclopédie libre *Wikipedia.* Elle contient prés de 659388 documents, et offre un ensemble de 126 requêtes pour les évaluations. Le tableau suivant montre les principales caractéristiques de cette collection:

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

Figure 11: Caractéristiques de la collection INEX 2006

IV.2. Tests et évaluation

IV.2.1. Environnement d'évaluation

Dans cette section nous allons décrire l'environnement et les conditions expérimentales qui vont nous permettre d'évaluer les différentes requêtes que nous avons exécuté.

IV.2.1.1. les requêtes utilisées :

Pour tester notre formule un ensemble de 8 requêtes illustrées dans la table ci-dessous est exécuté et voici un récapitulatifs de ces requêtes :

Numéro requête	**Titre requête**
289	emperor "Napoleon I"Polish
290	"genetic algorithm"
292	Italian Flemish painting Renaissance -French -German
293	wifi security encryptions
303	fractal applications -art
323	founder ikea
331	figure tulips
339	Toy Story

Nous n'avons pas pu faire varier les paramètres (α ,λ ,β) vu la collection gigantesque d'INEX 2006 et le temps d'exécution de chaque requête qui est très lent. Pour calculer les différents scores nous avons fixé ces valeurs comme suit:

$$\begin{cases} \alpha=0.6 \\ \lambda=0.02 \end{cases}$$

β=0.4

Nombre de niveaux=3

IV.2.1.2. Les mesures utilisées

Les jugements de pertinence pour chaque requête sont effectués par les différents participants selon deux dimensions : *l'exhaustivité* et *la spécificité*.

Dans notre expérimentation nous avons utilisé les mesures officielles d'INEX 2006 qui sont : les mesures XCG (XML Cumulative Gain), la précision moyenne interpolée (AiP[x]) et la quantité du texte souligné retournée. [Lalmas M & al,2006]

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 :

$$[\ ] = \frac{\sum\ (\ )}{\sum\ (\ )}$$

Le rappel à un rang r est définie par :

$$[\ ] = \frac{\sum\ (\ )}{(\ )}$$

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

$$[\ ] = \begin{matrix} \max_{|\ |}(\ [\ ] \wedge\ [\ ] \geq\ ) & \leq\ [|\ \ |] \\ 0 & \geq\ [|\ \ |] \end{matrix}$$

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.

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 :

$$= \frac{1}{} \qquad [\ ]$$

- **La mesure nxCG**

Pour evaluer notre système nous avons utilisé la mesure nxCG définie précédemment

$$(\ ) = \frac{(i)}{(i)}$$

Le principe consiste à calculer la valeur nxCG pour 4 rangs (10, 20, 30, 50) qui représente le nombre des premier résultats considérés. Cette valeur est calculée pour les deux systèmes (Maxplanck et notre modèle).

xCG représente la somme des scores des i premiers résultats obtenus les deux systèmes.

xCI représente la somme des i premiers scores affectés au jugement de pertinence de INEX 2006.

IV.2.1.3. Outil d'évaluation Evalj

EvalJ [Piwowarski, 2005] est une application entièrement écrite en java exploitable sous Linux via les lignes de commandes. 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 :

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

Avec :

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

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

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

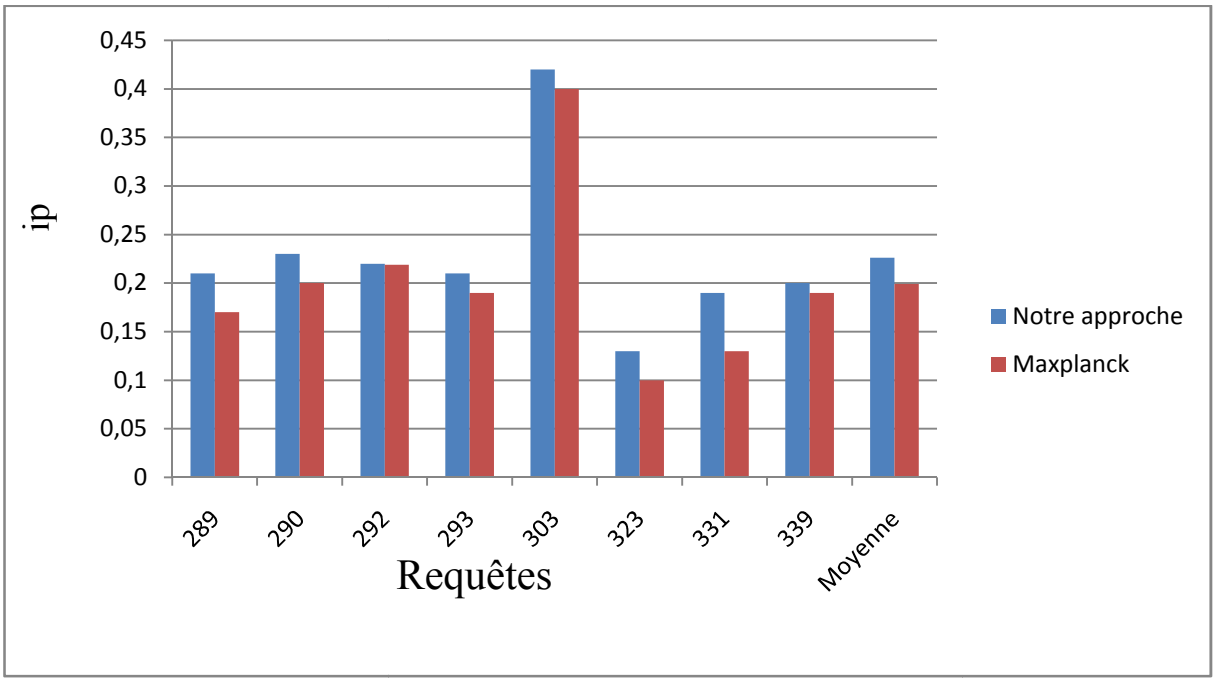
IV.2.2. les résultats de l'évaluation

Les résultats présentés dans cette section ont été obtenus à partir de l'évaluation des requêtes montrées dans le tableau précédant.

IV.2.2.1. Résultats de l'évaluation en calculant le nxCG

Voici un graphe représentatif des résultats de calcul de la mesure nxCG pour ces requêtes :

IV.2.2.2. Résultats de l'évaluation en utilisant Evalj:



Les résultats montrent une amélioration de l'ordre de 7,95% pour la moyenne calculé avec nxCG et de 11,94% pour la Précision interpolées.

A partir des résultats que nous avons obtenu en évaluant les 8 requête précédentes on remarque une amélioration au niveau de la précision pour chaque requête exécutée en exploitant le titre et l'information portée par le lien et cela en comparant avec les résultats obtenu avec le système Maxplanck qui n'exploite pas de liens.

On constate que la prise en compte des liens et le titre de document est imporante pour améliorer la qualité de réponse aux besoins des utilisateurs.

V. Conclusion

Nous avons décrit au cours de ce chapitre les différentes étapes que nous avons suivi pour implémenter notre méthode ainsi que les résultats obtenus par son évaluation. Nous avons pu constater une amélioration légère des valeurs de nxCG. Cependant cette approche devrait être

testée avec plusieurs requêtes pour pouvoir conclure de façon certaine sur l'efficacité de cette méthode.

Conclusion

Notre travail se situe dans le contexte de la recherche d'information, plus particulièrement la recherche d'information dans des documents semi structurés de type XML. Actuellement, la recherche d'information a évolué de l'accès à un document ou un ensemble de documents vers l'accès à des informations (des nœuds de documents XML) répondant à un intérêt particulier de l'utilisateur. Dans le cadre de cette nouvelle problématique de nouveaux modèles ont été proposés afin d'exploiter l'information structurelle contenue.

L'objectif de notre travail était d'implémenter une méthode proposé par Mme.Fellag pour l'exploitation des liens dans les documents XML, qui consistait à exploiter le contenu du texte ancre des liens et le titre des documents afin de calculer les différents scores, et ce à partir des éléments retournés par la recherche classique.

Les expérimentations effectuées sur la collection INEX 2006 ont montré que notre modèle donne de meilleurs résultats comparé aux résultats du système classique sans prise en compte des liens.

Perspectives :

Comme perspective envisagées à ce travail on trouve :

- La première perspective consiste à améliorer le temps d'exécution des requêtes pour ainsi réduire l'impact du nombre immense des liens contenus dans la collection dans le calcul des scores par l'utilisation du parallélisme.
- La deuxième perspective consiste à intégrer la sémantique dans le calcul des scores des textes ancres des liens et les titres des documents. En effet, un lien contenant par exemple le texte « *Informatique* » devrait être considéré comme pertinent pour une requête contenant le terme « *ordinateur* ».
- La dernière perspective consiste à faire varier les paramètres (α ,λ ,β).

[Amann, 2003] Bernd Amann, Cours *"XML et les bases de données : Introduction à la gestion de contenus Web et XML",* Module Données et services sur le Web 2003/2004.

[Balpe & al, 1995] Balpe, J., Lelu, A., and Saleh, I. Hypertextes et hypermédias : réalisations, outils et méthodes. Paris : Hermès, 1995

[Bourne &Anderson, 1979] C**.** Bourne, and B. Anderson, *"Dialog: Labworkbook. In Lockheed Information Systems"*, pages 640–644. PaloAlto, Californie (USA), 1979.

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

[Clark. j & S. Derose, 1999] XML Path Language (XPath) , version 1.0.Technical report, World Wide Web Consortium (W3C), W3C Recommendation, Novembre 1999.

[Cleverdon CW, 1970] Evaluation tests of information retrieval systems. Journal of Documentation. 1970:55–67

[Fagan, 1987] Fagan, Joel L. 1987. Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-syntactic methods, PhD thesis, Dept. of Computer Science, Cornell University, Sept. 1987.

[Fellag S, 2006] Fellag. S : « recherche d'information dans des documents semi-structurés XML » « Thèse de magister » Université Mouloud Mammeri de Tizi-Ouzou.

[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.

[Frakes & Baeza-yates, 1992] W. B. Frakes. Stemming Algorithms, pages 131–160. Frakes W B, Baeza-Yates R (eds) Prentice Hall, New jersey, 1992.

[Grosso et al., 2003] Grosso P., Maler E., Marsh J., Walsh N., « Xml pointer language (xpointer) », *Technical report, World Wide Web Consortium (W3C), W3C Recommendation*, 2003.

[Harter S, 1992] Psychological relevance and information science. Journal of the American Society for Information Science (JASIS). 1992:602–615.

[Hiemstra, 2002] D. Hiemstra, F. De Jong, *"Term specific smoothing for the language modeling approach to information retrieval: The importance of a query term"*, In proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 35-41, 2002.

[Hunter & al, 2001] D.Hunter, C.Cagle, D.Gibbons, N.Ozu, J.Pinnock, P.Spence, *"Initiation à XML avec trois études de cas détaillées"*, Edition Eyrolles 2001, traduit de l'anglais par F.Lemainque, L.Adam, C.Raspaud.

[Jaap & al, 2008] Jaap Kamps and MarjinKoolen, *The Importance of Link Evidence in Wikipedia*, In lecture
Notes in Computer Science, pp.270-282, Heidelberg, 2008.

[Järvelin & Kekäläinen, 2002] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems, 20(4): 422–446. 2002.

[Kent & al, 1955] Allen Kent, Madeline M. Berry, Luehrs, and J. W. Perry. Machine literature sear-ching viii, operational criteria for designing information retrieval systems. American Documentation, 6(2): 93–101, 1955. page 81.

[Khairun & al, 2008] Khairun Nisa Fhairun Nisa Fachry, Jaap Kamps, Marijn Koolen, and Junte Zhang, Using and Detecting Links in Wikipedia, In: Focused Access to XML Documents, pp.388-403, 2008.

[Kleinberg, 1999] J. Kleinberg: *Authoritative sources in a hyperlinked environment*.

Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 668-677. Washington.

[Lalmas M, 2006] Mounia Lalmas: INEX 2006 Evaluation Measures, Queen Mary, University of London, United Kingdom.

[Lelu & François, 1992] A.Lelu et C.François, *" Information retrieval based on a neural unsupervised extraction of thematic fussy clusters, neuro-nîmes 92 : les réseaux neuro mimétiques et leurs applications",* 2-6 novembre 1992, nîmes, France.

[Lin & al ,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.

[Luhn, 1957] Luhn, H. A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development 4, 1 (1957), 309–317.

[Maniez et al, 1991] Maniez, J., and de Grolier, E. A decade of research in classification. International Classification 18, 2 (1991), 73–77.

[Maron &Kuhns, 1960] M.Maron, J.Kuhns, *"On relevance, probabilistic indexing and information retrieval"*, Journal of the Association for computing Machinery, pages 216-244. 1960.

[Martin, 2000] Daniel Martin, *"Stockage et Interopérabilité en XML"* département ASI, 2000.*http://worldserver2.oleane.com/dmartin.*

[Martin T & al, 2007] Martin Theobald, Andreas Broschart, Ralf Schenkel, Silvana Solomon, and Gerhard Weikum. TopX – AdHoc and Feedback Tasks. Max-Planck-Institut fur Informatik Saarbrucken, Germany, 2007.

[Mattaoui M, 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.

[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*).

[Piwowarski &M. Lalmas, 04] B. Piwowarski and M. Lalmas. Interface pour l'évaluation de systèmes de recherche sur des documents XML. In Actes de CORIA 2004, Toulouse, France, pages 109–121, 2004.

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

[Ponte &Croft, 1998] Ponte, J. M., and Croft, W. B. A language modeling approach to information retrieval. research and development in information retrieval. In Proc. of the International ACM-SIGIR Conference (1998), Proc. Of the International ACM-SIGIR Conference, pp. 275–281.

[Porter, 1980] M.F. Porter. An algorithm for suffix stripping. pages 130–137, 1980.

[Ren & al.,1999] F. Ren, L. Fan, and J. Nie. Aak approach : How to acquire knowledge in an actual application system. IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, pages 136–140, 1999.

[Robertson & al., 1997] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pages 16–24. ACM Press, 1997.

[Robertson & al, 1998] «XML Query language (XQL) » Proceedings of W3C QL'98 (Query Languages 98).Massachusetts, 1998.

[Salton & al., 1968]: G. Salton, and M. Lesk. Computer Evaluation of Indexing and Text Processing. J. ACM 15(1): 8-36 (1968).

[Salton, 1970] Gerard Salton - Automatic processing of foreign language document –Journal of the American Society for Information Science, May 1970.

[Salton, 1971] Gerald . Salton, *"The SMART Information retrieval system"*, Prentice-Hall, Englewood Cliffs, 1971.

[Salton, 1971b] G. Salton. "A Comparison between manual and automatic indexing methods". Journal of the American Documentation, 20(1), pp. 6171, 1971.

[Salton & McGill, 1983] G.Salton et M.McGill, *" Introduction to modern information retrieval "*, McGraw-Hill, New York, 1983.

[Salton & al, 1983] Salton.G, E.A.Fox,H.Wu: Introduction to modern information retrieval, Mc Craw Hill international book company ISBNO-07-y665266-5, 1983.

[Salton, 1988] Salton, G. Syntactic approaches to automatic book indexing. In Proc. of the annual meeting on Association for Computational Linguistics (ACL) (1988), Department of Computer Science, Cornell University, Ithaca, New York, pp. 204–210.

[Savoy & Rasolofo, 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 & al., 1997] A. Singhal, M. Mitra, and C. Buckley. (1997). Learning routing queries in a query zone. In Proceedings of the 20th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Philadelphia, Pennsylvania, United States, July 27 - 31, 1997). N. J. Belkin, A. D. Narasimhalu, P. Willett, and W. Hersh, Eds. SIGIR '97. ACM Press, New York, NY, 25-32.

[Sparck, 1979] K. Sparck Jones. Experiments in relevance weighting of search terms. Inf. Process. Manage. 15(3): 133-144, 1979.

[Tittel, 2003] Ed Tittel, *" XML"*, EdiScience 2003, Schaum's, Traduit de l'américain par Patrick Fabre

[Urfist, 2004] Cours *"Problématique Générale de la Recherche d'Information"* URFIST Bretagne-Pays de Loire, Alexandre Serres, 2002

http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html

[Vieillard, 2000] Daniel VIEILLARD, "*Les Technologies associées à XML",* 2000.
http://daniel.veillard.com/Talks/200011XML/Overview.html

[W3C XQuery, 2003] M Fernandez et al, « *XQuery 1.0 and XPath 2.0 Data Model* », W3C Working Draft, Mai 2003. Disponible sur : *http://www.w3c.org/TR/xpath-datamodel/.*

[Xavier, 2007] Xavier Tannier. Indexation et Recherche d'Information " Recherche d'information semistructurée ". Université PARIS-SUD 11.

I. Les types de DTD (Définition de Type Document)

Une DTD peut être stockée dans deux endroits différents. Elle peut être incorporée au document XML (elle est alors dite interne), ou bien être un fichier à part (on parle alors de DTD *externe*). Cette dernière possibilité permet de la partager entre plusieurs documents XML.

I.1. La DTD interne

C'est-à-dire, on peut inclure son propre DTD dans le code source du fichier XML. une DTD interne suit la syntaxe suivante :

Syntaxe : < ! DOCTYPE élément-racine [déclaration des éléments]>

Exemple :

< ?xml version =''1.0''standalone=''yes'' ?>	DTD interne : fichier indépendant (standalone).
< !DOCTYPEparents[	Début du DTD avec parents comme élément racine.
< !ELEMENT parents(garçon, fille)>	Cet élément racine parents contiendra les sous éléments garçon et fille.
< !ELEMENT garçon(#PCDATA)>	#PCDATA indique au Parseur XML que l'élément garçon contient des données exprimées en chiffres ou en lettres.
< !ELEMENT fille(#PCDATA)>	Idem pour l'élément fille.
]>	Fin du DTD.
<parents> <garçon>David</garçon> <fille>Maria</fille> </parents>	Début de document XML. Fin du document XML

I.2. La DTD externe

Le document peut aussi faire référence à une DTD stockée dans une entité externe. L'avantage est alors la réutilisation possible de la DTD.

L'exemple précèdent serait alors :

```
< ?xml version="1.0" encoding="ISO-8859-1"standalone="No">
< !DOCTYPE parents SYSTEM "parents.dtd">
<parents>
<garcon>David</garçon>
<fille>Maria</fille>
</parents>
```

Le fichier de DTD externe (ici dans le même répertoire) "parents.dtd" est le suivant :

```
< !ELEMENT parents(garçon,fille)
< !ELEMENT garçon(#PCDATA)>
< !ELEMENT fille(#PCDATA)>
```

Le fichier de DTD peut être situé dans un autre répertoire au quel en fait référence, comme par exemple :

```
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML1.0 Strict//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

L'attribut standalone doit prendre la valeur « no » afin d'indiquer au processeur que des déclarations ou des parties de déclarations sont contenues dans des entités externe.

II. Les espaces de noms

La spécification des espaces de noms XML se trouve à l'adresse http://www.w3/org/TR/Rec-xml-names.

Les espaces de nom 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.

Un espace de nom est déclaré au moyen d'un attribut **xmlns** dont la valeur est une adresse

URI (**U**niform **R**esource **I**ndentifier), pointant sur l'espace de noms qui peut être préfixé comme suit : **xmlns :prefix.** Notons que lorsque l'attribut xmlns n'a pas de préfixe, il établit ainsi un espace de nom implicite pour l'élément considéré ainsi que pour ses sous éléments.

Exemple :

Un document qui contient les informations d'un produit :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<produit>
    <numero>p7325</numero>
    <nom>Alienware M18x</nom>
    <quantite>112</quantite>
    <prix>1999,00</prix>
</produit>
```

Un document qui contient les informations sur le client :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<client>

    <numero>c16212</numero>
    <nom>Dupont, François</nom>

</client>
```

Les deux balises *nom et numero* proposent des caractéristiques différentes selon qu'elles soient utilisées dans le document produit ou le document client. Si l'on désire créer un document commande unique décrivant la commande d'un produit par un client, un moyen permettant d'identifier les balises est nécessaire, c'est l'objectif des espaces de noms.

Le document XML unique la commande est alors :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<commande xmlns:produit="http://mon-entreprise.com/XML/produit"
xmlns:client="http://mon-entreprise.com/XML/client">
<produit:numero>p7325</produit:numero>
<produit:nom>Alienware M18x</produit:nom>
<produit:quantite>1</produit:quantite>
<produit:prix>1999,00</produit:prix>
<client:numero>c16212</client:numero>
<client:nom>Dupont, François</client:nom>
```

III. XML Schema

La DTD (Document type definition) est l'outil de validation courant pour les documents XML [Tittel, 2003], elle contient des informations de structure et de typage des données de ces documents. Bien que la DTD possède plusieurs avantages, elle a aussi certains inconvénients comme : avoir une syntaxe différente de celle d'XML et un typage non détaillée des données. Pour résoudre ces problèmes, le W3C propose en recommandation depuis 2001 le langage XML schema.

Xml schema présente de nombreuses améliorations par rapport aux DTD, notamment une plus grande flexibilité et un typage plus important et plus détaillé des données (string, décimal, boolean,…).

Xml schéma est défini formellement par trois documents conservés par le W3C, *Xml Schema Part 0* qui décrit l'utilisation de Xml schema, *Xml schema Part 1* qui concerne les structures (Structure recommandation) et *Xml schema part 2* relatifs aux types de données (Datatype recommandation).

Exemple :

Soit le fragment de DTD suivant :

```
< !ATTLlST 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.

IV. XSL (Extensible Stylesheet Language)

Le XSL 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 :

IV.1. XSLT

XSLT (*eXtensible Stylesheet Language Transformations*) défini dans la recommandation XSL du W3C est un langage XML qui permet de passer d'un document dans un format XML à un document dans un autre format texte (XML, XHTML/HTML, CSV …). Il est apparu en deux versions, la première se base sur XPath1, et la deuxième sur XPath2.

L'algorithme de transformation de XSLT se résume à parcourir l'arbre du document XML grâce à des expressions XPath, et à exécuter des blocs d'instructions sur les éléments ainsi retournés pour construire le nouveau document. Pour ce faire, XSLT offre plusieurs instructions de bases présentes dans l'espace de noms «*http://www.w3.org/1999/XSL/Transform* ». D'autres instructions provenant d'autres langages peuvent aussi être utilisées en spécifiant l'espace de noms d'où elles proviennent.

IV.2. XSL-FO

XSL-FO (*eXtensible Stylesheet Language - Formatting Objects*) fait aussi partie de la recommandation XSL du W3C. Son rôle est d'offrir un format de présentation, comme HTML, mais servant à générer des formats plus complexes, comme les formats PDF.

Il vient ainsi compléter XSLT qui n'aurait pas pu être utilisé comme tel pour des formats manipulant des données binaires.

Un document XSL-FO est composé d'instructions de mise en page et d'instructions d'affichage contenues dans l'espace de noms « *http://www.w3.org/1999/XSL/Format* ». Les instructions de mise en page concernent le dimensionnement des pages du document et le découpage des pages en régions. Les instructions d'affichages concernent le contenu des pages et permettent de définir le contenu de chaque région d'une page. Grâce à ce système, XSL-FO permet de gérer facilement les régions statiques comme les pieds de pages et la numérotation des pages.

Un exemple de XML+XSL :

Soit le fichier XML suivant contenant une compilation MP3 que l'on voudrait représenter sous forme d'un tableau trié par ordre alphabétique du nom des artistes.

Ce fichier fait référence à un fichier XSL en ligne 2 de nom *xsldemo*

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsldemo.xsl"?>
<compilation>
<mp3>
    <titre>Foule sentimentale</titre>
    <artiste>Alain Souchon</artiste>
 </mp3>
<mp3>
    <titre>Solaar pleure</titre>
    <artiste>MC Solaar</artiste>
</mp3>
<mp3>
    <titre>Chambre avec vue</titre>
    <artiste>Henri Salvador</artiste>
</mp3>
</compilation>
```

Le fichier *xsldemo* qui va transformer le document XML en tableau est le suivant :

```
<?xml version='1.0'?>
<xsl:stylesheet
version="1.0"xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table border="1" cellspacing="0" cellpadding="3">
<tr bgcolor="#FFFF00">
  <td>Artiste</td>
  <td>Titre</td>
</tr>
<xsl:for-each select="compilation/mp3" >
<tr>
   <td><xsl:value-of select="artiste"/></td>
   <td><xsl:value-of select="titre"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Artiste	Titre
Alain Souchon	Foule sentimentale
MC Solaar	Solaar pleure
Henri Salvador	Chambre avec vue

XML + XSL constitue une puissante solution d'interopérabilité [Martin, 2000].

V.les langages d'interrogation de documents XML :

L'interrogation des corpus de documents XML diffère de l'interrogation habituelle en RI,
Afin de permettre une prise en compte optimale des informations de contexte XML (textuelle et structurelle), un langage de requêtes est utilisé, l'interface utilisateur transforme la requête en un langage interne .Dans sa forme générale la recherche se fait en trois phases qui sont la formulation de requête utilisateur, la recherche dans la collection de documents, et en fin la représentation des résultats.
De nombreux langages de requêtes ont été proposés dans la littérature, les plus utilisés sont :

V.1. XPath (XML Path Language)

Xpath est un langage de spécification d'expressions régulières décrivant un chemin ou une famille de chemins dans une arborescence XML. Xpath 1.0 est une recommandation du W3C [Clark. j & S. Derose, 1999].

Il s'agit d'un langage permettant de sélectionner des sous-arbres d'un document XML. Il possède une syntaxe simple et non ambigüe et implémente les types usuels (chaines, nombres, booléens, variables, fonctions). Il permet aussi de manipuler des nœuds et des ensembles de nœuds. Xpath est utilisé par Xpointer et XSLT.

Les requêtes XPath se basent sur la représentation sous forme d'arbre du document XML afin d'y sélectionner des éléments en se basant sur différents critères. Une requête XPath est exprimée sous forme d'un *chemin de localisation* lui-même constitué de *pas de localisations* séparés par des slashs « / ». Chaque pas de localisation est composé d'un *axe*, un *type d'élément* et des *prédicats*.
L'axe indique la direction dans laquelle se déplacer dans le document XML. Parmi les axes XPath on trouve:

- « *ancestor* » : tous les éléments ancêtres de l'élément courant (parent de l'élément, parent du parent de l'élément …);
- « *attribut* » : tous les attributs de l'élément courant. Le symbole « @ » peut être utilisé comme abréviation pour cet axe;
- « *child* » : tous les enfants de l'élément courant. Si l'axe n'est pas spécifié, *child* est considéré par défaut;
- « *descendant* » : tous les éléments descendants de l'élément en cours (fils de l'élément, fils du fils de l'élément…);
- « *descendant-or-self* » : tous les éléments descendants de l'élément en cours, plus l'élément en cours. La chaine de caractères « // » est une abréviation pour cet axe;
- « *parent* » : l'élément parent de l'élément courant. Utiliser deux points « .. » revient à utiliser cet axe;
- « *self* » : l'élément courant. Le point « . » est une abréviation pour cet axe.

Le test de l'élément permet de sélectionner des éléments suivant leur nom ou leur type. Il s'agit donc du nom d'un élément ou d'une expression plus générale tel que:

- « *text()* » : pour sélectionner les éléments textes;
- « *node()* » : pour sélectionner tout type d'élément;

Les prédicats sont écrits entre crochets (« [» et «] ») et permettent de filtrer les éléments sélectionnés par l'axe et le test de l'élément. Si la valeur du prédicat est numérique, elle indique la position de l'élément à sélectionner. Par exemple, « titre[5] » va sélectionner le cinquième titre dans l'élément courant. En revanche, si le prédicat n'est pas numérique, il est évalué comme une expression logique qui est testée avec chaque élément, lequel est rejeté s'il ne la satisfait pas. Par exemple, « chapitre[@titre='RI'] » va sélectionner tous les éléments «chapitre » dans l'élément courant, qui ont en plus un attribut « titre » contenant le texte

« RI ».

V.2. Le langage XQuery

XQuery (pour *XML Query Langage*) [W3C XQuery, 2003] 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* (prononcé *flower*), 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, on préfère la première syntaxe car elle est plus simple et plus lisible contrairement à XQueryX, qui est destinée à des manipulations formelles par des programmes (éventuellement eux-mêmes écrits en XQuery).