République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Conduite de Projets Informatique**

Présenté par
Samira LANDRI
Rahma ALILOUCHE
Dahmane AIT CHEKDHIDH

Thème

Implémentation et évaluation d'un modèle de langue pour la Recherche d'Information dans les documents XML

Mémoire soutenu publiquement le 09/10/ 2016 devant le jury composé de :

Président :	**M^{me} G.SINI**
Encadreur :	**M^{me} Samia FELLAG**
Examinateur :	**M^{lle} Y.YESLI**
Examinateur :	**M^{lle} S. AIT-ADDA**

Dédicaces

Àla mémoire de mon père, que mon dieu l'accueille dans son paradis.

Àla lumière de ma vie, ma chère maman.

Àla source de ma force, mes frères (yassine, ali , idris).

À le secret de ma joies, mes sœurs (assia, mariem et ferroudja).

Àtoute ma famille paternelle et maternelle.

Àtous mes adorables amis (es).

Rahma

Je dédie ce travail à mon père et ma mère,

Mes frères et mes sœurs,

Mes amis,

Dahmane

C'est avec un immense plaisir que j'écris ces lignes, et que je tiens à exprimer ma profonde gratitude à mes très chers parents qui n'ont cessé de me soutenir, et de croire en moi
A ma chère sœur ainsi que son mari pour leur amour et soutien
A mes Chers frères qui ont été toujours a mes cotés
A mes belles sœurs
A mon cher mari et ma belle famille
A mes chers neveux sarrah nouara, yassmine, djamila, mouhammed, anais

Samira

Remerciements

Nous tenons à remercier en premier lieu notre promotrice ***Mme FELLAG****, pour son encadrement continu, et pour les remarques constructives qu'elle nous' a fournies ainsi que pour la confiance qu'elle nous a donnée, et sa patience durant toute la période de travail. Nous la remercie également pour ses qualités humaines, son hospitalité et son soutien qui a su nous faire profiter de ses nombreuses connaissances et compétences dans le domaine de la recherche d'information.*

Nous remercions très sincèrement les membres du jury : ***Mme SINI.G*** *et* ***Mlle AIT_ADDA.S, Mlle YESLI.Y*** *qui nous font l'honneur d'accepter d'évaluer ce travail. Nous les remercions aussi pour leurs enseignements et orientations, ainsi pour leurs conseils durant tous ces années d'études.*

Nos vifs remerciements s'adressent à ***Mlle BELKACEM.T*** *pour son aide à tout moment, et sa contribution immédiate malgré l'éloignement.*

Et enfin, merci à tout ceux qui' ont contribué de prés ou de loin à la réalisation de ce travail.

SOMMAIRE

CHAPITRE I : LA RECHERCHE D'INFORMATION ET STRUCTUREE ET

CHAPITRE II : LE MODELE DE LANGUE

II .1Partie (1) : Modèle de langue pour la RI.

II.2 Partie (2) : Modèle de langue pour la RI-Structurée

CHAPITRE III : UN MODELE DE LANGUE POUR LA RIS DANS LES DOCUMENTS XML

CHAPITRE IV : IMPLEMENTATION ET RESULTATS

Annexe 1 : Les concepts de base de la RI.

Annexe 2 : Les technologies de XML.

Annexe 3 :.L’environnement de développement.

La Table des Figures :

La Liste des Tableaux :

Introduction Générale

Introduction Générale :

Tous les domaines scientifiques connaissent d'énormes essors grâce aux efforts et recherches de l'être humain. Dans ce contexte « la recherche d'information » abrégée en RI est une science apparue avec l'apparition des ordinateurs en 1940, qui évolue avec le développement de web et ayant pour objectif de retourner l'information la plus pertinente dans une grande masse d'information stocké sous format numérique qui satisfait un besoin utilisateur.

Plusieurs caractéristiques des documents ont été explorées pour estimer la pertinence de ce derniers en rapport avec le besoin en information de l'utilisateur, comme : le contenu, la longueur…etc. Ce qui a mené la communauté de RI à développer des systèmes de recherche d'information (SRI) pour manipuler ces nouvelles caractéristique.

Cependant, le développement du web a fait émerger des nouveaux formats de documents traités par les SRI, ces derniers sont passés de la RI classique ne traitant que des documents textes dits plats, à la RI structurée ou semi-structurée (RIS) traitant les documents à formats variables et structurés, tel que le format XML, qui est traité par des SRI dit structurés (SRIS).

Grâce à la structure des documents XML, l'évaluation de la pertinence d'un document se rapportant à une requête utilisateur ne se définit plus au niveau du document entier comme c'était le cas dans la RI standard ; mais plutôt au niveau de l'élément ou d'un ensemble d'éléments. On parle alors de recherche ciblée.

La structure de ces documents, nous permet aussi l'interrogation de ces derniers, de deux manières. En effet, on peut définir des requêtes portant sur le contenu (CO Content Only) ou sur le contenu et la structure (CAS Content and structure). Plusieurs modèles de la RI classique ont été adaptés à la RIS pour les documents XML.

Dans ce travail, nous nous sommes intéressés au modèle de langue qui est une variante du modèle probabiliste de base. Un modèle de langue représente la similarité entre un document est une requête du fait que le modèle du document puisse être généré par celui de la requête.

Dans ce cadre, notre contribution porte principalement sur l'implémentation et l'évaluation d'un modèle de langue proposé par [Belkacem, 2015] pour les documents semi-structurés XML. Dans [Belkacem, 2015] plusieurs modèles de langue ont été proposés, mais un seul a été testé. Notre travail consistait à implémenter, et tester le modèle avec les formules [3.2 et 5.2] et le comparer à celui testé et évalué dans [Belkacem, 2015] avec les formules [3.1 et.5.1].

Contexte de travail

À la différence de la recherche d'information classique qui retourne comme résultats un document dans sa globalité à une requête utilisateur, la recherche d'information dans les documents XML (RIS) retourne un élément ou un ensemble d'éléments de ce document.

Notre contribution porte principalement sur l'implémentation et l'évaluation d'un modèle de langue proposé par [Belkacem, 2015] pour les documents semi-structurés XML, afin de tester la similarité éléments/requête. Nos tests intégrons deux formules proposé par [Belkacem, 2015], la première pour estimer la probabilité d'avoir un élément comme résultat pour une requête $P(e_j)$ notée dans le rapport [3.2] et la deuxième calcule la probabilité de vraisemblance maximale d'un terme t_i dans le modèle de l'élément e_j ainsi notée [5.2]. Ce modèle est testé sur la base de documents INEX, concerne principalement le développement de moteurs de recherche documentaire pour la RIS, et les requêtes de test est de type CO (Content Only).

Chapitre I
La Recherche d'Information et la Structure

Introduction :

La recherche d'Information (RI) est une discipline de l'informatique qui date des années 40, est qui s'est développée avec l'évolution de l'internet. La RI étudié la manière de retrouver des informations dans un corpus (base documentaire), qui consiste à la recherche, le stockage, le filtrage d'information répondant aux besoins utilisateur et de satisfaire ce dernier.
L'évolution du web a favorisé l'apparition et la progression de nouveaux formats de documents, du document plat (texte) au document structuré et semi-structuré (contenu multimédia).
La diversité d'information contenue dans les documents structurés et semi-structurés a poussé le word wide webconsortium (3WC) a standardisé un langage structuré XML, qui est considérer comme étant un intermédiaire entre le SGML et le HTML.
Dans cette partie nous allons étudier la recherche d'information généralement et la RIS (recherche d'information structuré) particulièrement en présentant les notions de base de la recherche d'information appliquée aux documents semi-structurée XML.

1. La recherche d'information classique :

1.1 Définition :

Une des premières définitions de la RI a été données par Salton ‘‘*La recherche d'information est un domaine qui a pour objectif la représentation, l'analyse, l'organisation, le stockage et l'accès à l'information''.*
D'après AFNOR *‘‘La recherche d'information (RI) est un ensemble de méthodes et 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'informations en réponse à une question sur un sujet précis''.*[L'afnor, 1979]
La RI est une discipline qui s'inspecte à la recherche, l'acquisition, la collecte, le stockage et l'organisation d'informations répondant aux besoins des utilisateurs en utilisant un « Système de Recherche d'Information ».
Donc la RI dans le cas générale comporte deux acteurs principaux :

- Utilisateur qui a besoin d'information.
- Une masse d'information préalable.

Du point de vue utilisateur, *c'est l'aspect d'acquisition des unités informatives les plus précises à sa requête.* [Sauvagnat, 2006].

1.2 Système de recherche d'information (SRI) :

Un système de recherche d'information intègre un ensemble de techniques et de processus permettant de sélectionner dans une collection de documents ceux qui sont susceptibles de répondre au besoin d'un utilisateur. Ces processus permettent :

- la représentation des informations et des besoins,
- l'interrogation, la recherche et la sélection des informations pertinentes répondant aux besoins d'un utilisateur.

L'objectif principal d'un SRI classique est de retourner pour toute requête utilisateur les documents les plus pertinents pour satisfaire son besoin en information.

Dans un SRI l'utilisateur exprime son besoin sous forme de requête et le SRI renvoie l'ensemble des documents qui sont similaires à cette requête. Pour cela, il met en œuvre deux processus : un processus d'indexation et un processus d'appariement.

La figure suivante, présente les différentes étapes d'un processus de recherche d'information :

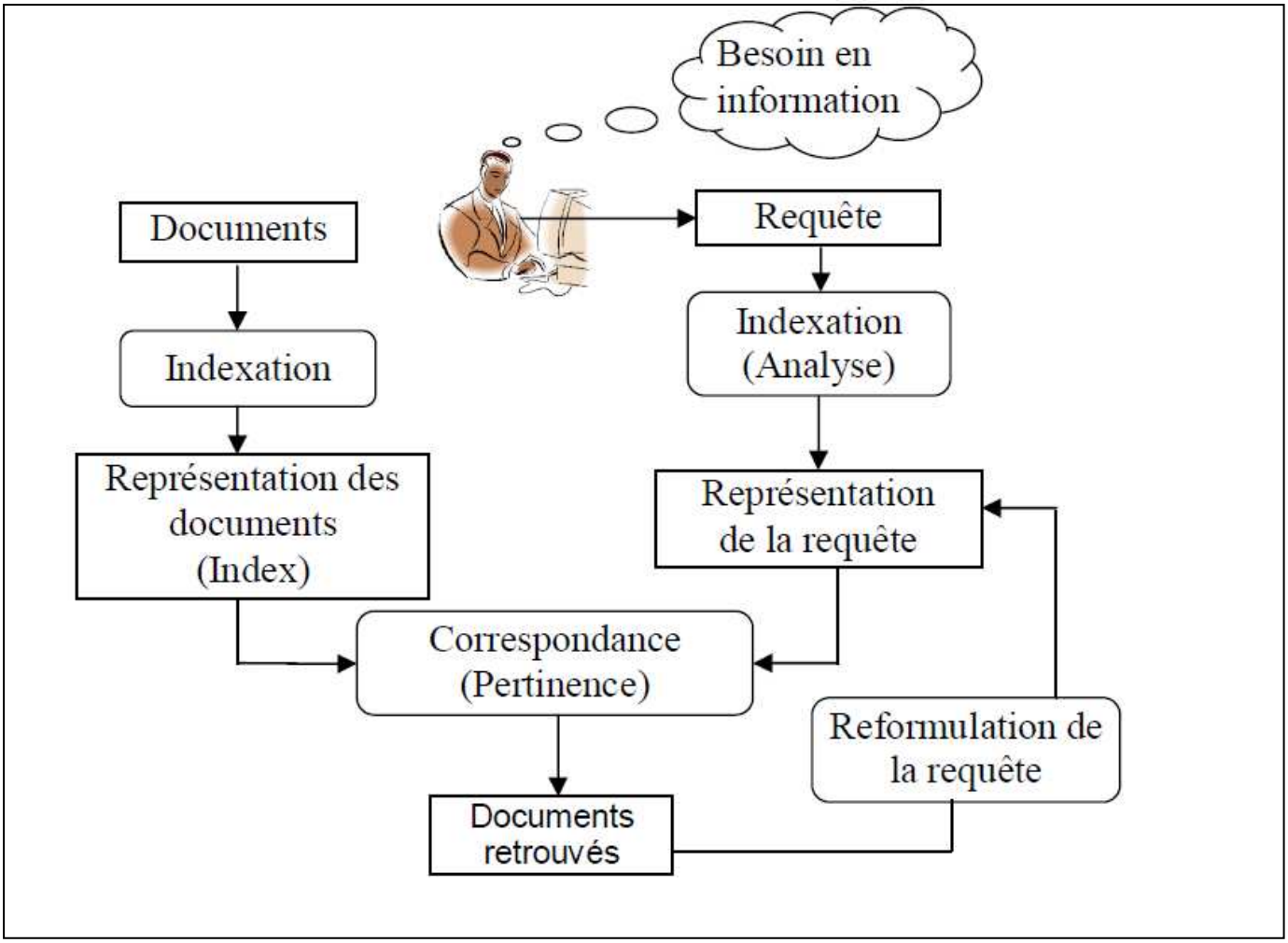


Figure I.1: le processus en U [HAMMACHE, 2014]

2. La RI structurée et semi-structurée :

2.1 Définition de la structure et du contenu :

2.1.1 ***La structure*** **:** considérée comme un élément intéressant qui permet de bien affiner la représentation des documents, On distingue deux types de structures d'un document semi-structuré :

- ***La structure logique*** **:** est spécifiée par l'auteur, elle lui permet de décomposer et d'organiser le document pour mieux exprimer ses idées, elle décrit la nature des éléments et les relations hiérarchique qui les relient.
- ***La structure physique*** **:** un document est composé d'un ensemble de pages, elles-mêmes composées d'un en-tête, de lignes, de figures, etc. La structure physique dépend de l'environnement de présentation du document, Elle traite les propriétés typographiques (police, taille, couleur …) associée à chaque élément du texte,

2.1.2 .Le contenu **:** d'un document structuré désigne le contenu textuel ou multimédia compris entre des balises, c'est l'élément qui porte l'information. Il est représenté sous la forme d'un ensemble de fragments, comme par exemple des paragraphes, des figures ou des images.

3. Présentation de XML :

XML (eXtensible Markup Language) : est un langage de balises extensible, définit comme un nouveau standard de représentation et d'échange de données sur internet, a été développé en 1996 ensuite recommandé par W3C en février 1998.

XML se qualifié d'extensibilité car il permet l'utilisation des nouvelles balises qui possèdent une structure qui n'est pas imposée par une norme, un standard ou une recommandation, Mais une structure que le créateur peut déterminer lui-même au moment de la conception du document, contrairement en langage HTML qui a des balise prédéfinies.

Le **XML** est un métalangage, c'est-à-dire un langage pour écrire d'autre langage, permet de séparation stricte entre contenu et présentation :

• Simplicité, universalité et extensibilité.
• Format texte avec gestion des caractères spéciaux.
• Structuration forte.
• Modèles de documents (DTD et Schémas XML).

XML apparaît comme un format d'échange de données universel. Il garantit à ses utilisateurs l'indépendance de leurs documents de toute technologie propriétaire, [Allorge, 2000].

3.1. Structure d'un document XML :

- ***Un Prologue :*** contient deux déclarations facultatives mais cela ne coûte rien de les ajouter.
- ***Un arbre d'éléments*:** représente le véritable contenu du document XML.
- ***des commentaires* :** informations supplémentaires servant de documentation de programmeurs.

La figure sous-dessus est un exemple d'un document semi-structuré au format XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<repertoire>
   <!-- John DOE -->
   <personne sexe="masculin">
      <nom>DOE</nom>
      <prenom>John</prenom>
      <telephones>
         <telephone type="fixe">01 02 03 04 05</telephone>
         <telephone type="portable">06 07 08 09 10</telephone>
      </telephones>
   </personne>
</repertoire>
```

Figure I.2 : exemple de document XML.

3.1.1 Le prologue : est constitué de la déclaration XML <?xml version="1.0" encoding="UTF-8"?> Permet de spécifier explicitement l'encodage Par défaut, il s'agit de l'UTF-8.Et la seconde déclaration c'est la déclaration du type du document (DTD) *Document Type Definition* ou *Définition de type de document* permet d établir les règles de définition de la structure documentaire XML, elle précise en particulier quels éléments peut contenir chacun des éléments. Cette déclaration de type peut prendre plusieurs formes suivant que la définition du type est interne, c'est-à-dire incluse dans le document ou externe. Lorsqu'une DTD est associée à un document XML, on dit qu'il est ***valide.***

3.1.2 Un arbre d'éléments: C'est la partie essentielle d'un document XML constitué d'une hiérarchie d'éléments comportant éventuellement des attributs, ce sont les sources d'information en XML.

- ***L'élément :*** est un nœud commence par une balise ouvrante <élément> et se termine par une balise fermante </élément>.

<repertoire> ceci est un élément</repertoire>

Un élément vide est représenté par : **<nom_élément/>**

- ***Les Attributs* :** Quand on veut préciser ce que contient un élément, on peut placer un attribut sur la balise ouvrante. La valeur d'un attribut est une chaîne de caractères encadrés par des apostrophes (' ') ou par des guillemets (« »).

Dans notre exemple est : **<telephone type= "fixe">01 02 03 04 05</telephone>.**

3.1.3 Les commentaires : Une chaine de caractère qui sont délimités par les chaînes de caractères '**<!--**' et '**-->**' comme en HTML. Le contenu d'un commentaire ne sera pas interprété.

Le commentaire de notre exemple : **< !--john DOE -->**

La chaîne de caractères **« -- »** ne peut apparaître dans le contenu d'un commentaire (un interpréteur XML considère que ce signal annonce la fin d'un commentaire).

L'arbre d'éléments suivant nous montre une représentation graphique de document XML précédent :

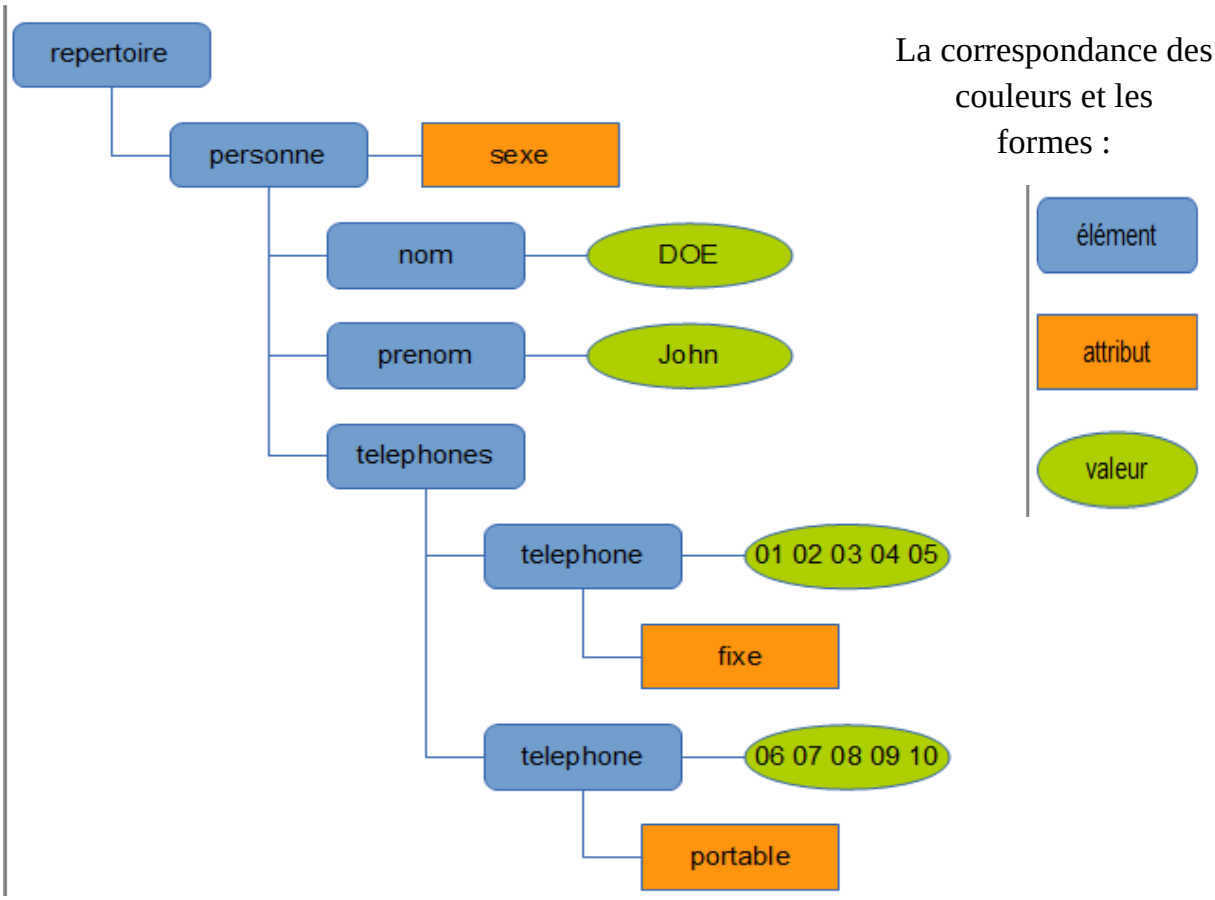


Figure I.3 : l'arbre d'éléments de documents XML.

3.2. Analyseur du document XML :

Le XML n'est pas un langage de programmation, c'est uniquement un langage de structuration et de représentation de données, pour cela, il existe deux types d'analyseur ou parser en anglais (le parser**DOM** (Document Object Model) orienté hiérarchie et le parser SAX (Simple API for **XML)** orienté événement), permettent de récupérer dans une structure XML, des balises**,** leur contenu**,** leurs attributs et de les rendre accessibles.

3.2.1. ***DOM: ''****Document Object Model''* est une API (**A**pplication ***P****rogramming* ***I****nterface ou interface de programmation d'application*) constitue un objet en mémoire de la totalité d'un document XML. Cette API permet l'accès direct à tous les nœuds de l'arbre (éléments, texte, attributs), pour les lire, ou les modifier. Le modèle DOM est une recommandation du W3C depuis octobre 1998.

3.2.2. ***SAX :*** Afin de parcourir un document XML, l'API SAX fournit une interface événementielle, ce qui signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML en envoyant à l'application qui manipule le document les événements comme l'ouverture de balise, la fermeture de balise, contenu textuel…etc.

SAX utilise, pour son fonctionnement, des méthodes de gestion des événements telle que :

- **startDocument() :** méthode invoquée lors de la rencontre d'un nouveaux document.
- **startElement() :** méthodes invoquée lors de la rencontre d'une balise ouvrante.
- **endElement() :** méthodes invoquée lors de la rencontre d'une balise fermante.

L'avantage de SAX est sa faible consommation en mémoire car il n'est pas nécessaire d'avoir une représentation complète des données en mémoire.

4. Problématiques liées à la RI structurée :

Dans le cadre de la recherche d'information dans les documents XML, une nouvelle unité d'information potentielle présente qui se caractérise par la présence d'une structure organisant leurs contenus textuels. Donc l'information dans un document n'est plus un simple paragraphe (granule d'information indivisible) comme c'est le cas dans la RI traditionnelle, mais une donnée complexe regroupant l'information textuelle et son emplacement ou encore son niveau (des contraintes structurelles -les balises -), c'est ce qu'on appelle *la granularité de l'information*. La problématique engendrée, en RIS, par ce type de document (XML) est liée à la nature des traitements des données et informations retournées aux utilisateurs répondant à leurs requêtes.

Le but des SRI est alors d'identifier des parties (un élément ou un ensemble d'éléments) du document les plus pertinentes en implémentant :

- L'unité d'information pertinente
- L'interrogation des documents.
- L'indexation.

L'unité d'information pertinente: La question, Comment, la recherche d'information structurée évalue le degré de similarité entre la requête et les unités d'information (nœuds de l'arbre XML) et attribue à ces dernières un score de pertinence.

L'interrogation des documents : Elle consiste dans sa forme générale en trois phases qui sont la formulation de la requête de l'utilisateur, la recherche dans le document (XML) et la présentation des résultats.

L'indexation : ce processus en RI traditionnelle se base sur la pondération des termes qui représentent le contenu du document tout entier, mais dans la RIS, les informations (texte) dans un document, ont des contraintes structurelle (Balises) ce qui pose les questions suivante: *Que doit-on indexer de la structure du document ? Et comment relier cette structure au contenu texte du document ?*

4.1. Pondération des termes* :** Les mesures classiques ***Idf et ***tf*** (voir formules dans l'annexe 1) ont été adaptées pour la RIS (document XML), car, en RIS, s'ajoute l'importance du terme au niveau de l'élément qui le contient. Plusieurs travaux ont été présentés pour répondre à la problématique d'évaluation de la similarité entre une partie d'un document et une requête utilisateur, en étudiant la structure des documents : les nœuds, les éléments, les feuilles….

Ief (Inverted element frequency) est une adaptation d' ***Idf*** à la nouvelle granularité d'information traitée dans les documents XML, qui exprime l'importance d'un terme dans l'élément (nœud) de document et sa formule de calcul est :

$$Ief = log\left(\frac{Ne}{ne}\right) + \alpha \qquad \text{avec} \quad 0.5 \leq \alpha \leq 1$$

Ief : fréquence inverse d'éléments.
Ne : Nombre total de nœuds feuilles dans le document.
ne : nombre de nœuds feuilles contenant le terme ***i*** dans le document.

***4.2. L'indexation des documents* :** [Luk et al.,89], ont proposé des possibilités pou l'indexation de la structure.

- L'indexation par les champs où chaque terme est associe au nom de la balise dans laquelle il apparait. Cette méthode permet de savoir dans quel contexte l'information textuelle est énoncée. Il est également possible d'indexer selon les chemins, c'est-a-dire d'indiquer le chemin d'accès logique plutôt que le nom de la balise. Ceci permet de faciliter et d'accélérer l'accès dans certains cas, mais apporte peu d'information supplémentaire (sauf dans le cas ou beaucoup de balises de même nom sont présentes à divers niveaux de la structure).
- L'indexation par les arbres, où un identifiant unique est attribué à chaque nœud (élément) du graphe représentant le document XML. Les termes sont donc associés à cet identifiant, ce qui permet de localiser de façon précise l'endroit où ces termes sont apparus et de retrouver les relations hiérarchiques entre les éléments.

.

5. Evaluation des SRI structurés :

L'évaluation des systèmes de recherche d'information structurée vise principalement à mesurer la capacité d'un système à retrouver des éléments à la fois exhaustifs et spécifiques au sujet de la requête. L'exhaustivité décrit à quel point l'élément traite du sujet de la requête et la spécificité décrit à quel point l'élément se focalise sur le sujet de la requête.

Les métriques d'évaluation sont loin d'être stabilisées et consensuelles pour la RI structurée [Pehcevski, 2006],[Tannier, 2006]. Leur évolution est indissociable de la campagne d'évaluation INEX. INEX (INitiative for the Evaluation of XML), est une campagne d'évaluation qui a pour but l'évaluation et la comparaison des systèmes de recherche d'information dans les collections XML.

5.1. La compagne INEX:

INEX (INitiative for the Evolution of XML retrieval): une compagne d'évaluation des systèmes de recherche d'information structurée, qui a eu lieu depuis avril 2002, elle utilise un langage de requête NEXI (Narrowed Extend XPath I) et propose plusieurs taches telle que ad-hoc, la tâche multimédia et, tache Relevance feedback et la tache hétérogène…etc.

INEX fournit une collection de documents, un ensemble de requêtes et des jugements de pertinence. La collection de test proposée par la compagne INEX ne cesse d'évoluer, afin d'améliorer la qualité de l'évaluation .INEX à utiliser une collection d'article appartenant à la revue scientifique (IEEE Computer Society), entre 2002 et 2004, cette collection est de tille d'environs 500 MO.

En 2005, la collection a été étendue pour atteindre 12000 articles issus de 18 revues. A partir de 2006, la collection a été complété par des documents extraits de

l'encyclopédie en ligne « Wikipédia », cette collection est composée de 659388 documents d'environs 6 GO.

En 2007, il Ya eu un changement concernant la permission de retourner des parties arbitraires d'un document et l'évaluation de la pertinence d'un texte comme réponse à diverses requêtes.

En 2012, INEX et CLEF ont procédé à l'étude différents aspects d'accès à l'information ciblée et à l'établissement des tâches de base dans l'évaluation des systèmes de recherche d'information à base des collections appropriés.

5.2. Les éléments d'évaluation :

***5.2.1. Les Requêtes* :** (Topic) INEX introduit deux types de requêtes :

- ✓ ***les requêtes de type CO*** (Content Only) : relatives au contenu. Elles décrivent uniquement le contenu souhaité des éléments XML recherchés.
- ✓ ***Les requêtes de type CAS*** (Content And Structure) : ces requêtes combinent le contenu et la structure, mais elles contiennent plus de contraintes sur la structure des documents XML. Pour chaque Topic ont constituées de quatre parties : titre, description, narration et mots-clés.

5.2.2. Les Tâches : Différentes tâches ont été proposées depuis le début de la compagne.

Les participations se sont focalisées sur la tâche principale « *ad hoc* ».

- ✓ *La tâche ad-hoc* **:** L'objectif de cette tâche est évaluer la capacité des systèmes à répondre à des requêtes sur le contenu textuel seul ou des requêtes combinant contenu textuel et contraintes sur la structure. Cette principale tâche est fragmentée en trois sous tâches distinctes :
  - *a) La tâche CO :*(Content Only task) pour le traitement des requêtes uniquement sur le contenu.
  - *b) La tâche SCAS* **:** (Strict Content And Structure task) pour le traitement des requêtes mêlant contenu et structure. L'évaluation VCAS se base sur les requêtes CAS.
  - *c) La tâche VCAS* : (Vague Content And Structure task) Répondre aux requêtes CAS de manière vague. ie : avec des granules satisfaisant globalement les requêtes.

6. Evaluation :

L'évaluation de la performance des différents systèmes de RI XML proposés par les participants a tout d'abord utilisé des méthodes basées sur les mesures de rappel et précision, en cherchant à prendre en compte la structure des documents XML, mais ces mesures présentent cependant un inconvénient majeur : elles ne prennent pas en

compte l'imbrication (*overlap*) des éléments et évaluent le retour d'un élément pertinent sans prendre en compte le fait qu'il ait été déjà peut-être vu entièrement ou en partie par l'utilisateur.

Depuis 2003, des nouvelles mesure ont été fournies pour essayer de résoudre ce problème, telle que L'exhaustivité et La spécificité (*On dit qu'une unité d'information est exhaustive à une requête si elle contient toutes les informations requises par la requête et qu'elle est spécifique si tout son contenu concerne la requête*. [LAL, 97]), ces mesures incorpore la taille des éléments et le concept d'imbrication dans les mesures de rappel et précision. Au lieu de mesurer le rappel et la précision après qu'un certain nombre d'éléments aient été retrouvés, la taille totale de l'élément retrouvé est utilisée comme paramètre de base, alors que l'imbrication est traitée en ne considérant que les parties de l'élément qui n'aient pas déjà été vues (on considère alors que l'information pertinente est répartie uniformément au sein d'un élément).

Afin de résoudre ce problème, [Gabriella Kazai *et al]* établissent dans [KAZ 04] la définition d'une base de rappel idéale, qui supporterait la procédure d'évaluation suivante : les éléments de la base de rappel idéale doivent être retournés par les systèmes, les éléments proches de ceux contenus dans la base de rappel idéale peuvent être vus comme des succès partiels, mais les autres systèmes ne doivent pas être pénalisés s'ils ne les renvoient pas.

Les mesures proposées pour répondre à ces besoins sont :

- ***XML Cumulated Gain (XCG):***Cumulation des scores de pertinence des éléments de la liste des résultats. Etant donnée une liste d'éléments triée par ordre décroissant dans laquelle, les éléments sont présentés par leurs scores de pertinence :

$$XCG(i) = \sum_{j=1}^{i} XG(j) \qquad \textbf{I.1}$$

i: le rang de l'élément dans la liste.

XCG(i) : somme des scores de pertinence des documents ***j (j = 1, i).***

XG(j) : le score du document de rang ***j.***

Après avoir calculé le gain cumule des éléments, pour chaque requête on calcule un vecteur de gain idéal ***XCG(i)*** à partir de la base de rappel, et le ***XCG*** peut être alors comparé au ***XC(i)*** avec le ***nXCG :***

$$nXCG(i) = \frac{XCG(i)}{XCI(i)} \qquad \textbf{I.2}$$

Tel que : pour l'élément de rang **i**, le score de pertinence cumulé acquis sur le score de pertinence idéale voulu nous donne une valeur de norme comprise entre 0 et 1 tel que :

Si ***nxCG(i)*** : 0 élément non pertinent
Si ***nxCG(i)*** : 1 élément pertinent
Et il reflète le gain relatif que l'utilisateur accumule jusqu'à ce rang si le système avait produit une liste triée optimale.

- ✓ ***L'effort précision (ep)* :** Elle représente l'effort (en nombre de liens à visiter) qu'un utilisateur doit fournir pour parvenir à un gain donné ***r,*** e_{system} (respectivement e_{ideal}) est le rang auquel le gain **r** est atteint par le système (respectivement par la liste optimale).

Cette mesure dépend du gain, car elle est calculée par :

$$ep(r) = \frac{e_{ideal}}{e_{system}} \qquad \textbf{I.3}$$

Tel que :
r: c'est le gain
e: le rang correspondant au gain
e_{ideal}***:*** le rang auquel le gain est idéal
e_{system} : le rang auquel le gain est celui retourné par le système évalué.

- ✓ ***Mesure de precal* :** Elle a été utilisée lors de la compagnie d'évaluation 2002 pour définir la probabilité qu'un élément retrouvé et retourné à l'utilisateur soit pertinent, est calculée par :

$$p(pert/\,retr)(x) = \frac{x.n}{x.n + esl_{x.n}} \qquad \textbf{I.4}$$

Tel que :
pert: document X pertinent.
retr*:* document retrouvé.
$esl_{x.n}$: nombre attendu d'éléments non pertinents retrouvés jusqu'à ce qu'un point de rappel *X* soit atteint.
n: le nombre de documents pertinents dans la collection par rapport à une certaine requête.

Conclusion :

XML est de plus en plus reconnu comme un format standard de documents et l'on peut penser que dans un futur proche, un nombre important de documents et de données seront disponibles en format XML. L'avantage de ces documents est qu'ils possèdent une structure qui facilite leur présentation, ainsi que leur interprétation et leur exploitation dans des contextes présentant différents besoins.

Chapitre II
Le Modèle de Langue

PARTIE I : Modèle de langue pour la RI.

Introduction :

L'approche probabiliste traditionnelle tourne autour de la modélisation du concept de la pertinence, aussi le formalisme probabiliste offre un cadre pouvant mieux expliquer les problèmes de la RI.

Les documents contiennent des éléments qui caractérisent l'informationtraitée, c'est-à-dire les termes importants du sujet, mais aussi beaucoup d'autres éléments généraux de la langue qui ne sont pas informatifs (du bruit). Donc il est important de dégager les éléments significatifs par un moyen résistant au bruit. Les modèles de langue sont appropriés pour jouer ce rôle.

Au niveau de cette partie nous allons étudier largement les modèles de langue comme étant une dérivée du modèle probabiliste, et occupant une place de premier plan dans le domaine de la recherche d'information depuis la fin des années 90[Ponte et al, 1998].et nous allons étudier précisément ce modèle et leur exploitation en RI, car, notre contribution se base sur ses différent concepts.

1. Le modèle probabiliste de base :

Le modèle probabiliste, utilise un modèle mathématique fondé sur la théorie de la probabilité. De manière générale, le modèle probabiliste présente l'intérêt d'unifier les représentations des documents et des concepts.

Le modèle probabiliste tente de déterminerla probabilité de pertinence d'un document d par rapport à la requête q notée $P(per|q,d)$.Le premier modèle probabiliste était proposé par MaronKuhns en 1960. Les concepteurs de ce modèle ont bâti leur approche sur la théorie de probabilités ainsi que celle des statistiques pour évaluer ces probabilités de pertinence. Un document est alors sélectionné si la probabilité qu'il soit pertinent à q, notée $P(per|d)$ est supérieure à la probabilité qu'il soit non pertinent à $q$notée $P(\neg per|d)$, le score d'appariement entre le document$d$et la requête q, noté RSV(d, q) est donné par [Robertson et al, 1994] :

$$RSV(d,q) = \frac{P(per|q,d_i)}{P(\neg per|q,d_i)} \textit{(II.1)}$$

Avec :

$P(per|q,d_i)$Etant la probabilité que le document*d*soit pertinent pour la requête q et $P(\neg per|q,d_i)$est la probabilité qu'il soit non pertinent.

Et *per*est l'événement de pertinence et$\urcorner per$ est l'événement de non pertinence.

Donc : il revient à sélectionner les documents ayant à la fois une forte probabilité d'être pertinent est une faible probabilité d'être non pertinent,avec l'application de la formule de bayes qui est une formule mathématique utilisée pour le calcul des probabilités conditionnelles, sa forme est la suivante :

$$P\left(A/_B\right) = \frac{P(A|B) * P(A)}{P(B)} \qquad (II.2)$$

On obtient :

$$P(per|q,d_i) = \frac{P(per|q)P(d_i|per,q)}{P(d_i)} \textit{(II.3)}$$

Et:

$$P(\urcorner per|q,d_i) = \frac{P(\urcorner per|q)P(d_i|\urcorner per,q)}{P(d_i)} \qquad (II.4)$$

$P(d_i)$ Est la probabilité que le document soit choisi (si on prend au hasard un document dans le corpus, la probabilité de tomber sur *d*), sion considère qu'elle est constante.

$P(per|q,d_i)$Respectivement$P(\urcorner per|q,d_i)$ est la probabilité que le document $d_i$fait partie des documents pertinent ou non pertinent pour la requête$q$.

$P(per|q)$Respectivement $P(\urcorner per|q)$est la probabilité de pertinence ou non pertinence d'un document quelconque pour la requête q.

Avec : $$P(Per\backslash q) + P(\urcorner Per\backslash q) = 1 \qquad (II.5)$$

Un document, comme une requête, sont représentés par un ensemble de termes qui représentent des « événements ». La présence ou l'absence de ces termes sont les seules caractéristiques observables dans les documents comme dans les requêtes.

Ainsi un événement indique soit la présence ou l'absence d'un terme dans un le document ou la requête.

Plusieurs solutions ont été proposées pour représenter le document d et pour estimer les paramètres du modèle. La plus connue est celle du modèle BIR (BinaryIndependanceRetrieval) [Roelleke et al., 2009].

Un document est sélectionné si la probabilité, que le document $d$soit pertinent, notée $P(per|q,d_i)$ soit supérieure à la probabilité que d soit non pertinent notée$P(\neg per|q,d_i)$.

On considère le document d_i= (t_1, t_2,…,t_n),$d_{i(i=1,n)}$ des termes tel que :

(t_i=1)le terme est présent dans le document, et (t_i=0) le terme est absent dans le document, le score d'appariement entre le document$d$ et la requête q, noté *RSV(q,d)*est donné par :

$$RSV(q,d) = \frac{P(t_i = 1|t_1, t_2, \dots t_n)}{P(t_i = 0|t_1, t_2, \dots t_n)} \qquad (II.6)$$

Le modèle Okapi BM25 a été développé par [Robertson ,1994], il implémente le modèle probabiliste, en utilisant la fonction BM25 (largement discutée dans les travaux actuels de RI), pour calculer la probabilité de pertinence d'un document Dvis-à-vis de la requête Q:

$$RSV(d_i,q) = \sum tf_i * log\left(\frac{N - df_i + 0.5}{df_i + 0.5}\right) * \frac{(K1+1)tf_{ij}}{k1\left(\frac{b*ID}{AvgID+tf_{ij}}\right)} \qquad (II.7)$$

Avec :

tf_i = la fréquence d'un terme $t_i$dans la requête q.

tf_{ij} = la fréquence de$t_j$dans le documentd_i.

df_i= le nombre de documents contenant le terme t_j.

N = le nombre de documents dans le corpus.

ID= la longueur du documentd_i(le nombre de termes d'indexation).

AvglD= la moyenne des longueurs des documents dans le corpus.

*K1*et *b*sont deux constantes.

2. Le modèle de langue en linguistique informatique:

Le terme « Modèle de Langue » est emprunté de la linguistique informatique (le domaine qui sert à analyser une langue par un ensemble de probabilités qui calculent l'estimation d'inférer une séquence de mots à partir des mots d'une langue donnée).

Les premiers travaux qui ont porté sur l'utilisation des modèles de langue sont basé s sur les techniques de l'informatique linguistique, car ce dernier et la RI ont en commun la spécificité de manipulation de grandes masses de texte.

[Ponte et al ,1998], [Boughanem et al,2004] depuis leur introduction en RI, les modèles de langue se sont distingués par leur efficacité et leur fondement mathématique solide, contrairement aux autres modèles de la RI, les modèles de langue ne modélisent pas directement la notion de pertinence d'un document *d* face à une requête*q*. La pertinence d'un document vis-à-vis d'une requête est vue comme la probabilité que la requête soit générée par le modèle de langue du document.

2.1. L'idée de base :

Le modèle de langue est un concept probabiliste, Quelle est la probabilité qu'une séquence de mots d'une requête apparaît dans un document ou dans un corpus donné ? Si nous avons un modèle de la langue, nous saurons le dire au moins approximativement[Boughanem&al, 2004].

Par « modèle de langue», On désigne une fonction de probabilité *P*qui assigne une probabilité *P(s)* à un mot ou à une séquence de mots **s**dans une langue. Une fois cette fonction définie, il est possible d'estimer la probabilité d'une séquence de mots quelconque dans la langue, ou d'un point de vue générative, d'estimer la probabilité de générer cette séquence de mots à partir du modèle de la langue.

Considérons la séquence scomposée des mots suivants : m_1, m_2,…, m_n. La probabilité *P(s)* peut être calculée comme suit :

$$P(s) = \prod_{i=1}^{l} (m_i | m_1, m_2 \dots m_{i-1}) \qquad (II.8)$$

Avec : l = le nombre de mots prédécesseurs de m_i.

Pour des raisons de complexité, on simplifie en ne prenant que *n-1* prédécesseurs d'unmot appelé modèle '*n-gramme*'.

Dans cette probabilité, un problème lié à la multiplicité des paramètres surgit, d'où les étudesnous ont conduites à la simplification suivante :

$$P(s) = \prod_{i=1}^{l} (m_i | m_{i-n+1}, \dots m_{i-1}) \qquad (II.9)$$

Les modèles souvent utilisés sont les modèles uni-gramme, bi-gramme et tri-gramme comme suit :

Uni-gramme $P(s) = \prod_{i=1}^{l} P(m_i)$ $(II.10)$

Bi-gramme: $P(s) = \prod_{i=1}^{l} P(m_i | m_{i-1})$ $(II.11)$

Tri-gramme : $P(s) = \prod_{i=1}^{l} P(m_i | m_{i-2} m_{i-1})$ $(II.12)$

L'estimation se faitpar rapport à un corpus de documents*C*, parce que c'est impossible d'estimer ces probabilités pour une langue dans l'absolue. Si le corpus est suffisamment grand, nous pourrons ainsi dire que le modèle de langue établit est approximativement le modèle de cette langue.*P(α|C)* par hypothèse qu'il reflète cette langue en général, et on peut calculer les probabilités assez facilement par la méthode de vraisemblance maximale ML. Il suffit simplement de trouver toutes les occurrences d'un n-gramme (*α)* dans le corpus C, sa probabilité $P(\alpha|C)$ peut être directement estimée comme suit :

$$P(\alpha) = \frac{|\alpha|}{\sum_{\alpha_j \epsilon C} |\alpha_j|} = \frac{|\alpha|}{|c|} \qquad (II.13)$$

Avec :

$|\alpha|$: fréquence d'occurrence du n-gramme α dans le corpus C

α_j: n-gramme de même langueur que α

$|C|$: taille du corpus par nombre de mots = somme de toutes les occurrences de tout les termes.

Problème :

L'estimation des probabilités par vraisemblance maximale a ses limites : quand il s'agit d'estimer de nombreuses probabilités avec une grande quantité d'exemples, mais non infinie, il y aura toujours des problèmes, parfois systématiques, dans les faits, il y aura toujours des mots qui sont correctes et comprises, mais qui ne figurent pas dans un corpus. On veut donc que notre calcul de probabilité n'associe pas une valeur nulle à un des n-grammes quin'apparaît pas dans le corpus. Pour cela, destechniques sont invoquées et exploitées pour éviter à pénaliser les documents pouvant être pertinents dans la RI et la linguistique informatique, ces techniques sont connues par : ***techniquesde lissage.***

2.2. Les techniques de lissage :

Le lissage est une procédure qui consiste à attribuer des probabilités non nulles à des séquences contenant des mots absents dans quelques éléments d'un corpus d'entrainement, afin de généraliser les modèles de langue. Et pour cela une série de méthodes proposées dans la littérature. Nous présentons les plus classiques.

2.2.1. Lissage de Laplace ou (Ajouter-un) :

Cette méthode consiste à ajoute « 1 »à la fréquence d'occurrences de tous les n-grammes, Pour un n-gramme 'α', sa probabilité est estimée comme suit :

$$P_{Laplace}\ (\alpha|c) = \frac{|\alpha| + 1}{\sum_{\alpha_{i\epsilon V}}(|\alpha| + 1)} \qquad (II.14)$$

Avec : Vest le vocabulaire de la langue.

S'il y a trop de mots ou de n-grammes qui n'apparaissent pas ou peu dans le corpus, le lissage de Laplace devient inefficace, parce que tous les mots ont pratiquement la même probabilité « 1 » sur le nombre de mots.

2.2.2. Lissage Good-Turing :

Consiste à remplacer la fréquence d'un n-gramme 'α'observé 'r' fois dans le corpus par 'r*' suivante :

$$r^* = (r+1)\frac{n_{r+1}}{n_r} (II.15)$$

Où : n_r= le nombre de n-grammes apparus 'r'fois dans le corpus.

Donc sa probabilité est estimée comme suit :

$$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_{i\epsilon c}} |\alpha_i|} \qquad (II.16)$$

Ainsi, la fréquence des n-grammes leur est appliquée une diminution d'ordre (r*/r).

2.2.3. Lissage de Backoff :

Cette technique consiste à utiliser un modèle de langue du même ordre que le n-gramme recherché dans le corpus d'entrainement, si le n-gramme recherché est observé sinon celui d'ordre inferieur, c'est le cas du lissage Katz qui combine un modèle bi-gramme avec un modèle uni-gramme :

$$P_{katz}(m_i|m_{i-1}) = \begin{cases} P_{GT}(m_i|m_{i-1}) \; si \;\; |m_i m_{i-1}| > 0 \quad (Bi-gramme) \\ \alpha(m_{i-1}) P_{katz}(m_i) \qquad\qquad\qquad\qquad sinon \end{cases} \; II.17$$

Avec :$\alpha(m_{i-1})$:est un paramètre de normalisation par distribution des fréquences diminuées dansP_{GT}, calculé par :

$$\alpha(m_{i-1}) = \frac{1 - \sum_{m_i:|m_{i-1}m_i|>0} P_{katz}(m_i|m_{i-1})}{1 - \sum_{m_i:|m_{i-1}m_i|>0} P_{ML}(m_i)} \qquad (II.18)$$

2.2.4. Lissage par interpolation :

Le premier modèle d'interpolation qui a été appliqué par Jelinek Mercer, ce type consiste à combiner un modèle de langue d'ordre **N** avec un ou plusieurs modèles d'ordre inférieur, plutôt que d'utiliser ce dernier seulement dans le cas de fréquence 0 comme dans « Backoff ».

Une combinaison du modèle bi-gramme avec le modèle uni-gramme, on a :

$$P_{JM}((m_i|m_{i-1}) = \lambda_{m_i} P_{ML}(m_i|m_{i-1}) + \left(1 - \lambda_{m_{i-1}}\right) P_{ML}(m_i) \qquad (II.19)$$

Avec : $\lambda_{m_{i-1}}$est un paramètre dépendant du mot m_{i-1}ou il peut avoir une valeur fixe pour tous les mots de la séquence et sert à mettre en valeur la séquence recherchée.

Typiquement, dans le cas de collection de document, on pourrait estimer le modèle de document en le combinant avec le modèle de la collection, dans ce cas le modèle de document est exprimé comme suit :

$$P_{JM}(m_i|d) = (1 - \lambda) P_{ML}(m_i|d) + \lambda P_{ML}(m_i|c) (II.20)$$

Les modèles $P_{JM}(m_i|d)$ et $P_{ML}(m_i|d)$ sont estimés selon le maximum de vraisemblance.

2.2.5. Lissage de Dirichlet :

Le lissage précédent ne prend pas en considération la taille des échantillons d'entrainement, pour cela, Dirichlet a proposé d'exploiter les valeurs de λ (II.20) en fonction de la taille des échantillons. Dans ce cas la formule s'écrit comme suit :

$$P_{Dir}(m_i|d) = \frac{|d|}{|d| + \mu} P_{ML}(m_i|d) + \frac{|d|}{|d| + \mu} P_{ML}(m_i|c)$$
$$= \frac{|d| P_{ML}(m_i|d) + \mu P_{ML}(m_i|c)}{|d| + \mu}$$

$$(II.21)$$

Avec :

$$P_{ML}(m_i|d) = \frac{|m_i|}{\sum_{m_j \in d} |m_j|} = \frac{tf(m_i, d)}{d} \qquad (II.22)$$

On obtient

$$P_{ML}(m_i|d) = \frac{tf(m_i, d) + \mu P_{ML}(m_i|c)}{|d| + \mu} (II.23)$$

Avec : |*d*| est la taille du document *d*.

$tf(m_i, d)$est la fréquence brute du terme m_i dans le document *d*.

Plusieurs études en RI ont montré que le choix d'une méthode à un grand impact sur les performances du SRI.[Zhai&Lafferty, 2001] ont rapporté que la méthode de lissage de Dirichlet donne de meilleurs résultats que les autres méthodes de lissage.

Pour traiter les différents problèmes auxquels les approches symboliques traditionnelles ne donnent pas de résultat satisfaisant, des modèles de langue sont appliqués en RI, donc *: Quelle est la performance ajoutée par ces modèles dans le monde de la RI ? Et comment sont traités ces modèles ?* Ce sont des questions auxquelles répond la section suivante.

3. Les modèles de langue en RI :

3.1. Le principe :

Le principe des approches utilisant un modèle de langue est différent à celui des approches traditionnelles, où on tente de modéliser la relation de pertinence entre un document et une requête (la probabilité de pertinence d'un document face à une requête). Mais, le principe c'estd'utiliser le modèle de langue où on considère que la pertinence d'un document face à une requête est en rapport avec la probabilité que la requête peut être générée par le modèle de langue d'un document.

Donc : on suppose qu'à chaque document est associé un modèle de langue typique *Md* et soit la requête fournie est q, donc : la pertinence du document *d*vis-à-vis de la requête *q* est déterminée par : *P(q|Md)*qu'on lit : « probabilité de génération de la requête *q*par le modèle de langue du document *d*».

3.2. Approches d'exploitation des modèles de langue en RI :

Les approches de modélisation des modèles de langue en RI se classe en trois grandes catégories qui se basant sur la représentation des documents et des requêtes, et la fonction de similarité par le calcul de probabilité.

1. Génération de la requête par le modèle de document (QueryLikelihoodModels) : sont des méthodes appliquant le principe expliqué ci-dessus. Les documents sont alors classés selon leurs probabilités de génération de la requête *P(q|Md).*

2. Génération de document à partir de la requête (Document LikelihoodModels) : contrairement à l'approche précédente, cette approche associe à chaque requête un modèle et calcule la probabilité de génération du contenu document de cette requête.

3.Similarité document-requête: qui consiste à associer un modèle de langue pour chacun des deux entités « document et requête » donc ces derniers sont ordonnés selon la similarité de leurs modèles. Cette similarité est estimée en calculant l'entropie croisée KL (Kullback-Leiber).

Dans ce qui suit, nous présentons plus en détail ces approches pour implémenter ces principes.

3.2.1. Génération de la requête par le modèle de document :

[Ponte &Croft, 1998], ont supposé que l'utilisateur a eu des idées sur les termes qui sont présents dans le document pertinent ou modèle de ce document $M_{D,}$ et à base de ça génère sa requête en utilisant ces termes, avec la probabilité suivante qui représente le score de pertinence de ce document :

$$Score(d,q) = P(q|Md) \qquad (II.24)$$

Ce modèle ne prend pas en compte seulement les mots présents dans la requête, mais aussi les absents sont pris en compte, cette distribution faite par le modèle de bernoulli, Parce que dans les modèles de la RI, la requête est considérée comme étant une suite de mots clés : $q=t_1,t_2\ldots t_n$, indépendants les uns des autres.

- ✓ Supposons, que la requête Qest composée de l'ensemble de mots t_1, t_2, … t_n, et que les mots t_{n+1}, t_{n+2}, … $t_l$sont absents de la requête. *P(q|Md)* peut être ré-exprimée comme suit :

$$Score(q,d) = \prod_{t_i \in Q} P(t_i|d) \times \prod_{t_i \notin Q} (1 - P(t|Md)) \qquad (II.25)$$

Puis ils ont proposé de calculer la probabilité $P(t_i/M_D)$ en s'inspirant du lissage de Backoff comme suit :

$$P_{PC}(t_i|Md) = \begin{cases} P_{ML}(t_i|d)^{1-\hat{R}(t_i,d)} \times P_{avg}(t_i)^{\hat{R}(t_i,d)} \; si \; tf(t_i,d) > 0 \\ P_{ML}(t_i|c) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad sinon \end{cases}$$

$$(II.26)$$

Avec : $tf(t_i,d)$ est la fréquence brute du terme $t_i$dans le document *d*.

$P_{ML}(t_i|d)$est la probabilité par vraisemblance maximale de $t_i$dans *d*.

$$R(t_i,d) = \frac{1}{1+f_{t_i}} \times \left(\frac{f_{t_i}}{1+f_{t_i}}\right)^{tf(t_i,d)} \qquad (II.27)$$

f_{t_i}= la fréquence moyenne de t_i dans le document *d*.

R = le risque d'apparition du terme t_i dans le document *d*.

$P_{avg}(t_i)$=la probabilité moyenne du terme t_i dans le document*d*.

$P_{ML}(t_i|c)$ = la probabilité par vraisemblance maximale du terme t_i dans la collection C.

L'avantage dans cette approche est qu'elle met en évidence l'aspect de « spécificité du document » car elle considère les termes absents dans la requête.

3.2.2. Génération de document à partir de la requête :(document likelihoodModèl)

[Lavrenko&Croft, 2001] ont proposé de modéliser explicitement le modèle de pertinence, donc estimer le modèle du document à partir de celui de la requête sans utiliser aucun corpus d'entrainement, ils considèrent en effet que pour chaque requête, il existe un modèle permettant de générer le sujet (thème) abordé par le requête c'est ce que les auteurs appellent le modèle de pertinence, noté :θ_R

Alors la probabilité à estimer est : $P(t/\theta_R)$,est une probabilité de générer un terme à partir du modèle de pertinence θ_R, mais ce dernier n'est pas en corps connu, ceux qui amène les auteurs à exploiter les documents retournés les mieux classés (**feedback document).**

Le modèle de pertinence est alors exprimer comme suit :

$$P(t|\theta_R) = \sum_{d \in R} \frac{P(t|d)P(q|d)P(d)}{P(q)} = \sum_{d \in R} P(t|d)P(q|d) \qquad (II.28)$$

Avec : Rest l'ensemble des documents feedback.

$P(q|d)$est le score de pertinence du document *d*vis-à-vis de la requête*q*.

3.2.3. Similarité modèle de document-modèle de requête :

Le modèle « Uni-gramme » de 1er ordre est le modèle utilisé dans tous les modèles présentés, justifiée par l'hypothèse d'indépendance entre les termes, mais elle ne reflète pas la réalité, car dans cette dernière les termes dans un document ou d'une requête sont liés, donc il s'avère évident de penser à un modèle combinant les termes,

par intégration des relations potentielles entre termes dans les modèles de langue, ces relations sont de deux types :

✓ ***Les relations de proximité :***

Relation de proximité de surfaces ou de dépendance entre les termes, dans l'objectif de remédier au problème d'ambigüité des termes. Plus de détails dans [HAMMACHE, 2014].

✓ ***Les relations sémantiques :***

Relation sémantique entre termes c'est-à-dire on exploite « L'expansion du modèle de la requête » ou en ajoutant les termes liés au modèle de la requête, ou on exploite « L'expansion du modèle du document» où en attribuant une plus grande probabilité aux termes liés aux termes du document, ou ces deux approches conjointement. Pour plus de détails [HAMMACHE, 2014].

4. Le modèle de langue et la longueur du document :

[Ponte &Croft, 1998] ont proposé dans cette approche de générer la requête utilisateur par le modèle de langue du document comme suit :

$$Score(d,q) = P(d,q) \qquad (II.29)$$

$P(d,Q)$: Probabilité conditionnelle.

$Score(d,Q)$ = score de pertinence du document*d*pour la requête *Q*.

Si on applique la formule de bayes on obtient :

$$P(d,Q) = \frac{P(Q|D)P(d)}{P(Q)} \qquad (II.30)$$

$P(q)$est la probabilité de la requête$q$**.**

Mais l'utilisateur génère sa requête indépendamment du document *d*,

C'est-à-dire*P(q)* n'a aucun influence sur la pertinence du document *d*, on obtient :*P(d|q)=P(q|d) P(d)*

Donc la longueur du document est incorporée dans sa probabilité *P(D)*

$$P(d) = \frac{|d|}{\sum_{i=1}^{n}|d_i|} = \frac{|d|}{|C|} \qquad (II.31)$$

Avec : $|d|$est la longueur du document *d*,et|***C***|est la longueur de la collection.

De ce fait, la probabilité d'obtenir le document *d*est proportionnelle à sa langueur.

La prise en compte de la taille du document dans la construction du modèle de langue correspondant à un document, et le calcul de score de pertinence suite à une requête utilisateur, et aussi la prise en compte la distribution des mots dans les documents de la collection, cecia été largement traité dans[Achmoukh, 2006],Cette approche a été comparée au modèle de langue basique et les résultats obtenus ont montré son efficacité.

5. Evaluation des modèles de langue :

La qualité d'un modèle de langue a été exprimée par [Estève, 2002] : c'est une valeur de « perplexité », notée PP est calculée : PP = 2^{LP}

Avec : LP ou logProgdésigne l'approximation moyenne d'émission d'une séquence de mots S de l'historique des mots h, sa valeur est donnée par :

$$LP = (-1/h)logP(s) = -(1/h)\sum logP(s)\ (II.32)$$

<u>Interprétation :</u>

Plus la valeur de perplexité est petite plus le modèle de langue évalué est performant, ie : sa valeur de prédiction est importante, car cette valeur représente le degré d'incertitude du modèle de langue.

Deux types de perplexité existent :

- ✓ Perplexité calculée sur le corpus d'apprentissage du modèle, elle permet de décider la qualité des approximations ou probabilités utilisées pour définir le modèle ainsi que la qualité de ses paramètres.
- ✓ Perplexité calculée sur le corpus de test, elle permet de juger la capacité à la généralisation associée au modèle.

Conclusion :

Nous avons présenté au cours de cette partie le modèle probabiliste de base et l'idée de base en linguistique informatique du modèle de langue ainsi que les techniques de lissage exploitées par ce dernier. Nous avons par la suite présenté le principe du modèle de langue en RI traditionnelle (documents plat), ainsi que les différentes approches d'exploitation de ce modèle en RI. Cette partie a été clôturée par une brève présentation des techniques d'évaluation de ce modèle.

Le modèle de langue a été largement utilisé en RI traditionnelle et a prouvé son efficacité dans la satisfaction des besoins de l'utilisateur en informations pertinentes. Dans le chapitre suivant nous allons présenter le modèle de langue utilisé dans la RI dans les documents XML et voir si ce dernier a eu autant de succès en RIS qu'en RI.

PARTIE II : Modèle de Langue pour la RIS.

Introduction :

La Recherche d'information structurée utilise les différents modèles de la RI traditionnelle tout comme le modèle de langue, dont l'objectif est de retourner l'information pertinente au sein des documents XML.

Les modèles de langue utilisent l'information structurelle afin de retrouver les unités d'information répondant de manière **exhaustive (**qui décrit à quel point l'élément traite du sujet de la requête), et **spécifique (**qui décrit à quel point l'élément se focalise sur le sujet de la requête)à la requête utilisateur.

Cette partie résume, comment le modèle de langue a été adapté à la RI structurée et appliquer aux documents de type XML?, et Comment satisfaire l'utilisateur avec l'information pertinente en profitant des avantages de la structure du document *?*

1. Utilisation de la structure dans le processus d'appariement :

Comme la granularité de l'information en RIS, peut être un paragraphe, un texte ou un élément,…etc. Il s'agit donc de remplacer dans le modèle de langue standard, le modèle du document *Md* par le modèle d'élément *Me*. De ce fait, le modèle*Me* est la plateforme basique de l'estimation d'une requête*q* à partir d'un document *d*est donnée par : [Baeza& Ribeiro, 2011].

$$P(q|Md) = P_{t_{i\in Q}}((t_1, t_2, \dots t_n|Me) = \prod_{i=1}^{n} P(t_i|Me) \qquad (b,1)$$

La probabilité d'aboutir à un élément en réponse à une requête est évaluée dans [Sigubjörnsson&al., 2004] au travers d'un lissage de Jelinek-Mercer (pour obtenir la formule suivante qui fait l'interpolation entre la probabilité du terme *t* dans l'élément *e* et sa probabilité dans la collection *C* :

$$P(t|Me) = \lambda P(t|e) + (1-\lambda)P(t|C) \qquad (b,2)$$

λ: est une constante de pondération de la séquence de mots recherchés.

[Lalmas, 2009] a intégré la taille de l'élément lors de la création du modèle, dans la probabilité associée à un élément donnée par :

$$P(e) = \frac{|e|}{\sum_{i \in c} |e_i|} \qquad (b,3)$$

|C|: collection de test ou d'apprentissage

|e| : taille de l'élément en nombre de termes.

2. Utilisation de la structure du document dans les modèles de langue :

Les documents semi-structurés sont caractérisés par la présence d'une structure organisant leurs contenus textuels, et pour utiliser cette structure dans les modèles de langue appliqués à la RIS, les auteurs ont défini deux familles d'approches :

- ✓ *La contextualisation des éléments* : un modèle d'élément est interpolé comme une couverture de tout le document en générale.
- ✓ *L'agrégation* : Le texte de chaque élément est vu comme l'agrégation du texte contenu dans ses descendants.

o *La contextualisation des éléments :*

Selon [Trotman, 2009] c'est une famille de techniques cherchant à évaluer la pertinence d'une unité de texte au moyen d'informations obtenues en dehors de l'unité d'information elle-même et a travers des unités textuelles qui l'entourent.
La contextualisation d'un nœud (ou élément) permet d'évaluer la pertinence d'un élément en combinant le score de ce dernier avec les scores d'autres éléments de niveaux différents. Un lissage par interpolation est considéré comme un exemple de contextualisation, puisque un paramètre contrôlel'influence de la probabilité du terme dans une collection par apport à celui de l'élément.

$$P_{\mu,l}(t_i|e_l) = \frac{tf(t_i, e_l) + \mu P(t_i, C_l)}{|e_l| + \mu} \qquad (b,4)$$

Avec :

μ *:* une constante et $\mathbf{e_l}$ l'élément de label particulier.

$\boldsymbol{C_l}$ *:* est la collection dont les éléments sont de label **l**.

t_i *:* est le terme recherché.

Le concept de recherche par élément introduit par [Hiemstra, 2003] a été étendu dans [Ganguly&al., 2011]par l'interpolation du modèle de langue d'un élément et celui de son père, cela à permis de favoriser un élément non pertinent dans un document pertinent dans un corpus XML.

$$P(t_i|Me) = \lambda \times P(t_i|e) + (1-\lambda) \times (\beta \times P(t_i|d) + (1-\beta) \times P(t_i|Mc)$$

$$(b,5)$$

Avec λ **:** paramètre de pondération de document.
Et β : paramètre de pondération de collection.

2.1. L'agrégation :

Dans [Larson, 2006],l'agrégation est le fait de combiner des représentations des éléments de niveau inférieur (fils) avec celles de l'élément en cours en fonction de la structure du document.
L'agrégation identifie les relations entre éléments structurels. Le score d'un élément est calculé par l'interpolation entre les modèles de langue des fils qui y sont associés, et ces scores sont calculés pour chaque élément à partir de leur contenu textuel. La formule est la suivante :

$$P(t_i|Me) = \lambda_0 P(t_i|Me) + \sum_{i=1}^{e} \lambda_j P(t_i|Mej) \qquad (b,6)$$

Avec :
e : est l'élément ayant l fils,
Me : est le modèle associé à e.
$Me_{j\ (j=1..l)}$: sont les modèles associés à chacun des fils **ej**de **e**.
λ : est un paramètre d'influence de chaque modèle d'un élément, tel que :

$$\lambda_0 + \sum_{j=1}^{e} \lambda_j = 1 \qquad (b,7)$$

3. Traitement des contraintes structurelles des requêtes CAS:

Les CAS (Content And Structure): Ce sont des requêtes qui contiennent de plus des contraintes sur la structure des documents. Ces contraintes structurelles ont été considérées comme vagues de conditions, qui peuvent être traitées de trois manières :

- ✓ Classification des éléments selon leur label (type de balise) : c'est le fait de regrouper des labels ou des ensembles de labels pour construire des classes d'équivalence afin d'en faciliter la représentation.
- ✓ Calcul du score selon les conditions de structure : faire varier le score en prenant en compte le label de l'élément vis à vis de celui exprimé en tant que contrainte dans la requête, est couramment utilisé en modèle de langue [Cyril, 2013].
- ✓ Prise en compte de la notion de la hiérarchie et de la distance entre les éléments, en général c'est le fait de déterminer les modèles des éléments ***Me*** pris en compte lors

de l'interpolation (ascendant, parent ou descendant) par le processus de calcul de score.

4. *Utilisation explicite de la structure des documents vis-à-vis des contraintes des requêtes :*

Les auteurs appellent l'appariement entre les requêtes et la structure des documents basant sur le modèle de langue « *le processus de relaxation de structure* ».
Ce processus selon [ALILAOUAR, 2007]. Consiste à assouplir les contraintes portées sur la structure du document cible, et ce type de relaxation peut être subdivisé en trois sous types : la relaxation d'ordre, la relaxation des nœuds ainsi que la relaxation des arcs.
De ce fait, la construction d'un modèle du document oblige l'inclusion de sa structure.
Cette approche est restée dans le cadre théorique puisqu'elle n'a été ni implémentée ni testée [Cyril, 2013].

5. *Un modèle de langue basé sur la structure de document :*

D'après [Cyril, 2013]les modèles de langue adaptés à la RIS sont basés sur la prise en compte de la structure des documents, et l'évaluation de la similarité structurelle document-requête, et l'utilisation les techniques de lissage.

a. *Prise en compte de la structure du document XML :*

Les meilleures approches sont celles basées sur des modèles de langue qui prennent en compte l'utilisation de la structure d'un document, et l'emplacement d'un élément pertinent dans un arbre XML.

Donc, le score de pertinence d'un nœud qui répond spécifiquement à une requête cible un élément est donné par la formule suivante :

$$RSV(n,Q) = \begin{cases} 0 & si\ l(n) \neq EC \\ \lambda \times score_c(n,Q) + (1-\lambda) \times score_s(n,Q)\ sinon \end{cases} \qquad (b,8)$$

Avec :
*Score $_c$*est score de contenu pour le nœud n.
*Score $_s$*est le score de similarité structurelle pour n.
$\boldsymbol{\lambda \epsilon [0,1]}$est le paramètre d'apport du $score_c$et $score_s$.
Ec (Q) est la fonction qui retourne la balise spécifique par la requête *Q*.
l(n) est la balise du nœud n.
[Sparck Jones, 1973], on trouve une mesure de score dite « *score de contenu* » basée sur l'approche *(tf * idf),*et qui évalue les nœuds feuilles qui contiennent l'information textuelle. Ce score est calculé comme suit :

$$score(x) = \sum_{t_{i \in Q}} tf(t_i|X) \times idf_{t_i} \qquad (b,9)$$

Avec : *x* est un nœud feuille de la collection considéré.

5.1. Estimation de la probabilité de génération de la structure de la requête :

Les éléments d'un document XML sont présentés sous forme d'un arbre, et le modèle d'éléments est remplacé par le modèle d'arbre noté : M_T(le modèle de langue associé à l'arbre *T*), et la probabilité de similarité « Sim $_s$(M_T,*Q)* »entre la requête Q et le modèle de langue d'un arbre T est donnée par la formule suivante :

$$P(Q|M_T) = \prod \prod P\left(e_{i,j\in q}\middle|M_T\right)^{w(e_{i,j},Q)} \qquad (b,10)$$

Avec :

$\boldsymbol{e}_{i,j}$est l'arc $u \longrightarrow v$ de la requête pour lequel on calcule la probabilité $P\ (e_{i,j}|M_T)$ que $\boldsymbol{e}_{i,j} \in T$.

w(e $_{i,j}$, Q) est le poids calculé selon la distance entre les nœuds u et v.

5.2. Lissages du Modèle de Langue basé sur la structure:

Leur objectifs est de pallier au problème des probabilités nulles assignées aux arcs absents dans un arbre de requête $\boldsymbol{e}_{i,j} \in T$, les techniques de lissage basées sur la structure d'arbre sont utilisées :

- ***Lissage de Jelinek-Mercer :***

A partir de tous les arcs de la collection, en lissant le modèle d'arbre M_T *:*

$$P\left(e_{i,j}\middle|M_T\right) = (\alpha \times P\left((e_{i,j\in Q}\middle|M_T\right) + (1-\alpha) \times P\left(e_{i,j}\middle|M_C\right)^{W(e_{i,j},Q)} \qquad (b,11)$$

Avec :

$P\left(e_{ij}\middle|M_T\right) = \frac{w(e_{ij},T)}{w(T)}$ *:*la probabilité de pertinence d'un arc dans l'arbre de document.

W($e_{i,j}$*,T)*est Le poids de la relation dans l'arbre calculer :

$$w(T) = \sum_{e_{ij\in T}} w\left(e_{i,j},T\right)\ et\ w(C) = \sum_{e_{i,j}\in c} w\left(e_{i,j},C\right) \qquad (b,12)$$

$$P\left(e_{ij}\middle|M_C\right) = \frac{w(e_{i,j},C)}{w(C)} \qquad (b,13)$$

- ***Lissage de Dirichlet* :**

Se base sur la longueur de document, et donnée par la probabilité que la structure de la requête soit générée par l'arbre document et calculer :

$$P\left(e_{i,j}\middle|M_T\right) + \frac{w\left(e_{i,j},T\right) + \mu \times \frac{w\left(e_{i,j},Md\right)+\lambda P(e_{i,j},Mc)}{w(d)+\lambda}}{w(T)+\mu} \qquad (b,14)$$

avec :

λ et μ les paramètre de lissage utilisés pour contrôler l'influence de la collection sur le score final.

6. *Discussion :*

La recherche d'information est une science qui s'intéresse à la collecte, le stockage et la sélection des informations afin de retourner celles qui sont pertinentes pour une requête utilisateur, de ce fait il existe un grand nombre de modèle de langues pour la recherche d'information.

Le modèle de langue est définit comme étant une variante du modèle probabiliste de base, qui se base sur le principe de la probabilité d'inférer la requête utilisateur par un document particulier.

Dans la RI classique diverses techniques d'appariement entre un document appartenant a un corpus et une requête utilisateur sont proposées.

Dans la recherche d'information XML, les travaux présentés pour les modèles de langue sont peu. Du fait que l'information portée par les documents XML n'est plus un simple texte, le principe d'inclure la structure du document dans les modèles de langue pour la RIS reste d'actualité.

Dans les modèles de langues pour la RIS les modèles des documents sont traduit sous forme d'arbre d'éléments (à savoir que l'élément est la plus petite information contenue dans un document structuré), la RIS vise à retourner un élément comme résultat d'une requête (l'élément le plus pertinent).

Le paramètre pris en compte dans les modèles de langue adaptés à la RIS est le modèle de l'élément en prenant compte la taille du document, ou la fréquence des termes ou bien les relations hiérarchiques, par contre les relations entre les termes et les éléments (le nombre de termes dans un élément, le nombre d'élément dans un document ainsi que le nombre d'éléments la collection de documents) ne sont pas incluse, or que ces paramètres peuvent être efficacespour les méthodes d'appariement élément- requête pour les modèles de langue pour la RIS.

Dans le cadre de la RIS, nous voulons adapter le modèle de langue proposé par [Belkacem, 2015] qui se base sur l'appariement document requête, pour réaliser l'appariement entre l'élément et la requête et cela en prenant compte la taille de l'élément dans le document, le nombre de nœuds feuilles, et la taille moyenne de

l'élément dans le document, pour réaliser un modèle de langue remarquable pour la RIXML

Conclusion :

Cette partie a pour objectif de présenter les notions élémentaires des modèles de langue appliquée à la RIS. Ainsi que les différentes notions et techniques proposées par plusieurs auteurs, et qui prennent en compte la structure des documentsXML exploité par la RIS.

Le chapitre suivant consiste à présenter notre contribution qui se base sur certains concepts du modèle de langue traités dans cette partie.

Chapitre III
Un Modèle De Langue Pour La RIS Dans Les Documents XML

Introduction :

La structure des documents XML est la caractéristique principale qui a incité les auteures de la RI dans les documents XML a exploiter les modèles existant dans la littérature et plus précisément le modèle probabiliste, à réaliser des approches, et des systèmes de recherche d'information flexibles, répondant au mieux à des requêtes utilisateurs.

Notre contribution consiste à implémenter et évaluer un modèle de langue proposé pour la RI dans les documents XML qui exploite la structure de ces documents et calcule la probabilité qu'un modèle structuré puisse générer une requête, et lui attribue un bon score de pertinence.

Dans ce chapitre, nous nous intéressons à la nouvelle granularité d'information retournée aux requêtes utilisateurs, par l'adaptation d'un modèle de langue proposé, et à la problématique liée à ce modèle quand il traite la structure des documents XML.

1. Problématique :

Tous les travaux présentés dans la RI et la RIS ont pour but de rapprocher la pertinence système de la pertinence utilisateur.
Dans les modèles de langue, plusieurs facteurs de pertinence ont été étudiés et mis en œuvre, sous forme de paramètres intégrés dans les formules d'appariement et de calcul de score de pertinence d'un document recherché. En effet, le modèle de langue perçoit la pertinence sous un format différent dans le calcul des probabilités : contrairement au modèle de base, le modèle de langue se base sur la probabilité qu'une requête utilisateur soit générée par un modèle de document.

Dans cette optique, plusieurs travaux ont été présentés dans la littérature pour la RI dans les documents plats pour le modèle de langue, dont l'approche se basant sur la taille du document [Achmoukh, 2006]. Mais concernant la RI dans les documents XML, nous avons constaté que certains auteurs se sont intéressés à proposer des approches combinant à la fois la contrainte structurelle d'un document XML et le principe d'appariement dans les modèles de langue.

Alors qu'un document XML est une arborescence d'éléments : le nœud feuille (qui contient l'information textuelle qui est exploitée dans le processus d'appariement), le nœud interne (ayant des descendants) et le nœud unique racine (qui représente le document XML complet). Mise à part la contrainte structurelle, ces caractéristiques n'ont pas été prise en compte dans les différents modèles de langue proposés dans la RI dans les documents XML, alors qu'il serait plus judicieux de les considérer.

De ce fait, le processus de calcul du score de pertinence dans la RI XML, doit prendre en compte ces notions :

- Un document XML est une arborescence d'éléments,
- Un nœud feuille est un ensemble de termes,
- Un élément peut être un nœud interne ou feuille,
- La collection XML est un ensemble de documents XML, donc un ensemble d'arborescences.

C'est précisément dans cette optique que [Belkacem, 2015] a proposé sa contribution. Le calcul des probabilités de pertinence d'un composant retourné à l'utilisateur doit prendre en compte la pertinence d'un élément (d'un nœud feuille ou d'un nœud racine) et l'importance d'un terme dans une collection de documents structurés sera le résultat de combinaison de sa probabilité dans les éléments cités ci-dessus, ainsi, l'importance des termes ou la probabilité d'avoir la requête utilisateur à partir d'un document XML sera le résultat de distribution de l'importance des termes la constituant.

Notre travail consiste à implémenter et à évaluer le modèle de langue appliqué aux documents XML proposé par [Belkacem, 2015].

2. Le modèle de langue proposé pour la RIXML :[Belkacem, 2015]

2.1. Contribution :

L'auteur dans [Belkacem, 2015], a défini un ensemble de paramètres à prendre en considération pour l'adaptation d'un modèle de langue à la nouvelle granularité d'information, celle contenue dans les documents semi-structurés XML, car selon la structure d'un document et son contenu textuel que nous pourrions extraire un contenu informatif combinant à la fois la structure et l'information recherchée par l'utilisateur.

L'ensemble des paramètres utilisés dans l'établissement de l'approche proposée dans [Belkacem, 2015] sont :

- *dcel* = La taille d'un document en nombre d'éléments.
- $|e_j|$ = La taille de l'élément *ej* en nombre de termes.
- $\Delta|e|$ = La taille moyenne d'un élément dans le document en cours, calculée par :

$$\frac{\text{taille du documents en nombre de termes}}{\text{taille du documents en nombre d'éléments}} = \frac{\sum_{i=1}^{n} |ei|}{dcel}$$

tf_{ji} = Fréquence du terme t_i dans l'élément e_j.
Nf = Nombre de nœuds feuilles dans le document en cours.
Nfc = Nombre de nœuds feuilles dans la collection.

$|D|=\sum_{j=1}^{Nf}|ej|$La taille du document en nombre de termes.

$|ed_i|$ = Nombre d'éléments qui contiennent le terme t_i *dans* le document en cours

$|ec_i|$= Nombre d'éléments qui contiennent le terme *ti* dans la collection des documents XML.

2.2. Idée de base de la contribution:

[Belkacem, 2015] s'est inspiré du modèle langue de [Achmoukh, 2006] pour proposer son modèle en exploitant la taille d'un document plat (texte). Par adaptation à la nouvelle granularité d'information.

Pour commencer la recherche, et répondre à une requête utilisateur, l'auteur propose d'examiner les nœuds constituant un document de la collection, pour calculer la probabilité d'avoir l'élément *ej* comme résultat ;

Ensuite, l'auteur estime une probabilité conditionnelle reliant la requête *Q* à un élément ***ej*,** c'est la probabilité de similarité élément-requête, en deux phases :

1. Calculer la probabilité d'importance d'un terme $\boldsymbol{t_i}$ dans un élément ***ej***, au biais de deux algorithmes différents ;
2. Calculer le produit de ces probabilités pour tous les termes de la requête : c'est la distribution aux termes de la requête *Q ;*

Et au final, calcule le produit entre la valeur de similarité obtenue et la probabilité associée à l'élément *ej*, ce qui donne le score de pertinence de l'élément *ej* pour la requête *Q***.** Le résultat *ej* sera retourné si son score n'est pas nul.

2.3. Le détail de travail :

Pour répondre à une requête utilisateur, le SRI dans les documents XML retournera un élément de score calculé par la formule présentée [Achmoukh, 2006] et qui a été adapté à la RI XML, du fait que dans un document XML l'unité recherchée n'est plus un document entier mais c'est l'élément répondant à la requête, donc :

$$Score(D,Q) = P(e_j|Q) = P(e_j|Q) \times P(e_j) \qquad (2)$$

Avec $P(e_j)$ est la probabilité d'avoir l'élément e_j comme résultat pour une requête, cette probabilité peut être calculée de deux façons :

par :

La formule de base : $$P(e_j) = \frac{|ej|}{|D|} \qquad (3.1)$$

Celle que l'auteur a utilisée et testée.

Ou par : La variante : $$P(e_j|Q) = \frac{|ej|}{\Delta|e| \times Nf} \qquad (3.2)$$

Du fait que l'importance d'un élément ne dépend pas seulement du rapport entre sa taille et celle du document le contenant mais peut dépendre des autres éléments et des nœuds feuilles dans le document vue que l'information textuelle est présente dans les nœuds feuilles ainsi que par hypothèse de dépendance entre les éléments du même document, ie : les éléments d'un même document appartiennent au même contexte même s'ils ne traitent pas forcément le même sujet mais leurs thèmes sont en relation.
Notre travail s'est basé sur cette dernière formule (variante). En effet l'auteur l'a proposé mais ne l'a pas testé.
Notre travail consiste à implémenter et tester cette variante et comparer les résultats obtenus avec la formule de base que l'auteur ait implémenté et testé.
Ainsi que la probabilité conditionnelle d'avoir la requête utilisateur à partir de l'élément encours est :

$$P(Q|e_j)\prod_{i=1}^{n} P(t_i|e_j) \qquad (4)$$

Cette probabilité permet la distribution des probabilités des termes de la requête *Q* étant une séquence de mots *: Q = t_1,t_2,…,t_n* et le modèle de langue
Me_j =t'_1, t'_2,…, t'_n = e_j.
Ainsi, la probabilité d'avoir le terme t_i dans le modèle de l'élément e_j dans *(3)* est donné par son importance par :

$$P_I(t_i|Me_j) = P(t_i|e_j) = \frac{P_{MLE}(t_i|Me_j)}{\sum_{k=1}^{n} P_{MLE}(t_i Me_k)} \qquad (5)$$

Avec :
$N = dcel$.
P_{MLE} est la probabilité de vraisemblance maximale adaptée à la nouvelle granularité « élément XML » qui peut être calculé soit par la formule de base :

$$P_{MLE}(t_i|Me_j) = \frac{tf_{ji}}{|e_j|} \qquad (5.1)$$

Soit en exploitant la formule de pondération des F4 proposée par [fellag, 2006] pour calculer le poids du terme ti dans l'élément ej dans la probabilité de génération de génération :

$$F4 = tf_{ji} \times ief \times \frac{1}{h1+h2+\frac{|e_j|}{\Delta|e|}} \qquad \text{avec:} \qquad ief = log\left(\frac{Nf}{tf_i} + \alpha\right)$$

D'où la probabilité de vraisemblance est :

$$P_{MLE}(t_i|Me_j)\frac{F4}{|e_j|} \qquad (5.2)$$

Avec: $\alpha = 0{,}5$, $h_1 = 0{,}7$, $h2 = 0{,}3$

2.4. Problème des probabilités nulles :

Dans la formule *(4)*, si un terme t_i de la requête *Q* absent dans l'élément e_j, alors une probabilité nulle est attribuée dans *(5)*. De ce fait, toute la séquence de termes *Q* ainsi que l'élément e_j sont pénalisés par une probabilité nulle. Pour pallier à ce problème, l'auteur a adapté la technique de lissage utilisée par
[Achmoukh, 2006] à notre modèle pour les éléments de documents XML:

- ✓ **Le lissage et redistribution des probabilités :** Cette technique permet de calculer la probabilité d'importance d'un terme t_i selon son absence ou sa présence dans le modèle Me_j de l'élément e_j:

$$P_I(t_i|Me_j) = P(t_i|Me_j)\begin{cases} P_s(t_i|Me_j)\ si\ \ t_f(t_i|e_j) > 0 \\ P_U(t_i|Me_j)\ si\ t_f(t_i|e_j) = 0 \end{cases} \qquad (6)$$

D'où,(4) devient :

$$P(Q|e_j)\prod_{t_i \in e_j} P_S(t_i|Me_j) \times \prod_{t_i \notin e_j} P_U(t_i|Me_j) \qquad (4')$$

Avec :

$P_S(t_i|Me_j) = P_I(t_i|Me_j)$ **:** la probabilité associée à un terme de la requête t_i présent dans le modèle de langue de l'élément e_j calculée par la formule *(5)*.

$P_U(t_i|Me_j)$ **:** La probabilité associée à un terme de la requête absent dans l'élément e_j. Pour l'estimation de cette probabilité l'auteur dans [Belkacem, 2015], à utiliser le modèle de document M_D, vu qu'un terme de la requête absent dans un élément en cours peut apparaitre dans les éléments voisinage ou dans autre éléments de même document, car l'importance d'un terme absent dans un élément est donnée par l'importance de sa présence dans le document tout entier, ce qui donne:

$$P_U(t_i|Me_j) = P(t_i|M_d)$$

Ainsi :

$$P_U(t_i|Me_j) = P(t_i|M_d) = \frac{\sum_{i=1}^{Nf} t_f(t_i,e_j)+1}{\sum_{i=1}^{Nf} |e_j|+1} \qquad (7)$$

Qui est un rapport entre la fréquence du terme t_i dans tous les nœuds feuilles du document d, pour laquelle nous avons appliquée un deuxième lissage étant le lissage de Laplace pour la fréquence « ajouter-un » et la taille du document en nombre de termes augmenter de un, afin d'éviter d'avoir des probabilités supérieur à 1.vue que les termes d'un document ne sont pas distribués équitablement entre les éléments constituant le document, une distribution des termes s'avère nécessaire . Cette distribution des termes et appliquée pour les documents XML tout en adaptant la nouvelle granularité « élément » dans le calcul de la probabilité d'importance $P_I(t_i|M_D)$ d'un terme ti dans le modèle du document *Md*.

$$P_I(t_i|M_d) = \frac{P(t_i|M_d)}{\sum_{j=1}^{Nf} P\left(t_i\middle|M_{e_j}\right)} \qquad (8)$$

Avec:

$P(t_i|M_d)$ Calculée par *(7)* et $P\left(t_i\middle|M_{e_j}\right)$ calculée par *(6).*

2.5. Expansion du modèle de l'élément avec le modèle du document et celui de la collection :

Les termes d'une requête ne sont pas tous présent dans un élément, tendis qu'ils peuvent apparaitre dans d'autres éléments voisins ou dans l'arbre du document, ou encore ne pas apparaitre dans tous le document, mais peuvent être présent quelque part dans l'un des arbres de la collection.

Afin d'éviter de pénaliser une requête ayant un terme absent dans un document, les auteurs de la RI classique ont adaptés une expansion au modèle de la collection.

Dans [achmoukh, 2006], l'auteur a proposé un modèle qui combine le modèle du document et celui de la collection :

$$P_I(t_i|D) = \lambda_1 P_I(t_i|M_D)\lambda_2 P_I(t_i|M_c) \qquad (9)$$

Avec :

$\lambda_1 + \lambda_2 = 1$ *et* $\lambda_1 = 0{,}5$ $\lambda_2 = 0{,}5$

et

$$P_I(t_i|e_j) = \lambda_1 P_I(t_i|M_{e_j}) + \lambda_2 P_I(t_i|M_d) + \lambda_3 P_I(t_i|M_c) \qquad (10)$$

Avec :

$\lambda_1 + \lambda_2 + \lambda_3 = 1$ *et* $\lambda_1 = 0{,}5$ $\lambda_2 = 0{,}3$ $\lambda_3 = 0{,}2$

Les valeurs de $\lambda_1, \lambda_2, \lambda_3$ sont des constantes.

$\lambda_1 = 0{,}5$ la va leur associée au modèle d'élément l'intéressant dans notre cas.

$\lambda_2 = 0,3$ la valeur associée au modèle de document le fait qu'un élément fait partie d'un document.

$\lambda_3 = 0,2$la valeur associée au modèle de la collection est utilisé juste comme lissage des probabilités nulles.

$$P_I(t_i M_c) = \frac{\sum_{j=1}^{Nfc} tf(t_i e_j)}{\sum_{j=1}^{Nfc} |e_j|} \times \frac{1}{|ec_i|} \qquad (11)$$

Avec :

Nfc est le nombre de nœuds feuilles dans la collection.

$|e_{ci}|$est le nombre d'éléments contenant le terme *ti* dans la collection.

D'où, (1) devient :

$$Score(e_j, Q) = P(e_j|Q) = \prod_{t_i \in c} P_i(t_i, e_j) \times P(e_j) \qquad (12)$$

3. *Algorithmes de recherche* :

Pour lancer la recherche avec notre modèle de langue, le SRI doit suivre une succession d'algorithmes implémentant l'ensemble des formules précédente.

3.1 Algorithme (1): « Calcul des scores »

Le processus de recherche par notre modèle se base sur cet algorithme, qui implémente la formule (3) « la formule de base et la variante », et englobe les autres algorithmes, sa structure est la suivante :

dcel: est la taille du document en nombre d'éléments.

Entrées = {collection C de documents XML , requête Q}

Sorties = {$élément_i$, $score_i$}

Debut

Tant que C n'est pas vide faire

Pour tout document d_i contenant *dcel* éléments faire

Pour (j=*1* à $dcel(d_i)$) faire

Debut

Calculer la probabilité $P(e_j)$ *par la formule 3.1 ou 3.2*

Calculer la probabilité d'inférer Q à partir de e_j: $P(Q/e_j)$ algorithme (2) pour la formule (4) ou algorithme (6) pour la formule (4')

//selon le lissage choisi

Calculer la probabilité de score de e_j pour Q: formule (12)

Fin

Fait

Fait

Fait

Fin

3.2 Algorithme(2) : «Calcul de similarité »

Cet algorithme permet de calculer la probabilité de similarité entre la requête *Q* est l'élément *ej* représentant la probabilité d'inférer la requête *Q* de l'élément *e* il implémente la formule (4) et sa structure est la suivante:

Entrées = {*élément* e_j , *requête* Q = $t_1, t_2, \dots t_n$}

Sortie = valeur de similarité *P*

Constantes: $\lambda_1 = 0.5$, $\lambda_2 = 0.3$, $\lambda_3 = 0.2$

Variable: *P*

Debut

Tant que Q ≠ null faire

Debut

Pour le terme t_i faire

Debut

Calculer $P_l(t_i/Me_j) = P_{le}$ par l'algorithme (3)

Calculer $P_l(t_i/Md) = P_{ld}$ par l'algorithme (4)

Calculer $P_l(t_i/Mc) = P_{lc}$ par l'algorithme (5)

$P_i := \lambda_1 P_{le} + \lambda_2 P_{ld} + \lambda_3 P_{lc}$

// c'est la probabilité $P_l(t_i/e_j)$ *calculée par la formule (10)*

Fin

Fait

$Q := Q \setminus \{ t_i \}$

Fin

Fait

Calculer et retourner le produit des $\prod P_{i(\overline{1,\dots n})}$

Fin

3.3 Algorithme(3) :«Calcul d'importance d'un terme dans un élément donné»

Cet algorithme permet de calculer pour un terme donné de la requête, son importance dans le modèle associé à un élément de l'arborescence du document XML concerné, en utilisant la formule (6), sa structure est la suivante:

Entrées = {un modèle d'élément: Me_j , un terme t_i}

Sortie = valeur de $P_I(t_i/Me_j)$

Variables: P,S

Debut

  Pour le terme t_i faire

    Debut

      Si $tf(t_i/e_j) \neq 0$ alors

        Debut

          P := $P(t_i/ej)$ calculé par la formule (5)

        Fin

      Si non

        P := $P(t_i/Md)$ calculé par la formule (7)

      Fsi

    Retourner P

  Fin

  Fait

Fin

3.4 Algorithme(4) :«Calcul d'importance d'un terme dans un document»

Cet algorithme nous permet de calculer la probabilité d'importance d'un terme ti dans le modèle Md. associé à un document d donné, par la formule (7), sa structure est la suivante :
Nf(d): est le nombre de nœuds feuilles du document *d*.

Entrées = { document d = ensemble d'éléments e_j , le terme t_i }

Sortie = valeur de $P_I(t_i/Md) = P_I$

Variables: Ptd, $Pte = 0$

Debut

$Ptd := P(t_i/Md)$ calculée par la formule (7)

Pour k := *1* à Nf(d) faire

$Pte := Pte + P_I(t_i/Me_j)$

//avec $P_I(t_i/Me_j)$ calculée par l'algorithme (3)

Fait

$P_I := Ptd / Pte$

Retourner P_I

Fin

3.5. Algorithme(5) :«Calcul de l'importance de terme dans la collection»

Cet algorithme nous permet de calculer la probabilité d'importance d'un terme de la requête utilisateur dans le modèle de la collection de documents XML, il implémente la formule (11), sa structure est la suivante :

*Nf(C) :*est le nombre de nœuds feuilles dans toute la collection C.

Entrées = { ensemble des nœuds feuilles e_j de la collection XML , un terme t_i }

Sortie = valeur de $P_I(t_i / Mc) = P$

Variables: $Stf_C = o$, $St = o$

// *Stfc:* Somme des tf du terme dans tous les nœuds de la collection

// *St:* somme des tailles des nœuds feuilles de la collection en nombre de termes

Debut

Pour j:= 1 à Nf(C) faire

$Stf_C := Stf_C + tf\ (t_i / e_j)$

//avec $tf\ (t_i / e_j)$ *est la fréquence brute de* t_i *dans* e_j

Fait

Pour k := 1 à Nf(C) faire

$St := St + |e_k|$

Fait

$P := Stf_C / (St * |ec_i|)$

Retourner P

Fin

3.6. Algorithme(6) : «Calcul de la probabilité P(Q/ej)»

Cet algorithme permet de calculer la probabilité conditionnelle *P(Q/ej)* pour chacun des éléments de la collection XML, dans le cas du lissage par distribution des probabilités, il implémente la formule (4'), sa structure est la suivante:

Entrées = { *requête* $Q = (t_1, t_2, \dots t_n)$, *élément* e_j }

Sortie = valeur de $P(Q/e_j)$

Variable $P := 1$

Debut

Pour $i := 1$ à n faire

Si ($t_i \in e_j$) alors $P := P * P_I(t_i/Me_j)$

//*avec* $P_I(t_i/Me_j)$ calculée par la formule (5) algorithme (3)

Si non

$P := P * P_I(t_i/Md)$

//*avec* $P_I(t_i/Md)$ calculée par l'algorithme(4)

Fsi

Fait

Retourner P

Fin

Conclusion :

Dans ce chapitre, nous avons présenté la démarche à suivre pour rechercher dans une collection du documents XML, par le modèle de langue proposé, commençant par une description détaillée de notre contribution, ensuite, la définition d'un ensemble de formules pour le calcul de score d'un élément pour une requête, ainsi une succession d'algorithmes qui décrit le processus de recherche par un SRI basé sur ce modèle de langue pour retourner l'élément le plus pertinent possible à une requête utilisateur. Et comme nous avons vu, ce modèle de langue exploite complètement la structure du document XML, le considérant comme un document plats basant sur la taille d'un document en nombre d'éléments (car le document XML est un arbre d'éléments), la taille moyenne d'un document et sur la taille d'un document en nombre de nœuds feuilles (vu que l'information textuelle se trouve au niveau des nœuds feuilles).

Chapitre IV
Implémentations et Tests

Introduction :

Dans ce chapitre nous allons présenter l'environnement de développement et les outils d'implémentation utilisés pour la recherche par le modèle de langue proposé, ainsi que l'ensemble d'éléments de test nécessaires exploités pour la procédure d'implémentation de ce dernier, ensuite interpréter les résultats obtenus en utilisant des mesures de la compagne d'évaluation INEX, afin de bien comprendre le but de notre contribution

1. Environnement et outils d'implémentation :

L'application à été réalisé sous une plateforme Unix (UBUNTU 10.10) montée dans une VMware Workstation. Eclipse comme environnement de développement, et le choix de Java comme un langage de programmation, car le code source de notre système de recherche Xfirm est entièrement écrit en java.

2. Protocole d'évaluation :

Pour la réalisation de nos tests, nous avons utilisé des échantillons des éléments d'évaluation de la compagne INEX :

a. Les requêtes de tests :

Pour pouvoir comparer fidèlement les résultats obtenus par les formules variantes utilisées dans notre cas et celles de base testés dans [belkacem, 2015] ; nous avons utilisé les mêmes requêtes et les meme corpus documentaire que dans [belkacem, 2015].

Les tests sont effectués dans le cadre d'INEX. A cet effet, nous avons utilisé un échantillon de 10 requêtes CO (Content Only) extraites des124 requêtes de la compagne INEX 2006.

Le tableau suivant montre l'ensemble des requêtes utilisées :

Le numéro de la requête	Le titre de la requête
289	emperor "Napoleon I" Polish
292	"genetic algorithm"
295	software intellectual property patent license
297	"cool jazz" "West coast" musician
323	founder ikea
367	true story films best director or movie award
393	Wireless devices "Health Hazards"
408	"electroconvulsive therapy" depression
410	Hybrid Vehicles -biology "fuel efficiency" "fuel sources" model engine
413	Coordinates and Population of capital cities of Europe

Tableau IV.1 : Les 10 requêtes de Test

b. La collection de Test :

Pour nos tests, nous avons utilisé un échantillon de 1000 documents différents, extraits de toute la collection d'INEX 2006 :

	La collection INEX	**La collection de Test**
La taille de la collection :	**4.6 Go**	**17,1 Mo**
Le nombre de documents dans :	**659388 répartis en22 ensembles**	**1000**
Le nombre d'éléments dans :	**50.000.000**	**191549**

Tableau IV.2 : Caractéristiques de la collection : INEX / Test.

3. La phase d'implémentation :

Pour la recherche avec le modèle de langue proposé dans le cadre d'INEX sur le système de recherche XFIRM, et renvoie les résultats obtenus par le calcul de similarité, nous avons mis en place les algorithmes de recherche présentés dans le chapitre précédent, ainsi développé des modules correspondant à les méthodes de calcul des probabilités de similarité associées à ce modèle dans la classe « *Similarity* », afin de récupérer le score associés à tous les nœuds de tous les documents de la collection pouvant répondre à la requête utilisateur.

4. Lancement des tests :

Le lancement des tests dans le cadre d'INEX nécessite le démarrage d'Oracle et de lancer la console sqlplus en administrateur, ainsi le lancement de serveur de BDD sous un terminal avec les commandes suivantes :

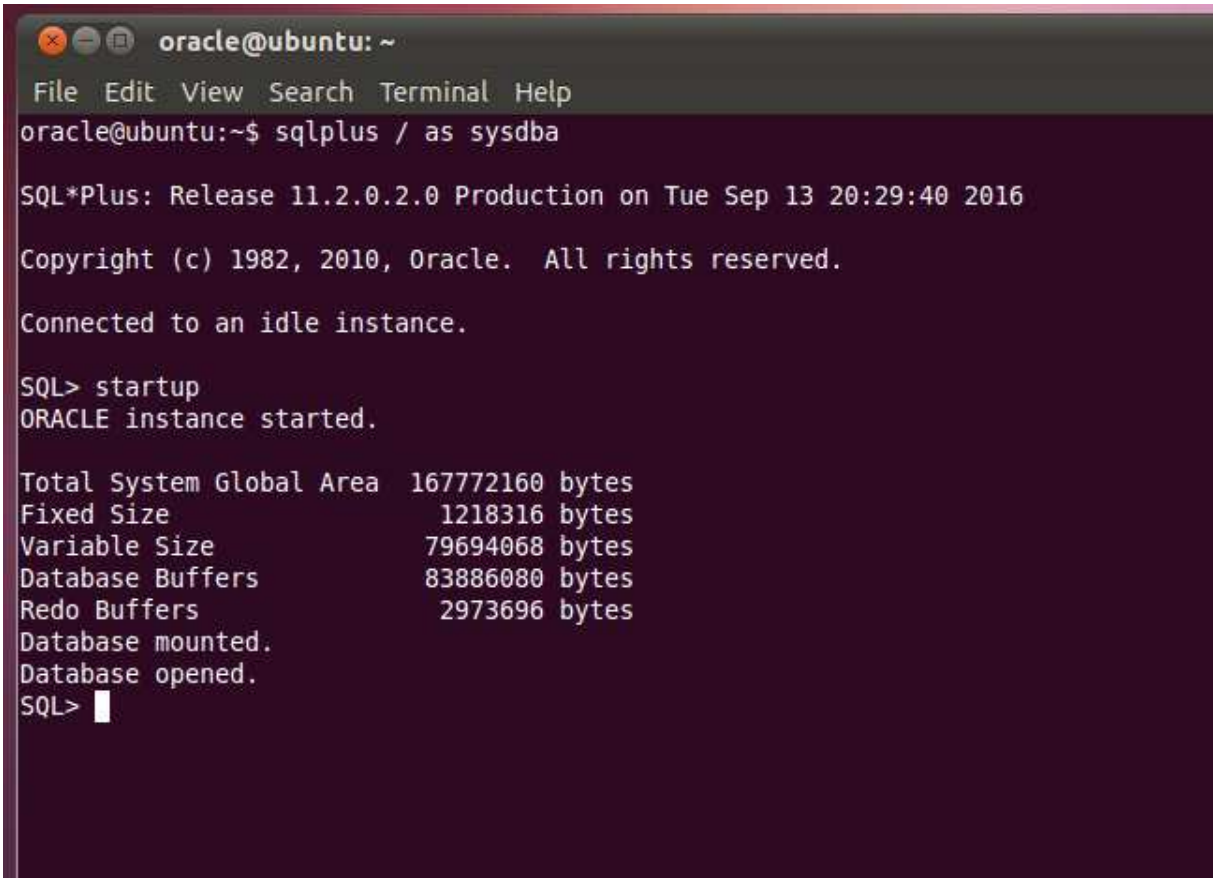


Figure IV.1 : Illustration des commandes de lancement du SGBD du système en lignes de commande.

À partir d'un deuxième terminal, se placer dans l'environnement d'exécution nous avons configuré un listener pour les requêtes de la BDD, par la commande suivante : *˜$ netca.*
Après les configurations de listener nous pouvons lancer le modèle de recherche par la commande suivante :
˜$ java xfirm/inex/InexLang p1 p2 p3 p4
Avec :
p1 : chemin vers le fichier de configuration de la BDD Oracle

p2 : chemin vers le fichier contenant les requêtes de test à exécuter

p3 : chemin vers le fichier résultat qui va contenir les résultats de recherche

p4 : le nom de la collection de recherche indexée dans la BDD.

La figure suivante montre un exemple de lancement d'une requête de test :

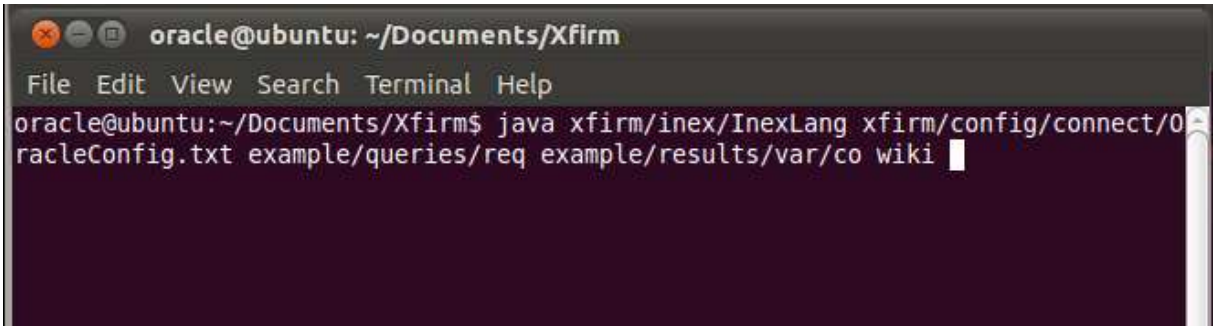


Figure VI.2 : la commande correspond au lancement de la recherche d'une requête avec la formule variante.

5. La mesure d'évaluation :

Pour évaluer les résultats de la recherche nous avons utilisé la classe *Resultat_final* pour calculer la valeur *nxCG* des scores :

$$nxCG[I] = \frac{xCG[I]}{xCI[I]} \qquad (IV.1)$$

xCG représente la somme des *i* premiers scores obtenus pour les résultats de les formules variantes [3.2 et 5.2].
xCI représente la somme des *i* premiers scores obtenus pour les résultats de les formules de base [3.1 et 5.1].
Le principe consiste à calculer la valeur *nxCG* pour 5 rangs, ainsi *i* aura les valeurs (5, 10, 25, 50, max) qui représentent respectivement les nombre des premiers résultats considérés et la totalité des résultats.

Cette valeur est calculée pour les résultats des tests obtenus en utilisant «les formules de base 3.1 et 5.1 », ainsi pour ceux obtenus en utilisant «les formules variantes 3.2 et 5.2 », dans le but de comparer les résultats de ces deux formules et de décider des meilleurs formules du modèle proposé à considérer.

6. Tests et résultats :

Pour décider de la meilleure proposition pour le calcul de la probabilité associée à un élément résultat (la formule 3.1 ou sa variante 3.2, chapitre III), et le calcul de la probabilité de vraisemblance maximale (la formule 5.1 ou la 5.2, chapitre III) nous avons lancé des tests intégrant les formules [3.2 et 5.2] sur le système de recherche, afin de les comparer avec les résultats obtenus par les tests intégrant les formules [3.1 et 5.1].

Et dans ce qui suit l'ensemble des tableaux résumant les résultats obtenus pour les différentes formules de ce modèle de langue proposé et l'histogramme associé aux valeurs du Gain pour les différents niveaux de chaque test.

Gain / Requête	Gain à 5	Gain à 10	Gain à 25	Gain à 50	Gain totalité
289	**0.9901**	**0.5386**	**0.5871**	**0.5427**	**0.3272**
292	**0.9983**	**0.5445**	**0.5795**	**0.5441**	**0.2634**
295	**0.9778**	**0.5271**	**0.5624**	**0.4878**	**0.0724**
297	**0.9961**	**0.5521**	**0.6562**	**0.6483**	**0.0435**
323	**0.9102**	**0.4649**	**0.4677**	**0.3973**	**0.0525**
367	**1**	**0.5611**	**0.4771**	**0.3736**	**0.2372**
393	**1**	**0.5573**	**0.6597**	**0.4936**	**0.1107**
408	**1**	**0.5573**	**0.6597**	**0.4936**	**0.1334**
410	**1**	**0.5611**	**0.6767**	**0.7073**	**0.0598**
413	**1**	**0.5611**	**0.6032**	**0.5517**	**0.1370**
La moyenne	**0.9895**	**0.5425**	**0.5929**	**0.5240**	**0.1437**

Tableau IV.3 : Les résultats obtenus pour le modèle de langue avec les formules variantes (3.2 et 5.2)

Gain / Requête	Gain à 5	Gain à 10	Gain à 25	Gain à 50	Gain totalité
289	0.8888	0.4237	0.3672	0.3105	0.1457
292	0.8820	0.4124	0.2736	0.1678	0.0159
295	0.9877	0.5414	0.5461	0.4436	0.0604
297	0.9224	0.4719	0.4980	0.4531	0.0669
323	0.7907	0.3171	0.1817	0.1081	0.0129
367	1	0.5611	0.6002	0.5553	0.0422
393	0.9396	0.4813	0.4365	0.3708	0.0313
408	0.9914	05203	0.4597	0.3245	0.0246
410	0.9877	0.5362	0.5784	0.5146	0.3545
413	0.9234	0.4904	0.4296	0.2789	0.0208
La moyenne	0.9319	0.4756	0.4371	0.3527	0.0775

Tableau IV .4: Les résultats obtenus pour le modèle de langue avec les formules de base (3.1 et 5.1).

Tableau comparatif :

Niveau / Systèm		Gain à 5	Gain à 10	Gain à 25	Gain à 50	Gain sur la totalité
Le modèle de langue proposé	**Les formules de base (3.1et 5.1)**	**0 .9319**	**0.4756**	**0.4371**	**0.3527**	**0.0775**
	Les variantes (3.2 et 5.2)	**0.9895**	**0.5425**	**0.5929**	**0.5240**	**0.1439**
Taux d'amélioration		**5 ,761 %**	**6,691 %**	**15,584 %**	**17,13 %**	**6.637 %**

Tableau IV.5 : Tableau comparatif des résultats obtenu par le modèle proposé.

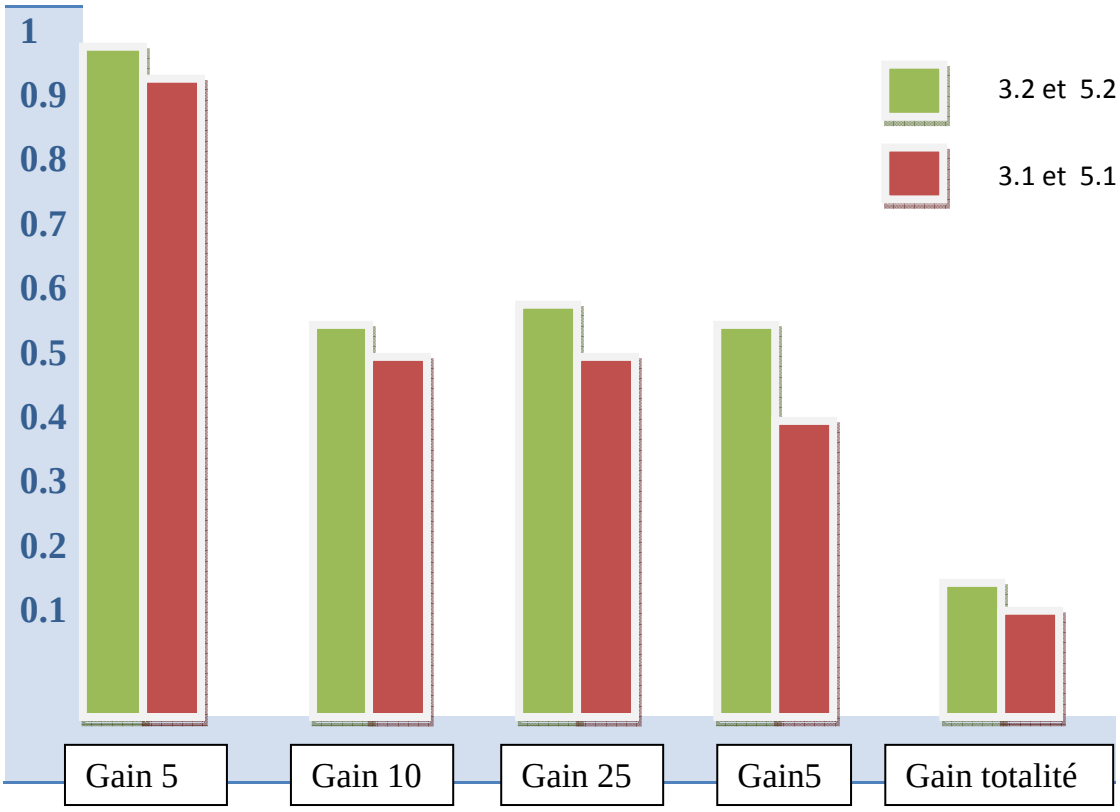


Figure VI.3: L'histogramme associé au tableau comparatif entre les deux Formules de modèle proposé.

Interprétation :

L'objectif de la comparaison des résultats obtenus avec les formules de base avec ceux des formules variantes est de pouvoir décider des paramètres à prendre en compte pour l'implémentation du modèle de langue proposé dans [Belkacem, 2015].

L'histogramme et le tableau comparatif ci-dessus montrent la différence entre le Gain obtenus pour les cinq niveaux (à 5, 10, 25, 50 et le Gain totale) des premiers résultats, nous remarquons que l'adaptation de les formules [3.2 et 5.2] a aboutis à des améliorations jusqu'à 5 ,761 % pour les 5 premiers résultats, et 6,691% , 15,584 %,17,13 % pour les 10 et 25 et 50 premiers résultats respectivement, et d'une amélioration de **6.637 %** sur la totalité par apport à l'adaptation des formule [3.1 et 5.1].

A savoir que Les formules de bases prennent en compte, dans le calcul de la probabilité d'avoir un élément comme résultat, la taille de l'élément en nombre de termes ainsi que la taille du document en nombre de termes comme paramètres, tandis que les formules variantes, en plus de la taille de l'élément en nombre de termes, elle prend en compte le fait que l'information textuelle réside au niveau des nœuds feuilles, et considère le nombre de nœuds feuilles et la taille moyenne de l'élément dans le calcul de la probabilité, ce qui nous a permet d'aboutir à des améliorations remarquables dans les résultats obtenus.

Donc, grâce à l'exploitation des différentes caractéristiques des documents XML nous avons aboutis à des résultats satisfaisants.

Conclusion :

Dans cette partie, nous avons présenté la démarche d'implémentation, les tests, l'ensemble des résultats obtenus par les formules variantes, ainsi qu'une étude comparative par rapport aux formules de base. Nous avons constaté que le modèle de langue avec les formules variantes présente une amélioration du gain sur les différents niveaux considérés, par rapport au modèle de base.

Conclusion Générale

Conclusion Générale :

Le travail présenté dans ce mémoire s'inscrit dans le cadre de la Recherche d'information (RI) et plus particulièrement la Recherche d'information structurée (RI XML). Nous nous sommes principalement intéressés à l'adaptation d'un modèle de langue à la nouvelle granularité d'information. Vu que les documents XML sont un ensemble d'éléments, donc les résultats des recherches seront soit un élément, un ensemble d'éléments ou le document entier.

Dans ce contexte, les concepts de la recherche d'information classique doivent être adaptés à la recherche d'information structurée. Un document XML est vu come un arbre d'élément, de ce fait, une collection est un ensemble d'arborescence dont les nœuds feuilles sont porteur de l'information textuelle ce qui implique que la recherche d'information dans les documents XML doit prendre en compte ces nouvelles caractéristiques.

L'objectif d'un modèle de langue est d'estimer la probabilité qu'une requête utilisateur puisse être générée par un documents (un modèle de document), dans ce contexte plusieurs travaux ont été présentés, et parmi ces derniers on trouve celui présenté dans [Achemoukh,2006], qui prend en compte la taille du document. Pour l'adaptation à la nouvelle granularité d'information peu de travaux sont présentés, parmi lesquels nous avons l'approche proposée dans [Belkacem , 2015] , qui propose un modèle de langue qui détermine la probabilité d'inférer la requête utilisateur par un modèle d'élément.

Notre travail consistait aux tests et à l'évaluation du modèle de langue présenté dans [Belkacem, 2015] , ce modèle de langue adapté aux documents XML estime le score de pertinence des éléments d'un document en prenant en compte la taille d'éléments en nombre termes, et le nombre de nœuds feuille ainsi que la taille moyenne d'un élément dans ce document.

Afin de répondre à une requête utilisateur en lui fournissant un élément ou un ensemble d'élément comme résultat, nous avons estimé la probabilité de similarité (élément- requête) qui revient à calculer la probabilité d'avoir un élément comme résultat qui peut être calculer soit par la formule de base [3.1] (qui a été déjà testé et évalué par l'auteur avec la probabilité de vraisemblance [5.1], soit par la formule variante [3.2] (que nous avons testé, avec la formule de vraisemblance [5.2]), et cela, en commençant par le calcul de la probabilité d'importance d'un terme de la requête dans un élément, ensuite appliqué le produit de ces probabilités pour tous les termes de la requête (c'est la distribution aux termes de la requête), et enfin le produit entre la valeur de similarité et le produit associée à l'élément, ce qui nous donne le score de pertinence d'un élément pour une requête utilisateur.

Après avoir implémenté et testé l'approche proposée dans [Belkacem,2015],nous avons effectué une comparaison entre les résultats des scores obtenus par les formules

de tests dans [Belkacem,2015] et les résultats de tests obtenus par les formules considérés dans notre travail en utilisant la mesure du gain que nous avons calculée sur les 5 rangs (5,10,25,50,max).

Nous avons remarqué que les résultats de nos tests ont abouti à un gain de +6% sur la totalité des résultats et jusqu'à +17% sur les 50 premier résultats.

Enfin nous concluons qu'avec le modèle de langue implémenté nous avons obtenus des résultats satisfaisants, grâce à la prise en compte des caractéristiques des documents XML comme paramètres dans les formules de calcul des scores.

Perspectives :

Nos perspectives se résument à ce qui suit :

- Dans l'approche proposée nous avons deux algorithmes de lissage, nous avons testé l'algorithme (2) et nous souhaitons tester l'algorithme (6) et comparer les résultats obtenus par les deux algorithmes de lissage. [Belkacem,2015]
- Nous avons utilisé, dans nos tests, un échantillon de 10 requêtes extraites de 124 requêtes de la campagne INEX 2006, nous envisageons d'étendre la collection de tests jusqu'a 124 requêtes.
- Nous avons effectué des tests sur une collection de 1000 documents, nous souhaitons tester cette approche sur toute la collection INEX 2006.
- Les requêtes utilisé dans nos test sont de type CO (centent Only), nous envisageons d'utiliser des requêtes de type CAS (des requêtes portant sur le contenu et la structure).

Bibliographie

Bibliographie :

- **[ALILAOUAR, 2007]** Contribution à l'interrogationflexible de données semi-structurées., 2007
- **[Achemoukh, 2006] :** contenant un modèle de langue pour la RI classique se basantsur la taille d'un document. LARI-UMMTO 2006.
- **[Allorge, 2000]** Stéphane Allorge, *" XML"*, département ASI, 2000.
- **[Amirouche, 2008]** « Contribution à la définition de modèles de recherched'informationflexibles basés sur les CP-Nets »,Fatiha BOUBEKEURAMIROUCH. Toulouse, 2008
- **[Baeza et Ribeiro, 2011]** Baeyza-Yates, R, Ribeiro-Neto, B.A. modern information retreival. *Pearson Education Ltd*., Harlow, UK, 2nd edn, 2011
- **[Ben Aouicha, 2009]** « Une approche algébrique pour la recherche d'information structurée »., 2009
- **[BELKACEM, 2015]**THESE, Proposition et expérimentation d'un modèle de langue pour la RI dans les documents XML, 2015
- **[Boughanem, 2000] :** Mohand Boughanem, « *Modèles de langue pour la recherched'information* **».**Institut de Recherche en Informatique de Toulouse.
- **[Boughanem et al, 2004]** M. Boughanem, W. Kraaij, J.Y. Nie, *Modèles de langue pour larecherche d'informations*, Les systèmes de recherche d'informations - Modèles conceptuels, ed. M. Ihadjadene, Hermes, pp. 163-184, 2004
- **Cours Master Recherche 'Paris 13'** Modèles en Recherche d'Information Modèles de langue.
- **Cours XMLOlivier Carton,**L'essentiel de XML.
- **Cours septembre 2015**, La structure des documents XML.

- **[CYRIL, 2013]**THÈSE« Impact de la structure des documents XML sur le processus d'appariement dans le contexte de la recherche d'information Semi-Structurée », Université Toulouse, 2013
- **[Fellag, 2006**] THESE recherche d'information dans les documents semi-structurés XML, 2006
- **[Fellag et boughanem, 2010]**Traitement des requêtes CO (content only) sur un corpus de documents XML, COSI, 2010
- **[Ganguly et al, 2011]**Debasis GANGULY, Johannes LEVELING, Gareth JONES, Saupurna PALCHOWDHURY, Sukomal PAL et Mandar MITRA : DCU and ISI@INEX 2010 : Adhoc and data-centrictracks. *Dans* Shlomo GEVA, Jaap KAMPS, Ralf SCHENKEL et Andrew TROTMAN, éditeurs : *Comparative Evaluation of FocusedRetrieval*, volume 6932 de *Lecture Notes in Computer Science*, pages 182–193. Springer Berlin / Heidelberg, 2011.
- **[Boughanem et al,]**Recherche d'Information Structurée vers un modèle possibiliste pour la recherche d'information dans des documents structurés.
- **[Hammache, 2014] :** Arezki Hammache, « Recherche d'information : un modèle de langue combinant mots simples et mots composés ». *LARI*-UMMTO, 2014
- **[Hiemestra, 2003]**Djoerd HIEMSTRA : Statisticallanguagemodels for intelligent xmlretrieval. *Dans* Burkhard STILLER, Georg CARLE, Martin KARSTEN et Peter REICHL, éditeurs : *Group Communications and Charges. Technology and Business Models*, volume 2818 de *Lecture Notes in Computer Science*, pages 107–118. Springer Berlin / Heidelberg, 2003.
- **[Hugues Bouchard &Jian-Yun Nie, 2006]**Modèles de langue appliqués à la recherched'information contextuelle, 2006
- **[Sauvagnat et al, 2006]**Le langage de requêtes XFIRM pour la recherche d'information dans les documents XML, 2006
- **[Lalmas, 1997]**Lalmas M. Dempster shafer s theory of evidenceapplied to structureddocuments : modellinguncertainty. In Proceedings of annual international ACM SIGIR'97 Conference, pages 110-118. Philadelphia PA, USA, 1997.

- **[Lalmas, 2009]**Mounia LALMAS : *XML Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan &ClaypoolPublishers, 2009.
- **[NAFFAKHI Najeh , 2013]** Un modèle de recherche d'information agrégée basée sur les réseaux bayésiens dans des documents semi-structurés, Univrsité TOULOUSE, 2013
- **[Pehcevski, 2006]**Pehcevski J. Evaluation of Effective XML Information Retrieval. RMIT University, Melbourne. PhDthesisAustralia, 2006.
- **[Philippe Mulhem, Jean-Pierre Chevallet, 2012]**Un modèle decontexte documentaire par doxals pondérés. Application un modèle de langue contextuel pour la recherche de documents structurés, thèse, 2012
- **[Ponte et al, 1998]** Jay M. Ponte and W. Bruce Croft, A LanguageModeling Approach to Information Retrieval, Research and Development in Information Retrieval, Proc. ACM-SIGIR, pp. 275-281, 1998
- **[Rami HARRATHI, 2010]** THÈSE sur la Recherche d'information conceptuelle dans les documents semi-structurés, 2010
- **[Robertson et al, 1994]**. Robertson S. E. and Walker S. Some simple effective approximations to the 2-Poisson model for probabilisticweightedretrieval. In Proceedings of SIGIR 1994, pages 232–241, 1994.
- **[Roelleke et al. 2009]**. Roelleke T., Wang J., Robertson S. ProbabilisticRetrievalModels and Binary Independence Retrieval (BIR) Model. Encyclopedia of DatabaseSystems 2009: 2156-2160.
- **[Sauvagnat et al, 2006]**Karen Sauvagnat, Mohand Boughanem et Claude CHRISMENT : Answeringcontentand-structure-basedqueries on xml documents using relevance propagation. *InformationSystems, Special Issue SPIRE 2004*, janvier 2006
- **[Sigurbjornsson et al , 2004]**B. Sigurbjornsson, J. Kamps, and M. de Rijke. An Element-basedApproach to XML Retrieval, 2004
- **[Tannier, 2006]**. Tannier X. Extraction et recherche d'information en langage naturel dans lesdocumentssemi-structurés.Thèse de doctorat en informatique

effectuée à l'Ecole Nationale Supérieure des Mines de Saint-Etienne et de l'Université Jean Monnet, 2006

- **[Zhai et al, 2001]**.J. Laferty and C. Zhai. Do cumentlanguage mo dels, query mo dels, and riskminimization for information retrieval. In Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.
- **[Kazai et al, 2004].**G. Kazai, M. Lalmas, and A. P. de Vries. The overlapproblem in content-oriented XML retrievalevaluation. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, United Kingdom, 2004. ACM Press, New York City, NY, USA
- **[Luk et al,2002]**R. W. Luk, H. Leong, T. S. Dillon, A. T. Chan, W. B. Croft, and J. Allan. A survey in indexing and searching XML documents. Journal of American Society for Information Science and Technology (JASIST), 53(6) :415-437, 2002.
- **[Urfist, 2004]** Cours *"Problématique Générale de la Recherche d'Information"* URFIST BretagnePays de Loire, Alexandre Serres, 2004 *http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html*

Webographie:

- http ://www.Wikipedia.com
- http://hal.inria.fr/hal-00954051
- http://www.eclipse.org
- http://www. Wikilingue_Encydia.com
- http://www.iro.umontreal.ca/~nie/IF
- http://www.commentcamarche.net/contents/702-oracle-introduction-au-sgbd-oracle
- http://msdn.microsoft.com/fr-fr/library/ms256190%28v=VS.80%29.aspx

Annexe

ANNEXE 1 : Les concepts de base de la RI

***Document* :**tout objet qui peut apporter de la connaissance exploitable et accessible par un SRI afin de constituer une réponse à un besoin en information exprimé par l'utilisateur.
C'est l'élément central de tout SRI qui est en évolution continue avec le développement des technologies du Web, peut être un texte, paragraphe, image, vidéo, page web…etc.
***Requête* :**un ensemble de mots formés par un utilisateur pour exprimer son besoin en information, peut être exprimée en langage naturel, booléen ou graphique, et considérée comme un élément déclencheur de la recherche.
***Appariement document/requête* :**C'est la fonction de comparaison entre le document et la requête, elle revient à calculer un score, supposé représenterla pertinence du document vis-à-vis de la requête, ceci en se basant sur un modèle RI bien déterminé **(voir section 4)**. Cette valeur est calculée à partir d'une fonction ou d'une probabilité de similarité notée **RSV(Q,d)**(Retrieval Status Value), entre la requête **Q** et le document **d**.
***Masse d'information (Base documentaire)* :**ou encore un corpus c'est un ensemble de documents à des contenus diversmanipulés par un SRI.
***La reformulation* :**Comme l'utilisateur exprime son besoin en information par une requête en langage naturel, il est habituel de préciser la requête sur laquelle se base le SRI pour retourner des documents plus qui répondent le mieux aux besoins en informations de l'utilisateur, car elle se base sur des mots de la requête utilisateur et les documents disponibles. Et on distingue deux types:

- ✓ ***La reformulation manuelle* :**

Il s'agit de la stratégie de reformulation de la requête la plus populaire. On la nomme aussi **réinjection de la pertinence** ou **relevance feedback**.
L'idée principale de la réinjection de pertinence est de sélectionner les termes importants appartenant aux documents jugés pertinents par l'utilisateur, et de renforcer l'importance de ces termes dans la nouvelle formulation de la requête.

- ✓ ***La reformulation automatique* :**

Dans ce type de reformulation l'utilisateur n'intervient pas.
L'extension de la requête est faite à partir d'un thésaurus qui définit les relations entre les différents termes de l'index et permet de sélectionner de nouveaux termes à ajouter à la requête initiale.
***La notion de Pertinence* :** notion clé de la RI, c'estle degré de relation (correspondance) entre le document et la requête.

La capacité (d'un système de recherche d'information) à récupérer des éléments qui satisfirent les besoins d'un utilisateur. *Dictionnaire Merriam-Webster (2012)*
TefkoSaracevic [Saracevic, 1970] propose la définition suivante de la pertinence :
La pertinence est la A d'un B existant entre un C et un D jugé par un E.
Où :
A = intervalle de la mesure
B = aspect de la pertinence (la pertinence absolue)
C = un document
D = contexte dans lequel la pertinence est mesurée (y compris le besoin d'information)
E = le juge (l'utilisateur).
Deux types de pertinence sont définis : **la pertinence système** et **la pertinence utilisateur.**
La pertinence systèmeest l'ensemble des principes qui caractérisent la fonction de correspondance dans un système de recherche d'information.
La pertinence utilisateur correspond à l'ensemble des jugements de pertinence que produit l'utilisateur du système de recherche d'information.
***L'indexation* :** Est une opération permettant d'extraire d'un document ou une requête une représentation paramétrée qui couvre au mieux son contenu sémantique.
Le résultat de l'indexation constitue le descripteur du document ou de la requête, pour objectif de construire une représentation simplifiée des documents et requêtes et à les stocker dans des structures d'index qui facilitent la recherche.

- ✓ ***Mode d'indexation* :**

Techniquement, l'indexation peut être manuelle, automatique ou semi-automatique :

- **Indexation manuelle** : chaque document est analysé par un spécialiste du domaine ou un documentaliste.
- **Indexation automatique** : chaque document est analysé à l'aide d'un processus entièrement automatisé.
- **Indexation semi-automatique (mixte)** : c'est une combinaison des deux méthodes précédentes : un premier processus automatique permet d'extraire les termes du document. Cependant, le choix final reste au spécialiste du domaine ou au documentaliste pour établir les relations entre les mots clés et choisir les termes significatifs.

- ✓ ***processus d'indexation*** : est un processus de représentation d'information, il permet la traduction de la requête ou du document d'une description brute, souvent en texte libre vers une description structurée, aussi conduit à élaborer des outils de recherche documentaire qui seront ensuite consultés pour permettre la sélection des documents qui ont répondu à la requête, regroupe un ensemble de traitements à effectuer sur un document comme l'analyse lexicale, la lemmatisation, l'élimination des mots vide, la pondération des termes.

- ***L'analyse lexicale***: Une phase qui consiste à éliminer les variantes morphologiques (genre, nombre, dérivations, ...) des mots.
- ***L'élimination* des mots vides :** Les mots composant le texte sont extraits et les mots vides (prépositions, pronoms personnels,…) éliminés, l'élimination de ces mots peut se faire en utilisant une liste dressée de mots vides (également appelée anti-dictionnaire).
- ***La lemmatisation* :** opération consiste à regrouper les formes occurrentes d'un document ou d'un texte sous des adresses lexicales (mots de la même famille), et le but de cette étape c'est faciliter à l'utilisateur la façon de former sa requête (chaque mot prendre sa forme canonique (exp : pluriel, singulier ou masculin, féminin).

✓ ***La pondération des termes* :** permet d'exprimer le pouvoir discriminant d'un terme dans un document, en effectuant à chaque terme $\mathbf{t}_i$ d'un document **d** un poids $W_{t,d}$ qui exprime le degré de représentativité du terme dans le document, dans le but de distinguer les documents les uns des autres. Intuitivement, plus en terme apparait dans un document plus est important dans ce document.

La fonction de pondération s'exprime en $[tf * Idf]$.

***tf* :** terme frequency, est la fréquence d'occurrences du terme **t**dans le document **d**

***Idf* :**inversed document frequency,est la fréquence documentaire du terme **t**(i.e. la proportion de documents de la collection qui contiennent **t).**

- ***Calcul du poids d'un terme dans un document :***

- ***Mesure simple***: inverse du nombre de documents de la collection contenant le terme :

$$W_{i,d} = tf_{id} * \left(\frac{1}{df_i}\right) \quad (1)$$

- ***Mesure fréquemment utilisée*** : log du quotient du nombre de documents dans la collection par le nombre de documents contenant le terme :

$$W_{i,d} = tf_{id} * log\left(\frac{N}{df_i}\right) \quad (2)$$

- Le poids d'un terme augmente :

• avec la fréquence du terme dans un document.

•avec la rareté du terme dans les documents du corpus.

- Quelques fonctions de calcul de tf**:**

$$tf = \frac{n_{oc}}{tf_{max}} \quad (3)$$

$$tf = 0.5 + 0.5\left(\frac{n_{oc}}{tf_{max}}\right) \quad (4)$$

Et d'Idf :

$$idf = log\left(\frac{N}{N_i}\right) \quad (5)$$

$$idf = log\left(\frac{N-n_i}{n_i}\right) \quad (6)$$

$$idf = log\left(\frac{N}{n_i}\right) + 1 \quad (7)$$

Avec :

N : Nombre total de documents dans le corpus.

n_i: Nombre total de documents contenant le terme i.

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

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

***Organisation de l'index* :**c'est un processus nécessaire pour organiser et stocker les informations sélectionnées.

- ✓ ***Les modèles de base de la recherche d'information :*** c'est la fonction principale d'un SRI, dont la finalité est de fournir la liste des documents correspondant à la requête d'un utilisateur donnée, on peut spécifier trois modèles :

- Le modèle ensembliste, Le modèle algébrique, Le modèle probabiliste.

- ***Modèle booléen* (*modèle ensembliste*) :**est le premier modèle utilisé dans le domaine de la recherche d'information [Salton, 1971]. Il se base sur la théorie des ensembles. Dans ce modèle, la requête est représentée sous forme d'une expression logique. Dans cette expression, les termes d'indexation sont reliés par des opérateurs booléens NON, ET et OU. Le document est représenté par un ensemble de termes d'indexation. [Harrathi, 2010].

La vérification de l'implication logique $D_i \rightarrow Q_k$indique que le document corresponde bien à la requête.

***Evaluation des requêtes*:**un ensemble de termes construit un document D, et un ensemble de requête sous forme d'une expression logique, La Similarité *RSV(Q,D)*(Retrieval Status Value).se présente comme se suit :

RSV(ti, D) = 1 si ti ϵ d; 0 sinon.

· *RSV(Q1 ∧ Q2, D) = 1 si RSV(Q1, D) = 1 et RSV(Q2,D) = 1; 0 sinon.*

· *RSV(Q1 ∨ Q2, D) = 1 si RSV(Q1,D) = 1 ou RSV(Q2,D) = 1; 0 sinon.*

· *RSV(¬Q1, D) = 1 si RSV (Q1,D) = 0; 0 sinon.*

- ***Le modèle booléen étendu* :** comme le modèle de base présente un ensemble d'inconvénients, salton et al [salton & al, 1983] ont introduit le booléen étendu pour compléter ce dernier, En intègrent des poids d'indexation dans le document et l'expression de la requête pour que, la sélection de documents sur la base d'un appariement rapproché. La Similarité RSV(Q,D)(Retrieval Status Value).se présente :

Opérateur OR :

$$RSV(D_j, Q_k) = \left(\frac{\sum_{i=1}^{n} q_{ik}^{p} d_{ij}^{p}}{\sum_{i=1}^{n} q_{ik}^{p}}\right)^{1/p} \quad (8)$$

Opérateur AND :

$$RSV(D_j, Q_k) = 1 - \left(\frac{\sum_{i=1}^{n} q_{ik}^{p}\left(1 - d_{ij}^{p}\right)}{\sum_{i=1}^{n} q_{ik}^{p}}\right)^{1/p} (9)$$

P : une constante (0<= P <=∞)

q_{ik} : le poids du terme $\mathbf{t}_i$dans la requête $\mathbf{Q}_k$.

- ***Le modèle vectoriel* (*modèle algébrique*) :** C'est le plus utilisé dans la RI. Il repose sur l'aspect quantitatif des documents et requêtes par rapport aux termes, dont l'idée de base est d'utiliser une représentation géométrique pour classer les documents par ordre de pertinence par rapport à une requête.

On a : T = {t_i}, i ϵ [1,…N]

Tous les documents sont décrits suivant ce vocabulaire :

- Un document Di est représenté par un vecteur $\overrightarrow{D_i}$décrit dans l'espace vectoriel R^N défini par T : $\overrightarrow{D_i}$ = (wi,1, wi,2, …, wi,j, …, wi,N), avec w_{kl} le poids d'un terme pour un document.
- Une requête Q est représentée par un vecteur Q décrit dans l'espace vectoriel R^N défini par T : $\vec{Q}$ = (w_{1q}, $w_{2,q}$,$w_{q,j}$,… $w_{q,n}$,).

La pertinence du document *Di* pour la requête *Q* est mesurée comme le degré de corrélation des vecteurs correspondants. Cette corrélation peut être exprimée par l'une des mesures suivantes :

Produit scalaire :

$$RSV(\overrightarrow{D_j}, \vec{Q}) = \sum_{i=1}^{n}(WD_{ji} * WQ_{iq}) (10)$$

Mesure de cosinus :

$$sim(D_j, Q_k) = \frac{\sum_{i=1}^{n}(WD_{ij} * WQ_{ik})}{\sqrt{\sum_{i=1}^{n} D_{ij}^2 * \sum_{i=1}^{n} WQ_{ik}^2}} (11)$$

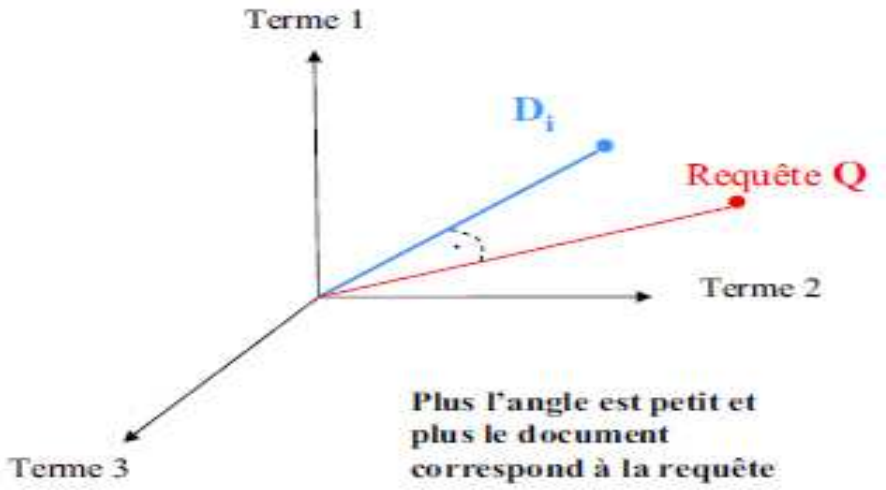
Mesure de Dice :

$$sim(D_j, Q_k) = \frac{\sum_{i=1}^{n}(WD_{ij} * WQ_{ik})}{\sum_{i=1}^{n} D_{ij}^2 * \sum_{i=1}^{n} WQ_{ik}^2} (12)$$

***Mesure de Jaccard* :**

$$sim(D_j, Q_k) = \frac{\sum_{i=1}^{n}(WD_{ij} * WQ_{ik})}{\sum_{i=1}^{n} D_{ij}^2 + \sum_{i=1}^{n} WQ_{ik}^2 - \sum_{i=1}^{n}(WD_{ij} * WQ_{ik})} (13)$$

Soit l'exemple suivant : le document D et la requête Q et les trois termes « terme1, terme 2, terme3 », plus l'angle est petit plus le document correspond à la requête.



Le graphe de correspondance.

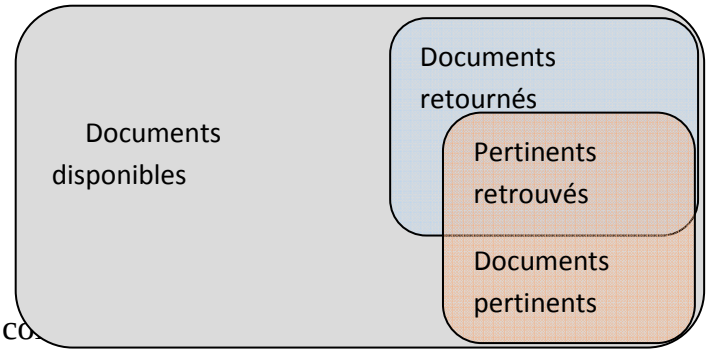
- ***Le modèle de langue (Modèle Probabiliste):*** Se base sur les probabilités.et en recherche d'information, les modèles de langue ont été introduits en 1998 par Ponte et Croft (Ponte *et al.*1998) qui voient le score d'un document face à une requête comme la probabilité que la requête soit générée par le modèle du document. En pratique et pour simplifier les calculs, il est supposé dans les modèles de langue, qu'un terme ti ne dépend que de ses (i-1) prédécesseurs. Ainsi, les modèles les plus utilisés sont les modèles unigramme, bigramme et trigramme. Nous allons étudier en détails ce modèle dans le chapitre suivant.

***Evaluation des SRI* :** pour garantir la satisfaction de l'utilisateur il est nécessaire de lui renvoyer des informations pertinentes, pour ce but une étape importante qui consiste à évaluer la performance et la capacité de tous les SRI a été apparue. On va présenter les mesures les plus utilisés :

***Rappel et précision* :**ce sont les deux paramètres utilisés pourévaluer la capacité et l'efficacité d'un SRI. La précision détermine l'aptitude d'un SRI à rejeter les

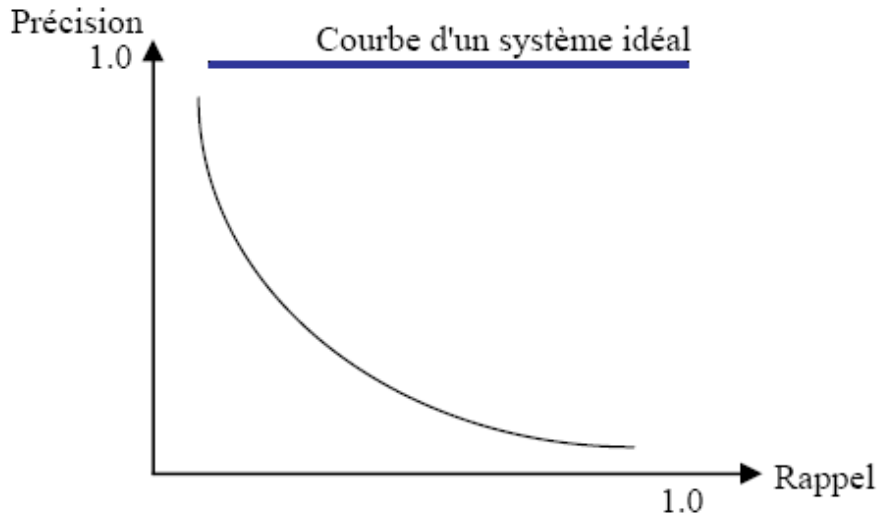
documents non pertinents vis à vis à d'une requête utilisateur **Précision=pertinents retrouvés/documents retrouvés.**

Le rappel exprime la capacité d'un SRI à sélectionner tous les documents pertinents vis à vis de cette requête.
***Rappel=pertinents retrouvés/documents pertinents*.**



Partition d'une co

La courbe précision/rappel : Le comportement d'un système peut varier en faveur de précision ou en faveur du rappel. Un système idéal devrait retourner tous les documents pertinents c'est à dire un taux de précision et de rappel égal à 100%. Ainsi, pour un système, on a une courbe de précision-rappel qui a en général l'aspect suivant :



Courbe de rappel/précision.

La collection de tests : Depuis 1970 des compagnies spécialisées a commencés des tests en RI pour évaluer les stratégies de recherche. L'intérêt est de pouvoir comparer des SRI entre eux à l'aide des critères objectifs (les réponses idéales que l'utilisateur souhaite recevoir).

La compagnie d'évaluation TREC (Text Retrieval Conference), est un grand projet qui a été initié par la DARPA (Defense Advanced Research Project Agency), et co-organisé par le

NIST Nationnal Institut of Standards and Technology), afin d'encourager le domaine de la recherche documentaire sur des grandes collections de test.

Parmi les plus importants projets aussi : la collection ISI. La compagne CLEF (CrossLanguage Evaluation Form), ou La collection CACM.

Actuellement la collection TREC (***T****ext****RE****trieval****C****onference*) commencée en 1992, est la plus utilisée en RI, [Harman, 1992] [Voorhees, 2000] est un programme international qui bénéficie du support du NIST (***N****ational* ***I****nstitute of* ***S****tandards and* ***T****echology*) et du DARPA (***D****efense****A****dvanced* ***R****esearch****P****rojects****A****gency*), il offre une plateforme d'évaluation à grande échelle pour les techniques de RI,

Et une très large collection de documents de sources très variées : Financial Time, Résumés de publications etc… organisées en sous collections, qui évoluent d'année en année. Pour chaque session de TREC, un ensemble de documents et de requêtes est fourni.

ANNEXE 2 : Les technologies XML

. XSL (eXtensibleStylesheetLanguage) est le langage qui permet d'écrire des feuilles de style. Une feuille de style est constituée d'un ensemble de règles de transformations, s'appliquant chacune à un ou plusieurs nœuds de l'arbre et permettant de transformer ce nœud en un nouveau nœud de l'arbre résultat. Est un langage déclaratif utilisé pour transformer un document XML en un autre html ou texte ou encore XML.

. *Le langage XSLT* décrit des règles pour transformer un document XML. Ces règles s'appliquent chacune à un ou plusieurs nœuds de l'arbre et spécifient la transformation à effectuer sur un nœud pour le transformer en un nouveau nœud de l'arbre résultat.

. ***Le langage XHTML*** est destiné à devenir le successeur de HTML. C'est un sous-ensemble de XML, largement compatible avec HTML et qui remédie aux principaux défauts de ce dernier auquel il apporte la rigueur et la clarté. XHTML combine tous les éléments de HTML avec la syntaxe de XML.

. **SVG** est le langage **S**calable**V**ector**G**raphics (graphiques vectoriels redimensionnables) est un langage permettant de décrire des graphiques à deux dimensions en XML.

. MathMLil s'agit d'un langage de syntaxe XML permettant de décrire la structure d'un texte mathématiques et de ses formules et d'en permettre l'affichage par un navigateur Web, mais qui permet aussi de décrire un contenu mathématique effectif.

. *Les Espaces de Noms (Namespaces*) 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 URI19 (**U**niform **R**esource **I**ndentifier).

. XPointer (XML Pointer) est utilisé pour adresser des fragments de documents XML. Xpointer est construit sur le langage Xpath, de qui il a adopté les mêmes fonctionnalités et la même, syntaxe cependant il offre des multiples extensions à Xpath.

. XLinkest une Recommandation du W3C depuis 2001, il permet la construction de liens XML semblables aux liens HTML mais avec davantage de puissance et d'extensibilité.

XQueryest un langage intéressant avec quelques idées insolites, où tout est une expression qui renvoie une valeur. Un programme ou un script XQuery est juste une expression, avec une fonction facultative et d'autres définitions.

. *RDF* : (Ressource Description Framework) est un standard décrivant les ressources : personnes, lieux, documents… et est un framework contenant un modèle de données : langage et syntaxe. Il a été créé en 1999 en tant que norme sur XML pour les

métadonnées. Les métadonnées peuvent être : l'auteur d'une page web, à quelle date une entrée de blog a été publiée, … etc.

. XPathest un sous-langage de XSLT, il est venu dans son comme un élément clé de XML. XPath 2.0 a émergé comme un langage robuste deux fois la taille de son prédécesseur, complexe et capable de triompher.

ANNEXE 3 : L'environnement de développement

***Le système de recherche XFIRM* :** [Sauvagnat, 2006]

XFIRM (XML Flexible Information Retrieval Model) est l'un des modèles de RIS qui présente la possibilité d'implémentation de nombreux modèles de RI, c'est un système orienté pertinence. Tel présenté dans le chapitre IV, XFIRM est constitué principalement d'une BDD (Base De Données) MySQL qui consiste en l'élément centrale de tout le système et contenant ses index, autour de la BDD sont placés des éléments ayant des accès à celle-ci via un pilote JAVA, ces élément sont des modules qui construisent la partie dynamique du système et sont : *Le module d'indexation, le module d'interrogation* et *le module de traitement des requêtes* (orientées contenu et orientées contenu et structure). Pour répondre aux exigences utilisateur en structure et contenu en lui renvoyant une liste triée d'éléments répondants à sa requête.

VMware Workstation :

VMware Workstation est un environnement de test et de développement qui permet aux administrateurs système de créer et d'exécuter des machines virtuelles (VM) directement sur un bureau. Il permet aussi l'évaluation des hyperviseurs1.

***XFIRM* :** C'est un système de recherche d'informations développé en 2005 par Karen Sauvagnat, ce système se base sur le modèle vectoriel [Sauvagnat, 2005] et est orienté vers la RI structurée dans des corpus documentaires XML. Il a apporté de la flexibilité dans la recherche car il utilise une représentation des documents par un modèle générique de données (description des formats des documents dans une base de données) et un langage de requêtes en permettant l'expression du besoin en information de l'utilisateur avec des simples expressions du langage naturel puis une reformulation des requêtes.

L'environnement de développement Eclipse :

C'est un environnement de développement intégré (IDE) développé par I.B.M. pour des applications telles que java, dans le site2 d'I.B.M associé à Eclipse on trouvera la définition suivante : *«Le projet Eclipse est un projet de développement de logiciels open source dédié à fournir un robuste, complet, une qualité commerciale et plate-forme de l'industrie pour le développement d'outils hautement intégrés »*. Donc, par définition, Eclipse est une plateforme ouverte pour l'intégration de l'outil, par l'intégration des plugins à un IDE.

***Le langage Java* :**né en 1991 À cette époque, des ingénieurs de chez**Sun Microsystems** ont cherché à concevoir un langage applicable à de petits appareilsélectriques (*code embarqué*). Pour ce faire, ils se sont fondés sur une syntaxe très proche decelle de C++, en reprenant le concept de machine virtuelle déjà exploité auparavant par lePascalUCSD. L'idée consistait à traduire d'abord un programme source, non pas directementen langage machine, mais dans un pseudo langage universel, disposant des fonctions communes à toutes les machines.

Java est un langage de programmation à usage général mais avec des fonctionnalités qui lerendent plus simple :

- Création des applications web, les servlet, Les applets.
- Programmation procédurale, événementielle et implémentation flexible del'orienté objet.
- Bibliothèques riches de méthodes.

***Le SGBD Oracle10g EX* :**Un SGBD (Système de Gestion des Bases de Données) est un logiciel permettant la création, manipulation et la modification des BDD, ainsi que le contrôle et la confidentialité des données, dans notre cas, le SGBD utilisé est Oracle 10g.

Oracle est un SGBD écrit en langage Cet édité par la société du même nom, qui est un leader mondial des bases de données et a été créée en 1977 par Lawrence Ellison, Bob Miner, et Ed Oates. Elle s'appelle alors *Relational Software Incorporated (RSI2)* et commercialise un Système de Gestion de Bases de données relationnelles (SGBDR ou RDBMS pour *RelationalDatabase Management System*) nommé *Oracle*. En 1984 la première version d'Oracle(Oracle4) est commercialisée sur les machines IBM. Et en 1985 Oracle 5 permet une utilisation client-serveur grâce au middleware *SQL*Net*. Et en 1988 Oracle 6 est disponible sur un grand nombre de plates-formes et apporte de nombreuses nouvelles fonctionnalités, ainsi qu'une amélioration notable des performances. Et ce n'est qu'en 1992, qu'Oracle 7 sort sur les plates-formes UNIX (elle ne sortira sur les plates-formes Windows qu'à partir de 1995). Cette version permet une meilleure gestion de la mémoire, du CPU et des entrées-sorties. La base de données est accompagnée d'outils d'administration (SQL*DBA) permettant une exploitation plus aisée de la base.

✓ ***Les fonctionnalités d'Oracle :***

Oracle est un SGBD permettant d'assurer :

· La définition et la manipulation des données.
· La cohérence des données.
· La confidentialité des données.
· L'intégrité des données.
· La sauvegarde et la restauration des données.
· La gestion des accès concurrents (transactions).

***L'interface SQL*PLUS* :** Oracle propose également de nombreux outils de développement permettant d'automatiser la création d'applications s'interfaçant avec la base de données, une interface interactive permettant d'envoyer des requêtes SQL et PL/SQL à la base de données, la créer et de manipuler interactivement des objets de la base via une interface en ligne de commandes à travers le SGBD Oracle.

Les outils de connexion au SGBD et à la BDD :

✓ ***Le pilote de connexion au SGBD Oracle :***

La classe permettant la connexion au SGBD est : *java.sql.DriverManager*qui se charge de la gestion, du contrôle et de la connexion au SGBD en fournissant les méthodes principales suivantes :

Static void register Driver(Driver driver): enregistre le driver pour un type de SGBD particulier ;

Static ConnectiongetConnection (String *url*, String *user*, String *password*): Crée une connexion permettant d'utiliser une base de données.

Avec:

driver: est un pilote dépendant du SGBD utilisé, dans notre cas il s'agit du pilote d'oracle sous linux : *jdbc:oracle:thin:@oracleserv.irit.fr:1521:test*

url: identification de la base considérée sur le SGBD à format dépendant du SGGB utilisé

user: nom de l'utilisateur qui se connecte à la base.

password: mot de passe de l'utilisateur.

✓ ***Le JDBC :***(Java Data Base Connectivity)

C'est un outil de connexion à la BDD conçu par Sun. JDBC est un Framework pour le langage Java permettant l'accès aux bases de données relationnelles dans un programme Java indépendamment du type de la base utilisée (mySQL, Oracle, Postgres...) et seule la phase de connexion au SGBDR change. Il permet de faire tout type de requêtes : sélection des données dans des tables, création et insertion d'éléments dans les tables et gestion des transactions.Les packages java le configurant : *java.sql*et*javax.sql*

✓ ***L'ODBC :***

Abrégé en ODBC Open DatabaseConnectivityest un outil de connexion à une BDD, il construit un pont « *Bridge »* entre le pilote JDBC et la BDD configurée. Tel que le JDBC communique avec l'interface ODBC et non directement avec la BDD. L'intérêt réside dans le caractère standard de l'ODBC qui est utilisé dans les SGBD comme *Microsoft SQLServer* ou *Microsoft Access*. Ces différents drivers permettent d'accéder indifféremment à la plupart des SGBD.

Principales classes pour accéder à une BDD :

La classe Driver Manager charge et configure le driver de la base de données qui permet de gérer l'accès à un type particulier de SGBD.

La classe Connection réalise la connexion et l'authentification à la base de données.

La classe Statement et PreparedStatement) contient la requête SQL et la transmet à la base de données.

La classe ResultSet permet de parcourir les informations retournées par la base de données dans le cas d'une sélection de données.