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

Université Mouloud MAMMERI de Tizi-Ouzou
Faculté génie électrique et informatique
Département informatique

Mémoire de Master

Spécialité : **S**ystèmes **I**nformatiques

Sujet :

Implémentation et expérimentation d'une méthode de propagation des termes en utilisant de différentes formules de pondération

Réalisé par :

M^elle^ AIMENE Sonia

M^elle^ BELHOCINE Naoual

Proposé par :

M^me^ FELLAG SAMIA

Année Universitaire 2012/2013

Remerciements

Nos vifs remerciements accompagnés de toute notre gratitude vont tout d'abord à notre promotrice Mme S.Fellag pour son suivi et son engagement lors d'élaboration de ce projet. Nous la remercions pour ses orientations et suggestions efficaces et pour ses conseils judicieux.

Nos sincères salutations aux membres du jury qui nous font l'honneur d'examiner et de juger notre travail.

Finalement, nos remerciements vont à toute personne ayant contribué de près ou de loin à l'aboutissement de ce modeste travail.

Naoual & Sonya

Dédicaces

Je dédie ce modeste travail à :

Ma chère mère

Mes chères sœurs

Mon cher frère

Mes amis (es)

Mes oncles et tantes

Mes cousins et cousines

Mes pensées vont particulièrement à mon défunt père qui repose en paix.

Qu'ils trouvent ici toute ma gratitude

Naoual

Dédicaces

Je dédie ce modeste travail à :

Ma chère mère

Mon cher papa

Mes chères sœurs

Mes chers frères

Mes amis (es)

Mes oncles et tantes

Mes cousins et cousines

Qu'ils trouvent ici toute ma gratitude

Sonya

Liste des figures :

Liste des tableaux :

Introduction Générale

L'Homme étend aujourd'hui ses activités à un nombre croissant de secteurs. Alors qu'au siècle des Lumières les savants pouvaient se vanter de regrouper toute la connaissance humaine connue dans leur fameuse Encyclopédie, cette démarche, si on voulait la reconduire aujourd'hui, ne serait plus qu'une utopie. L'augmentation quasi-exponentielle des connaissances de l'Homme ainsi que leur spécialisation dans des domaines d'intérêt très variés, et le développement de l'Internet et de l'informatique dans tous les secteurs d'activité ont conduit à une explosion sans précédent du volume d'informations disponibles sous des formats hétérogènes, et produites par des sources d'informations distribuées. Ainsi l'´elaboration de systèmes automatisés pour gérer ces masses de données est devenue dans un tel contexte une nécessité.

Notre travail se situe dans le contexte de ces outils automatisés et plus précisément dans le domaine de la Recherche d'Information (RI). La RI est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage et la recherche des informations. Elle propose des outils, appelés systèmes de recherche d'information (SRI), dont l'objectif est de capitaliser un volume important d'information et d'offrir des moyens permettant de localiser les informations pertinentes relatives à un besoin en information d'un utilisateur exprimé à travers une requête.

Notre travail se situe dans le contexte de la recherche d'information (RI). L'objectif principal des Systèmes de Recherche d'Information (SRI) est de répondre au besoin en information des utilisateurs. Les utilisateurs interrogent, au moyen d'une requête, une base de documents numériques et les SRI leur renvoient une liste de documents susceptibles de répondre à leur besoin.

Aujourd'hui, la nature des sources d'information évolue, et les documents traditionnels "plats" ne contenant que du texte s'enrichissent d'information structurelle et d'information multimédia. En effet l'apparition du SGML, la naissance du HTML et du WEB et avec l'apparition du XML, la RI a vite évolué à la RIS, une recherche d'information qui ne

s'occupe pas que de l'information textuelle, mais elle occupe aussi de l'information structurelle du document.

C'est dans ce contexte de recherche d'information structurée que se situent plus particulièrement nos travaux. Nous nous plaçons plus précisément dans le cadre de documents semi-structures, c'est à dire de documents ne disposant pas d'une structure fixe et homogène, mais au contraire d'une structure flexible ainsi que de contenus hétérogènes. Nous utiliserons le format XML tout au long de ce mémoire pour illustrer nos propos.

L'arrivée d'XML à complètement réactualisé la problématique de la RI. En RI traditionnelle ou non structurée l'unité documentaire retournée à l'utilisateur est le document tout entier. Cependant, l'information structurelle contenue dans les documents XML soulève deux nouvelles problématiques.

Dans le cadre de la RI, on distingue deux principales problématiques. La première problématique est dans le cadre de l'indexation, elle se situe essentiellement au niveau de l'information structurelle. La dimension structurelle ajoutée au contenu des documents XML ainsi que les contraintes soulèvent de nouvelles questions :

- Comment indexer la structure des documents ?
- Que doit-on indexer de la structure des documents ?
- Comment relier cette structure au contenu du document ?
- Quels sont les paramètres à considérer pour la pondération des termes d'indexation?

La seconde problématique concerne la granularité de l'information à renvoyer à l'utilisateur en réponse à une requête donnée. Cette problématique est liée au modèle de recherche et le tri des unités d'information selon leurs pertinences vis-à-vis d'une requête ainsi qu'à la formulation même des requêtes. En effet, dans le cas des requêtes portant sur le contenu (CO) (cadre de notre travail) qui sont de loin les plus simples pour l'utilisateur, impose au SRI de décider de la granularité d'information appropriée à renvoyer à l'utilisateur et donc d'évaluer l'exhaustivité et la spécificité de l'information.

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

1) Réalisation d'un état de l'art sur la recherche d'information classique et structurée.
2) Présentation d'un modèle de propagation des termes et des formules de pondérations utilisées.
3) Implémentation de ce modèle.
4) Expérimentation et évaluation de ce modèle avec différentes formules de pondérations sur un ensemble de requêtes et de documents de la collections INEX 2006, et comparaison des résultats obtenus avec ceux obtenus avec la modèle XFIRM.

Chapitre 1

Recherche

D'Information

I.1. Introduction :

La croissance du volume de données textuelles comme les livres et les articles dans les bibliothèques durant des siècles a imposé de définir des mécanismes efficaces pour les localiser.

Les premières techniques, comme l'indexation et l'utilisation des catégories de classification ont marqué la naissance de la « Recherche d'Information » comme discipline de recherche, dite RI.

L'objectif de ce chapitre est de présenter les concepts de base de la RI, nous décrivons le processus global de la RI, communément connu sous le nom du processus en U, nous donnons l'utilité et l'importance des fonctions qui composent ce processus.

I.2. Notions de base :

I.2.1. Définition de la recherche d'information (RI) :

La recherche d'information est traditionnellement définie comme l'ensemble des méthodes, procédures et techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur.

D'après (l'AFNOR, 1979) [1], *"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"*

La RI selon Salton [2] est, *l'ensemble des techniques permettant de gérer des textes*.

Gérer des textes implique stocker, rechercher et explorer des documents ou parties de documents pertinents.

I.2.2. Les systèmes de recherche d'information (SRI) :

Les Systèmes de Recherche d'Informations documentaires sont nés de la nécessite d'automatiser la gestion et la recherche des informations documentaires.

Un SRI peut être défini comme un ensemble de programmes informatiques et de procédures qui ont pour but de sélectionner des informations pertinentes répondant à des besoins de l'utilisateur, exprimés sous forme de requêtes.

Autrement dit un SRI joue l'intermédiaire entre un utilisateur et une collection d'informations. Son but est de satisfaire au mieux le besoin en information de cet utilisateur.

I.3. Le processus de recherche d'information :

Le processus de la RI permet de mettre en correspondance les informations disponibles d'une part, et les requêtes (besoins) de l'utilisateur d'autre part. Ce processus cherche alors à mettre en relation les besoins utilisateurs et les informations, afin que le Système de Recherche d'information (SRI), ne retourne à l'utilisateur que le maximum de documents pertinents par rapport à son besoin (et le minimum de documents non pertinents).

Un document est dit ***pertinen****t* lorsqu'il satisfait le besoin en information de l'utilisateur, et non pertinent dans le cas contraire. La notion de pertinence est alors fortement subjective, c'est-à-dire dépendante de l'utilisateur.

Le processus de Recherche d'information, appelé **Processus en U de Recherche d'Information** est représenté dans la figure I.1.

Cette figure présente les principales tâches d'un SRI qui sont :

1. L'indexation des documents et des requêtes.
2. L'appariement requête-document, qui permet de comparer la requête et le document, et de calculer la similarité entre ces deux éléments.
3. La Reformulation de la Requête.

Ces trois tâches seront détaillées dans les sections suivantes.

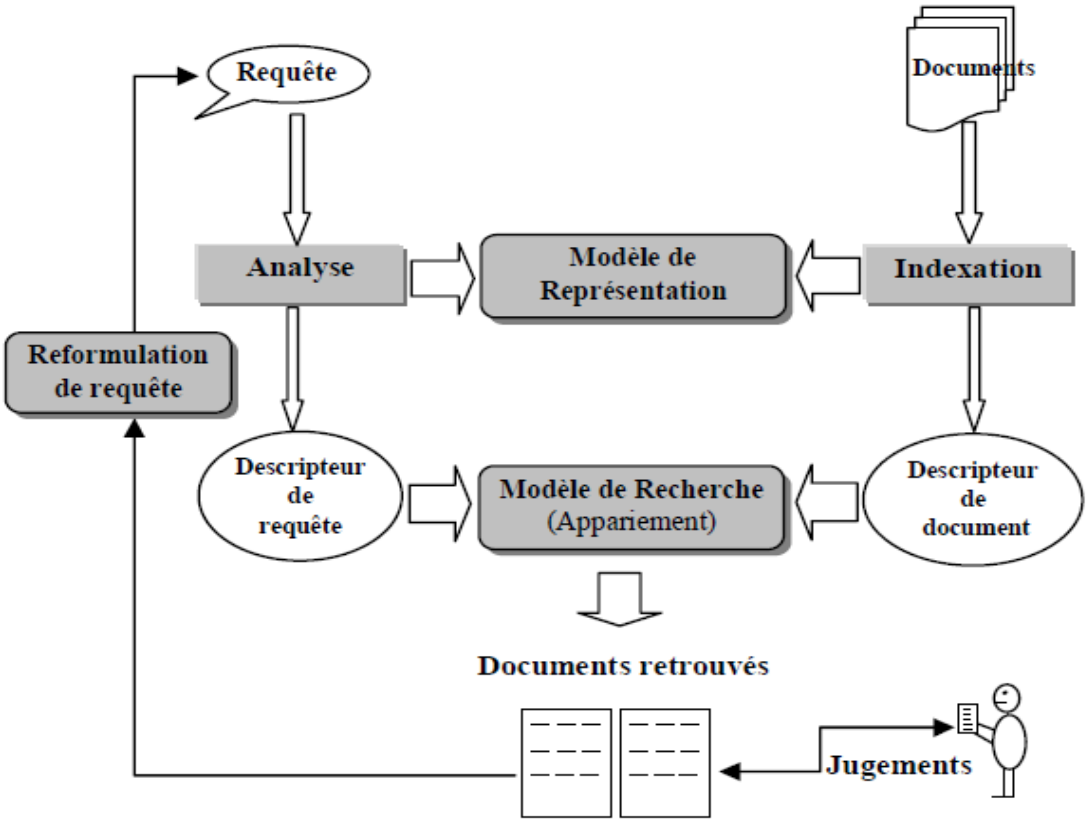


Figure I.1 : Processus en U de la recherche d'information [17].

Ce processus manipule des documents et des requêtes qui sont définis comme suit :

a. document :

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

b. requête :

D'après Wikipedia (l'encyclopédie libre), une requête (en anglais query) correspond à l'interrogation d'une base de données, dans notre cas les documents, pour en récupérer une certaine partie des données.

D'autre part, le besoin de l'utilisateur est l'expression mentale de ce qu'il recherche. Ce besoin est alors représenté à travers une requête qui sera traitée par le SRI.

Une requête peut être décrite :

- Par une liste de mots clés : cas des systèmes SMART [2] et Okapi [4].
- En langage naturel : cas des systèmes SMART [2] et SPIRIT [5].
- En langage booléen : cas des systèmes DIALOG [6].
- En langage graphique : cas du système issu du projet NEURODOC [7].

I.3.1. Le processus d'indexation :

Afin que le coût et le temps de réponse de la recherche soient acceptables, il convient d'effectuer une étape primordiale sur la base documentaire. Cette étape consiste à analyser chaque document de la collection afin de créer un ensemble de mots-clés : on parle de l'étape d'indexation. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche.

Le résultat de l'indexation constitue le descripteur du document ou requête, ce descripteur peut être composé de mots simples, mots composés ou groupes de mots.

I.3.1.1. Types d'indexation :

Il existe trois types d'indexation :

- Indexation manuelle.
- Indexation automatique.
- Indexation semi-automatique.

Ces trois types d'indexation sont définis comme suit :

1. Indexation manuelle :

Chaque document est analysé par un spécialiste du domaine ou par un documentaliste qui détermine, selon ses connaissances, les unités syntaxiques ou mots-clés qui lui semblent les plus significatifs pour représenter le contenu du document. Les mots clés sont ensuite regroupés dans une liste plus ou moins structurée.

Ce type d'indexation assure une meilleure pertinence dans le système puisqu'elle permet de repérer de façon plus précise les mots-clés décrivant un document, mais présente

néanmoins des inconvénients puisqu'elle est très coûteuse en termes de temps, puis elle est subjective puisqu'elle est fondée sur le jugement d'un être humain

2. indexation semi-automatique :

Dans ce type d'indexation, les termes significatifs du document sont choisis par un spécialiste ou documentaliste utilisant un thesaurus ou une base terminologique.
Un *thesaurus* ou une *base terminologique* est une liste organisée de descripteurs (mots-clés) obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques.

3. Indexation automatique :

Le processus d'indexation automatique est entièrement informatisé ; il a été mis au point dans le but de pallier les inconvénients des systèmes manuels.

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

Les Systèmes de Recherche d'Information actuels utilisent l'indexation automatique. Cette indexation suit plusieurs étapes :

- L'analyse lexicale ;
- L'élimination des mots vides ;
- La lemmatisation ;
- Repérage des groupes de mots.

Nous détaillons dans ce qui suit ces cinq étapes :

a. L'analyse lexicale :

Le but de l'analyse lexicale est de convertir le texte d'un document en un ensemble de termes. Un terme est une unité lexicale ou un radical .Ce processus permet de reconnaître les espaces de séparation des mots, des chiffres, les ponctuations, etc.

b. L'élimination de mots vides :

Cette étape a pour objectif de supprimer les termes non significatifs (pronoms personnels, prépositions,...) ou mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc.).

Il existe deux techniques pour éliminer les mots vides, la plus connue est l'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire). L'anti-dictionnaire est consulté lors du processus d'indexation; si le terme existe dans celui-ci, il sera systématiquement éliminé du texte sinon il sera considéré comme index.

L'élimination des mots vides permet donc de réduire le nombre de termes d'indexation, mais aussi peut réduire *le taux de rappel*, c'est-à-dire le rapport de documents pertinents renvoyés par Le système sur l'ensemble des documents pertinents, une notion que nous allons détailler dans la section **I.5.**

c. La lemmatisation :

Un mot donné peut avoir différentes formes dans un texte, mais le sens reste le même ou très similaire. Il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept. Une substitution des termes par leur racine, ou lemme, est utilisée, Frakes et Baeza-yates [8] citent cinq stratégies de lemmatisation dont:

- L'élimination des affixes : On peut par exemple citer l'algorithme de Porter [9].
- La troncature : Si on fait une requête du type « logiciel» et qu'un document contient les mots *logiciels*, il faut que le moteur de recherche puisse reconnaître que *logiciel* et *logiciels* sont en fait le même mot. Pour y arriver, les moteurs tronquent les mots.
- La méthode des n-grammes[1] [10].

Exemple de lemmatisation :

Le lemme de « révolution » et celui de « révolter » sont les mêmes.

1.Un n-gramme est une succession de n lettres. *Exemple* : pour le mot "recherche "

1-gramme: r,e, c, h, e, r, c, h, e; *2-gramme :* re, ec, ch, he, er, rc, ch, he; *3-gramme:* rec, ech, che, her, erc, rch, che,

d. Le repérage de groupes de mots :

L'unité de base lors du processus d'indexation est un groupe de mots, les mots ne sont pas pris séparément car ils perdraient tout leurs sens.

Cet argument a conduit à ne pas considérer les mots simples comme unités de base lors du processus d'indexation, mais des groupes de mots.

Exemple : la requête *«recherche d'information»* perd tout son sens lorsqu'elle est représentée par les mots *«recherche»* et *« information ».*

Afin d'obtenir une représentation plus fine que la simple présence ou absence d'un terme dans le document, on calcule un poids représentant l'importance qu'a le terme dans le document considéré.

I.3.1.2. Pondération de termes d'indexation :

La pondération des termes permet de mesurer l'importance d'un terme dans un document. L'objectif est de trouver les termes qui représentent le mieux le contenu d'un document.

Si on dresse une liste de l'ensemble des mots différents d'un texte quelconque classés par ordre de fréquences décroissantes, on constate que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste.

Cette constatation est énoncée formellement par la loi de Zipf [11] :

$$rang * fréquence = constante.$$

La loi de distribution des termes se présente comme le montre la figure suivante :

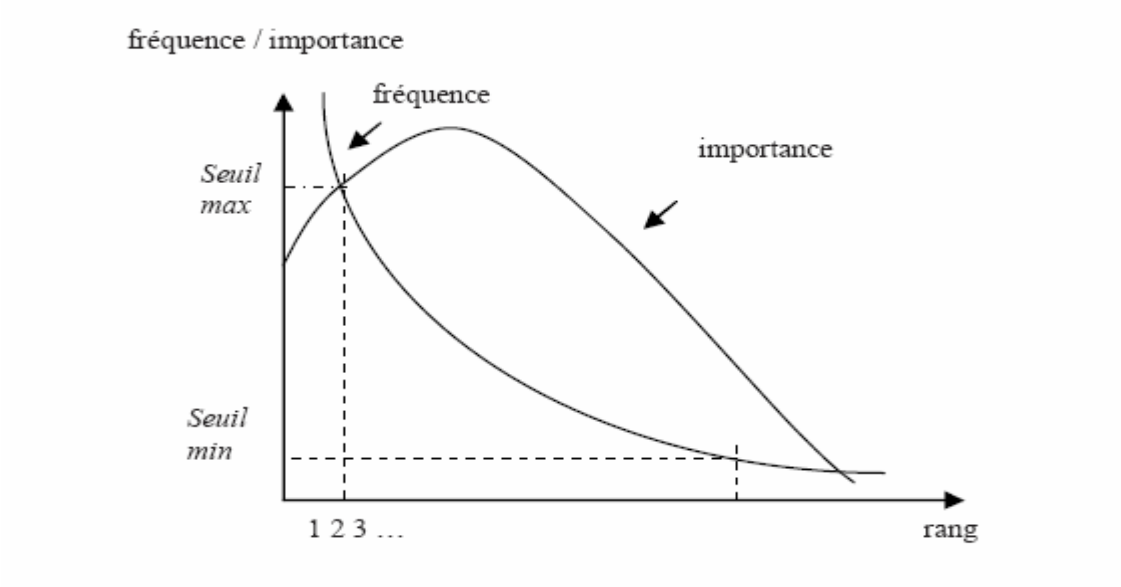


Figure I.2 : Importance d'un terme en fonction de sa fréquence d'apparition dans le document [12]

Un terme représentatif d'un document ne doit pas être de fréquence plus élevée ou plus petite, et il est sélectionné en fonction de sa fréquence et de son rang dans le document.

A partir de ces constatations, des techniques de pondération ont vu le jour. La plupart de ces techniques de pondération sont basées sur les facteurs **TF** et **IDF** qui permettent de combiner les pondérations locale et globale d'un terme, ils sont définis comme suit :

- **TF** (**Term Frequency**) :

C'est la pondération locale qui est proportionnelle à la fréquence d'un terme « dans le document ». Elle se calcule selon l'une des formules suivantes :

- tf = n_{oc}
- tf = [0|1] présence, absence
- tf = $\frac{n_{oc}}{tf_{\max}}$
- tf = $0.5 + 0.5\frac{n_{oc}}{tf_{\max}}$

Où :

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

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

- **IDF** (**Inverse Document Frequency**) :

Représente l'importance d'un terme dans toute la collection (pondération globale).Un terme ayant une fréquence élevée mais qui n'est pas concentrée dans un nombre limité de documents mais plutôt dans toute la collection, ne doit pas avoir le même impact ou le même poids qu'un terme moins fréquent.

$$\text{-idf} = \log \left[\frac{N}{n_i} \right]$$

$$\text{-idf} = \log \left[\frac{N \_ n_i}{n_i} \right]$$

Où :

***N* :** représente le nombre total de documents dans le corpus (la collection).

***n_i* :** représente le nombre total de documents contenant le terme *i*.

D'autres facteurs sont également considérés comme par exemple la longueur d'un document. En effet, la longueur d'un document doit être prise en compte, sinon les documents longs auront plus de chance d'être sélectionnés que les documents courts car dans les documents longs les fréquences des termes sont plus élevées.

Pour pallier ce problème, Robertson [13] et Singhal et al. [14] ont introduit **la normalisation** utilisée pour donner aux documents une chance égale d'être sélectionnés indépendamment de leurs longueurs.

Si c'est lors de l'indexation que sont choisis les termes pour représenter le contenu d'un document ou d'une requête, c'est au modèle de leur donner une interprétation. Le modèle joue un rôle central dans la RI. C'est lui qui détermine le comportement clé d'un SRI. Nous décrivons les modèles les plus souvent utilisés en RI dans la section **I.4.**

I.3.2. L'appariement document-requête :

Le but de tout SRI est de retourner à l'utilisateur le plus grand nombre possible de documents pertinents. La pertinence est basée sur une fonction d'appariement (matching) qui effectue une comparaison entre les représentants des documents et ceux des requêtes construits lors de la phase d'indexation.

La valeur de la pertinence est calculée à partir d'une fonction ou d'une probabilité de similarité notée **RSV (Q, D)** (Retrieval Status Value), ou *« Q » est une requête* et « *D » un document.* Cette mesure tient compte des poids des termes dans les documents, déterminés en fonction d'analyses statistiques et probabilistes.

La fonction de similarité permet d'ordonner les documents renvoyés à l'utilisateur, Cet ordonnancement joue un rôle primordial puisque l'utilisateur se contente la plupart du temps d'examiner les premiers documents affichés.

I.3.3. La reformulation de la requête :

La reformulation de la requête rentre en général dans le processus d'optimisation de la fonction de pertinence qui a pour but de rapprocher la pertinence système de la pertinence utilisateur.

La pertinence utilisateur correspond au jugement de celui-ci par rapport à la réponse rendue par le système en termes de documents, par contre la pertinence système correspond à la similarité entre le document et la requête indépendamment de l'utilisateur.

La reformulation de la requête peut être automatique ou manuelle. Dans le premier cas, l'utilisateur n'intervient pas. L'extension de la requête est effectuée à partir d'un **thesaurus** qui définit les relations entre les différents termes de l'index et permet de sélectionner de nouveaux termes à ajouter à la requête initiale.

Quand à la reformulation manuelle on parle de la réinjection de pertinence (Relevance feedback).

I.4. Les Modèles de Recherche d'Information :

Les modèles de RI se déclinent en trois grandes catégories qui sont les modèles booléens, les modèles vectoriels (algébriques) et les modèles probabilistes. Nous allons décrire dans ce qui suit chacun de ces modèles.

I.4.1. Le modèle booléen :

Le modèle booléen est le premier modèle utilisé en RI, il est basé sur la théorie des ensembles et l'algèbre de Boole.

Dans ce modèle, les documents et les requêtes sont représentés par des ensembles de mots clés. Ce modèle doit son nom à l'utilisation des opérateurs *et*, *ou* et *non* pour la représentation des documents et des requêtes.

Chaque document D est représenté par un ensemble de termes (non pondérés), et c'est la conjonction de ces termes t_i qui constitue l'index des documents.

Par exemple : $D = t_1 \wedge t_2 \wedge \ldots t_n$.

Une requête Q est une expression logique quelconque de termes. On peut utiliser les opérateurs (ou v, (et ∧) et (non *7*). Par exemple: **Q = (*t*1 ∧ *t*2) v (*t*3 ∧ *7t*4)**

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

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

Ce modèle présente le principal avantage de simplicité de mise en oeuvre.

On peut remarquer les problèmes suivants dans ce modèle :

- La sélection d'un document est basée sur une décision binaire car la correspondance entre un document et une requête est soit 1, soit 0. En conséquence, le système détermine un ensemble de documents non ordonnés comme réponse à une requête. Il n'est pas possible de dire quel document est mieux qu'un autre.
- La formulation de la requête n'est pas toujours évidente pour beaucoup d'utilisateurs.
- Problème de collections volumineuses : le nombre de documents retournés peut être considérable.

Afin de remédier à ces inconvénients une extension du modèle booléen a été proposée : le modèle booléen étendu.

I.4.2. Le modèle booléen étendu :

Le modèle booléen étendu a été introduit par Salton [15]. C'est une extension du modèle précèdent qui vise à tenir compte d'une pondération des termes dans le corpus. Cela permet de pallier aux problèmes du modèle de base en ordonnant les documents retrouvés par le SRI. Dans ce modèle la requête demeure une expression booléenne classique, tandis que les termes d'un document sont maintenant pondérés. En général le poids d'un terme dans un document est en fonction du nombre d'occurrences de ce terme dans le document. L'appariement document-requête est le plus souvent déterminé par les relations introduites dans le modèle P-norm basées sur les P-distances, avec $1 \leq P \leq \infty$.

La valeur de P est indiquée au moment de la requête.

Si **M** est le *nombre de termes dans la requête* et $X = q_{ik} * d_{ik}$; les fonctions de similarité se calculent comme suit :

$$RSV\ (D, Q_{ou}) = \frac{(X_1^P + X_2^P + ... + X_M^P)^{1/P}}{M}$$

$$RSV\ (D, Q_{et}) = 1 - \frac{((1 - X_1)^P + (1 - X_2)^P + ... + (1 - X_M)^P)^{1/P}}{M}$$

I.4.3. Le modèle vectoriel :

Dans ce modèle, les requêtes et les documents sont représentés dans l'espace vectoriel engendré par les termes d'indexation. L'espace est de dimension N (N étant le nombre de termes d'indexation de la collection de documents).

Soit la figure I.3 qui représente trois documents D_i et une requête Q dans un espace vectoriel :

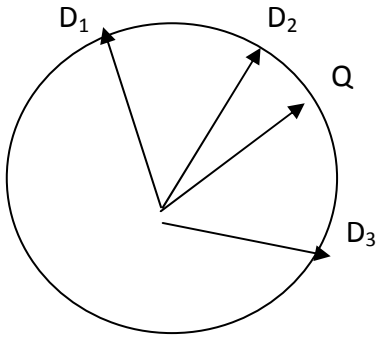


Figure I.3 : Le modèle vectoriel.
D'après cette figure le document D_2 est le document le mieux placé.

Soit l'espace vectoriel suivant : $<t_1, t_2, \ldots, t_n>$. Chaque document et requête sont respectivement représentés par un vecteur document et un vecteur requête comme suit:

$D_j = (d_{1j}, d_{2j}, \ldots, d_{Nj})$ avec d_{ij} : poids du terme t_i dans le document D_j.

$Q = (q_1, q_2, \ldots, q_N)$ avec q_i : poids du terme t_i dans la requête Q.

Le mécanisme de recherche consiste à retrouver les vecteurs documents qui s'approchent le plus du vecteur requête. Les principales mesures de similarité utilisées sont :

Produit interne (Inner product) :

$$RSV\ (D_j, Q_k) = \sum_{i=1}^{N} d_{ij}\, q_{ik}$$

Coefficient de Dice :

$$RSV\ (D_j, Q_k) = \frac{2\sum_{i=1}^{N} d_{ij}\, q_{ik}}{\sum_{i=1}^{N} (d_{ij}^2 + q_{ik}^2)}$$

Mesure du Cosinus :

$$RSV\ (D_j, Q_k) = \frac{2\sum_{i=1}^{N} d_{ij}\, q_{ik}}{\sqrt{\sum_{i=1}^{N} d_{ij}^2 \times \sum_{i=1}^{N} q_{ik}^2}}$$

Mesure de Jaccard :

$$RSV\ (D_j, Q_k) = \frac{\sum_{i=1}^{N} d_{ij}\, q_{ik}}{\sum_{i=1}^{N} d_{ij}^2 + \sum_{i=1}^{N} q_{ik}^2 - \sum_{i=1}^{N} d_{ij}\, q_{ik}}$$

- ## Avantage :

- La pondération améliore les résultats de la recherche.
- La mesure de similarité permet d'ordonner les documents selon leurs degrés de pertinence vis-à-vis de la requête.

- ## Inconvénients :

La représentation vectorielle suppose l'indépendance entre les termes, ce qui n'est pas souvent le cas.

I.4.4. Le modèle probabiliste :

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête. L'idée est de retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents.

Pour ce faire, ce modèle utilise deux probabilités conditionnelles :

- P (d_{ij}/pert) : Probabilité que le terme t_i apparaît dans le document d_j sachant que ce dernier est pertinent pour la requête.
- P (d_{ij}/Non Pert) : Probabilité que le terme t_i apparaît dans le document d_j sachant que ce dernier n'est pas pertinent pour la requête.

Si on suppose l'indépendance des variables documents pertinents et non pertinents la fonction de recherche peut être obtenue en utilisant la formule de Bayes suivante :

Soit d_j (t_1, t_2,…, t_N) ou t_i =

- **1 Si t indexe le document dj**
- **0 Sinon**

- **Avantage :**

- Les modèles probabilistes retournent de bons résultats par rapport aux modèles booléens [16].
- Ils sont indépendants du domaine d'application
- Les documents retrouvés sont classés selon l'ordre de pertinence décroissant

- **Inconvénients :**

-Difficulté de calcul de probabilités conditionnelles.

-Les jugements de pertinence étant propres à un utilisateur particulier, l'apprentissage effectué a partir de ses données ne sont pas valables que pour cet utilisateur. [16]

I.5. Evaluation de performance d'un SRI :

En général un SRI est évalué par sa capacité à sélectionner des documents pertinents d'une part et de rejeter les documents non pertinents d'autre part, mais une mesure d'évaluation appropriée doit estimer la capacité des systèmes à trouver des documents

pertinents, sans pour autant favoriser ceux qui retournent plus de documents. Il est donc nécessaire de considérer aussi les documents non pertinents dans une métrique.

Les documents dans la base peuvent être décomposés comme suit :

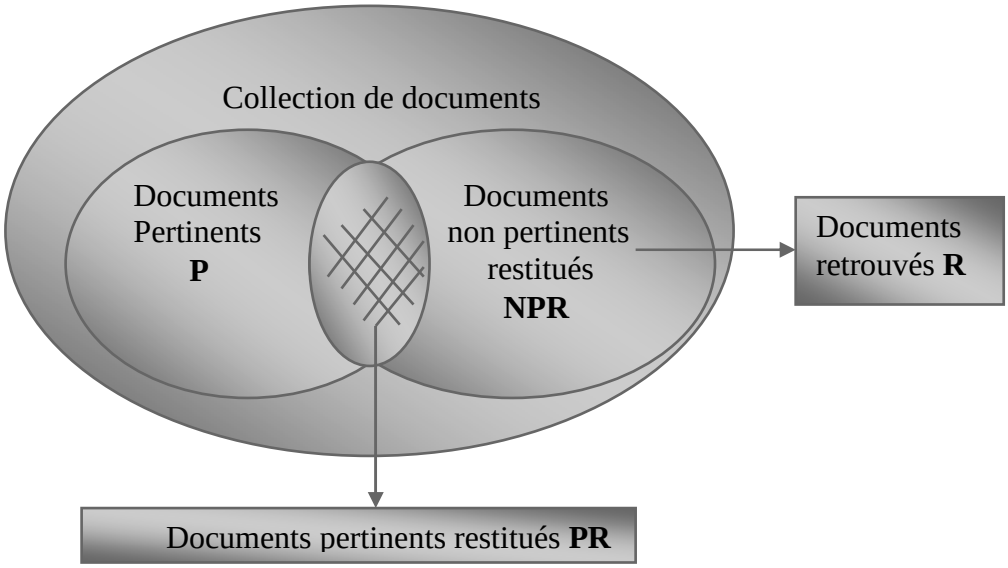


Figure I.4 : Répartition des documents face à une requête [17]

D'une façon générale, tout SRI a deux objectifs principaux : retrouver tous les documents pertinents et rejeter tous les documents non pertinents pour une requête donnée. Ces deux objectifs sont généralement évalués par **les mesures de rappel et précision, bruit et silence.**

Soient :

|R| : Le nombre de documents retrouvés dans la collection pour cette requête.

|P| : Le nombre de documents pertinents dans la collection pour cette requête.

|NPR| : le nombre de documents non pertinents restitués dans la collection pour cette requête.

|PR| : Le nombre de documents pertinents restitués dans la collection pour cette requête.

Comme représentés dans la figure I.4.

- La précision mesure le taux de documents pertinents parmi tous les documents retrouves par le système :

$$\Pr écision = \frac{|PR|}{|R|}$$

- Le rappel mesure le taux de documents pertinents retrouves par le système parmi l'ensemble des documents pertinents de la collection :

$$Rappel = \frac{|PR|}{|P|}$$

- Le bruit mesure le taux de documents non pertinents par rapport à la totalité des documents extraits.

$$Bruit = \frac{|NPR|}{|R|}$$

- Le silence mesure le taux de documents non pertinents restitués par rapport à la totalité des documents pertinents contenus dans la base documentaire.

$$Silence = \frac{|NPR|}{|p|}$$

Tel que : $|\ |$ désigne le cardinal d'un ensemble.

Remarque :

- La précision et le rappel s'opposent : l'augmentation de l'une implique la diminution de l'autre.
- La performance d'un SRI peut être appréciée si d'un coté le taux de précision et de silence est élevé, et de l'autre le taux de bruit et de rappel est bas.
- Les quatre facteurs peuvent être présentés avec les relations suivantes :

  Précision+bruit=1.

  Rappel+silence=1.

I.6. Conclusion :

Dans ce chapitre nous avons essentiellement présenté les concepts de base de la recherche d'information. Nous concluons que la RI tient compte non seulement de la manière de représenter les informations mais aussi et surtout du choix de la technique utilisée pour la sélection des documents pertinents. Les modèles de RI utilisés dans un SRI sont donc influents sur la crédibilité des documents renvoyés en réponse à une requête utilisateur.

A la fin, nous avons présenté les métriques d'évaluation d'un SRI à savoir le rappel et la précision.

La recherche d'information classique permet seulement de déterminer et de sélectionner les documents pertinents pour une requête utilisateur, afin de pouvoir extraire les informations ciblées de ces documents, la recherche d'information structurée donne le moyen de le faire, c'est ce que nous allons présenter dans le prochain chapitre.

Chapitre 2

XML

II.1. Introduction :

XML (Extented Markup Language) est un langage de description de structures de documents recommandé par l'organisme de normalisation W3C[1] (World Wide Web Consortium) dès 1996. Très rapidement, après la publication de la recommandation XML 1.0, par le W3C de nombreuses propositions ont émergé afin d'étendre le champ d'application d'XML, comme par exemple les DTD, le langage de requête XPath, les langages de lien et d'adressage XPointer et XLink ou encore des interfaces de programmations (SAX, DOM). Nous reviendrons dans ce chapitre sur ces différentes technologies.

II.2. Présentation d'XML :

XML (langage à balises étendues, ou langage à balises extensibles) est un nouveau langage de balisage à caractère significatif, permettant de décrire le contenu plutôt que la présentation. Ainsi, **XML permet de séparer le contenu de la présentation**.

XML a une syntaxe générale qui permet de décrire des données hiérarchiques, lisibles pour l'homme, compréhensibles pour les machines, applicable à un large étendu des applications.

Le W3C regroupe sous l'acronyme XML une boîte à outils extensible dont le but est de simplifier la gestion des données (c'est-à-dire l'organisation, l'échange et la transformation). Il est important de noter que XML n'est pas réellement un langage, mais un métalangage[2] permettant l'élaboration de balisages spécialisés. C'est à dire qu'en fonction du contenu qu'on souhaite publier, on définit nos propres balises tout en respectant les critères XML.

II.2.1. Structure d'un document XML :

Les documents XML possèdent un certain nombre de particularités structurelles. L'un des objectifs de XML sur lesquels insiste le W3C est de faire en sorte que « la conception du XML soit formelle et concise ».

Chaque document XML [18] possède à la fois une structure logique et une structure physique.

[1] W3C: est un organisme fédérateur du développement de la toile. Pour en savoir plus voir le site : http://www.w3.org/tr

[2] metalangage:un langage qui permet de définir d'autres langages

- La *structure logique* définit les unités et les sous-unités des conteneurs de données (les éléments) qui définissent le type de données, les attributs, etc.
- La *structure physique* fournit les données qui sont contenues dans les éléments, comme le texte, les images ou les autres types médias, selon les spécifications établies par la structure logique. Tout document XML se compose:

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

Soit le document doc.xml de la figure II.1 suivante :

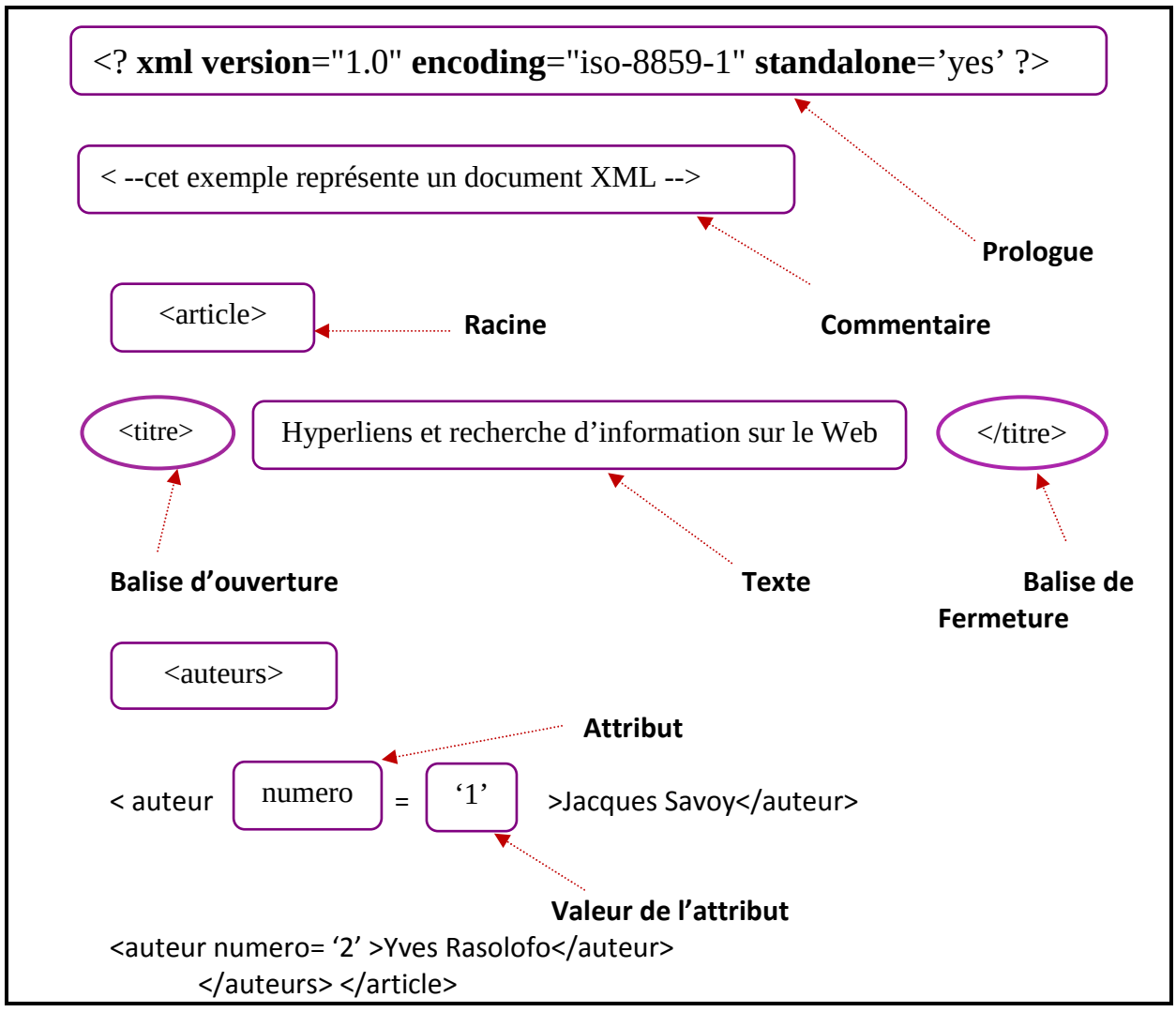


Figure II.1 : doc.xml

II.2.1.1. Le prologue :

Ce sont les premières lignes qui forment le document XML, il consiste en la déclaration XML, et éventuellement en la déclaration d'une DTD ou de XML-Schéma.

a. Déclaration XML :

Il est souvent très pratique de pouvoir identifier un document comme appartenant à un certain type. XML dispose d'une déclaration pour signaler qu'un document est de type XML. Cette déclaration fait partie des instructions de traitements.

Syntaxe :

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

Les informations fournies dans cette déclaration sont de trois sortes :

- **version** : version de XML utilisée dans le document (version 1.0).
- **encoding** : jeu de codage de caractères utilisé. Le jeu de caractère standard pour la langue française est l'ISO-8859-1.
- **Standalone** : dépendance du document par rapport à une déclaration de type de document (DTD). Si standalone a la valeur « yes », le processeur de l'application n'attend aucune déclaration de type de document extérieure au document. Sinon, le processeur attend une référence de déclaration de type de document.

Remarque : Les attributs version, encoding et standalone doivent être placés dans cet ordre.

b. Déclaration éventuelle de type de document (DTD) ou de XML-Schéma :

❖ **DTD**

XML fournit le moyen de vérifier la syntaxe d'un document grâce aux DTD (Document Type Definition). La DTD est donc l'ensemble des règles et des propriétés que doit suivre le document XML. Ces règles définissent généralement le nom et le contenu de chaque balise et le contexte dans lequel elles doivent exister. Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML. (Voir section II.4 pour plus de détails sur les DTD).

- **XML-Schéma :**

XML- Schéma publié comme recommandation par le W3C en mai 2001 est un langage de format de document XML permettant de définir la structure d'un document XML. La connaissance de la structure d'un document XML permet notamment de vérifier la validité de celui-ci. (Pour plus de détails voir section II.4).

II.2.1.2. Les commentaires :

les commentaires permettent d'insérer dans un document XML, du texte ne faisant pas réellement partie du document, mais étant plutôt destiné aux personnes examinant le code source XML.

En XML, les commentaires commencent par **<!--** et se terminent par **-->**.

Exemples: Voici deux types de commentaires qu'on peut trouver dans un document XML:

- **<!--ceci est un commentaire -->**
- **<exemple>** texte **<!-- ceci est un texte --> </exemple>**

Le corps du commentaire peut contenir n'importe quel caractère à l'exception de la chaine «--».

On ne peut donc pas inclure un commentaire dans un autre. Un commentaire ne peut être inclu à l'interieur d'une balise.

II.2.1.3 Les instructions de traitement :

Les instructions de traitement sont utilisées par les applications qui génèrent le code XML ou par les applications qui interprètent le code XML. Les instructions de traitement qui servent le plus souvent sont la déclaration XML ainsi que la déclaration de feuille de style.

Exemple : **<? Xml-stylesheet type='**type_fichier**' href='**nom_doc**'?>**

II.2.1.4. L'arbre d'éléments:

Tout document XML est représenté sous la forme d'un arbre d'éléments. Comme tout arbre, il comporte une racine, des branches et des feuilles, qui sont les éléments.

Voici l'arbre d'éléments pour le document de la figure II.1:

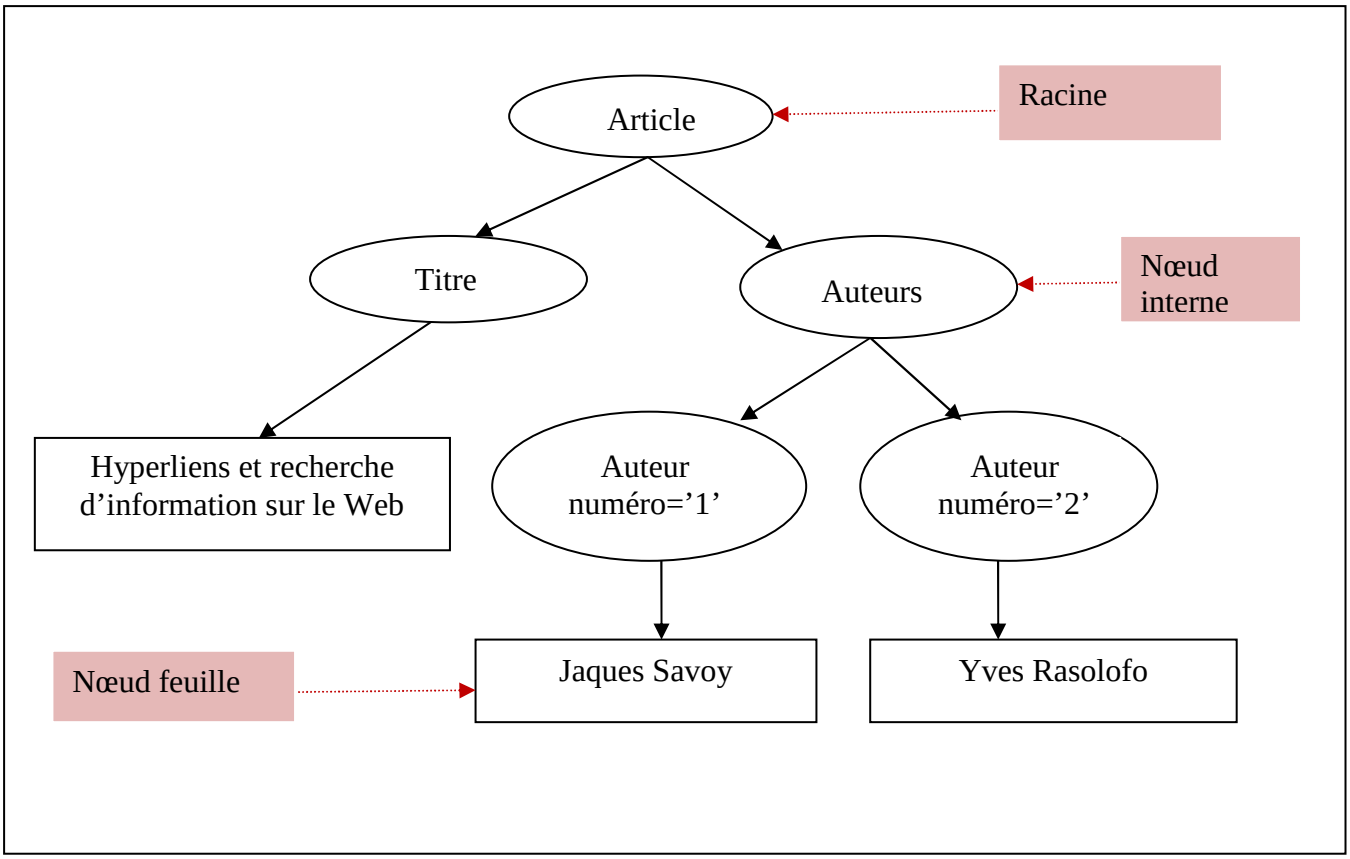


Figure II.2 : Arbre d'éléments

a. Elément racine:

L'élément racine est le premier élément déclaré dans un document XML, qui va contenir tous les autres éléments.

Dans l'exemple l'élément racine est l'elément **article**.

b. Les éléments :

Les éléments sont les composants majeurs du contenu du document XML. En effet, fondamentalement, les documents XML sont des hiérarchies d'éléments.

Syntaxe :

```
<nom_de_balise> contenu element </nom_de_balise>
```

les éléments dans cet exemple sont: titre, auteurs, auteur.

C.L'élément vide:

Un élément vide est un élément dont le contenu entre la balise d'ouverture et la balise de fermeture est vide.

Syntaxe: **</nom_de_balise>**

*D.*Les attributs:

La balise d'ouverture d'un élément peut inclure des attributs sous la forme de paires nom="valeur".

La valeur d'un attribut est une chaine de caracteres encadrée par des apostrophes ou bien de guillemets.

Exemple:

<auteur numero='1'> Jacques Savoy</auteur>.

II.3. La DTD:

On distingue deux types de DTD :

- DTD interne.
- DTD externe.

II.3.1. DTD interne :

Le contenu ne change pas suivant le type de DTD, mais les déclarations d'une DTD interne sont écrites à l'intérieur du prologue du document XML. Cette solution est rarement utilisée. Elle suit la syntaxe suivante :

< !DOCTYPE élément –racine[déclaration des éléments]>

Voici un exemple de document XML (Repertoire.xml) muni d'une DTD interne

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE repertoire [
        <!ELEMENT repertoire (service)*>
        <!ELEMENT service (nom, tel*)>
        <!ELEMENT nom (#PCDATA)>
        <!ELEMENT tel (#PCDATA)>
]>
<repertoire>
  <service>
    <nom>pompiers</nom>
    <tel>18</tel>
  </service>
  <service>
    <nom>police</nom>
    <tel>17</tel>
  </service>
  <service>
    <nom>samu</nom>
    <tel>15</tel>
  </service>
</repertoire>
```

Document XML (Repertoire.xml) muni d'une DTD interne

II.3.2. DTD externe :

Il existe deux types de DTD externes, **SYSTEM** ou **PUBLIC**. Le mot-clef SYSTEM indique que le fichier qui suit se trouve sur l'ordinateur local et qu'il est disponible uniquement à titre privé. Par contre, le mot-clé PUBLIC indique une ressource disponible pour tous sur un serveur web distant.

- Syntaxe de déclaration de DTD externe de type SYSTEM :

<!DOCTYPE élément-racine **SYSTEM '**nom-du-fichier.dtd**'>**

- Syntaxe de déclaration de DTD externe de type PUBLIC :

<!DOCTYPE élément-racine **PUBLIC '**URL.dtd**'>**

Remarques :

- URL(Uniform Ressource Locator).
- La déclaration **DOCTYPE** est un mot-clef ; qui associe une DTD à une instance de document XML.

Voici un exemple de document XML (Parent.xml) ainsi que la DTD externe correspondante :

```
< !--Exemple de document XML-->
<parent>
        <garçon>sami<garçon>
        <fille>yasmine<fille>
</parent>
```

```
< ? xml version="1.0" ?>
< !—DTD externe pour Parent.xml-->
< !DOCTYPE parent[
<!element parent(garçon,fille)>
<!element garçon(#PCDATA)>
<!element fille (#PCDATA)> ]>
```

Document XML muni d'une DTD externe

II.4. XML-Schéma:

Une instance d'un *XML*-Schéma est un peu l'équivalent d'une *définition de type de document* (DTD). XML-Schéma amène cependant plusieurs différences avec les DTD : il permet par exemple de définir des domaines de validité pour la valeur d'un champ, alors que cela n'est pas possible dans une DTD ; en revanche, il ne permet pas de définir des entités. Voici un exemple de document XML "repertoire.xml" , et le document XML-Shéma qui lui correspond:

```
<?xml version="1.0" encoding="UTF-8"?>
<repertoire>
      <nom>pompiers</nom>
      <tel>18</tel>
</repertoire>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="repertoire">
        <xsd:complex Type>
                <xsd:sequence>
                        <xsd:element name="nom" type="xsd:string"
                />
                        <xsd:element name="num_tel"
                type="xsd:string" />
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
</xsd:schema>
```

document XML "repertoire.xml" , et le document XML-Shéma qui lui correspond

Remarque :

Un document XML est dit **bien formé** s'il obéit aux règles syntaxiques du langage XML (voir "annexe A" pour plus de détails sur ces règles). Un tel document sera correctement traité par un programme adéquat, comme un parseur XML. Si le document n'est pas bien formé son traitement provoquera un message d'erreur ou un arrêt de l'application qui le traite.

Un document XML **valide** est forcément un document bien formé mais il obéit en plus à une structure type définie dans une DTD ou dans XML-schéma.

II.5. Analyseur (parser) XML:

XML est uniquement un langage de structuration et de représentation de données. Il ne comporte pas d'instructions de contrôle et donc ne permet pas d'exploiter directement les données. Afin de traiter ces données, il faut disposer d'un analyseur.

Un analyser XML (en anglais parser XML) est un outil logiciel qui permet de parcourir un document XML et de récuperer de sa structure des balises, leur contenu, leurs attributs et de les rendre accessibles.

XML dispose de deux types de parseurs:

- Le parser **DOM** (Document Object Model).
- Le parser **SAX** (Simple API for XML).

II.5.1. DOM (Document Object Model):

Le modèle DOM (ou Modèle Objet de Document) est une recommandation du W3C (World Wide Web Consortium) depuis octobre 1998 qui permet de représenter un document XML sous forme d'arbre d'objets reliés entre eux, Chaque objet représente une entité (document, élément, attribut, texte,…) d'un document XML. Il définit une API (Application Programming Interface ou interface de programmation d'application) pour parcourir et modifier l'arborescence.

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

Exemple de document XML et l'arbre DOM qui lui est associé [19] :

```
<? xml version='1.0' standalone='yes'?>
<universite>
        <nom>Mouloud Mammeri  </nom>
        <ville >Tizi-Ouzou </ville>
        <pays>Algerie </pays>
</universite>
```

L'arbre DOM du document XML :

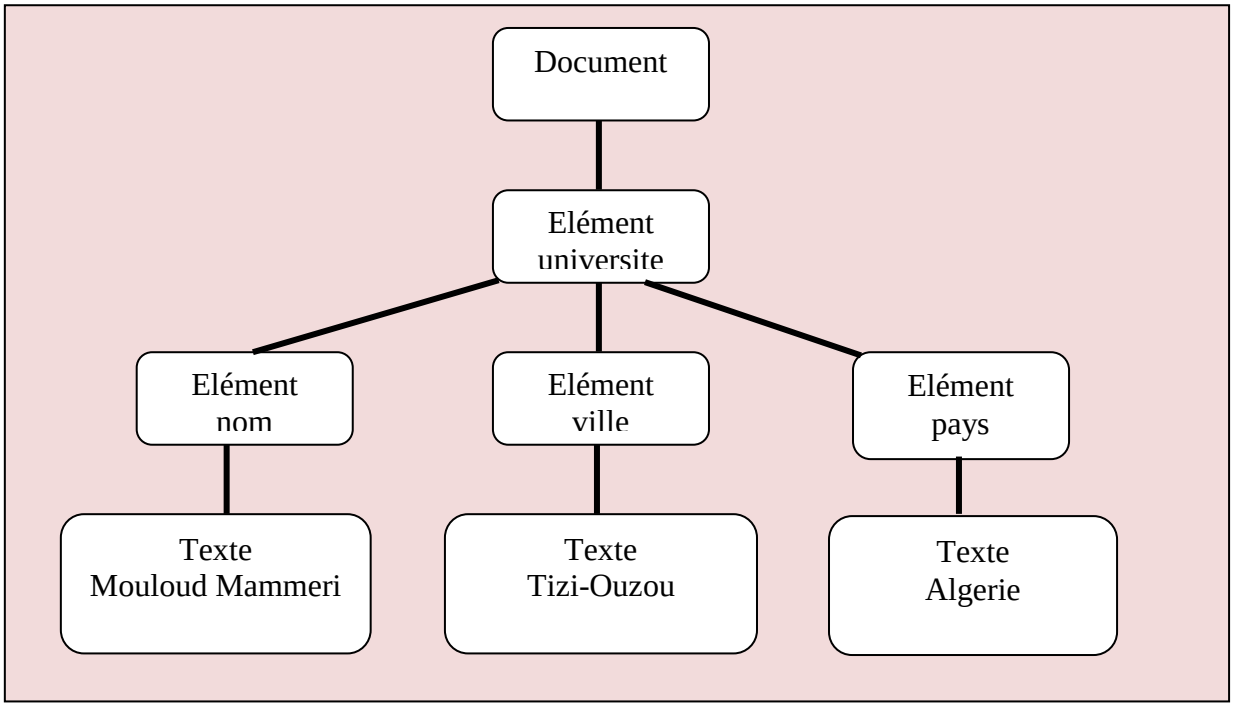


Figure II.3 : Exemple d'arbre DOM

II.5.2. SAX (Simple API for XML):

SAX fournit une interface événementielle pour parcourir un document XML. Cette API renvoie en effet, à l'application qui manipule le document XML des « *événements* » (Ouverture de balise, fermeture de balise, contenu textuel...).

SAX est bien adapté pour les traitements qui ne nécessitent qu'une seule passe sur le document, ou dans le cas de gros volumes de données, s'il n'est pas nécessaire d'avoir une représentation complète des données en mémoire.

L'API SAX définit les quatre interfaces suivantes :

- **DocumentHandler :** possédant des méthodes renvoyant des événements relatifs au document :
  - startDocument() : renvoyant un événement lié à l'ouverture du document.
  - startElement() : renvoyant un événement lié à la rencontre d'un nouvel élément.
  - characters() : renvoyant les caractères rencontrés.
  - endElement() : renvoyant un événement lié à la fin d'un élément.
  - endDocument() : renvoyant un événement lié à la fermeture du document.
- **ErrorHandler :** possédant des méthodes renvoyant des événements relatifs aux erreurs ou aux avertissements.
- **DTDHandler :** renvoie des événements relatifs à la lecture de la DTD du document XML.
- **EntityResolver :** permet de renvoyer une URL lorsqu'une URI est rencontrée.

Afin de comprendre comment une API événementielle fonctionne, considérons le document défini dans l'exemple suivant:

Exemple:

Soit le document XML suivant:

```
< ?xml version=''1.0'' ?>
<personne>
<nom>Samy</nom>
</personne>
```

L'interface événementielle décomposera la structure de ce document en une série d'événements linéaires:

Start document
Start element: personne
Start element: nom
Characters: Samy
End element: nom

End element: personne

End document

II.6. Conclusion :

Au cours de ce chapitre nous avons décrit le langage XML en commençant par définir sa syntaxe, un aperçu sur la validation des documents avec la DTD, puis la présentation des parseurs (DOM, SAX) qui permettent de les manipuler.

Chapitre III

La recherche d'information dans les documents XML

III.1. Introduction :

Le but des SRI pour les documents structurés, est de renvoyer des parties de documents les plus exhaustives[4] et spécifiques[5] possibles à l'utilisateur. Du point de vue de ces SRI, l'accès à ce type de documents soulève de nouvelles problématiques liées à la coexistence de l'information structurelle et de l'information de contenu. En effet, la prise en compte de l'information structurelle, devrait permettre, de mieux cibler l'information requise par les utilisateurs. Cette problématique sera détaillée dans le présent chapitre, ainsi que les différentes solutions proposées pour sa résolution. Nous présentons par la suite les différentes techniques d'indexation des documents XML avec quelques langages d'interrogation qui lui sont liés.

III.2. Les problématiques spécifiques à la Recherche d'Information Structurée (RIS) :

Pour permettre les différentes recherches dans les documents structurées, les techniques de la recherche d'information traditionnelle doivent être adaptées ou de nouvelles méthodes doivent être proposées pour l'indexation, l'interrogation ou encore la recherche et le tri des unités d'information. Ces différentes problématiques sont détaillées de la manière suivante :

a) La première problématique se rapporte à l'indexation des documents et concerne surtout la structure d'un document XML, car le document est représenté par un contenu informationnel (texte) et des contraintes structurelles (balises), et les questions suivantes se posent alors:

   Comment indexer la structure des documents ? Que doit-on indexer de la structure des documents ? Comment relier cette structure au contenu du document ? Quels sont les paramètres à considérer pour la pondération des termes d'indexation ?

---

[4] On dit qu'une unité d'information est exhaustive à une requête si elle contient toutes les informations requises par la requête

[5] Une unité d'information est spécifique à une requête si tout son contenu concerne la requête.

b) La deuxième problématique liée aux langages d'interrogation qui réside essentiellement sur la manière et les moyens à utiliser pour permettre à l'utilisateur d'exprimer des besoins diversifiés (concernant le contenu et la structure des documents)

c) La dernière problématique concerne les modèles de recherche et de tri des unités d'information. Cette problématique est liée à l'évaluation de la pertinence d'une information vis-à-vis d'une requête.

III.3. Techniques d'indexation des documents structurés :

L'indexation est l'un des processus pilier de la RI. Dans le cas des documents textes « Plats », le contenu textuel des documents est traité afin de trouver et de pondérer les termes les plus représentatifs des documents. Dans le cas des documents semi-structurés comme le document XML, la dimension structurelle s'ajoute au contenu.

Les techniques d'indexation dans les documents XML cherchent surtout à améliorer la vitesse et à diminuer l'espace occupé par l'index, elles doivent permettre :

- La restauration des documents, décomposée dans la structure de stockage.
- Le traitement d'expressions de chemin dans les documents XML
- Assurer une rapidité de navigation dans les documents XML.
- La recherche par mots clés.

Dans ce qui suit nous allons présenter, les différentes techniques adoptées pour l'indexation du contenu ainsi que l'information structurelle dans les deux sections qui suivent

III.3.1. L'indexation de l'information textuelle :

Dans la RI traditionnelle, cette technique consiste à extraire les termes les plus représentatifs des documents

III.3.1.1. portée des termes d'indexation :

Le problème de la portée des termes d'indexation se pose de la manière suivante :

- Comment rattacher les termes à l'information structurelle ?
- Doit-on chercher à agréger le contenu des nœuds ou au contraire à indexer tous les contenus des nœuds séparément ?

Afin de répondre à ces questions, deux solutions ont été proposée :

A) Sous arbres imbriqués :

Dans ces approches on considère que le texte complet de chaque nœud de l'index est un document atomique [20] et propagent donc les termes des nœuds feuilles dans l'arbre des documents

En d'autres termes, ces approches indexent tous les sous arbres (jugés potentiellement pertinents) des documents.

Comme les documents XML possèdent une structure hiérarchique, les nœuds de l'index sont imbriqués les uns dans les autres et l'index contient de nombreuses informations redondantes.

On présentera l'indexation des sous arbres dans la figure III.1 :

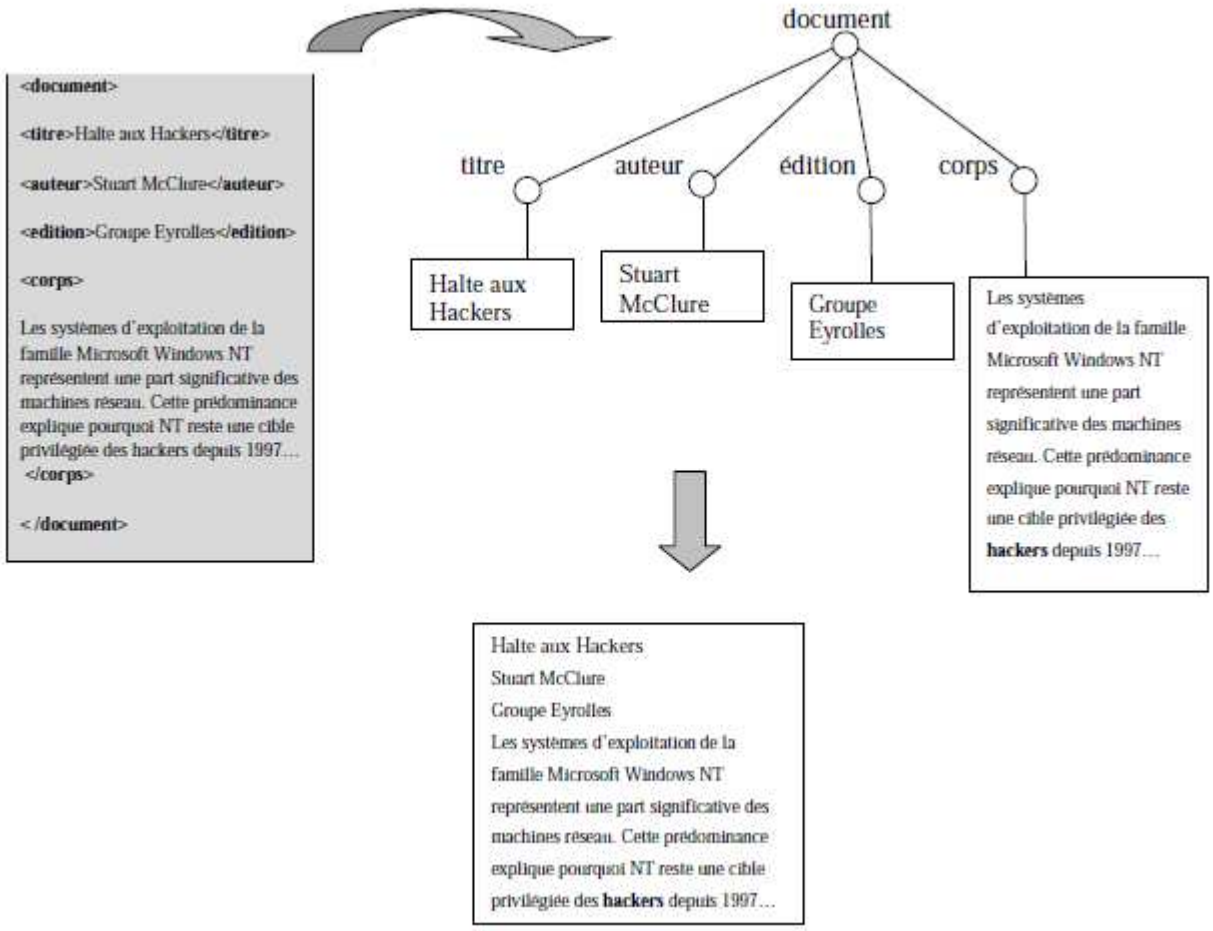


Figure III.1 Indexation de sous arbres imbriqués

Nous remarquerons que le terme « Stuart McClure » est relié au nœud /document/auteur et au nœud /document ; le terme « Hackers » est relié respectivement aux nœuds : /document/ titre , /document/corps et au nœud /document.

Nous voyons très bien sur la figure précédente les redondances d'informations induites par l'agrégation des différents nœuds imbriqués.

B) Unités disjointes :

Dans ces approches, le document XML est décomposé en unités disjointes, de telle façon à ce que le texte de chaque nœud de l'index est l'union d'une ou plus de ces parties disjointes. Les termes des nœuds feuilles sont uniquement reliés au nœud parent qui les contient.

Si on reprend en prend l'exemple l'arbre de la figure II.5, les termes " Halte aux Hacker" seront

uniquement reliés au nœud /document/titre, les termes « Stuart McClure » au nœud /document/auteur et les termes " reste une cible privilégiée des ***hackers*** " au nœud /document/corps.

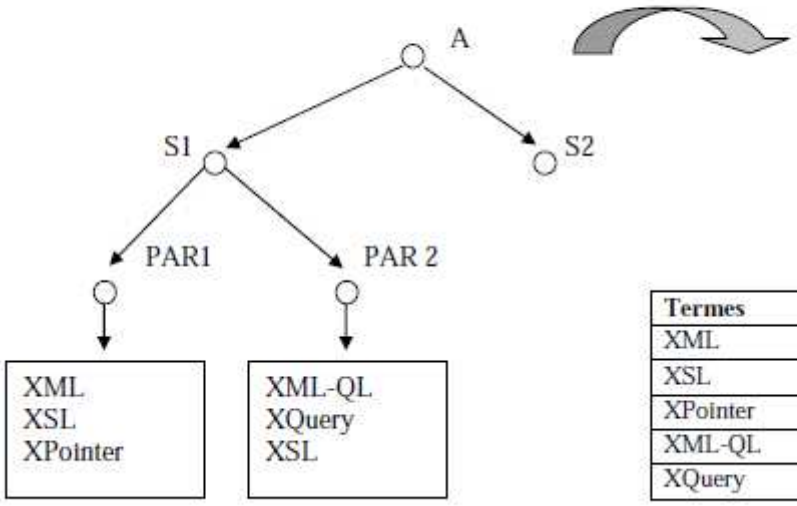
Ces différentes approches utilisées pour l'indexation textuelle entraînent l'utilisation des méthodes différentes pour l'interrogation des documents [22]

III.3.2. Indexation de l'information structurelle :

D'après [23], l'information structurelle peut être indexée selon des granularités variées, c'est à dire que toute l'information structurelle n'est pas forcément utilisée dans le processus d'indexation. Parmi les approches proposées dans la littérature, on distingue trois types d'approches pour l'indexation de l'information structurelle: l'indexation basée sur des champs, l'indexation basée sur des chemins, et enfin l'indexation basée sur des arbres. Nous allons détailler ces différentes approches dans les sections suivantes :

III.3.2.1. Indexation basée sur des champs :

Il s'agit certainement de la méthode d'indexation semi-structurée prenant en compte la structure la plus simple. Un document est représenté comme un ensemble de champs (par exemple titre, auteur, etc) et de contenu associé à ces champs. Pour permettre une recherche restreinte à certains champs, les termes de l'index sont construits en combinant le nom du champ avec les termes du contenu, comme l'illustre la figure III.2.

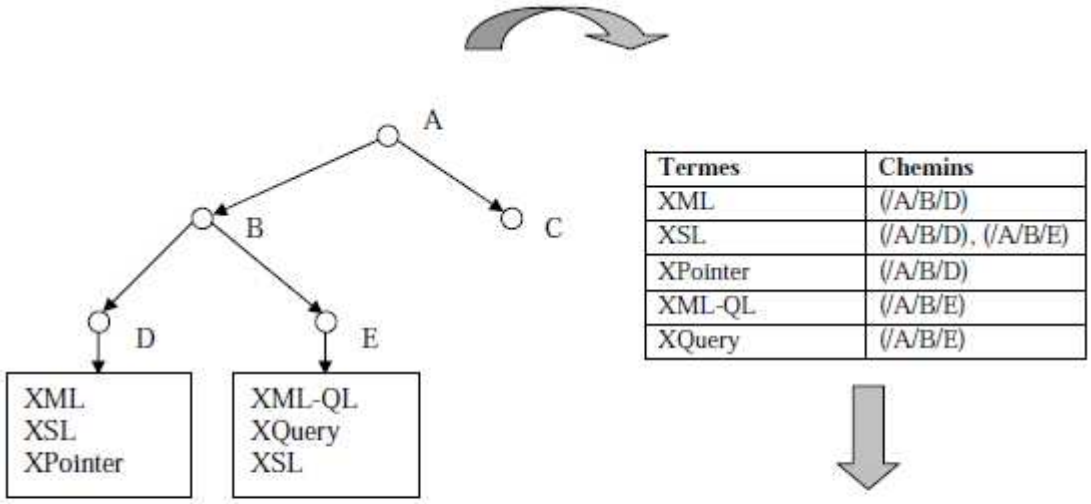


Termes	Champs
XML	(PAR1)
XSL	(PAR1), (PAR2)
XPointer	(PAR1)
XML-QL	(PAR2)
XQuery	(PAR2)

Figure III.2 – indexation basée sur des champs

III.3.2.2 Indexation basée sur des paths (chemins) :

Cette technique permet de faciliter la navigation dans les documents en permettant la résolution des expressions XPath, car le but de cette approche est de retrouver rapidement des documents ayant des valeurs connues pour certains éléments ou attributs, en utilisant des indexes de chemins, c'est des indexes donnant pour chaque valeur (élément, attribut) répertoriée d'un chemin de balises (de type XPath) la liste des documents possédante un élément accessible par ce chemin et ayant cette valeur, comme l'illustre la figure III.3



Termes	Chemins
XML	(/A/B/D)
XSL	(/A/B/D), (/A/B/E)
XPointer	(/A/B/D)
XML-QL	(/A/B/E)
XQuery	(/A/B/E)

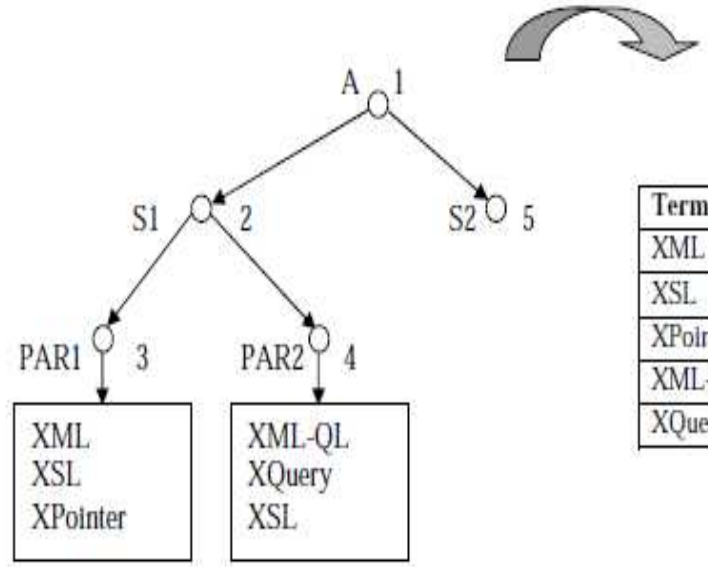
Chemins	Documents
(/A)	Doc1,…
(/A/B)	Doc1,…
(/A/B/D)	Doc1,…
(/A/B/E)	Doc1,…
…	…

Figure III.3 Indexation basée sur des chemins

III.3.2.3 indexation basée sur des arbres :

A

La figure III.4 donne un exemple d'indexation basée sur des arbres. Les nœuds de l'arbre sont numérotés dans les indexes de façon à pouvoir reconstruire la structure arborescente des documents.



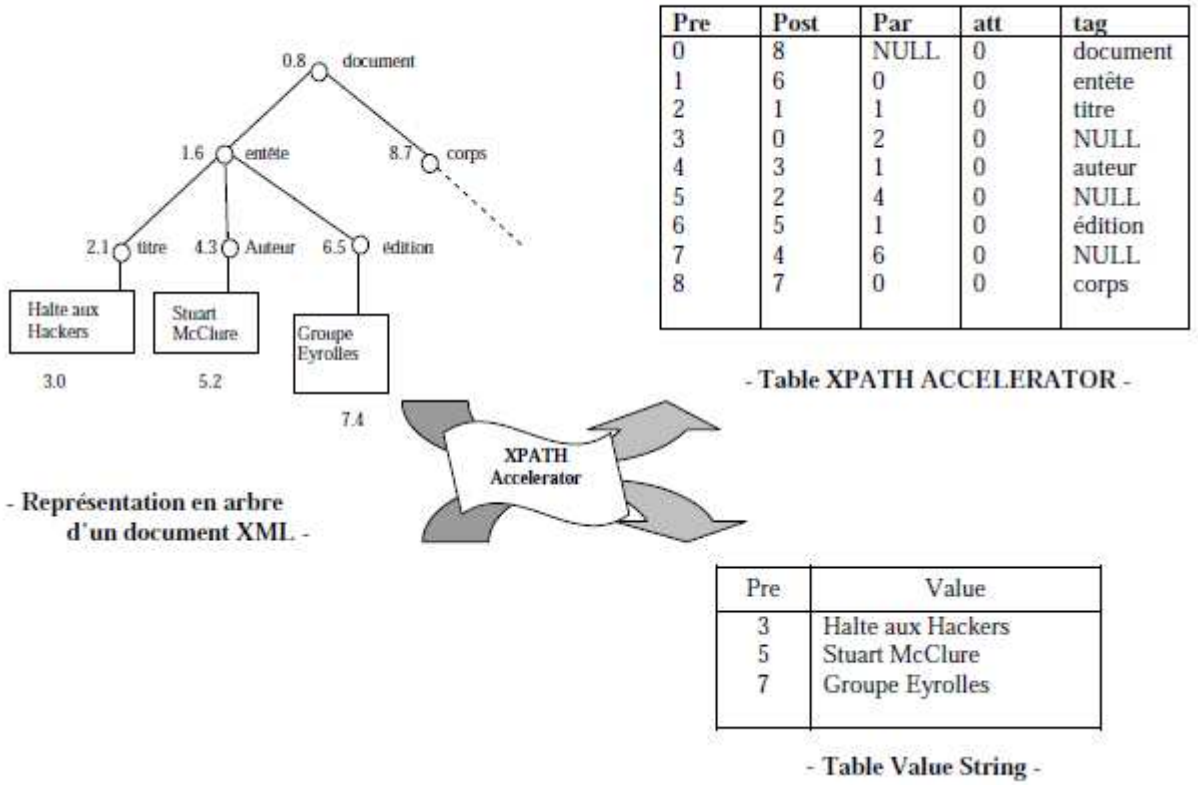
Termes	Noeuds
XML	3
XSL	3, 4
XPointer	3
XML-QL	4
XQuery	4

I I I . 3 . I I

Figure III.4 – Indexation basée sur des arbres

III.3.2.3.1 Xpath Accelerator:

La structure d'index du Xpath Accelerator [24] a été conçue pour l'évaluation des expressions de chemin. En chargeant un nouveau document XML, le Xpath Accelerator effectue une traversée de la représentation en arbre du document. Au cours de cette traversée, des valeurs croissantes de pré-ordre ou post-ordre sont assignées aux nœuds visités, comme le montre la figure III.5 ci-dessous.



Pre	Post	Par	att	tag
0	8	NULL	0	document
1	6	0	0	entête
2	1	1	0	titre
3	0	2	0	NULL
4	3	1	0	auteur
5	2	4	0	NULL
6	5	1	0	édition
7	4	6	0	NULL
8	7	0	0	corps

Pre	Value
3	Halte aux Hackers
5	Stuart McClure
7	Groupe Eyrolles

Figure III.5 – Transformation d'un document XML avec l'approche XPATH ACCELERATOR

Après avoir transposé les nœuds dans un espace à deux dimensions, basé sur les coordonnées de pré-ordre et post-ordre, le Xpath Accelerator exploite les propriétés suivantes, étant donné un certain nœud *v* :

Tous les ancêtres de *v* sont au-dessus à gauche de la position de *v* dans le plan, tous ses descendants sont en dessous à droite,

Tous les nœuds le précédant sont en dessous à gauche,

La partition du plan au dessus à droite comprend tous les nœuds successeurs.

En stockant de plus la dimension de prédécesseur du nœud parent, un champ indiquant la présence d'attributs et le nom de tag de chaque nœud, une navigation efficace devient possible.

Xpath Accelerator est particulièrement intéressant pour une navigation dans des documents XML et pour le traitement d'expressions Xpath.

III.4. Pondération des termes d'indexation :

La pondération des termes que nous avons définie dans le chapitre I sert à déterminer l'importance d'un terme au sein d'un document ou d'une collection de documents en lui attribuant un poids sémantique.

Dans le cadre de la recherche d'information dans les documents XML, les unités d'information sont imbriquées les une dans les autres, d'où l'importance d'un terme dans l'élément dépend fortement de l'importance de ce terme dans les sous éléments qui le composent.

Les techniques courantes de pondération des termes sont basées sur les notions de fréquence des termes dans le document ***tf*** (terme frequency) et de fréquence de ces termes dans l'ensemble des documents de la collection étudiée ***idf*** (inverse document frequency) et ***Ief*** (inverse element frequency).

Dans ce qui suit nous allons présenter quelques formules de pondération des termes dérivés des formules utilisées dans le cadre de la RI classique.

Formule 1 : c'est une adaptation de ***tf x Ief*** à l'information traitée dans les documents structurés (le nœud).

$$P_t = tf_t \text{ x } Ief_t$$

Formule 2 : reflète l'importance d'un terme au sein de l'élément et du document.

$$P_t = tf_t \text{ x } Ief_t \text{ x } Idf_t$$

tf : term frequency (fréquence d'un terme dans le nœud feuille)

Pt : Poids du terme t.

Ief : Inverted element frequency = fréquence inverse d'éléments

$= Log\left(\frac{Ne}{ne} + a\right)$ **avec 0.5 ≤ α ≤ 1** est une adaptation de *Idf* à la granularité de l'information à traiter qui n'est plus le document mais l'*élément* (le noeud).

Idf : Inverted document frequency = fréquence inverse de document $= Log\left(\frac{N}{n}\right)$

N : Nombre total de documents dans la collection

n : Nombre de documents contenant le terme t

Formules 3 :

$$w(t,e) = \frac{tf_t(e)}{\sum_{t' \in T} \sum_{e' \in E} tf_{t'}(e')} \times \log\left(\frac{N}{n_t}\right)$$

w(t,e): poids du terme ***t*** dans l'élément ***e*** ;

tft(e): fréquence du terme ***t*** dans l'élément ***e ;***

T : ensemble du vocabulaire ;

E: ensemble des éléments du document considéré. ;

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

n_t: Nombre total de documents contenant le terme ***t*** ;

Ensuite, le score de similarité entre une requête ***q*** et un élément ***e*** sera calculé avec les formule suivantes:

Si ***e*** est un élément feuille:

$$\boldsymbol{RSV(e,q) = \frac{W(e) \cdot W(q)}{|w(e)| \times |w(q)|} = \cos(W(e), W(q))}$$

Si ***e*** n'est pas un élément feuille, son score de similarité est calculé d'une manière récursive en partant des éléments feuilles:

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

Où:

p est généralement choisi avec une valeur égale à 2;

enf(e) représente l'ensemble des sous éléments de ***e***;

W(q) représente le vecteur de la requête dont les composantes (avec $\boldsymbol{1 \leq j \leq n}$) sont égales à 1 si le terme apparait dans la requête, et 0 sinon.

Tout en sachant qu'ils existent beaucoup de formules, et nous avons cité que les ces précédentes.

III.5 Les modèles de recherche dans les documents XML :

Nous présentons dans cette section les différents modèles qui répondent aux besoins et problématiques du RIS.

III.5.1 Le modèle vectoriel étendu :

Dans les approches issues du modèle vectoriel, une mesure de similarité de chaque élément à la requête est calculée, et ce à l'aide de mesures de distance dans un espace vectoriel. Les éléments sont représentés par des vecteurs de termes pondérés.

Pour ce faire, la plupart des approches indexent des sous-arbres imbriqués, c'est-à-dire propagent les termes des nœuds feuilles dans l'arbre du document. Les éléments sont renvoyés à l'utilisateur par ordre décroissant de pertinence [26]. La similarité d'un nœud ***n*** à une requête ***q = {t1, t2, ...tT}*** est exprimée selon la formule suivant :

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

Où

- ***α(T)*** : est un facteur permettant de prendre en compte le type du nœud,
- ***s*** : est le nombre de nœuds enfants $n_k$de ***n ;***
- ***β*** est un paramètre permettant d'assurer que le nombre d'enfants n'introduit pas un biais dans la formule.
- La fonction ***cosm*** est définie de la façon suivante :

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

Avec :

- w_i^qet w_i^nrespectivement le poids du terme $t_i$dans la requête ***q*** et dans le nœud n,
- $|n|$ le nombre de termes dans le nœud ***n,***

Une autre adaptation utilisée dans le système **JuruXML** [27] propose d'indexer les éléments selon leur type (un index par type d'éléments) et d'appliquer ensuite le modèle vectoriel pour la pondération des éléments.

Elle se base sur la décomposition et la représentation de la requête et des éléments en un ensemble de chemins ($c_{i\ q}$et$c_{i\ e}$), où sont respectivement le chemin de t_i dans le requête ***q*** et

dans le document ***d***. Ensuite, une mesure de similitude entres les chemins est calculée par la formule suivante:

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

Où $|c_i^q|$et $|c_i^e|$ sont respectivement la longueur du cheminc_i^q *et* c_i^e, elle se mesure en nombre d'éléments.

Par exemple, *cr(article/bibl, article/bm/bib/bibl/bb) = 3/6 = 0.5*

Le score de similarité entre une requête et un élément est alors calculé selon la formule suivante:

$$RSV(e,q) = \frac{\sum_{(t,c_i^q)\in q}\sum_{(t,c_i^e)} w_t^q \times w_t^e \times c_r(c_t^q, c_t^e)}{|e| \times |q|}$$

Où:

- $\boldsymbol{w_t^q}$et$\boldsymbol{w_t^e}$ représente respectivement, le poids du terme ***t*** dans la requête ***q*** et l'élément***e*** ;
- $|\boldsymbol{q}|\ et\ |\boldsymbol{e}|$sont les nombres de termes dans l'élément et la requête ;

III.5.2Le modèle booléen étendu :

Le modèle booléen étendu [28] se base, pour la prise en compte de la structure, sur les ***p distances*** et la propagation des scores depuis les feuilles de l'arbre jusqu'aux éléments.

Ce modèle est principalement adapté pour les requêtes orientées contenu, et permet de définir l'unité d'information à renvoyer à l'utilisateur.

Chaque élément feuille (noté) est alors représenté par un vecteur:

$$\boldsymbol{w(e_i) = (w_{i,1}, w_{i,1}, , , w_{i,n})}$$

Où ***n*** représente le nombre de termes d'indexation dans la collection de documents, et

$\boldsymbol{w_{i,j}}$ représente le poids du terme dans l'élément feuille qui est donné par:

$$w(t,e) = \frac{tf_t(e)}{\sum_{t'\in T}\sum_{e'\in E} tf_{t'}(e')} \times \log\left(\frac{N}{n_t}\right)$$

w(t,e): poids du terme ***t***dans l'élément ***e*** ;

tft(e): fréquence du terme ***t***dans l'élément ***e ;***

T : ensemble du vocabulaire ;

E: ensemble des éléments du document considéré. ;

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

n_t: Nombre total de documents contenant le terme ***t*** ;

Ensuite, le score de similarité entre une requête ***q*** et un élément ***e*** sera calculé avec les formule suivantes:

Si ***e*** est un élément feuille:

$$RSV(e,q) = \frac{W(e) \cdot W(q)}{|w(e)| \times |w(q)|} = \cos(W(e), W(q))$$

Si ***e*** n'est pas un élément feuille, son score de similarité est calculé d'une manière récursive en partant des éléments feuilles:

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

Où:

p est généralement choisi avec une valeur égale à 2;

enf(e) représente l'ensemble des sous éléments de ***e***;

W(q) représente le vecteur de la requête dont les composantes (avec $1 \leq j \leq n$) sont égales à 1 si le terme apparait dans la requête, et 0 sinon.

III.5.3Modèle probabiliste

Dans le modèle probabiliste [29], le classement des documents est basé sur la probabilité que le document retrouvé d implique la requête donnée q. Pour étendre le modèle probabiliste aux documents XML, les probabilités doivent tenir compte de l'information structurelle. Deux approches ont été développées.

La première approche permet d'utiliser des probabilités conditionnelles de jointure, par exemple :

P (d|t) devient ***P (d|p contains t)***

Où :

d : représente un document ou une partie du document,

t est un terme ;

p est un chemin dans l'arbre XML,

La deuxième permet d'étendre la logique propositionnelle à la logique prédicationnelle et prend en compte les problèmes liés à la structure : cette approche est basée sur la définition des relations entre les tuples des tables d'une base de données, et la modélisation des prédicats avec des formules logiques. Comme pour les modèles ensemblistes, cette formalisation ne permet pas d'ordonner une liste de documents et n'accorde aucune place à l'imprécision de la formulation de la requête.

Ce problème est résolu par Fuhr [30] qui s'est intéressé à la RI dans des bases de données. Il a proposé de combiner les approches de RI et les bases de données. Il propose une algèbre relationnelle probabiliste qui est une généralisation de l'algèbre relationnelle. Cette algèbre a pour but d'assigner des poids probabilistes aux tuples d'une relation. Ces poids donnent la probabilité qu'un tuple appartient à une relation. Cette approche présente deux avantages :

Permet de représenter les valeurs de données avec imprécision ;

Permet classer les documents par leurs poids probabilistes ;

Cette méthode est basée sur le langage de requête XIRQL[41], et a été implémentée au sein du moteur de recherche HyRex. Les termes sont propagés jusqu'au nœud indexable le plus proche. Le poids de pertinence des nœuds est calculé grâce à la propagation des poids des termes dans l'arbre du document. Le poids de chaque terme diminue par multiplication par un facteur nommé facteur d'augmentation. Considérons la structure du document illustré par la **Figure III.6**, nous attribuons à chaque terme un poids selon sa probabilité d'apparition dans un nœud. Nous voulons calculer le poids du terme modèle dans l'élément racine du document présenté par la **Figure II.6** (article) :

P ([article] ? [0.6, modèle] ? [modèle, 0.8])

Par la suite en introduisant le facteur d'augmentation (0.7) on aura :

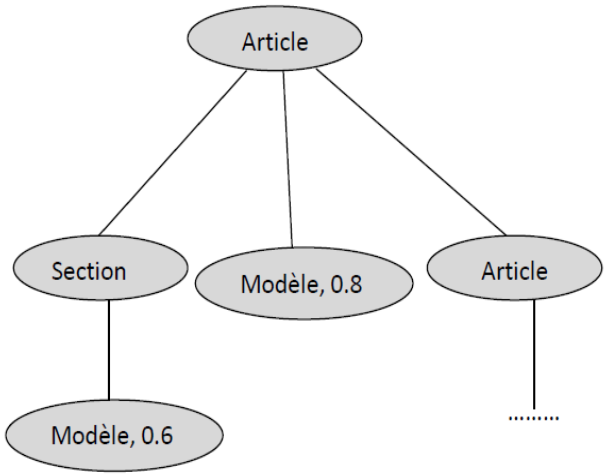


Figure III.6 : ***Modèle d'augmentation***

$$P([article] \square [0.6, modele] \square [modele, 0.8]) = P([article, modele]) + P([section, modele]) \times P([section]) - P([article, modele]) \times P([section]) \times P([section, modele])$$

$= 0.6 + 0.7 \times 0.8 + 0.6 \times 0.7 \times 0.8$

$= 0.824$

Pour des requêtes contenant des conditions structurelles, des sommes pondérés de ces probabilités sont cumulés.

III.5.4Modèle XFIRM

L'un des modèles de RIS qui se base sur le modèle vectoriel est le modèle XFIRM [31][32], il est basé sur un modèle de données générique permettant l'implémentation de nombreux modèles de RI et le traitement de collections hétérogènes (c'est-à-dire contenant des documents ne suivant pas la même DTD).

Le traitement des requêtes est effectué en deux temps : une première étape consiste à évaluer la similarité des nœuds feuilles de l'index à la requête (on parle alors de calcul des

poids des nœuds feuilles) et une seconde étape consiste à rechercher les sous-arbres pertinents.

La pertinence des sous-arbres est évaluée en effectuant la propagation des poids des feuilles dans l'arbre du document.

Les requêtes composées de conditions de structure sont décomposées en requêtes élémentaires de type *nom_element[condition contenu]* et chacune de ces requêtes est traitée de manière indépendante : on évalue la similarité des nœuds feuilles à la condition de contenu et une première propagation est effectuée pour répondre à la contrainte de structure.

Calcul du score des nœuds feuilles Les scores des nœuds feuilles identifiés dans l'arbre de document sont calculés grâce à la fonction de similarité **RSV** (*q, nf*).

Si la requête est composée de termes et des poids associés, on a :

$$RSV(q,nf) = \sum_{i=1}^{T} w_i^q \times w_i^{nf}$$

$$\text{Avec} \boldsymbol{w_i^q = tf_i^q} \text{ et } \boldsymbol{w_i^{nf} = tf_i^{nf} \times ief_i \times idf_i}$$

Où $\boldsymbol{w_i^q}$ et $\boldsymbol{w_i^{nf}}$ sont respectivement le poids du terme $\boldsymbol{i}$ dans la requête $\boldsymbol{q}$ et le nœud feuille $\boldsymbol{nf}$, avec $\boldsymbol{tf_i^q}$ et $\boldsymbol{tf_i^{nf}}$ sont respectivement la fréquence du terme $\boldsymbol{i}$ dans la requête $\boldsymbol{q}$ et dans le nœud feuille $\boldsymbol{nf}$, $\boldsymbol{idf_i = log\ (\frac{|D|}{|d_i|})}$ permet d'évaluer l'importance du terme $\boldsymbol{i}$ dans la collection de documents, où $|\boldsymbol{D}|$ est le nombre total de documents de la collection et $|\boldsymbol{d_i}||$ est le nombre de documents contenant le terme $\boldsymbol{i}$, et $\boldsymbol{ief_i = log\ (\frac{|F_c|}{|nf_i|})}$ permet d'évaluer l'importance du terme $\boldsymbol{i}$ dans la collection de nœuds feuilles, où $|\boldsymbol{Fc}|$ est le nombre total de nœuds feuilles de la collection, et $|nf_i|$ est le nombre de nœuds feuilles contenant le terme $\boldsymbol{i}$.

III.6 Compagne d'évaluation INEX :

INEX (INitiative for the Evaluation of Xml retrieval) est créée en 2002 dans le but d'évaluer les systèmes de recherche d'information structurée. Le but était le développement d'une méthodologie et des outils (collection de test, requêtes et mesures d'évaluation), pour tester des systèmes et approches de recherche XML, en utilisant un langage de requêtes qui est NEXI4 (Narrowed Extend XPath I) et un ensemble de tâches telles que : ad-hoc, la tâche multimédia, la tâche « Relevance Feedback » et la tâche hétérogène, et par la suite plusieurs

compagnies INEX dérivées de celle-ci à plusieurs évolutions et améliorations sont utilisées pour évaluer les SRI participant, telles que :

III.6 .1INEX 2005-2006 :

III.6 .1 .1 La collection de test :

En 2005 la collection de base est une collection d'outils provenant de la « IEEE Computer Society », ciblée au format XML. Cette collection est de volume d'environ 500 à 700 Mo composée de 8 millions éléments, équivalent à 12000 articles, d'environ 1500 éléments à profondeur moyenne de 6,9 (nombre de niveaux), provenant de 18 magasines ou revues différentes et on trouvera dans [12] un exemple de document complet de la collection INEX .

Les articles sont généralement structurés de la façon suivante :

- Chaque article est composé d'un en-tête (<fm>), un corps (<body>) et d'annexe.

Un élément, parmi ceux cités est conçu de la manière suivante :

- Le corps est composé de sections (<sec>)
- Une section est composée de paragraphes (<p>)
- Les annexes sont composées de références bibliographiques (<bibl>) et de curriculum vitae (<vt>)

A partir de 2006, la collection de base utilisée pour les tests étant la collection *Wikipedia*, qui est utilisée dans la plupart des tâches. Cette collection de 6 Go, est composée de 659.388 documents d'une profondeur (nombre de niveaux) moyenne de 6.72. Le nombre moyen de nœuds XML par document est 161,35. Cette collection est également utilisée dans la tâche multimédia, elle contient environ 246.730 images.

III.6 .1 .2 Les requêtes :

Sont appelées aussi *Topics*, elles sont créées par des différents participants et représentent les demandes moyennes de l'utilisateur, ainsi elles sont alors de différents types :

Les CO (Content Only) : Des requêtes en langage naturel telles que celles de TREC.

Les mots-clés de ces requêtes sont reliés par (+) pour impliquer l'obligation de présence du mot et (-) pour impliquer l'interdiction de sa présence, comme le montre l'exemple suivant :

```
<inex_topic topic_id="98" query type="CO">

<title> "Information Exchange" +"XML" "Information Integration"</title>

<description> How to use XML to solve the information exchange (information
integration) problem, especially in heterogeneous data sources ? </description>

<narrative> Relevant documents/components must talk about techniques of
using XML to solve information exchange (information integration) among
heterogeneous data sources where the structures of participating data sources
are different although they might use the same ontologies about the same
content. </narrative>

<keywords> Information exchange, XML, information integration,
heterogeneous data sources </keywords>
</inex topic>
```

Figure III.7: Exemple de requête CO, issue du jeu de test 2003

Les CAS (Content And Structure) : A l'indication du contenu bref du document voulu, on ajoute des contraintes de structure telles que les exigences de contenu des éléments du document XML

Pour chaque Topic, est construite une présentation sous forme de différents champs pour expliciter sa portée :

- Title : définition de la forme générale (CO)
- Key words : donne l'ensemble des mots clés de la requête qui ont permit l'exploration du corpus.
- Les champs Description et Narrative : expression en langage naturel des désires de l'utilisateur.
- Le champ Castitle pour la forme structurée CAS.

Et voici un exemple de requête orientée structure et contenu :

```
<inex_topic topic_id="205" query_type="CO+S" ct_no="12">
<InitialTopicStatement>McLuhan</InitialTopicStatement>
<title>marshall mcluhan</title>
<castitle>//bdy//*[about(., "Marshall McLuhan")]</castitle>
<description>
Find information about the relevance of Marshall McLuhan's ideas for current
digital technologies.
</description>
<narrative>
I am writing an essay on the inuence of new media icon Marshall McLuhan
on digital technologies. I'm seeking information describing how McLuhan's views
have inuenced current digital technologies. To be relevant, a retrieved item should
discuss some aspect of Marshall McLuhan's visionary ideas or famous one-liners
in the context of current digital technologies. Retrieved elements that merely cite
some of McLuhan's work are non-relevant, as are elements that discuss ideas not
originating from McLuhan.
</narrative>
```

Figure III.8: Exemple de requête CAS (Topic 205 d'INEX 2005)

III.6 .1 .3 Les tâches [INEX 2005]:

- La tâche ad-hoc :

C'est une recherche qui se fait en simulation de l'utilisateur d'une bibliothèque composée de documents XML, qui est interrogé par des requêtes utilisateur conçues dans la compagnie et portent sur le contenu et la structure. Cette principale tâche est fragmentée en trois sous tâches distinctes :

a- La tâche CO : (Content Only task)

Sert à répondre aux requêtes utilisateur de type CO par des granules d'information XML. Dans cette tâche, aucune indication de structure n'aide les SRI à savoir le type d'unité à retourner.

b- La tâche SCAS : (Strict Content And Structure task)

Répondre aux requêtes CAS avec des granules XML de manière stricte, ie : obéir aux exigences de structure et contenu à la fois, dont le champ Title des requêtes est basé sur une syntaxe XPath.

c- La tâche VCAS : (Vague Content And Structure task)

Répondre aux requêtes CAS de manière vague. ie : avec des granules satisfaisant globalement les requêtes.

Autres tâches:

En 2004, ils ont proposé quatre nouvelles tâches :

- La tâche « Relevance Feedback », usage des exigences de structure et contenu pour la reformulation des requêtes.
- La tâche « Language Natural », formulation des requêtes en langage naturel.
- La tâche « Interactive », étude du comportement utilisateur face aux corpus XML pour cerner son besoin.

La tâche *hétérogène*, proposée aux participants de nouvelles collections pour qu'ils développent des approches indépendantes des DTDs.

III.6 .1 .4 L'évaluation :

L'évaluation de pertinence des SRI XML passe premièrement par une validation des éléments/documents qui sont jugés à la main. Par la suite, en 2002 une échelle à deux dimensions a été proposée et basée sur les degrés de pertinence et couverture, qui ont été remplacée par les notions d'exhaustivité et de spécificité, depuis 2003. Telles que :

L'exhaustivité est mesurée selon une échelle à quatre niveaux, un élément est :

Pas exhaustif : il ne traite pas du tout du sujet de la requête.

Marginalement exhaustif : il traite peu d'aspects du sujet de la requête.

Assez exhaustif : il traite de nombreux aspects du sujet de la requête.

Très exhaustif : il traite la plus part ou touts les aspects du sujet de la requête.

La spécificité décrit à quel point l'élément est focalisé sur la requête, ie : la couverture du document/élément au sujet et elle est aussi mesurée sous quatre niveaux, l'élément peut être à :

Pas de couverture : le thème traité n'a rien à avoir avec celui de la requête, ie : n'est pas du tout spécifique.

Couverture trop large : ou marginalement spécifique où le thème de la requête est traité exactement dans un sous élément.

Petite couverture : si l'élément renvoyé contient juste une partie de l'information pertinente.

Couverture exacte : ou l'élément est très spécifique et le seul sujet qui traite est celui de la requête.

L'usage de l'échelle à deux dimensions, est impliqué par le besoin de mesurer la pertinence d'un élément par rapport à son descendant, et lorsqu'un élément n'est pas pertinent, il n'a pas de couverture et inversement.

Mesures d'évaluation :

Jusqu'à 2004, les seules mesures appliquées dans l'évaluation des SRI étaient le rappel et la précision, par la suite dans les compagnies INEX 2005 et 2006 d'autres mesures ont été définies pour mettre une meilleure évaluation des SRI en RI structurée [Xavier] :

Le gain cumule (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 :

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

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 xCGI à partir de la base de rappel, et le xCG peut être alors comparé au xCI avec le nxCG :

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

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 :

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

Tel que : *r* : c'est le gain

e : le rang correspondant au gain

$e_{idéal}$: 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 :

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

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.

Jugement de pertinence :

Pour juger la pertinence d'un SRI, il faut l'évaluer suivant les mesures d'évaluation conçues dans les compagnies. Mesurer la performance d'un SRI structuré, revient à mesurer sa capacité de retrouver et restituer les documents à la fois *exhaustifs* et *spécifiques* à la requête.

III.6 .2 INEX 2007 : [42]

Le changement principal dans INEX 2007 concerne la permission de retourner des parties arbitraires d'un document et l'évaluation de la pertinence d'un texte d'un élément en fournissant des requêtes diverses telles que :

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

Figure III.9: Exemple d'une requête INEX 2007

L'évaluation s'effectue par l'étude des trois tâches ad hoc suivantes :

III.6 .2 .1 Focused tasck (la tâche concentrée) :

Demande aux systèmes participants de renvoyer une liste triée de tout les éléments ou unités d'informations sans chevauchement, par exemple dans le cas du renvoie des éléments XML, un paragraphe et sa section conteneur ne devraient pas être renvoyés à la fois. Pour cette tâche, à partir de toutes les parties pertinentes estimées pour un document, les systèmes participants doivent choisir les éléments non-chevauchants qui représentent les unités les plus appropriées à la recherche.

III.6 .2 .2 In context tasck (tâche de mise en contexte) :

Elle correspond à la tâche « User » dont les réponses focalisées sur la recherche sont groupés par document dans leur ordre original et permettant à l'utilisateur d'y accéder via

d'autre moyens de navigation. Ceci, en supposant que l'utilisateur considère le document comme l'unité de recherche la plus naturelle et souhaite avoir un aperçu de celle-ci. Cette tâche est composée de deux sous tâches principales :

Relvant in context : (appropriée dans le contexte), demande aux systèmes le renvoi des parties non chevauchées des documents (éléments XML ou passages) groupés par documents où elles sont contenues.

Best in context : (mieux dans le contexte), demande aux SRI de renvoyer une simple unité XML par document, qui doit correspondre au meilleur point d'entrée pour commencer la lecture du texte approprié.

III.6 .2 .3 L'évaluation de pertinence :

Pour chaque partie pertinente d'un document (passage ou élément XML), l'évaluation de pertinence enregistre la taille du texte mis en évidence contenu dans cette partie aussi bien que la partie texte de tout le document. Dans cette procédure, il est demandé de mettre en évidence les phrases représentants l'information pertinente dans un ensemble de documents XML de la collection Wikipedia utilisée. Un programme d'évaluation calcule ensuite la pertinence des parties jugées du document (y compris le document tout entier), ainsi les valeurs de pertinence associées aux parties sont triées dans une échelle continue de 0 à 1 où :

0 correspond à une partie documentaire ne contenant aucune information pertinente,

1 correspond à une partie pleine d'information pertinente et enregistrer la taille du texte pertinent correspondant.

III.6 .2 .4 Les mesures d'évaluation :

Depuis 2007, les mesures officielles sont basées sur l'interpolation de Rappel/Précision sur 101 niveaux. Soit p_r la partie (élément XML ou passage) correspondante au rang r de la liste triée des parties $\mathcal{R}$ d'un document retournée par un SRI :

Avec $|\mathcal{R}| = 1500$ éléments ou passages. Et soient $rsize(p_r)$ la quantité du texte pertinent contenu dans p_r (si pas de texte pertinent, $rsize(p_r) = 0$: mise en évidence de la pertinence du texte), *Trel* la taille totale (par nombre de caractères) d'un texte pertinent répondant à une requête INEX 2007 et $size\ (p_r)$ la taille de p_r en nombre de caractères.

Pour la tâche concentrée, les systèmes sont demandés de retourner une liste triée des parties-document par leurs valeurs estimées de pertinence et de même décider quelles sont les moins chevauchées :

Mesurer la pertinence de la fraction de texte recouvré au rang *r* par la fonction suivante comme mesure de précision :

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

Pour réaliser un score de haute précision au rang *r*, on doit mesurer la pertinence de la fraction en tenant compte de la quantité du texte non-pertinent contenu dans cette fraction par la fonction suivante comme fonction de rappel :

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

Et puis appliquer la précision moyenne (*AP : Average Precision*), en calculant la précision à chaque niveau de rappel (après avoir retrouvé une partie pertinente d'un document) :

$$AP = \frac{\sum_{r=1}^{|R|} rel(p_r).p[r]}{\sum_{r=1}^{|R|} rel(p_r)} . R[|\mathcal{R}|]$$

Avec : [$|\mathcal{R}|$] = nombre total de parties du document correspondant

$$rel\ p_r = \begin{cases} 0\ si\ la\ partie\ p\ du\ document\ ne\ contient\ pas\ d'information\ pertinente\ soulignée \\ 1 \end{cases}$$

sinon

Et indique la pertinence de la partie p_r du document.

D'autres mesures sont utilisées pour évaluer les résultats des autres tâches :

Soit $\mathcal{P}_d$ l'ensemble des parties d'un document et *p* une partie de cet ensemble.

$$P(d) = \frac{\sum_{p \in \mathcal{P}_d} rsize(p)}{\sum_{p \in \mathcal{P}_d} size(p)}$$

Pour montrer que pour atteindre la haute précision, il faut qu'il contienne le moindre possible de textes non-pertinent dans un document.

Le rappel-document :

$$R(d) = \frac{\sum_{p \in \mathcal{P}_d} rsize(p)}{Trel(d)}$$

Pour montrer que pour atteindre le haut rappel dans un document il faut qu'il contienne le maximum possible de texte pertinent.

Le F-score du document :

$$F(d) = \frac{2.P(d).R(d)}{P(d) + R(d)}$$

Cette mesure est aussi utilisée dans la tâche in-context pour mesurer le score *S* d'un document *d* par la fonction suivante : $S(d) = F(d) \in [0,1]$ (0 : document sans texte pertinent et 1 : tout les textes pertinents sont recouverts sans avoir à recouvrir du texte non-pertinent).

Il y a aussi la mesure d'évaluation de la tâche *Best in context* où le score du document d est calculé par une mesure de distance de similarité :

$$S(x,b) = \frac{A.L}{A.L+d(x,b)}$$ Avec :

L : la longueur du document *d*

A > 0 est un paramètre de contrôle

b : est le meilleur point d'entrée (élément) pour lire le texte pertinent nommé BEP (Best Entry Point)

x : est un point d'entrée quelconque

Par la suite, on aura une liste triée de documents par leurs scores de pertinence et calculer d'autres valeurs telles que la précision généralisée et le rappel généralisé.

De 2006 à 2008, la collection de tests utilisée a resté la même étant la collection *Wikipedia* aux mêmes caractéristiques.

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

D'autres collections sont aussi fournies par la compagnie d'évaluation pour évaluer d'autre tâche telles que la collection mmWikipedia pour une sous tâche de la tâche multimédia. Et avec le développement continu des SRI et l'apparition des nouvelles techniques dans le domaine de la RI, la compagnie INEX est vue s'améliorée jusqu'à INEX 2012 où de nouvelles mesures sont apparues et de nouvelles collections et tâches sont fournies aux SRI.

III.6. 3 INEX 2012 : [43]

En 2012, INEX et en collaboration avec CLEF, ont étudié les différents aspects pour l'accès à l'information concentrée et ils ont établi les tâches de base pour l'évaluation des SRI en se basant sur des collections appropriées:

La tâche des données liées LDT (Linked Data Track) : examinant la récupération sur une collection de documents fortement structurés, ses dernier sont tirés de
DBpedia et Wikipedia. La tâche ad hoc a des requêtes à satisfaire avec des entités de la collection composée des articles Wikipedia et des propriétés RDF[5] de DBpedia et
YAGO2[6]. La tâche de recherche de facettes demande de retourner une liste limitée d'éléments qui guideront l'utilisateur de façon optimale vers des informations pertinentes.

Relvance feedback track (RFT) : qui examine l'utilité du niveau de passage progressif du relvance feedback en simulant l'interaction d'un chercheur. Une évaluation non conventionnelle suit la trace où les soumissions sont des programmes informatiques exécutables plutôt que des résultats de recherche, en se basant sur la collection INEX Wikipedia de 50.7GB de 2.666.190 articles XML de Wikipedia.

Snippet Retrieval Track (SRT) : (tache de récupération des petits bouts) qui examine la façon de génération des petits bouts informatifs (unités informatives) et qui devrai fournir des informations pertinentes permettant à l'utilisateur de déterminer la pertinence de chaque document sans avoir à consulter toute la collection. En utilisant la collection Wikipedia d'INEX 2009 qui est une version
XML d'English Wikipedia.

Social book search track (SBST) : la tâche de recherche des livres sociaux, examine des techniques pour soutenir les utilisateurs dans la recherche et la navigation dans des collections numériques des livres (à des livres numérisés), des métadonnées et des médias sociaux complémentaires. Elle étudie la valeur relative des métadonnées autorisées et le contenu créé par l'utilisateur en se basant sur une collection de test à des données de l'Amazone et Library Thing et la tâche demande des pages confirmant ou réfutant l'utilisation d'un corpus de textes pleins.

Tweet contextualization track (TCT) : sert à répondre à des requêtes de forme « What is this tweet about ? » avec un résumé synthétique d'informations contextuelles saisies de Wikipedia et évaluées selon la pertinence du texte retourné « le dernier point d'intérêt » (the

last point interest) en se basant sur une collection tirée de Wikipedia 2011 ainsi que 1000 requêtes en anglais.

III.7.Conclusion :

Dans ce chapitre nous avons vus La recherche d'information dans les documents semi structurés, qui a réactualisé de nombreuses problématiques de la recherche d'information. En effet la prise en compte de la structure du document engendrait de nouvelles problématiques au niveau : de l'indexation, de l'interrogation, du modèle d'appariement granules documentaires/requêtes, etc. Nous avons cité quelques approches qui ont été proposées dans la littérature et un certain nombre de solutions ont été trouvées. Pour finir, nous avons présenté la compagne d'évaluation INEX, qui est la compagne de référence pour l'évaluation des systèmes de RI structurée.

Dans ce qui va suivre nous allons détailler un modèle de RIS qui implémente la propagation des termes.

Chapitre IV

Méthode De Propagation

Des Termes

Et formules de pondérations

IV.1. Introduction :

Certains systèmes se sont intéressées à mieux répondre au besoin de l'utilisateur en considérant un niveau de granularité plus petit que le document. A cette fin, nous implémentons une méthode basée sur la propagation des termes. Cette méthode repose sur une propagation de termes des nœuds feuilles vers la racine des documents de manière Statique, c'est-à-dire avant la formulation d'une requête. Cette méthode proposée par S.Fellag dans son projet de magister **[17]**, permet de répondre aux requêtes utilisateur portant sur le contenu (composée de simples mots clés) et de résoudre le problème de la granularité d'information à renvoyer à l'utilisateur, puisqu'il s'agit de trouver le sous arbre de taille minimale qui répond spécifiquement et exhaustivement à la requête d'un utilisateur.

IV.2. Représentation logique des Documents :

Le modèle logique de représentation des documents qui a été utilisé est similaire au modèle logique utilisé dans le système XFIRM [12].

Ce modèle permet la navigation dans la structure en arbre des documents XML, de représenter le contenu et la structure, afin de pouvoir interroger ces documents et récupérer la partie de ces derniers qui répond le mieux à la requête utilisateur. *Un document XML est un arbre, composé de nœuds. Un nœud peut être un élément, un attribut, du texte, une instruction ou alors un commentaire.*

Pour naviguer aisément dans l'arbre, permettre l'accès rapide à un nœud, et déterminer rapidement les relations ancêtres-descendants l'approche utilisée est XPath Accelerator [24] telle que, un nœud est défini grâce à ses valeurs de pré-ordre et post-ordre (*pre* et *post*), la valeur de pré-ordre de son nœud parent (*par*) et selon que ce soit un nœud simple ou un nœud feuille, par un champ indiquant la présence d'attributs (*attribut*) ou les termes qui le composent (*{t1, t2, …tn}).*

Un attribut est défini par la valeur de pré-ordre du nœud auquel il se rattache (*pre*) et par sa valeur *(val).*

Les valeurs de pré-ordre et post-ordre sont assignées aux nœuds comme suit : en chargeant un nouveau document, on effectue un parcours séquentiel de la représentation en arbre du document structuré. Un parcours préfixé permet d'assigner à chaque nœud visité une

valeur croissante de pré-ordre (*pre*) avant que ses nœuds descendants ne soient aussi récursivement visités de gauche à droite. D'une manière inverse, la valeur de post-ordre (*post*) d'un nœud lui est assignée lors d'un parcours post fixé, c'est à dire une fois que tous ses nœuds descendants ont été visités de gauche à droite.

La figure ci-dessous donne un exemple pour le document *article.xml* extrait de l'article de *Ronald Bourret* sur *"XML et les bases de données"*.

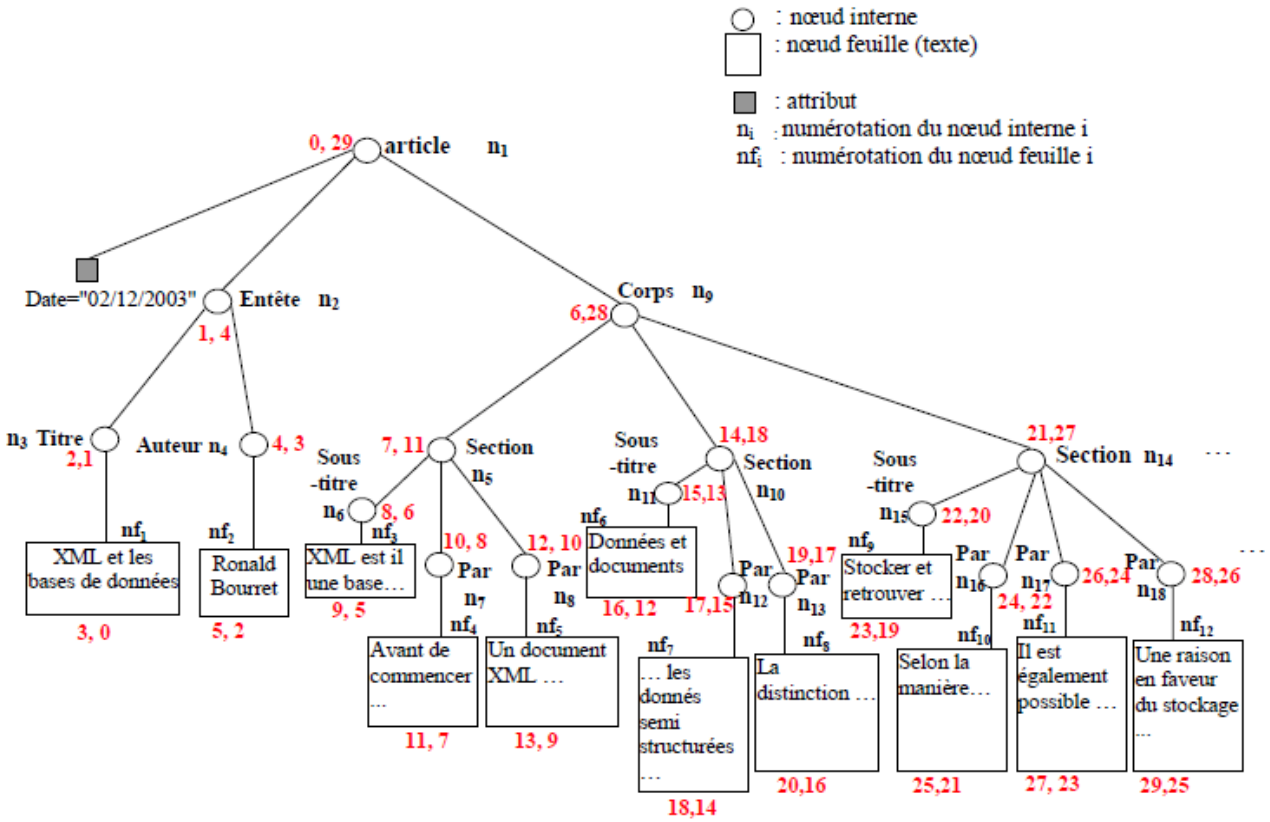


Figure IV.1 : Valeurs de pré-ordre et post-ordre assignées aux nœuds du document Article.xml

Si l'on transpose tous les nœuds dans un espace à deux dimensions basé sur les coordonnées de pré-ordre et post-ordre, on peut exploiter les propriétés suivantes illustrées par l'exemple de la figure V.1 Etant donné un certain nœud *n* (le nœud *article [1]/corps [1]/section [2]* dans l'exemple ci-dessous):

- tous les ancêtres de *n* sont au-dessus à gauche de la position de *n* dans le plan
- tous ses descendants sont en dessous à droite

- tous les nœuds le précédant dans la lecture séquentielle du document sont en dessous à gauche.
- La partition du plan au dessus à droite comprend tous les nœuds successeurs dans la lecture séquentielle du document.

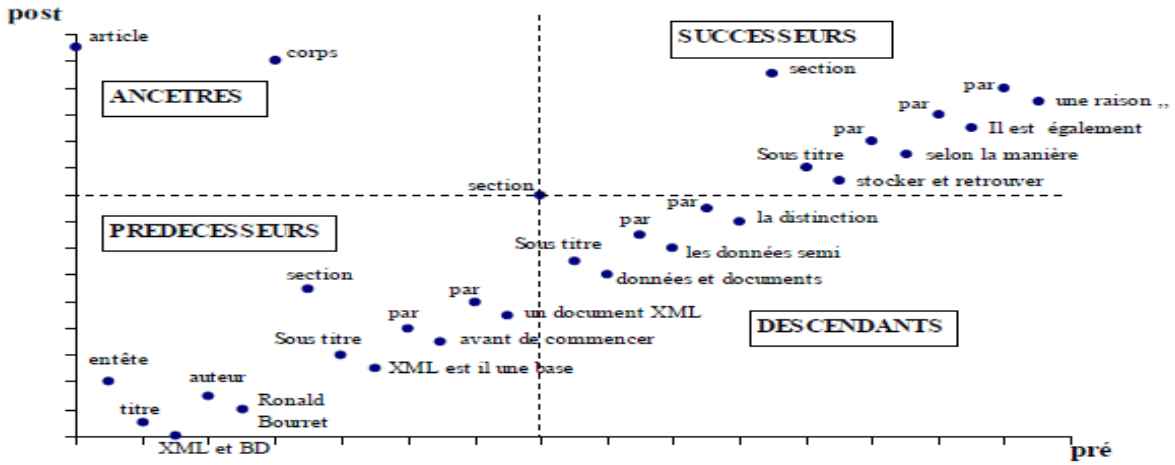


Figure IV.2 : Représentation du document article.xml dans un espace à deux dimensions basé sur les coordonnées de pré-ordre et post-ordre

Exemple : *Un nœud* ***n'*** *est ancêtre de* ***n*** *si pre(n') < pre(n) et post (n')> post (n)*

En plus du traitement des expressions XPath, cette représentation des nœuds est particulièrement intéressante pour une navigation dans la structure des documents.

IV.3. Indexation des termes des nœuds feuilles :

Dans les documents XML, le texte est situé au niveau des nœuds feuilles. Pour l'indexation du contenu de ces nœuds, une liste de mots vides est utilisé (*stoplist* en anglais) elle contient les mots vides comme les pronoms et les déterminants. Les nœuds feuilles sont parcourus séquentiellement, après suppression des mots vides, les termes d'index sont récupérés et rangés dans le fichier inverse après troncature.

L'algorithme de troncature utilisé, peut être celui de Porter si les termes sont en anglais, ou l'algorithme de troncature à 7, si les termes sont en français. Chaque terme ainsi rangé dans le

fichier aura une fréquence de un (01), celle-ci sera incrémentée à chaque fois que le terme est rencontré dans le même élément.
Les noms des balises et les noms d'attributs ne subissent aucun traitement.

IV.4. Représentation physique des documents :

Le stockage des documents XML est basé sur la représentation logique des documents du système XFIRM **[12]**, se fera suivant une approche bases de données relationnelles.

Algorithme de stockage d'un document XML :

Charger un document XML en mémoire
Lui attribuer un identifiant
Sauvegarder l'identifiant et le nom du document dans la table document
Parcourir le document avec un parser Sax
Attribuer les valeurs de pré-ordre et post-ordre à chaque nœud
Identifier chaque nœud visité avec idf_empl
Sauvegarder dans la table emplacement idf_empl, ainsi que les valeurs de pré-ordre et de post-ordre du nœud et la valeur du pré-ordre de son parent
Si le nœud est une balise alors
Si la balise existe déjà (à vérifier dans la table balise) extraire son identificateur de cette table
Sinon lui attribuer un identificateur, sauvegarder cet identificateur ainsi que le nom de la balise dans la table balise
sauvegarder dans le champ typ_elt de la table emplacement l'identificateur de la balise
Si la balise possède un attribut alors
Si l'attribut existe déjà (à vérifier dans la table attribut) extraire son identificateur de cette table
Sinon lui attribuer un identificateur, sauvegarder cet identificateur ainsi que le nom de l'attribut dans la table attribut
mettre le champ attr de la table emplacement à vrai
Sauvegarder l'idf_empl de l'attribut ainsi que son identifiant et sa valeur dans la table val_attribut
Sinon mettre le champ attr de la table emplacement à faux

Sinon si le nœud est un nœud feuille alors
sauvegarder dans le champ typ_elt de la table emplacement « texte » et mettre le champ attr à faux
Récupérer les termes un à un
Les indexer (lemmatiser), stocker dans le fichier inverse (la table termes)
Compter le nombre d'apparition du terme dans le nœud
Sauvegarder cette fréquence dans le champ Pds du fichier inverse et mettre le champ indiEc à faux (le champ indic sera mis à vrai après calcul du poids des termes).
Les étapes décrites ci-dessus sont réalisées pour tous les documents XML appartenant au corpus de test.

IV.5. Traitement des requêtes :

Le traitement des requêtes a pour but de renvoyer les unités d'informations les plus *spécifiques* et les plus *exhaustives* à l'utilisateur.
Les requêtes que l'utilisateur peut formuler, sont composées de *simples mots clés sans précision aucune sur la structure* et c'est au système, de décider de la granularité appropriée de l'information à renvoyer.
Le traitement de requêtes CO proposé par l'auteur, s'effectue comme suit :

- La première phase, consiste à pondérer les termes des nœuds feuilles.
- la seconde phase concerne, l'élagage de l'arbre du document, en ne conservant que les nœuds informatifs.
- la troisième phase, consiste à propager les termes bien distribués dans les nœuds feuilles, vers les nœuds ascendants. Afin de pouvoir identifier les unités d'information, pertinentes et informatives.
- enfin, la dernière phase consiste, à calculer le score de pertinence des unités d'information ainsi identifiées, avec la requête et à présenter les résultats obtenus, par ordre décroissant des scores.

Le calcul du score d'un nœud et la pondération des termes dans les nœuds sont des éléments prépondérants dans la phase d'évaluation de la pertinence d'un nœud vis-à-vis d'une requête. Avant de présenter le processus de propagation nous commençons par décrire ces deux points.

IV.6. Présentation des formules de pondérations utilisées :

Une formule de pondération sert à déterminer l'importance d'un terme au sein d'un document ou d'une collection de documents en lui attribuant un poids sémantique. Dans notre SRI nous allons intégrer onze formules de pondération qui se présente comme suit :

- $F1 = tf * idf$
- $F2 = tf * ief$

} Sauvagnat [12]

- $F3 = tf * idf * ief$ FELLAG.S [17]

Et celles proposées par FELLAF.S [17] et déjà intégrées par KESSIL[39] Dans XFIRM :

- $F4 = tf * ief * \frac{1}{h_1 + h_2 * \frac{det}{\Delta et}}$
- $F5 = tf * idf * ief * \frac{1}{h_1 + h_2 * \frac{det}{\Delta et} + h_3 * \frac{dct}{\Delta ct}}$

Notre camarade à greffé cinq nouvelles formules dans le SRI XFIRM en se basant sur la BM25 utilisée par Géry Mathias [40] et une sixième formule de pondération étant une normalisation de la *tf*ief*, pour mesurer le poids d'un terme dans un élément d'un document XML .

- $F6 = \frac{tf_{ij} * (K_1 + 1)}{K_1 * ((1-b) + (b * ndl)) + tf_{ij}} * log \frac{N - df_i + 0.5}{df_i + 0.5}$ [40] où :

$$ndl = \frac{|e_j|}{\Delta |e_j|} = \frac{det}{\Delta et}$$

Avec :

- $|e_j|$ est la taille de l'élément e_j en nombre de termes et $\Delta|e_j|$ est la taille moyenne d'un élément ;
- tf_{ji} : la fréquence de ti dans e_j ;
- N : le nombre d'éléments dans la collection ;
- df_i : le nombre d'éléments qui contiennent le terme t_i dans le document;
- ndl : le ratio entre la taille de e_j et la taille moyenne des éléments (en nombre de termes) ;
- $k_1 = 1.2$ et $b = 0.75$

Les formules suivantes concernent les paramètres *N* et *ndl* normalisés :

- *F7* = $\frac{tf_{ij}*(K_1+1)}{K_1*((1-b)+(b*ndl))+tf_{ij}} * log\frac{Ne-df_i+0.5}{df_i+0.5}$ où *Ne* est le nombre de nœuds feuilles dans le document ;
- *F8* = *F7* dont $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dct}{\Delta ct}$
- *F9* = *F7* dont $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel}$
- *F10* = *F7* dont $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel} + h_3 * \frac{dct}{\Delta ct}$ Où : *h1*, *h2* et *h3* sont des constantes dont les valeurs sont : *h1*=0.3, *h2*=0.5 et *h3*=0.2

Et la dernière formule intégrée, consiste en une normalisation avec des paramètres pondérés de la deuxième formule (*F2*) utilisée par Karen Sauvagnat dans le système XFIRM :

- $F11 = tf * ief * \frac{1}{h1+h2*\frac{det}{\Delta et}+h3*\frac{dct}{\Delta ct}+h4*\frac{dcel}{\Delta cel}}$ **[17]** Où :

  h1, *h2*, *h3* et *h4* sont des constantes dont les valeurs sont de *0.3, 0.2, 0.2* et *0.7*
  tf est la fréquence du terme associé dans l'élément en question.
  ief est la fréquence inverse de l'élément en question

  Avec :
  dct et Δ*ct* sont respectivement la taille et la taille moyenne d'un document (en nombre de termes)
  dcel et Δ*cel* sont respectivement la taille et la taille moyenne d'un document (en nombre d'éléments)
  Et :

$$\Delta et = \frac{\sum det}{Nf} = \frac{nombre\ de\ termes\ dans\ tous\ les\ noeuds\ feuilles}{nombre\ de\ noeuds\ feuilles\ dans\ la\ collection}$$

$$\Delta ct = \frac{\sum dct}{Nd} = \frac{nombre\ de\ termes\ dans\ tous\ les\ documents\ de\ la\ collection}{nombre\ de\ documents\ dans\ la\ collection}$$

$$\Delta cel = \frac{\sum dcel}{Nd} = \frac{nombre\ d'éléments\ dans\ toute\ la\ collection}{nombre\ de\ documents\ dans\ la\ collection}$$

- La première formule (*F6*) tient compte du nombre d'éléments dans toute la collection et un facteur de normalisation par la taille d'un élément pour calculer le poids d'un terme qu'il contient : $ndl = \frac{det}{\Delta et}$

Les normalisations dans les autres formules concernant cette formule se sont focalisées sur le paramètre *ndl* et *N* :

- La deuxième formule (*F7*) consiste en la normalisation par le nombre d'éléments (*Ne*) dans le document au lieu de se focaliser sur toute la collection : *N* := *Ne*
- La troisième formule (*F8*) consiste à ajouter au *ndl* de la *F7* un paramètre de normalisation par le ratio entre la taille *dct* d'un document et sa taille moyenne Δct (par le nombre de termes) : $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dct}{\Delta ct}$
- La quatrième formule (*F9*) consiste en la normalisation par l'ajout du ratio entre la taille $dcel$ d'un document et la taille moyenne Δcel de celui-ci (par le nombre d'élément) dans la formule *F7* : $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel}$
- La cinquième formule (F10) consiste en la normalisation par les deux facteurs précédents du paramètre *ndl* dans F7 :$ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel} + h_3 * \frac{dct}{\Delta ct}$
- La sixième formule (*F11*) consiste en une normalisation de la (*F5*) par la taille du document en nombre de termes et sa taille en nombre d'éléments avec pondération des paramètres d'arrangement : $\frac{1}{h1+h2*\frac{det}{\Delta et}+h3*\frac{dct}{\Delta ct}+h4*\frac{dcel}{\Delta cel}}$

IV.6.1. Calcul du score des nœuds :

La requête utilisateur est représentée par des mots clés éventuellement pondérés.
On peut avoir la représentation suivante de la requête :

$$q = \{(t_1, p_{q1}), \ldots\ldots, (t_M, p_{qM})\}$$

Avec *tk* un terme de la requête et *pqk* le poids de *tk* dans la requête *q* et *M* le nombre de termes dans la requête.
Le score d'un nœud *ni* identifié dans l'arbre, est calculé avec une fonction de similarité *RSV(q,ni)* du modèle vectoriel (Inner product) comme suit :

$$RSV(q, n_i) = \sum_{k=1}^{M} p_{qk} \times p_{nik}$$

Ou *pqk* et *pnik* sont respectivement le poids du terme *k* dans la requête *q* et le nœud *ni*.

Le poids *Pk* d'un terme *k* dans les nœuds feuilles et dans la requête est calculé grâce à une formule de pondération.

IV.7 Propagation des termes :

Afin d'identifier la partie du document qui répond le mieux à la requête utilisateur, nous allons implémenter une méthode de propagation des termes et des poids, en partant des nœuds feuilles jusqu'à la racine du document. **[17]**

La question qui en découle est : *quels termes propager et comment ?*

Dans ce but, l'auteur introduit une notion fondamentale, qui est *l'informativité d'un nœud*.

Un nœud est dit informatif s'il est porteur d'informations. S'il aisé de définir l'informativité, encore faut il la mesurer ! L'intuition guidant cette mesure étant la *taille* d'un nœud (c'est-à-dire le nombre de termes qu'il contient). En effet, un nœud qui ne contient *que* les termes de la requête, est s*pécifique* et *exhaustif* à cette requête.

Il est cependant, *non informatif* car il n'apporte pas l'information requise à l'utilisateur (un nœud *titre* par exemple peut être pertinent pour une requête mais pas informatif). Nous définissons, à cette fin, un *seuil* qui consiste en, le *nombre de termes minimal* qu'un nœud doit avoir pour être considéré comme informatif.

Deux cas dans la propagation des termes, sont à considérer : le premier, consiste à traiter les nœuds dont le nombre de termes est inférieur au seuil, et le second, consiste à prendre en compte les nœuds dont le nombre de termes est supérieur au seuil. Il est clair que nous n'avons aucun moyen théorique pour déterminer ce seuil. Dans notre cas, nous proposons de le fixer par expérimentation.

Les termes considérés sont les termes indexés, c'est-à-dire lemmatisés et sans mots vides.

A. Cas ou le nombre de termes des nœuds est inférieur à un seuil :

L'arbre du document est parcouru en commençant par les nœuds feuilles. Durant le parcours, lorsqu'un nœud visité a un nombre de termes inférieur à un seuil (à définir), ce nœud est supprimé de l'arbre et son contenu remonté vers son nœud parent. Ce procédé se fait de manière récursive jusqu'à atteindre (et éventuellement dépasser) le seuil, ou atteindre le nœud racine du document, ou atteindre un nœud interne, dont le nombre de termes d'au moins un de ses fils est supérieur ou égal au seuil.

L'algorithme illustratif est comme suit :

Algorithme de propagation cas A :

1. Commencer par les nœuds feuilles
2. Visiter un nœud de l'arbre
3. Si le nombre de termes du nœud est inférieur au seuil alors

3.1 remonter les termes vers le nœud parent avec leurs poids respectifs,

3.2 supprimer le nœud

4. reprendre les étapes à partir de 2 jusqu'à atteindre ou dépasser le seuil, ou atteindre le nœud racine ou atteindre un nœud interne dont le nombre de termes, d'au moins un de ses fils est supérieure ou égal au seuil.

B. Cas ou le nombre de termes des nœuds est supérieur au seuil :

L'idée sous-jacente est la suivante : *« des termes bien distribués dans les éléments enfants d'un élément peuvent être représentatifs pour cet élément »*.

Deux cas peuvent se présenter, un nœud peut avoir plusieurs nœuds fils, ou n'en posséder qu'un seul (seul un nœud feuille n'a pas de fils).

1. Cas ou un nœud possède plusieurs nœuds fils: Dans ce cas un terme d'un nœud peut être représentatif pour son parent, s'il existe au moins, dans un de ses frères. Cette intuition à elle seule ne suffit pas, car il faudrait tenir compte d'un facteur très important, qui est la pondération des termes dans les nœuds.

En effet, un terme peut appartenir à tous les nœuds fils d'un élément, mais si son poids est faible, par rapport à l'ensemble des termes des nœuds fils. Il ne pourra pas être discriminant pour ces nœuds. C'est dans cet ordre d'idée, que nous avons adjoins une autre intuition (en plus de la précédente) qui consiste à ne prendre en considération, que les termes dont le poids moyen au niveau des nœuds fils ou ils apparaissent, est compris entre, le poids moyen des termes des nœuds fils et leurs poids maximal.

2. Cas ou un nœud possède un seul nœud fils : Dans ce cas un terme d'un nœud fils, ne peut être discriminant pour son parent, que si son poids (du terme), est compris entre le poids moyen des termes du nœud fils et leurs poids maximal.

Le terme vérifiant les intuitions de l'un des cas, 1 ou 2, sera supprimé de son nœud d'origine et remonté vers son nœud parent considéré. Son poids dans le nœud parent, est égal à son poids moyen au niveau de tous les nœuds fils, dans le cas 1, ou à son poids dans le nœud fils considéré, dans le cas 2.

Ceci est formalisé comme suit :

1. Soit *e* un nœud possédant *plusieurs* nœuds fils *e'*. Soit *t* un terme d'un nœud fils *e'* de *e*. *P(t,e')* le poids du terme *t* dans le nœud *e'*, calculé avec la formule III.1. *t* peut être remonté vers *e*, si *t* existe dans au moins un nœud frère de *e'* et si la moyenne du poids de *t* dans les nœuds fils de *e* ou il apparaît, vérifie la condition suivante :

$$P_{moy} \leq \underset{e' \in enf(e)}{moy}(P(t,e')) \leq P_{max} \qquad (B.1)$$

Avec :

$$P_{moy} = \frac{\sum_{e' \in enf(e)} \sum_{i=1}^{Nte'} p(t_i, e')}{Nt} \qquad (B.2)$$

P_{moy} : le poids moyen des termes dans les nœuds *e'* fils de *e*

$$\underset{e' \in enf(e)}{moy}\left(P(t,e')\right) = \frac{\sum_{e' \in enf(e)/t \in e'} P(t,e')}{Ne'} \qquad (B.3)$$

Nte' : nombre de termes dans le nœud *e'*

Nt : nombre de termes dans tous les nœuds *e'* enfants de *e*

Ne' : nombre de nœuds *e'* contenant le terme *t*

Pmax : le poids maximum des termes dans tous les noeuds *e'* enfants de *e*

Le terme *t* sera supprimé des nœuds fils *e'*, et remonté vers le nœud père *e*, et son poids dans *e* sera :

$$P(t,e)=\underset{e'\in enf(e)}{moy}(P(t,e')) \qquad (B.4).$$

2. Soit *e* un nœud possédant *un seul* nœud fils *e'*. Soit *t* un terme du nœud *e'*. *P(t,e')* le poids du terme *t* dans le nœud *e'*. *t* peut être remonté vers *e*, s'il vérifie la condition (B.5) suivante:

Avec : $$\mathbf{P_{moy} \leq P(t,e') \leq P_{max}} \qquad (B.5)$$

$$P_{moy} : P_{moy} = \frac{\sum_{i=1}^{Nte'} P(t_i,e')}{Nte'} \qquad (B.6)$$

P_{moy} : le poids moyen des termes dans *e'*

P_{max} : le poids maximum des termes dans *e'*

Nte' : nombre de termes dans le nœud *e'*

Le terme *t* sera supprimé du nœud fils *e'* et remonté vers le nœud père *e* en conservant son poids, donc :

$$P(t,e)= P(t, e') \qquad (B.7).$$

Indiquons que, durant la remontée d'un terme *t* du nœud fils *e'* vers son parent *e*, celui-ci peut s'y trouver déjà. Dans ce cas, le terme *t* sera supprimé du (ou des) nœud(s) fils *e'*, et son poids dans le nœud *e* sera égal, à la moyenne de son poids, dans le(s) nœud(s) fils *e'* et le nœud parent *e*, c'est-à-dire

$$P(t,e) = \frac{P_{B.4/B.7}(t,e) + P_0(t,e)}{2} \qquad (B.8)$$

avec :

$P_0(t,e)$: poids initial du terme *t* dans le nœud *e*

$P_{B.4/B.7}(t,e)$: poids que devait avoir (s'il n'y existait pas) le terme *t* dans *e* calculé avec la formule (B.4) ou (B.7) selon le cas considéré.

Le processus de propagation des termes, se déroule de manière récursive des feuilles de l'arbre jusqu'à la racine. Nous résumons ces différents cas dans l'algorithme suivant :

Algorithme propagation cas B :

1. Commencer par les feuilles
2. visiter un nœud
3. lire un terme
4. si le nœud possède des frères alors

4.1 si le terme existe dans au moins un nœud frère alors

vérifier la formule (B.1)

4. sinon vérifier la formule (B.5)
5. si la formule (B.1 ou B.5) vraie alors supprimer le terme du (des) nœud(s) fils et le remonter vers son parent

5.1 si le terme existe dans le nœud parent alors le poids du terme se calcule avec la formule (B.8)

5.1 sinon si la formule (B.1) vraie alors le poids du terme se calcule avec (B.4) sinon avec (B.7)

6. reprendre à partir de l'étape 3 jusqu'à lire tous les termes du nœud considéré
7. reprendre à partir de l'étape 1 jusqu'à atteindre le nœud racine du document

Ce traitement peut s'effectuer de manière *dynamique* pendant le traitement de la requête, ou de manière *statique* indépendamment de la requête, dans ce dernier cas, ce traitement doit être appliqué à tous les termes du fichier inverse.

A l'issue de ce traitement, le score de similarité des termes de la requête avec les nœuds représentés par ces termes sera calculé, les résultats seront présentés par ordre décroissant des scores. Les fragments (sous arbres) pertinents et informatifs seront ainsi présentés à l'utilisateur.

IV.7.1. Exemple de traitement de requête :

Nous déroulons dans ce qui suit, un exemple de la méthode de propagation des termes. Sur le document précédant extrait de l'article de *Ronald Bourret* sur *"XML et les bases de données"*.

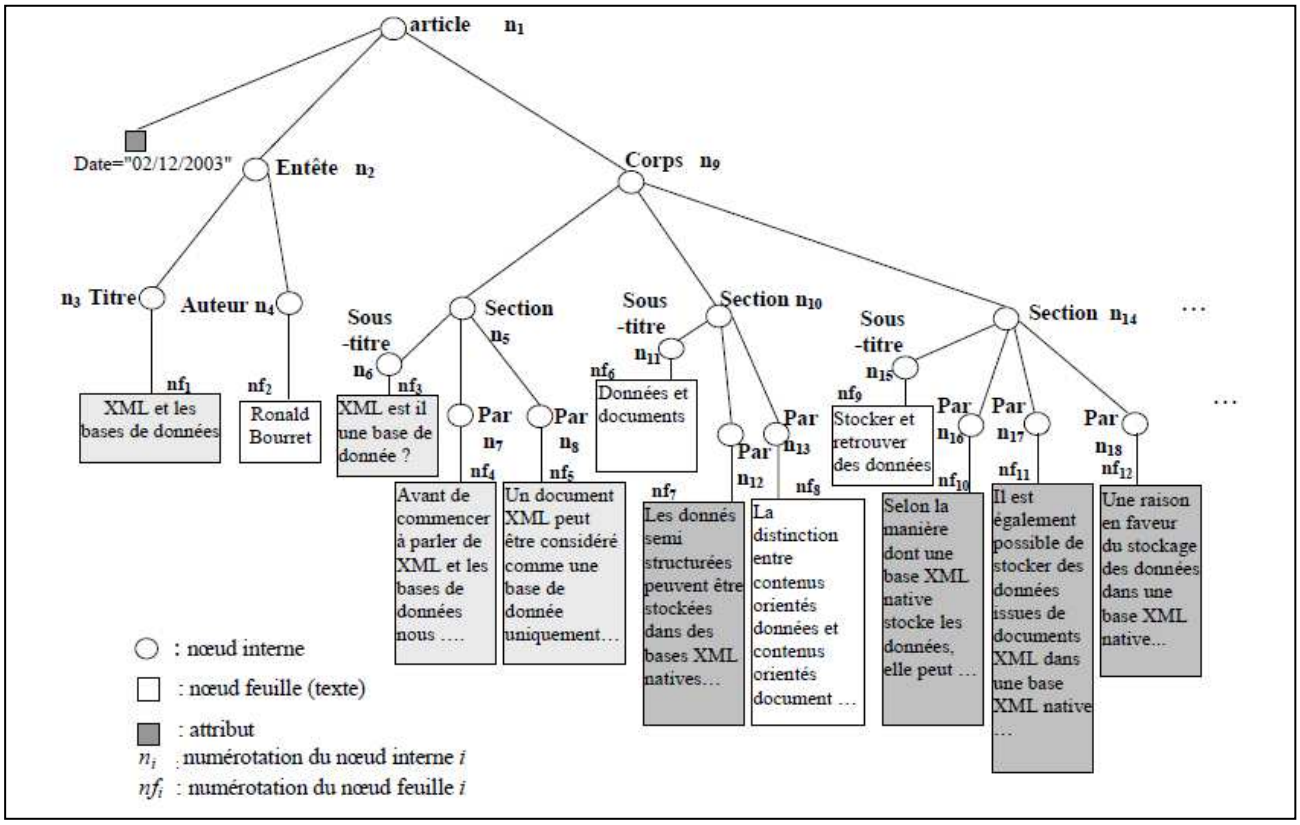


Figure IV.3 : ***Représentation en arbre du document article.xml***

Soit la requête suivante *"base XML native"*, composée de trois termes *base, XML* et *native.* Les nœuds feuilles contenant les termes de la requête sont : *nf1, nf3, nf4, nf5,* ***nf7,nf10, nf11, nf12,*** les quatre derniers nœuds cités contiennent *tous* les termes de la requête.

On a commencé par le calcul du poids des termes dans les nœuds feuilles. Ce calcul étant fastidieux, nous ne présentons que les résultats obtenus par rapport aux termes de la requête. Rappelons que, la formule de pondération utilisée est la suivante :

P(t,n)= tf*idf *ief (voir chapitre I pour plus détails)

En considérant,

N= nombre de documents de la collection = 10,

n =nombre de documents contenant les termes de la requête =3, les poids des termes de la requête dans les nœuds feuilles sont :

	nf_1	nf_3	nf_4	nf_5	nf_7	nf_{10}	nf_{11}	nf_{12}
XML	0,48	0,48	0,96	1,46	2,44	1,95	1,95	1,92
Base	0,48	0,48	0,96	0,97	4,39	1,95	2,44	0.96
Native	0	0	0	0	3,96	2,64	2,61	1,31
Pmoy				**1,63**	**2,20**	**1,55**	**1,59**	
Pmax				**3,99**	**5,16**	**4,55**	**2,99**	

Tab IV.1 : poids des termes dans les nœuds feuilles

-*Pmoy* : est le poids moyen des termes dans le nœud feuille considéré calcul avec la formule (B.6)

- *Pmax* : est le poids maximal des termes dans le nœud feuille considéré

Signalons que, pour le calcul du poids moyen (*Pmoy),* et du poids max (*Pmax),* d'autres termes (présents dans les nœuds feuilles), autres que ceux de la requête ont été pris en considération.

- **Propagation des termes :**

Pour appliquer le processus de propagation des termes, un seuil a été utilisé, sur le nombre de termes minimal qu'un nœud doit avoir pour être gardé (non supprimé). Nous considérons pour l'exemple, un seuil de *20 termes*. Les termes considérés sont les termes lemmatisés, sans aucun mot vide.

Deux cas sont pris en compte dans le traitement de la requête: le premier, consiste à ôter les nœuds possédant un nombre de termes inférieur au seuil. Le second, consiste à traiter les nœuds dont le nombre de termes est supérieur au seuil.

- **Traitement des nœuds dont le nombre de termes est inférieur au seuil**

En considérant ce seuil, les nœuds *nf1* et *nf2* seront supprimés, et leurs termes ainsi que leurs poids, remontés vers les nœuds parents respectifs *n3* et *n4*. Comme nous procédons de manière récursive, et que les nœuds *n3* et *n4* ne posséderons que les termes pondérés ainsi remontés (puisqu'ils n'en avaient pas d'autres), ces derniers seront aussi supprimés puisque leurs nombre de termes est inférieur au seuil. Leurs termes respectifs seront remontés vers le

nœud parent *n2* dont le nombre de termes est aussi inférieur à *20*. Il sera à son tour supprimé et transmettra ses termes pondérés à son parent le nœud *n1*.

Les nœuds *nf3, nf6 et nf9*, seront aussi supprimés, en transmettant leurs termes pondérés, respectivement aux nœuds *n5, n10 et n14*. Le processus de suppression s'arrêtera, car les nœuds ainsi atteints, possèdent des fils pourvus d'un nombre de termes supérieur au seuil.

Les nœuds nf_4 et nf_{12} disposent aussi d'un nombre de termes inférieur au seuil. Ils seront de ce fait, supprimés et leurs termes pondérés, remontés vers leurs parents respectifs n_7 et n_{18}. Le seuil n'étant pas atteint, les nœuds n_7 et n_{18} seront à leur tour supprimés. Ils transmettront leurs termes à leurs parents respectifs n_5 et n_{14}. Le processus d'élagage de l'arbre s'arrête, puisque les nœuds ainsi atteints, possèdent des fils dont le nombre de termes dépasse le seuil.

Lorsque, le nœud *n7* transmet son contenu au nœud *n5*, les termes *XML* et *base* s'y trouve déjà (ainsi que d'autres termes, mais dans le cas présent nous ne considérons que les termes de la requête). Leurs poids dans le nœud *n5* seront donc recalculés avec la formule (B.8). *XML* et *Base* auront donc un poids de *0,72* chacun dans *n5*.

Après suppression des nœuds, dont le nombre de termes est inférieur au seuil, l'arbre devient :

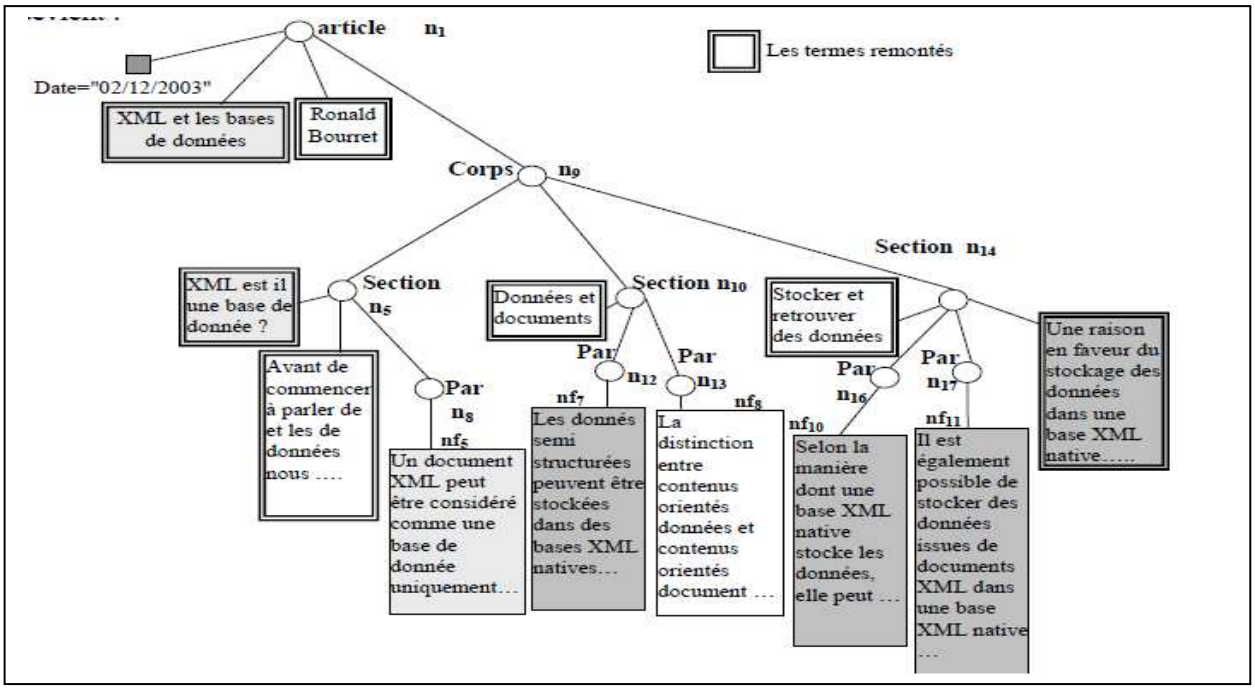


Figure IV.4 : ***le document article.xml après suppression des nœuds dont le nombre de termes est inférieur au seuil***

- **Traitement des nœuds dont le nombre de termes est supérieur au seuil**

Les nœuds à considérer dans ce cas sont : *nf5, nf7, nf10 et nf11*. Comme ils ne possèdent pas de frères; nous examinons si la condition (B.5) est vérifiée pour les termes de la requête dans ces nœuds.

	nf_5	nf_7	nf_{10}	nf_{11}
XML	1,46	2,44	1,95	1,95
Base	0,97	4,39	1,95	2,44
Native	0	3,96	2,64	2,61
Pmoy	**1,63**	**2,20**	**1,55**	**1,59**
Pmax	**3,99**	**5,16**	**4,55**	**2,99**
$P_{moy} \leq P(XML,nf) \leq P_{max}$	OK	OK	OK	OK
$P_{moy} \leq P(Base,nf) \leq P_{max}$	OK	OK	OK	OK
$P_{moy} \leq P(Native,nf) \leq P_{max}$	OK	OK	OK	OK

Tableau IV.2: Résultats de la condition (B.5) dans les nœuds nf5, nf7, nf10 et nf11

- *Pmoy* : est le poids moyen des termes dans le nœud feuille considéré calculé avec la formule (B.6)

- *Pmax* : est le poids maximal des termes dans le nœud feuille considéré

- *P(Xml,nf), P(Base, nf), P(native,nf)* : est le poids respectif des termes *XML,base* et *native* dans le nœud feuille considéré.

Les termes *XML*, *Base* et *native* seront supprimés des nœuds feuilles *nf7*, *nf10*, et *nf11* et remontés avec leurs poids, vers leur parents respectifs *n12, n16 et n17*. Ils demeureront par contre dans le nœud *nf5* puisqu'ils ne vérifient pas la condition.

L'arbre du document est transformé comme suit :

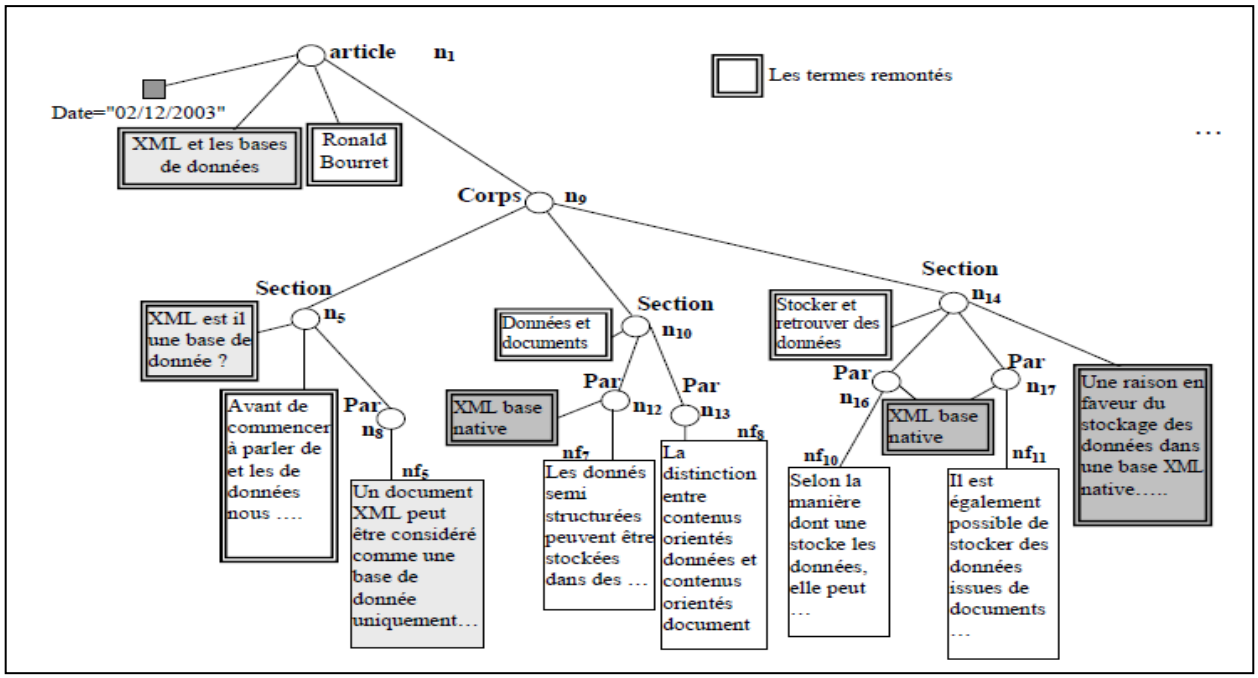


Figure IV.5: le document article.xml après remontée des termes XML, base et native de nf7, nf10 et nf11

Les termes *XML*, *base* et *native*, subsisteront dans le nœud *n12* puisqu'ils n'existent pas dans son frère, le nœud *n13.*

Les termes *XML*, *base* et *native* existent conjointement dans les nœuds *n16* et *n17,* nous allons voir s'ils vérifient la condition (B.1).

	n_{14}				
	n_{16}	n_{17}	Moy (P(t,e')) e'=n_{16}, n_{17}	Pmoy	Pmax
XML	1,95	1,95	1,95	1,90	4,55
Base	1,95	2,44	2,19		
Native	2,64	2,61	2,62		
$P_{moy} \leq$ Moy (P(XML,e'))$\leq P_{max}$	OK				
$P_{moy} \leq$ Moy (P(Base,e'))$\leq P_{max}$	OK				
$P_{moy} \leq$ Moy (P(native,e'))$\leq P_{max}$	OK				

Tableau IV.3: Résultats de la condition (B.1) dans les nœuds n16 et n17

- *Pmoy* : est le poids moyen des termes dans les nœuds *n16* et *n17* (formule (B.2)). Nous tenons à préciser que d'autres termes, autres que ceux de la requête, ont été pris en compte pour le calcul de ce poids.
- *Pmax* : est le poids maximal des termes des nœuds *n16* et *n17*
- *Moye' (P(t,e'))* : Poids moyen du terme *t* dans les nœuds *e'* (*n16* et *n17* dans ce cas)calculé avec la formule (B.3).

XML, *base* et *native* vérifient la condition (B.1), et peuvent ainsi remonter vers le nœud *n14.* Comme ils s'y trouvent déjà, ils ne seront donc pas réécrits, mais uniquement supprimés des nœuds *n16* et *n17.* Leurs poids dans le nœud *n14* seront calculés avec la formule (B.8). Ils sont de *1,93* Pour *XML, 1,57* Pour *Base* et *1,96* pour *native.*

L'arbre devient :

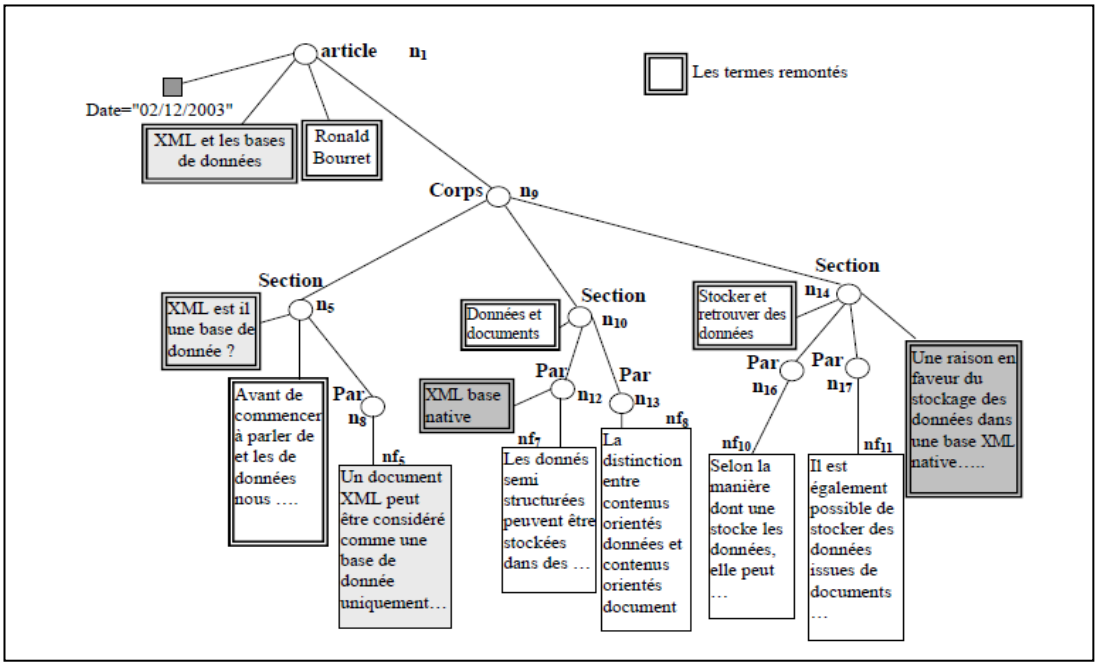


Figure IV.6: le document article.xml après suppression des termes XML, base et native de n16 et n17

Les termes *XML* et *base* existent à présent, dans les nœuds frères *n5* et *n14*, nous allons tester s'ils vérifient la condition (B.1) pour remonter dans le nœud *n9*.

	n_9				
	n_5	n_{14}	Moy (P(t,e')) $e'=n_5, n_{14}$	Pmoy	Pmax
XML	0,72	1,93	1,32	1,86	4,55
Base	0,72	1,57	1,14		
$P_{moy} \leq Moy(P(XML,e')) \leq P_{max}$	OK				
$P_{moy} \leq Moy(P(Base,e')) \leq P_{max}$	OK				

Tableau IV.4: Résultats de la condition (B.1) dans les nœuds n5 et n14

- *Pmoy* : est le poids moyen des termes dans les nœuds *n5* et *n14* (formule (B.2)).
- *Pmax* : est le poids maximal des termes des nœuds *n5* et *n14*
- *Moye' (P(t,e'))* : Poids moyen du terme *t* dans les nœuds *e'* (*n5* et *n14* dans ce cas) calculé avec la formule (B.3).

XML et *Base* ne vérifient pas la condition (B.1), ils ne vont donc pas remonter vers le nœud *n9*.

Le processus de propagation s'arrête puisqu'on a atteint la racine de l'arbre *n1*.
Nous allons calculer le score des termes de la requête avec les fragments ainsi repérés.

- **Le calcul de score**

A l'issue du processus de propagation, les termes de la requête sont représentatifs des sous arbres suivants :

- *XML* et *base* représentent, les sous arbre de racine n_5, n_{14} *et* n_{12}, l'arbre de racine n_1 et le nœud feuille nf_5.
- *Native* représente les sous arbres de racine n_{12} et n_{14}.

Nous allons calculer le score des termes de la requête avec les unités d'informations ainsi localisées.

$$RSV(q,n_i)=\sum_{k=1}^{M} p_{qk} \times p_{nik}$$

ou p_{qk} et p_{nik} sont respectivement le poids du terme *k* dans la requête *q* et le nœud n_i.

En appliquant la même formule de pondération pour les termes de la requête que celle des termes des nœuds feuilles, le poids de chacun des termes *XML*, *base* et *native* dans la requête est respectivement *0,48*, *0,48* et *1,32*. Le score de similarité de ces termes avec les nœuds considérés sera :

	n_5	n_{12}	n_{14}	n_1	nf_5
XML	0,72*0,48	2,44*0,48	1,93*0,48	0,48*0,48	1,46*0,48
Base	0,72*0,48	4,39*0,48	1,57*0,48	0,48*0,48	0,97*0,48
Native	0*1,32	3,96*1,32	2,62*1,32	0*1,32	0*1,32
RSV(q,n)	**0,69**	**8,50**	**5,13**	**0,46**	**1,16**

Tableau VI.5: Score de similarité requête nœuds du document

Les unités d'information pertinentes et informatives à retourner, suite au traitement de la requête *« base XML native »* sont par ordre décroissant des scores :

- le sous arbre de racine n12,
- le sous arbre de racine n14 (voir figure IV.7),
- le nœud feuille nf5,
- le sous arbre de racine n5,
- et enfin plus général l'arbre de racine n1.

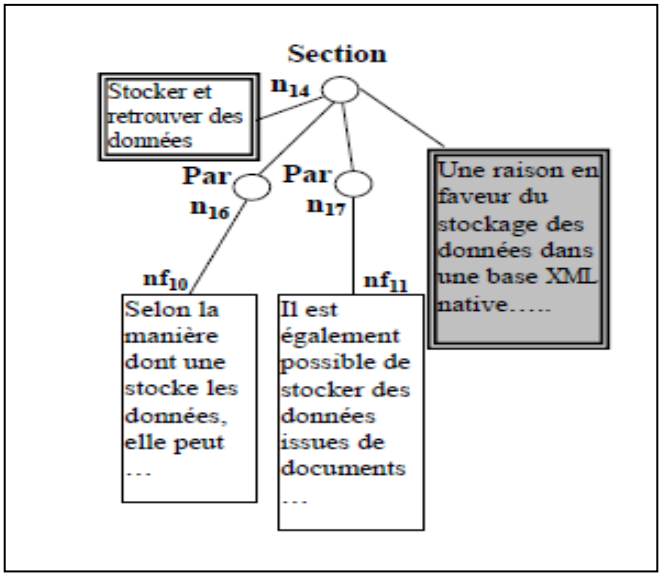


Figure IV.7 : Sous arbre de racine n14

Il est essentiel de noter que le traitement que nous venons d'effectuer peut se faire (est préférable d'être fait) de manière *statique* indépendamment de la requête. Ce traitement sera réalisé sur tous les termes du fichier inverse, les différents emplacements et poids des termes seront sauvegardés. A l'arrivée d'une requête, seuls les scores seront calculés. Précisons aussi, que le traitement statique permet un gain de temps considérable, et un temps de réponse du système à une requête très appréciable, ce qui constitue l'un des atouts d'un SRI.

IV.8. Conclusion

Dans ce chapitre nous avons présenté la méthode de propagation des termes que nous allons implémenter dans notre système de recherche d'information pour des documents XML, cette méthode se base sur deux contraintes à savoir l'informativité et la pertinence. Utilisée pour restituer l'unité d'information qui répond le mieux à une requête utilisateur.

Dans le chapitre suivant, nous allons présenter l'implémentation de cette méthode pour réaliser notre système de recherche d'information.

Chapitre 5

Implémentation

Et

Réalisation

V.1. Introduction :

Dans le chapitre précédent, nous avons présenté la méthode de propagation des termes ainsi que les formules de pondérations que nous allons utiliser dans cette dernière. Nous implémentons cette méthode en exploitant le module d'indexation du SRI XFIRM pour créer notre propre SRI qui prend en charge les documents semi structuré **XML.**

Dans ce chapitre, nous allons d'abord présenter l'environnement et les outils d'implémentation et quelques détails sur le mode de fonctionnement d'XFIRM, ensuite une description de notre modèle de SRI.

V.2. Environnement et outils d'implémentation :

Pour Implémenter notre méthode de propagation des termes et pouvoir la tester, nous allons utiliser un ensemble d'outils de développement dans un environnement adéquat.

Cette application a été réalisée dans une plateforme Unix (UBUNTU 10.10) montée sur un logiciel de machine virtuelle VMware Workstation, en utilisant les outils suivants : Xfirm comme système de recherche d'informations dont nous avons exploité le module d'indexation pour créer notre propre SRI de propagation des termes, Eclipse comme environnement de développement, Java comme langage de programmation, Oracle 10g XE comme SGBD pour la gestion de la BDD du système et les pilotes ODBC & JDBC pour la connexion à la BDD de notre système pour récupérer les données.

V.2.1. 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 hyperviseurs.

La version actuelle, VMware Workstation 9, est optimisée pour fonctionner avec les systèmes d'exploitation 64 bits et Windows 8 . Workstation utilise une interface Web pour connecter les utilisateurs de machines virtuelles locales et serveur hébergé depuis un PC, un smartphone ou une tablette. La figure suivante montre l'interface de la VMware Workstation7 :

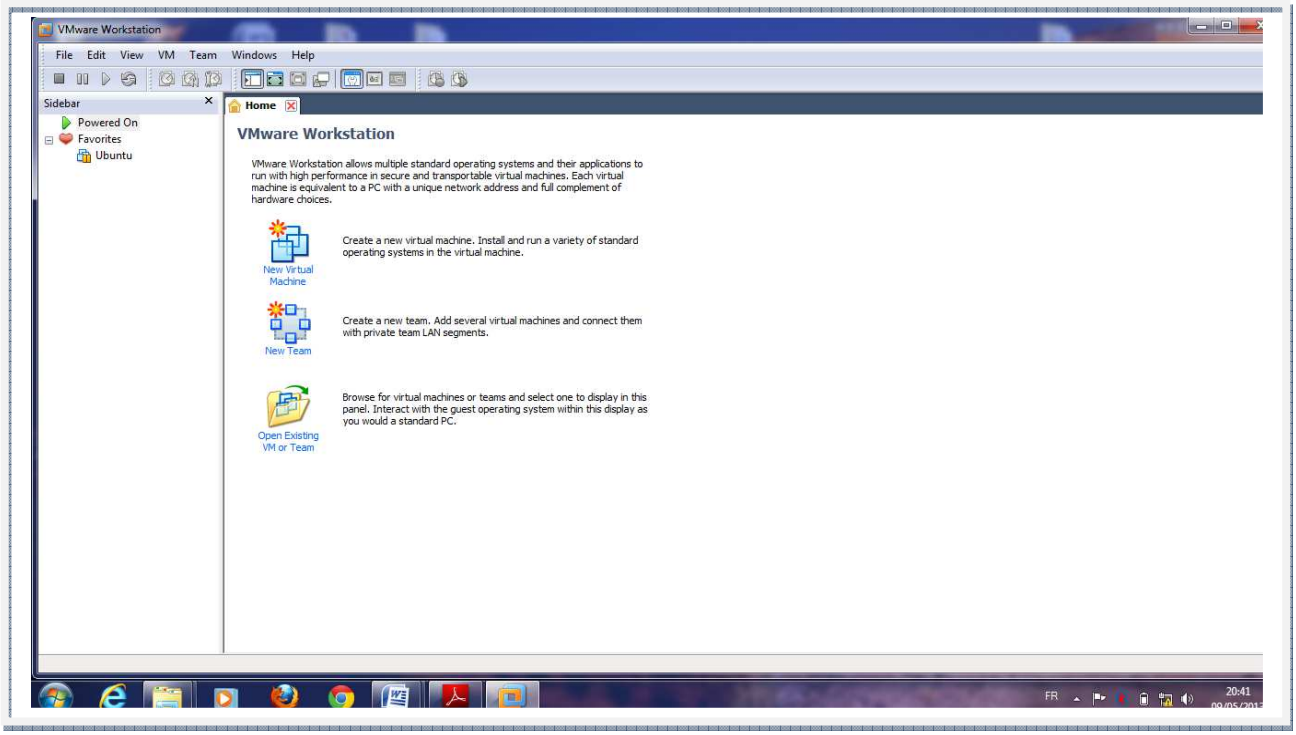


Figure V-1 : Interface de la VMware Worksation 7

V.2.2. L'environnement de développement Eclipse :

Est un environnement de développement intégré (IDE) développé par I.B.M. pour des applications telles que java et dans *http://www.eclipse.org* 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 plate-forme ouverte pour l'intégration de l'outil, pas un IDE. La question a été confondue car une force industrielle complète, y comprise la fonction IDE Java est fournie avec la plate-forme Eclipse, sous la forme de plug-ins qui étendent les installations du cadre de base de l'Eclipse. En outre, les plug-ins Eclipse peuvent s'étendre d'autres plug-ins. Quand une application basée sur Eclipse démarre, il découvre et active tous les plug-ins qui ont été configurés pour le poste de travail. La plate-forme Eclipse est littéralement la somme de ses parties, car il est capable d'effectuer n'importe quelle fonction qui a été ajoutée à elle par les plug-ins qu'il contient pour le moment.

Eclipse est dit universel et polyvalent, car il permet de créer des projets de développement, qui mettent en œuvre n'importe quel type de langage de programmation

(Java,C++, PHP, JavaScript, …). La figure suivante présente l'interface de l'IDE Eclipse sous linux :

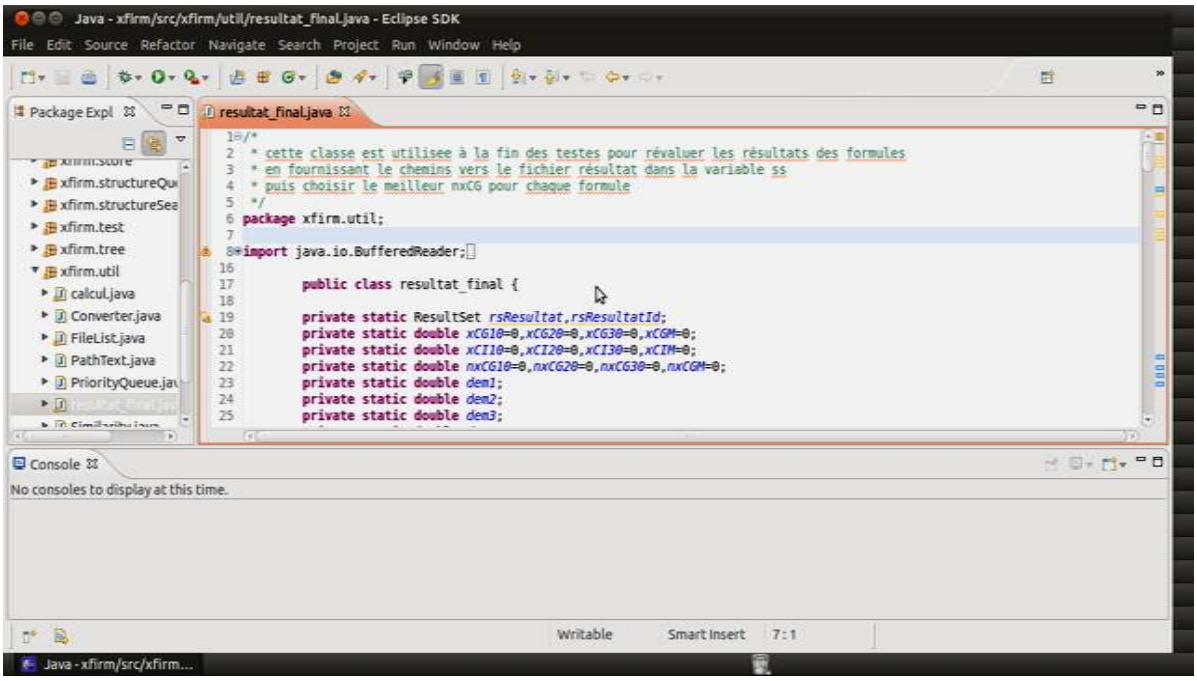


Figure V-2 : L'interface de l'IDE Eclipse

V.2.3. Le langage Java :

Le choix de java comme langage de programmation s'est imposé vue que le code source d'XFIRM est entièrement écrit en java.

On peut faire remonter la naissance de Java à 1991. À cette époque, des ingénieurs de chez **Sun Microsystems** ont cherché à concevoir un langage applicable à de petits appareils électriques (on parle de *code embarqué*). Pour ce faire, ils se sont fondés sur une syntaxe très proche de celle de C++, en reprenant le concept de machine virtuelle déjà exploité auparavant par le Pascal UCSD. L'idée consistait à traduire d'abord un programme source, non pas directement en langage machine, mais dans un pseudo langage universel, disposant des fonctionnalités communes à toutes les machines. Ce code intermédiaire, dont on dit qu'il est formé de *byte codes*, se trouve ainsi compact et portable sur n'importe quelle machine, il suffit simplement que cette dernière dispose d'un programme approprié (on parle alors de *machine virtuelle*) permettant de l'interpréter dans le langage de la

machine concernée. Il est dit aussi un langage multiplateforme selon la célèbre maxime «Write once, run everywhere ».

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

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

Et dans notre cas, on s'est basé sur la programmation procédurale par l'exploitation des différentes bibliothèques permettant l'implémentation des différentes phases du système.

Aujourd'hui, Java trouve une nouvelle niche dans la création d'applications RIA (Rich Internet Applications), des applications qui proposent des fonctionnalités, notamment des interfaces, plus évoluées à la fois sur Internet et sur les téléphones portables. Le langage JavaFx est un langage agile dérivé de Java, sous le contrôle de Sun Microsystems, qui met à profit la portabilité de Java ainsi que les vastes bibliothèques déjà disponibles dans le langage java.

V.2.4. Le SGBD Oracle10g EX :

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

V.2.4.1. Définition :

Oracle est un SGBD écrit en langage ℂ et édité par la société du même nom (*Oracle Corporation* http://www.oracle.com), 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 (RSI[2])* et commercialise un Système de Gestion de Bases de données relationnelles (SGBDR ou RDBMS pour *Relational Database Management System*) nommé *Oracle*.

En 1984 la première version d'Oracle (Oracle 4) 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.

V.2.4.2. 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)

V.2.4.3. 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. Et parmi ces outils on a utilisé l'SQL*PLUS qui est 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. SQL*PLUS est habituellement utilisé pour formuler une requête SQL et obtenir, sur un écran et de façon immédiate, le résultat attendu.

La figure suivante montre cette interface sur la plate forme linux :

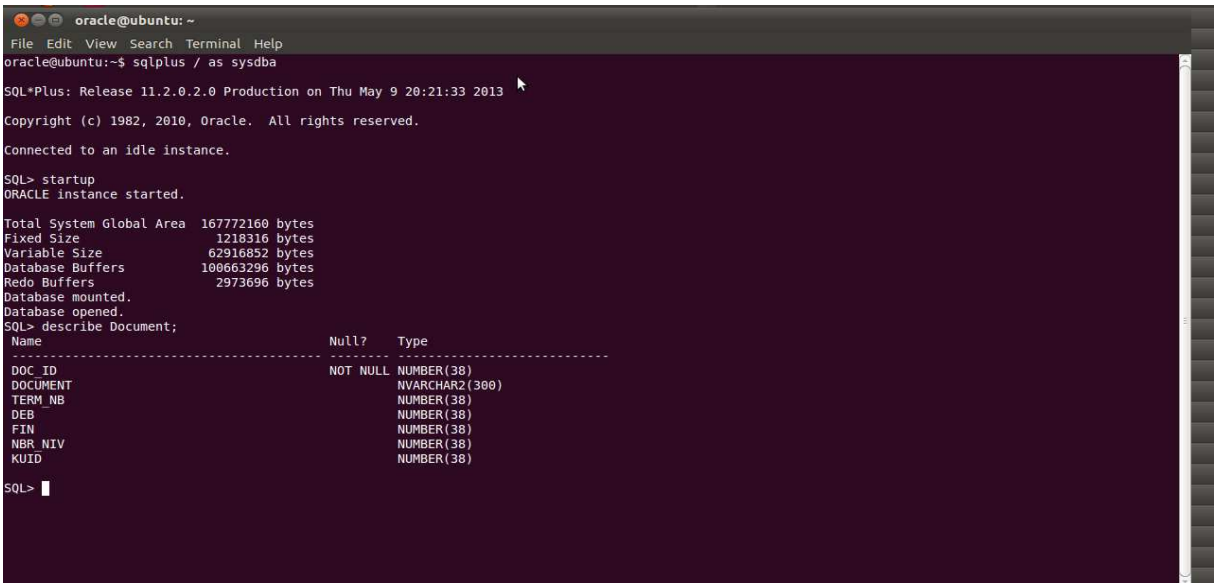


Figure V-3 : Illustration de l'interface SQL*PLUS sous linux.

V.2.4.3.1. Les outils de connexion au SGBD et à la BDD :

V.2.4.3.1.1. 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 Connection getConnection (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.

V.2.4.3.1.2. L'ODBC/JDBC :

- **Le JDBC :**

(Java Data Base Connectivity) 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 Database Connectivity est 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.

V.2.4.3.1.3. 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.

V.2.5. Le modèle XFIRM :

V.2.5. 1. Définition :

Le modèle XFIRM (XML Flexible Information Retrieval Model) a été proposé par K.Sauvgnat dans [12] pour la recherche d'information dans des documents semi-structurés, il se base sur une méthode de propagation de la pertinence. Il repose sur un modèle de

représentation des documents nous permettant de conserver à la fois toute l'information structurelle et toute l'information textuelle des documents.

V.2.5. 2. La plateforme XFIRM :

Le SRI XFIRM est un ensemble d'outils permettant de faire de la recherche d'information sur les documents semi-structuré en appliquant le modèle XFIRM.

V.2.5. 2. 1. Environnement de fonctionnement :

Le SRI XFIRM a été développée pour s'exécuter sur les OS Linux. La plateforme est écrite en langage java, ce qui implique que le système doit inclure un environnement d'exécution java ; comme la JVM.

XFIRM utilise une base de donnée à fin de stocker l'information structurelle et l'information de contenu des documents, ce qui lui permet de naviguer assez facilement dans la structure du document. La base de données que XFIRM utilise est la Oracle.

V.2.5.3. Schéma relationnel de stockage des index :

A fin de stocker l'information structurelle et l'information de contenu et de pouvoir la reconstituer en suite, les index sont stockés dans une base de données Oracle dont le schéma relationnel est le suivant :

TABLE	Rôle	Information sur les champs
Document (***doc_id*** , document, term_nb ,deb, fin nbr_niv, kuid)	contient les informations sur les documents de la collection	***Document :*** *URL du document* ***-kuid*** *=**K*** section 2.5 -***deb et fin*** : début et fin du numéro de pré-ordre des nœuds
Tag (***tag_id***,tag)	*contient des informations sur les balises de la collection*	
Path (***path_id***,***doc_id***,***tag_id***,ordre, nbrDirectFils, path_uid, nbrTermUniq, nbrTermsTot, nbrFilsText)	*contient des informations sur les nœuds (éléments) de la collection*	**Path_uid** = ***uid*** section 2.5 **nbrTermUniq**: nombre de terme unique. **nbrTermsTot** : nombre de terme. **nbrFilsText :** nombre de fils text

Termes (***term_id***, term ,freqColl, nbrDoc, nbrNoeuds, poids)	*contient des informations sur les termes de la collection*	Le champ **poids** est de type ***blob*** il permet de stoker les différent poids du terme de différent nœuds.
AttributsType : (***type_att_id***, att_name)	*contient des informations sur les noms d'attributs présents dans la collection*	
Attributs (***node_id,type_att_id***, att_val)	*permet de récupérer la valeur des attributs*	***node_id*** et ***type_att_id***sont deux clés étrangères, qui forme a eux deux la clé primaire
Dict : (***tag_id***, list_tag_id blob)	*permet d'établir des équivalences de balises dans la collection*	***tag_id*** est la clé primaire et aussi clé étrangère, elle référence la table Tag
Collection (nbrDoc, nbrNods, nbrNodText)	*contient des informations générales sur la collection*	
Images : (***nodeIm_id***,trmIm)	*contient des informations sur les images de la collection*	***nodeIm*** : le noeud a qui appartient l'image ***trmIm*** : le terme associé à l'image
Links (***path_id***,doc_name)	*contient des informations sur les liens de la collection*	
NodeToTerm (node_id, nbrTrmTot, listTerm)	*permet de connaître les termes présents dans un nœud donné cette table n'est utile que pour la reformulation des requêtes*	
termTemp (term_id,term,doc_id, node_id, pos);	*Cette table sera supprimée a la fin de l'indexation, elle permet de garder des informations sur tous les termes de la collection*	

Tableau V .1 : *Schéma de stockage des index XFIRM*

V.2.5.4. Les étapes d'indexation de XFIRM :

Le processus d'indexation de XFIRM passe sur certaines principales étapes avant de construire définitivement l'index de la collection. Les étapes se succèdent comme suit :

1. Création des tables pour stockage des index.

2. Traitement des documents de la collection, en remplissant les tables :

Document, Path, Link, Tag, Termtemp, Attributs, TypeAttribut, Collection, Image, NodeToTerm.

3. Remplissage de la table ***Terme*** à partir de la table ***TermTemp***, cette dernière sera supprimée a la fin de l'étape d'indexation.

4. Remplissage de la table ***Dict.***

5. Définition des contraintes des lignes des tables.

6. Suppression des tables temporaires.

V.3. Implémentation de la méthode de propagation :

Afin d'implémenter la méthode de propagation des termes, nous avons gardé le module d'indexation de XFIRM, et nous avons entièrement créé le module d'appariement document/ requête. Les différents changements que nous avons effectués sont des modifications :

1. Au niveau de la base de données.
2. Au niveau du processus d'indexation.

Le module **d'appariement requête/ document**, nous l'avons entièrement créé, le module ne prend en charge que les requête qui porte sur le contenu (**CO**).

V.3.1. Modification au niveau de la base de données :

Les modifications que nous avons effectué au niveau de la base de données sont l'ajout de deux nouvelles tables, que nous avons nommé ***trmpropag*** et ***trmpropag*****temp** leurs schéma relationnel est comme suit :

Tables	Rôles	Information sur les champs
***trmpropag* (*term_id*, term ,freqColl, nbrDoc, poids, doc_id,node_id);**	*contient des informations sur les termes de la collection, après avoir été propagé.*	Le champ **poids** est de type ***blob*** il permet de stoker les différent poids du terme dans différent nœuds
***trmpropag*temp (term_id, term, doc_id, node_id, weight);**	*Cette table sera supprimée a la fin de l'indexation, elle permet de garder des informations sur tous les termes de la collection*	Le champ weight est le poids du terme dans le document doc_id et le noeud node_id.

Tableau V.2 : Schéma relationnel des tables ajoutées

V.3.2. Changement au niveau du processus d'indexation :

Les changements au niveau de l'indexation sont l'ajout de quatre étapes au processus d'indexation, nous les avons ajoutées entre l'étape 2 et l'étape 4.

L'étape consiste en :

1. Création d'un fichier temporaire afin de contenir les informations sur les différents changements dans la structure du document l'hors du processus de propagation.
2. Remplissage de la table ***trmpropag*temp** à partir du fichier temporaire.
3. Remplissage de la table ***trmpropag*** à partir de la table temporaire ***trmpropag*temp**.
4. Suppression de la table **proptermtemp** et du fichier temporaire.

Les étapes d'indexation de notre système sont :

1. Création des tables pour stockage des index.
2. Traitement des documents de la collection, en remplissant les tables ***Document, Path, Link, Tag, Termtemp, Attributs, TypeAttribut, Collection, Image, NodeToTerm.*** (Les tables Attribut, TypeAttribut et NodeToTerm peuvent ne pas être pris en compte l'hors du processus d'indexation).
3. Remplissage de la table ***Terme*** à partir de la table ***TermTemp***, cette dernière sera supprimée a la fin de l'étape d'indexation.

4. Création d'un fichier temporaire afin de contenir les informations sur les différents changements dans la structure du document l'hors du processus de propagation.

5. Remplissage de la table ***trmpropag*temp** à partir du fichier temporaire.

6. Remplissage de la table ***trmpropag*** à partir de la table temporaire ***trmpropag*temp**.

7. Remplissage de la table ***Dict.***

8. Définition des contraintes des lignes des tables.

9. Suppression des tables temporaires et du fichier temporaire.

Pour se faire, nous avons ajouté des classes et apporté quelques modifications dans la classe BaseWriter.

A) Classes ajoutées

1. **SupNbrTerm :** traite les nœuds feuilles qui ont le nombre de terme supérieur au seuil fixé.

2. **InfNbrTerm :** traite les nœuds feuilles dont le nombre de terme est inférieur au seuil fixé.

Les deux classes sont instanciées par la class BaseWriter.

3. **CasFrere :** traite les noeuds qui ont des frères.

4. **CasNoFrere :** traite les noeuds qui n'ont pas de frère.

Les deux dernières classes sont instanciées dans la classe **SupNbrTerm.**

B) Modification de BaseWriter

La classe BaseWriter sera modifiée en lui ajoutant la méthode ***insertFinalNT (),*** cette méthode réalise l'étape 4 du nouveau processus d'indexation. On ajoute aussi une autre méthode qui est ***propageTerm(),***

V.3.3 Le module d'appariement Requête/ Document :

Afin de calculer la pertinence d'un élément vis-à-vis une requête, notre modèle de RIS dispose d'un module d'appariement qui est constitué de cinq classes, la classe Traitement, la classe SearchNode, la classe NodeDoc, la classe ScoreNode et la classe Order. Voici une description des rôles de chaque classe :

1) Traitement : cette classe lit et analyse la requête, celle-ci fait appelle à des classe qui se trouve dans la plateforme XFIRM ce sont : Analyser, Query et ScoreTerme. Le résultat de cette classe et l'identifiant du terme et son poids dans la requête.

2) SearchNode : cette classe prend en paramètre les termes de la requête, ensuite elle retrouve tous les nœuds qui possèdent les termes de la requête, sans calculer le poids des termes dans les nœuds ou le score de ces derniers. Le résultat de cette classe est l'ensemble des nœuds possédants les termes de la requête.

3) NodeDoc : cette classe prend en paramètre l'ensemble des nœuds et retourne un autre ensemble d'éléments regroupés dans leurs documents respectifs.

4) ScoreNode : cette classe prend le résultat de la classe NodeDoc et celui de la classe Traitement, elle retourne l'ensemble des éléments avec leur score respectifs.

5) Order : prend le résultat de ScoreNode et ordonne les éléments retourné selon leur score.

Récapitulatif :

Au total nous avons créé 9 classes et deux tables pour implémenter la propagation des termes.

V.4. Le modèle de RIS obtenu :

Notre SRI de propagation des termes est adapté aux requêtes de type CO et permet de retourner les sous arbres de tailles minimale qui répondent spécifiquement et exhaustivement aux requêtes utilisateurs, il ne prend pas en charge les requêtes de types CAS cela est du aux changements effectués dans la structure des documents lors du processus de propagations des termes.

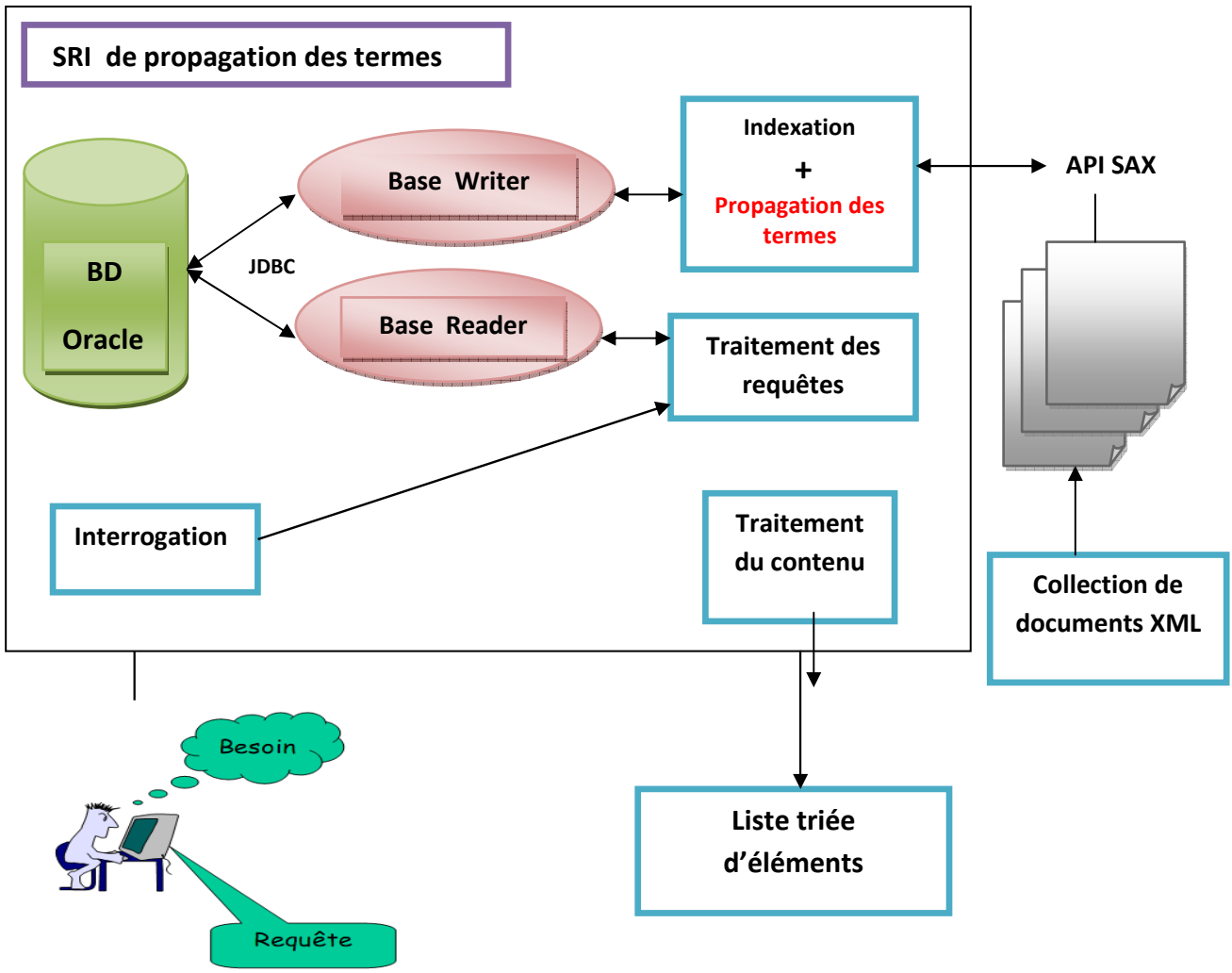


Figure V.6 : ***Schéma général du SRI de propagation des termes***

Synthèse :

Après l'implémentation, le SRI obtenu répond bien aux objectifs cités précédemment. Nous l'avons testé sur une collection de documents et le système exécute les différentes étapes citées précédemment sans inconvénients et nous avons aussi remarqué un gain considérable en temps d'exécutions des requêtes.

V.5.Conclusion :

Dans ce chapitre, nous avons présenté l'environnement de développement de notre système de recherche basé sur la propagation des termes, puis nous avons implémenté la méthode de propagation des termes en utilisant le module d'indexation de XFIRM et en lui créant un module de recherche propre a lui. Dans le chapitre suivant nous allons tester notre SRI pour voir s'il donne de meilleurs résultats en le comparant au SRI XFIRM.

Chapitre 6

Expérimentation

Des

Résultats

VI.1.Introduction :

Après avoir implémenté la méthode de propagation des termes dans le système XFIRM, nous avons effectué des tests sur ce dernier, afin d'avoir les résultats de l'exécution des formules de pondérations et les comparer avec ceux déjà obtenus par les formules intégrées dans XFIRM.

VI.2. Expérimentations des résultats:

Les expérimentations que nous avons menées ont été réalisées sur la collection INEX 2006. Dans le but d'analyser l'impacte du nombre de documents dans la collection sur les résultats obtenus, nous avons extrait plusieurs collections plus réduites en taille et gardant les documents concernés par les requêtes. Dans ce qui suit, nous commençons par présenter les collections de tests. Ensuite, nous décrivons le protocole d'évaluation utilisé lors des tests.

Enfin, nous présenterons les résultats de notre évaluation.

VI.2.1. Les requêtes :

Pour effectuer les tests des formules de pondération proposées dans notre système ou est implémenté la méthode de propagation des termes et pouvoir les comparer avec celles intégrées dans le système XFIRM, nous avons utilisé un échantillon de 32 requêtes CO (Content Only) extraites des 124 requêtes de la compagne INEX 2006 qu'on a pris aléatoirement de quatre sous-ensembles de ces requêtes.

Le tableau suivant montre l'ensemble des requêtes utilisées :

N° requête	Le titre de la requête
289	emperor "Napoleon I" Polish
290	"genetic algorithm"
292	Italian Flemish painting Renaissance -French –German
293	wifi security encryption
294	user interface design usability guidelines
295	software intellectual property patent license
296	"Borussia Dortmund" + European Championship Intercontinental Cup
297	"cool jazz" "West coast" musician
298	George Orwell life books essays 1984 Eric Arthur Blair Animal Farm
321	buildings designed Antoni Gaudi Barcelona architect
322	castles kasteel in the netherlands
323	founder ikea
324	composition of planet rings
325	"Cirque du Soleil" shows
326	Scotland tourism
327	cloning animals accepted "United States of America"
328	NBA European basketball player
361	"Europe after the second world war" + democracy
362	effect nuclear power plant accident
363	Bob Dylan Eric Clapton
364	+mushroom poisonous poisoning
365	economy peru international investment tourism
366	Fourier transform applications
367	true story films best director or movie award
393	Wireless devices "Health Hazards"
407	"Football World Cup" +"Miracle of Bern"
408	"electroconvulsive therapy" depression
409	Hybrid Vehicles -biology "fuel efficiency" "fuel sources" model engine
410	Routers and Switches +computer -travel -light network types history
411	+GSM, +CDMA, system,standard,clear battery coverage roaming price.
412	+Microsoft +Windows NT +Linux +SUSE +server +Network "Operating System" stability price security
413	Coordinates and Population of capital cities of Europe

Tableau VI-1 : Les 32 requêtes utilisées dans les tests des formules de pondération

VI.2.2. Collection de test :

Le tableau suivant montre les caractéristiques principales de la collection totale d'INEX 2006 :

La taille de la collection	4.6 Go
Le nombre de documents dans toute la collection	**659388 répartis en 22 ensembles**
Le nombre d'éléments dans toute la collection	**50.000.000**

Tableau VI-2 : Caractéristiques de la collection INEX 2006

Pour nos tests, nous avons utilisé un échantillon de 2200 documents différents, extraits de toute la collection (100 documents par sous-ensemble) :

La taille de la collection	17,1 Mo
Le nombre de documents dans la collection de test	**2200**
Le nombre d'éléments dans la collection de test	**421406**

Tableau VI-3 : Caractéristiques de la collection de test

VI.2.3.Les mesures d'évaluation :

Afin d'évaluer notre approche, nous avons utilisé les métriques d'évaluation de INEX 2006 (Voir chapitre II). Nous avons utilisé la classe ***Resultat_final*** permettant de calculer la valeur nxCG des scores :

$$nxCG[i] = \frac{xCG[i]}{xCI[i]}$$

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 nombres des premiers résultats considérés et la totalité des résultats.

Cette valeur est calculée pour le système XFIRM avec et sans la méthode de propagation pour permettre de faire une comparaison.

xCG représente la somme des scores des *i* premiers résultats obtenu par le système considéré à savoir XFIRM avec ces formules sans propagation et avec propagation des termes .

xCI représente la somme des *i* premiers scores obtenus pour les résultats de INEX 2006 (les scores idéales)

VI.2.4. Le calcul de score des différents tests :

Nous avons d'abord lancé l'indexation qui se fait en même temps que la propagation, qui donnera au final une collection indexée et des termes propagés.

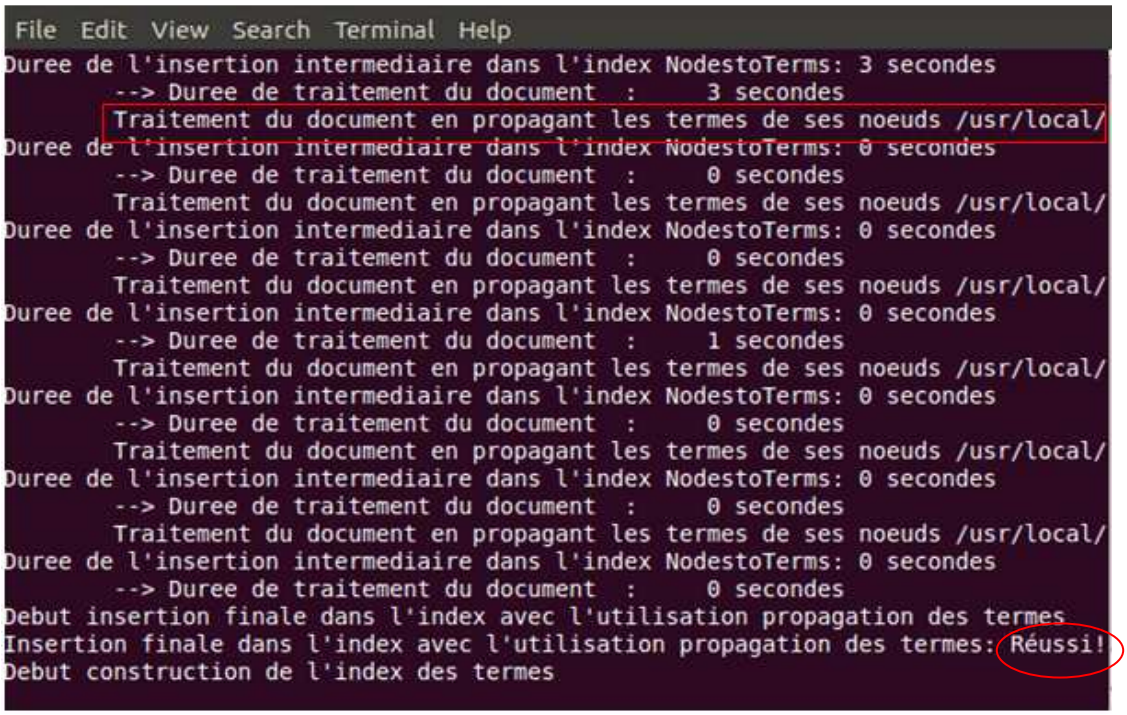


Figure VI-3 : Résultat de l'indéxation de la collection

Lancement d'une recherche :

Pour lancer une recherche l'utilisateur doit saisir la commande suivante :

```
>java  inex/TestInexRun config/ connect/OracleConfig.txt
example/queries/CO2004.2004.q  exemple/results/coZi/co  ''nom de la
collection''  ''le numero de la formule de ponderation''  CO
```

VI.2.4. 1. Protocole d'évaluation

Notre modèle est évalué et comparé sur 11 formules de 32 requêtes chacune avec les résultats de RIS de notre camarade de licence, nous nous somme basés sur la mesure de nxCG [i] pour déduire la différence entre ces dernières.

Résultats

Voici les résultats obtenus de toutes les requêtes évaluées, et la comparaison avec l'autre SRIS XFIRM. .

Voici les tableaux comparatifs des résultats des formules de pondération avec et sans propagation des termes :

Id_doc : identificateur du document.

Id_noeud : identificateur du nœud.

Score : score du terme dans le document.

Formule F1 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
30	5266	8.579908
75	14618	5.957853
1204	232052	4.232395
1311	250427	4.057469
270	56579	1.5619775
1858	361352	1.4859717
1258	240134	1.3688079
199	38681	1.0926802
144	24099	0.91223085
1115	216426	0.8402664
155	25664	0.67986035
499	95586	0.6529175
268	56379	0.48964694
1525	293586	0.42730775
176	31431	0.42354596
226	44166	0.42354596
1469	282260	0.41089743
1134	220529	0.39712277
1651	328334	0.38649365
667	126222	0.37230217

Formule F1 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
54	9670	6.7338924
1063	198013	5.608749
1063	197648	5.270141
1063	9665	5.1296487
54	194380	4.3916693
1063	194375	3.3454225
440	86404	3.316323
1251	239371	3.3031814
1103	214510	2.5158978
1103	214414	2.3565538
30	5266	1.7191781
1103	214417	2.1839314
440	86399	2.1628191
1251	239392	2.1573405
1103	214559	1.9980301
54	9680	1.8296759
1103	214579	1.8187681
1251	239462	1.7600595
54	9731	2.214339
1103	214494	1.6783338

Formule F2 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
30	5266	10.978679
75	14618	5.0143456
1311	250427	3.965136
1204	232052	3.5157042
270	56579	1.9986749
1858	361352	1.6171229
1258	240134	1.4903951
199	38681	1.3981714
155	25664	0.86993563
144	24099	0.8524642
1115	216426	0.84904045
499	95586	0.83546007
176	31431	0.54196084
226	44166	0.54196084
1134	220529	0.5081503
1525	293586	0.49110362
1469	282260	0.41260228
268	56379	0.40119228
667	126222	0.40067807
421	83653	0.3400619

Formule F2 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
1251	239332	5.979904
1063	198013	4.866941
1063	197648	4.5741644
1063	194380	4.5681453
54	9665	4.3984175
440	86404	3.2509425
1063	194375	2.9792242
1251	239371	2.977927
1103	214510	2.1753793
440	86399	2.12018
1103	214414	2.037602
1251	239392	1.9777863
30	5266	1.9146363
1103	214417	1.8883431
54	9680	1.8475833
1103	214559	1.727603
54	9731	1.6796834
1103	214579	1.5726035
1251	239462	1.540424
1103	214494	1.4864928

Formule F3 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
1103	214491	5.476862
1251	239332	5.388047
54	9665	4.393699
1063	198013	4.3404264
1063	194380	4.181974
1063	197648	4.0802884
440	86404	3.1766686
1251	239371	2.744001
1063	194375	2.727375
440	86399	2.0717409
1103	214510	1.9328666
54	9680	1.8501315
1251	239392	1.8478026
1103	214414	1.8104489
30	5266	1.701191
54	9731	1.6787398

1103	214417	1.6778295
1204	232052	1.5358523
1103	214559	1.5350088
1103	214579	1.3972888

Formule F3 (avec propagation)

Id_doc	Id_noeud	Score
54	9731	1.1912866
75	14623	1.0567756
54	9680	0.9094153
440	86399	0.9092268
1204	232134	0.88537896
1204	232169	0.71925175
268	56375	0.6415371
749	140922	0.6055854
1311	250427	0.6051873
1651	1651	0.57738334
1204	1204	0.56658185
54	54	0.56651855
1204	1204	0.564571
75	75	0.5201615
144	144	0.49109018
1204	1204	0.4737734
54	54	0.46862686
749	749	0.4606642
1204	1204	0.45107397
1204	1204	0.45107397

Formule F4 (XFIRM)

Id_doc	Id_noeud	Score
1103	214340	19.415333
1251	239332	15.262005
1063	194375	13.463289
1103	214335	12.662173
1103	214491	9.699186
1251	239371	7.1116915
1204	232052	5.2387238
1063	197660	4.7171087
54	9670	4.6392174
1251	239392	4.0737147
75	14618	3.9216237
1063	197657	3.9073281

1103	214414	3.8119216
1251	239439	3.7375996
75	14623	3.6759608
1063	197276	3.4926577
1204	232047	3.416559
1103	214510	3.4115696
1251	239462	3.3236547
1103	214579	3.2186444

Formule F4 (avec propagation)

Id_doc	Id_noeud	Score
1063	194380	1.7772818
54	9665	1.7444382
1204	232134	1.6083343
7	15104	1.159096
5	194375	1.2208807
1063	232169	1.1161476
1204	9685	1.0994679
54	328334	0.9639621
1651	232168	0.88391745
1204	14619	0.87315696
75	14633	0.87315696
75	14774	0.87315696
75	15120	0.87315696
75	15122	0.87315696
75	79127	0.8432001
397	15089	0.8158399
75	232260	0.79701865
1204	232137	0.77380943
1204	232139	0.78394634
75	14631	0.78394634

Formule F5 (XFIRM)

Id_doc	Id_noeud	Score
30	5266	22.497107
1204	232052	14.074618
1830	356747	9.197306
270	56579	4.8045397
1311	250427	4.1599746
199	38681	3.760474
1812	351918	3.015355
1858	361352	2.6344354
796	149495	2.5770395
1990	385346	2.0011587
1447	279077	1.8994627

845	162118	1.8741868
144	24099	1.6698602
226	44166	1.6585073
499	95586	1.5957545
1525	293586	1.4954765
1072	210519	1.4886106
1299	248926	1.3814368
1651	328334	1.3279462
1115	216426	1.3111373

Formule F5 (avec propagation)

Id_doc	Id_noeud	Score
54	9665	1.6099019
1204	232169	1.5313971
54	9680	1.4111078
75	15104	1.4090288
1651	328334	1.2953488
1204	232168	1.211397
440	86404	1.07239
1204	232260	1.0569344
1204	232137	1.0406575
1204	232139	1.0406575
54	9685	1.0040917
75	14619	0.9760641
75	14633	0.9760641
75	14774	0.9760641
75	15120	0.9760641
75	15122	0.9760641
1204	232149	0.9421366
1204	232252	0.8935044
75	15089	0.9046963
1204	232166	0.9267194

Formule F6 (XFIRM)

Id_doc	Id_noeud	Score
75	14618	20.067432
1830	356747	16.111975
1204	232052	10.125986
270	56579	7.9125385
199	38681	5.8138113
1812	351918	4.937683
1311	250427	4.231235
796	149495	4.0411897
1990	385346	3.8928685

1858	361352	3.3370233
845	162118	2.98168
226	44166	2.8961146
1447	279077	2.8492541
499	95586	2.4194162
1072	210519	2.3451
1299	248926	2.0996783
621	117180	2.0196433
1525	293586	1.8817434
144	24099	1.7202741
1861	362718	1.5089056

Formule F6 (avec propagation)

Id_doc	Id_noeud	Score
1251	239332	18.892391
1063	194375	18.395716
1103	214335	18.312826
1103	214491	13.627853
1251	239371	8.317092
1063	197660	6.638103
1103	214414	5.5144186
1063	197657	5.454519
1103	214510	4.025513
1103	214579	4.7586603
1103	239439	4.702437
1251	239392	4.577116
1063	197276	4.479143
1103	214559	4.4679923
1251	214417	4.1602774
1251	197157	4.159143
1063	196024	4.1525764
1063	239462	4.107965
54	9670	4.104553
1251	194402	4.8202243

Formule F7 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
54	9670	16.695236
75	14618	13.179829
1830	356747	7.8829503
1204	232052	5.697139
270	56579	3.5091906
1812	351918	3.0739322
796	149495	2.7220008
199	38681	2.6572747
1311	250427	2.4743938
1858	361352	2.413846
845	162118	1.7364724
499	95586	1.6258755
1990	385346	1.5604749
1447	279077	1.5516037
226	44166	1.5356824
621	117180	1.1070352
1115	216426	0.9718545
1525	293586	0.96324384
1299	248926	0.8764754
176	31431	0.8386957

Formule F7 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
1103	214335	7.3222656
1063	197660	6.7186584
1063	197657	5.5196257
1103	214491	5.4490085
1063	197276	4.521076
1063	197157	4.201076
1063	196024	4.2006893
1063	194402	4.071967
1063	197154	4.008844
1063	196021	4.009076
1063	196190	3.920915
1063	196083	3.920915
1063	197677	3.920915
1063	197693	3.920915
1063	197705	3.920915
1063	197761	3.920915
1063	197777	3.920915
1063	198014	3.920915
1063	198017	3.920915
1063	198021	3.920915

Formule F8 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
54	9670	6.11248
1103	214510	5.758706
1103	214559	5.257395
1103	214579	5.1915827
1103	214417	4.840534
1204	232052	4.650187
1103	214607	4.492631
1103	214356	4.421518
1103	214451	4.421518
1103	214415	4.376758
1103	214492	4.376758
1103	214576	4.376758
1103	214599	4.376758
1103	214604	4.3600783
1103	214646	4.3451343
1103	214645	4.27158
1103	214638	4.2274475
1103	214635	4.200968
75	14618	4.1581545
30	5266	4.1451054

Formule F8 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
1103	214491	19.941383
1063	194375	19.38965
1251	239337	8.654294
1103	214414	8.026573
1103	214510	7.0746202
1251	239332	7.0684996
1063	197660	6.875512
1103	214417	6.7555265
1103	214559	6.404991
30	5266	6.1562133
1103	214579	5.858325
1063	197657	5.6764793
1103	214607	5.235906
1103	214356	5.1725793
1103	214451	5.1725793
1103	214494	5.1717095

1103	214415	5.1327205
1103	214492	5.1327205
1103	214576	5.1327205
1103	214599	5.1327205

Formule F9 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
270	56579	8.3355465
1311	250427	6.660137
199	38681	6.1530075
1812	351918	4.077455
1447	279077	3.1522036
1990	385346	2.5143952
226	44166	2.510045
1299	248926	2.4112878
845	162118	2.3646865
1072	210519	2.1871347
144	24099	2.180045
1525	293586	1.9204767
176	31431	1.8569164
621	117180	1.6446759
1134	220529	1.4284067
667	126222	1.3870392
1469	282260	1.2257746
175	31195	1.0816525
1651	328334	1.0745983
687	129925	1.0636761

Formule F9 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
1103	214414	7.4532123
1103	214510	6.9278755
1103	214417	6.561927
54	9670	6.4949393
1251	239392	6.0536976
1103	214559	5.682174
1251	239439	5.2758727
1251	239462	5.1394844
1103	214579	5.131441
1251	239459	4.8366504

1103	214607	4.677956
1103	214494	4.674655
1103	214356	4.593029
1103	214451	4.593029
1103	214604	4.5796638
1103	214415	4.5781603
1103	214492	4.5781603
1103	214576	4.5781603
1103	214646	4.5658083
1103	214599	4.5781603

Formule F10 (XFIRM)		
Id_doc	**Id_noeud**	**Score**
1830	356747	13.783001
1204	232052	10.207474
270	56579	7.8478246
1311	250427	7.0545325
199	38681	5.792988
1812	351918	5.1033125
796	149495	4.088941
1858	361352	3.6129448
1447	279077	3.5615675
845	162118	2.9987295
499	95586	2.958348
226	44166	2.6892593
1990	385346	2.3672755
1299	248926	2.2702005
144	24099	2.2654648
176	31431	2.0716424
1072	210519	2.0591633
1525	293586	2.0349965
621	117180	1.9426883
1836	357577	1.7882929

Formule F10 (avec propagation)		
Id_doc	**Id_noeud**	**Score**
1103	9670	5.768486
54	214414	5.229312
1251	239439	5.2119584
1251	239392	5.146832
1251	239462	5.031146
1103	214510	4.89328
1251	239459	4.696494
1103	214559	4.4638

1251	239333	4.372459
1251	239338	4.372459
1251	239353	4.372459
1251	239362	4.372459
1251	239400	4.372459
1103	214579	4.35663
1251	239374	4.3486743
1251	239398	4.3486743
1251	239441	4.3486743
1251	239447	4.0459614
1103	214417	4.128685
1204	232052	4.3486743

Formule F11 (XFIRM)

Id_doc	Id_noeud	Score
199	38681	4.323601
75	14618	3.760245
1990	385346	2.7596946
1311	250427	2.500613
1072	210519	1.6936222
1651	328334	1.5702449
1299	248926	1.5201178
144	24099	1.1061387
397	79134	1.0
1483	283673	0.89035803
226	44166	0.8700271
1525	293586	0.8367304
2092	401328	0.8272236
41	7514	0.7479007
694	132765	0.7479007
1469	282260	0.7004478
690	131852	0.63116956
646	121518	0.59832054
667	126222	0.5894097
1134	220529	0.5048028

Formule F11 (avec propagation)

Id_doc	Id_noeud	Score
54	9665	0.60922
1204	232134	0.58181065
397	79124	0.52320004
440	86404	0.48558146
1651	328320	0.47531533
1204	232169	0.45079935
54	9680	0.4072874

75	14618	0.40639457
1651	328341	0.3606133
1204	232168	0.35516098
800	150232	0.34883288
75	14623	0.34619367
397	79076	0.3312
1204	232260	0.33114967
911	171927	0.32549104
440	86399	0.3166836
1651	328315	0.3099883
54	9731	0.3044207
2013	388149	0.3044207
487	94072	0.30647862

Tableau VI-4 : Illustration de vingt résultats consécutifs de la même requête avec les différentes formules ainsi la comparaison avec les résultats de notre système XPROPG.

VI.2.4.2. Evaluation des résultats :

Après la récupération des résultats des tests dans des fichiers résultats, les scores obtenus sont évalués en calculant le nxCG à 5 niveaux, tel que les montres le tableau suivant :

	FormuleF1(XFIRM)				
Requête	nxCG à 5	nxCG à 10	nxCG à 25	nxCG à 50	nxCG sur la totalité
289	0.3621045	0.1218828	0.0567855	0.0495381	0.0066874
290	-	-	-	-	-
292	0.8258805	0.3080352	0.1719771	0.1625821	0.0186942
293	0.3724965	0.1284815	0.0635317	0.0578528	0.0097742
294	0.7646156	0.2983244	0.1507398	0.1346997	0.0143648
295	0.4103860	0.1545866	0.0744257	0.0660276	0.0074086
296	0.4363120	0.1587046	0.0809986	0.0730953	0.0094714
297	0.4932019	0.1728044	0.0899073	0.0887320	0.0237743
298	0.4250383	0.1560680	0.0781126	0.0682658	0.0091111
321	0.5414870	0.1940157	0.9969680	0.0905297	0.0095567
322	0.6746190	0.2569784	0.1289420	0.1176420	0.0246724
323	0.3350510	0.1122590	0.0520860	0.4538727	0.0086676
324	0.3625262	0.1237141	0.0582966	0.0509431	0.0054042
325	0.6697184	0.2489669	0.1243478	0.1091207	0.0150042
326	0.3456339	0.1181565	0.0566951	0.0503355	0.0098895
327	0.7340081	0.2517986	0.1173278	0.1023165	0.0100262
328	0.3706305	0.1253363	0.0589084	0.0516254	0.0071566
361	0.4460065	0.1688091	0.1059077	0.1342621	0.0749726

362	0.5310962	0.1834039	0.0915784	0.0847555	0.0089095
363	0.3660586	0.1253794	0.0586279	0.0511471	0.0085911
364	1	0.6734006	0.6768189	0.6802721	0.6837608
365	0.4447564	0.1803063	0.1140463	0.1043128	0.0123808
366	0.5781974	0.2096360	0.1049128	0.0954021	0.0151412
367	0.4116472	0.1407725	0.0675888	0.0594850	0.0059945
393	0.3671029	0.1262425	0.0602877	0.0550794	0.0155848
407	-	-	-	-	-
408	0.6737162	0.2735221	0.2156053	0.3230620	0.1803992
409	0.5221977	0.1927663	0.0932880	0.0821714	0.0109848
410	0.5859378	0.2267698	0.1416703	0.1607386	0.0246026
411	-	-	-	-	-
412	-	-	-	-	-
413	0.6172031	0.2406015	0.1370272	0.1393560	0.0157340
La moyenne	**0.5240234**	**0.2026136**	**0.1189335**	**0.1174549**	**0.0445257**

Formule F1 (avec propagation)					
Requête	**nxCG à 5**	**nxCG à 10**	**nxCG à 25**	**nxCG à 50**	**nxCG sur la totalité**
289	**0.6857102**	**0.2720354**	**0.1895411**	**0.1464972**	**0.0136295**
290	-	-	-	-	-
292	0.8499310	0.4013192	0.2980096	0.2082601	0.0200101
293	0.6879149	0.2999708	0.1964400	0.1263257	0.0145125
294	0.8291675	0.4196394	0.3760690	0.2612296	0.0200060
295	0.8759939	0.4294789	0.3931872	0.2637954	0.0232213
296	0.6985499	0.3171734	0.2408701	0.1777999	0.0174795
297	0.8346962	0.4065048	0.2979748	0.2131781	0.0362130
298	0.6468221	0.2550657	0.1823135	0.1449494	0.0123432
321	0.8610127	0.4292136	0.3677372	0.2781472	0.0236944
322	0.7389679	0.3295300	0.2440650	0.1705714	0.0239832
323	0.7583365	0.3030495	0.2088961	0.1586470	0.0405418
324	0.8986374	0.4011407	0.2648624	0.1890691	0.0103671
325	0.7655997	0.3672011	0.2734316	0.1924845	0.0227520
326	0.7016653	0.2901118	0.2070891	0.1658405	0.0187012
327	0.8174676	0.3981387	0.2792658	0.1985104	0.0261101
328	0.8802709	0.4161644	0.2861430	0.1831080	0.0148392

361	0.6776251	0.3085548	0.2219250	0.1576235	0.0608926
362	0.7185541	0.3401106	0.2231092	0.1633196	0.0133527
363	0.6683132	0.2677687	0.1916888	0.1556324	0.0098893
364	0.9428009	0.4898062	0.3918215	0.3411416	0.3428911
365	0.7190256	0.3267839	0.2836280	0.2304811	0.0222144
366	0.8018451	0.3938905	0.3071003	0.2144201	0.0250102
367	0.8836666	0.4357659	0.3379651	0.2221088	0.0179797
393	0.7488909	0.2993799	0.2028838	0.1551890	0.0234356
407	-	-	-	-	-
408	0.8129752	0.3695209	0.2749730	0.2074003	0.0966404
409	0.7581292	0.3312073	0.2334889	0.1726926	0.0199016
410	0.8743074	0.4447297	0.3874397	0.3000273	0.0402883
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7054021	0.3208671	0.2514943	0.1896273	0.0213809
La moyenne	0.7800814	0.3594330	0.2719076	0.1995742	0.0368672

Formule F2 (XFIRM)					
289	0.3751069	0.1265321	0.0590079	0.3088115	0.0069610
290	-	-	-	-	-
292	0.8028231	0.3003549	0.1664630	0.0925507	0.0181026
293	0.3930419	0.1373867	0.0695770	0.0376450	0.0112837
294	0.7450016	0.2832349	0.1416002	0.0753184	0.0131472
295	0.4061633	0.1518447	0.0730608	0.0386208	0.0072596
296	0.4363120	0.1587046	0.0807495	0.4325556	0.0094340
297	0.4803179	0.1684881	0.8670056	0.0488462	0.0226792
298	0.4404199	0.1616634	0.8026202	0.0421858	0.0088463
321	0.6067417	0.2346753	0.1219663	0.0658874	0.0115716
322	0.6758247	0.2574811	0.1291879	0.0694777	0.0247212
323	0.3350510	0.1122590	0.0520860	0.0272282	0.0086676
324	0.3633573	0.1238703	0.0583409	0.0305723	0.0054102
325	0.6697184	0.2489669	0.1243478	0.0653860	0.0150042
326	0.3452564	0.1180163	0.0566125	0.0299131	0.0098680
327	0.7317597	0.2510567	0.1169782	0.0611937	0.0103321
328	0.3795165	0.1286896	0.0606393	0.0318790	0.0073786
361	0.4460065	0.1688091	0.1059077	0.7875698	0.0749726
362	0.5699012	0.1982138	0.0997564	0.5474454	0.0097309
363	0.3748843	0.1292331	0.0605536	0.0316832	0.0088838
364	1	0.6734006	0.6768189	0.6802721	0.6837606
365	0.4227687	0.1648970	0.0995371	0.0539724	0.0106257
366	0.4723759	0.1744818	0.0890599	0.0485370	0.0131224
367	0.4361902	0.1499469	0.0726035	0.0383469	0.0064162
393	0.3671029	0.1262425	0.0600149	0.0322302	0.0154755
407	-	-	-	-	-

408	0.6737162	0.2735221	0.2156053	0.1880407	0.1803992
409	0.5543178	0.2061655	0.1001424	0.0528487	0.0119059
410	0.5733233	0.2225437	0.1397099	0.0879829	0.0253399
411	-	-	-	-	-
412	-	-	-	-	-
413	0.6376641	0.2504299	0.1409266	0.0832949	0.0166148
La moyenne	**0.5257032**	**0.2036111**	**0.1192220**	**0.0793409**	**0.0445327**

Formule F2 (avec propagation)

Requête	nxCG à 5	nxCG à 10	nxCG à 25	nxCG à 50	nxCG sur la totalité
289	0.6857103	0.2727229	0.1901055	0.1467922	0.0137788
290	-	-	-	-	-
292	0.8504987	0.4037191	0.3015987	0.2105099	0.0202951
293	0.3764502	0.1340478	0.0686952	0.0384410	0.0030478
294	0.6182332	0.2300832	0.1361579	0.0815943	0.0045494
295	0.8964651	0.4430979	0.4077051	0.2722780	0.0243203
296	0.8346962	0.4065048	0.2979748	0.2131781	0.0377994
297	0.9092021	0.45532158	0.37863497	0.28216169	0.0238043
298	0.6453797	0.2546635	0.1821463	0.1448183	0.0123725
321	0.9092021	0.45532158	0.37863497	0.28216169	0.0238043
322	0.7426328	0.3315585	0.2457454	0.1716776	0.0241947
323	0.7583365	0.3030495	0.2088961	0.1586470	0.0405418
324	0.8973798	0.40042132	0.26502052	0.18913660	0.0104996
325	0.7655997	0.3672011	0.2734316	0.1924845	0.0227520
326	0.7016653	0.2901118	0.2070891	0.1658405	0.0189162
327	0.8137576	0.3951328	0.2771782	0.1974640	0.0259738
328	0.8903032	0.4136525	0.2793529	0.1781287	0.0142729
361	0.6776251	0.3085548	0.2219250	0.1576235	0.0608926
362	0.7181856	0.33989201	0.22302525	0.16327402	0.0134991
363	0.6682125	0.26771857	0.19156813	0.15549831	0.0099116
364	0.9428009	0.4898062	0.3918215	0.3411416	0.3428911
365	0.7304924	0.3409713	0.3043702	0.2498202	0.0239949
366	0.8011136	0.3934856	0.3066348	0.2141237	0.0249604
367	0.8815726	0.4345811	0.3395031	0.2242756	0.0185301
393	0.7488910	0.2993800	0.2028838	0.1551891	0.0236852
407	-	-	-	-	-
408	0.8129752	0.3695209	0.2749731	0.2074004	0.0966404
409	0.7592201	0.3322929	0.2336843	0.1728394	0.0199805
410	0.8934512	0.4511432	0.3923551	0.3034142	0.0405068
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7056222	0.3218775	0.2546516	0.1927631	0.0212780
La moyenne	0.7651794	0.3488459	0.2606428	0.1913696	0.0361204

Formule F3 (XFIRM)					
289	0.6857101	0.2697935	0.1874890	0.1453060	0.0132259
290	-	-	-	-	-
292	0.8448315	0.4013996	0.3024155	0.2108480	0.0203038
293	0.6891858	0.2956569	0.1886211	0.1160913	0.0121444
294	0.8639900	0.4387360	0.3999730	0.2794084	0.0224496
295	0.9011705	0.4489124	0.4175584	0.2780775	0.0250243
296	0.6985499	0.3171734	0.2412312	0.1782421	0.0176210
297	0.8615235	0.4441173	0.3804138	0.2734941	0.0572802
298	0.6446387	0.2546429	0.1821754	0.1448167	0.0125857
321	0.8935029	0.4490298	0.3612882	0.2642479	0.0219892
322	0.7421604	0.3312971	0.2455289	0.1715344	0.0241674
323	0.7583365	0.3030495	0.2088961	0.1586470	0.0405418
324	0.8979689	0.4007583	0.2649464	0.1891050	0.0104953
325	0.7655997	0.3672011	0.2734316	0.1924845	0.0227520
326	0.7016653	0.2901118	0.2070891	0.1658405	0.0189755
327	0.8145326	0.3957606	0.2776142	0.1976826	0.0255945
328	0.8746645	0.4021340	0.2648960	0.1685179	0.0132489
361	0.6776251	0.3085548	0.2219250	0.1576235	0.0608926
362	0.7193841	0.3391736	0.2221168	0.1628048	0.0133084
363	0.6679201	0.2675728	0.1912176	0.1551088	0.0094808
364	0.9428009	0.4898062	0.3918215	0.3411416	0.3428911
365	0.7590198	0.3662309	0.3352264	0.2784699	0.0264514
366	0.7949484	0.3900731	0.3027118	0.2116477	0.0245401
367	0.8775244	0.4276810	0.3287570	0.2157574	0.0176108
393	0.7488910	0.2993800	0.2028838	0.1551891	0.0239761
407	-	-	-	-	-
408	0.8129752	0.3695209	0.2749730	0.2074003	0.0966404
409	0.7639178	0.3354943	0.2354084	0.1737933	0.0202311
410	0.9066153	0.4555534	0.3957351	0.3057051	0.0406090
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7045907	0.3216009	0.2557382	0.1938879	0.0215655
La moyenne	**0.7862230**	**0.3635502**	**0.2772172**	**0.2033062**	**0.0377356**

Formule F3 (avec propagation)					
289	0.8463084	0.3584421	0.2613757	0.2060054	0.0162846
290	-	-	-	-	-
292	0.8372474	0.4035395	0.3144731	0.2184251	0.0209856
293	0.7849905	0.3344497	0.1924285	0.1104768	0.0096018
294	0.9675243	0.4802469	0.2530213	0.1464807	0.0177899
295	0.6680767	0.325268	0.284092	0.185124	0.0167170
296	0.7980418	0.3616154	0.2648627	0.1921173	0.0185123
297	0.8346962	0.4065048	0.3124115	0.2249264	0.0456588
298	0.6317922	0.2518710	0.1812337	0.1442046	0.0131615
321	0.7312453	0.3236247	0.2231660	0.1466558	0.0110461

322	0.7499735	0.3356217	0.2491112	0.1739725	0.02461835
323	0.7583365	0.3030495	0.2088961	0.1586470	0.0405418
324	0.8945983	0.3988517	0.2653700	0.1892857	0.0109502
325	0.7655997	0.3672011	0.2734316	0.1924845	0.0227520
326	0.7016653	0.2901118	0.2070891	0.1658405	0.0196059
327	0.8753595	0.4366018	0.3854854	0.2936300	0.0488571
328	0.7720465	0.3305940	0.1948077	0.1232734	0.0086704
361	0.6776251	0.3085548	0.1948077	0.1232734	0.0086704
362	0.6776251	0.3085548	0.2219250	0.1576235	0.0608926
363	0.7206545	0.335883	0.222761	0.163235	0.0138627
364	0.6673642	0.2672958	0.1905696	0.1543686	0.0090397
365	0.9428009	0.4898062	0.3918215	0.3411416	0.3428911
366	0.7864983	0.3853960	0.2973350	0.2083865	0.02396425
367	0.8190550	0.3710470	0.2639640	0.1996933	0.0191238
393	0.7488909	0.2993799	0.2028838	0.1551890	0.0245972
407	-	-	-	-	-
408	0.8129752	0.3695209	0.2749730	0.2074003	0.0966404
409	0.8007831	0.3621100	0.2526994	0.1828305	0.0210685
410	0.8019941	0.3976017	0.3293878	0.2518476	0.0313803
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8490876	0.3912812	0.2671401	0.1937811	0.0165084
La moyenne	0.7904571	0.3621441	0.2656813	0.1947163	0.0369804

Formule F4 (XFIRM)					
289	0.7143877	0.2985000	0.2250901	0.1780918	0.0187298
290	-	-	-	-	-
292	0.7119376	0.3260183	0.2410930	0.1662766	0.0180934
293	0.6883612	0.3032340	0.2057118	0.1301035	0.0139526
294	0.8028822	0.4189002	0.3437727	0.2223521	0.0175824
295	0.8803995	0.4539011	0.3852697	0.2618236	0.0256708
296	0.6924512	0.3196869	0.2531496	0.1890339	0.0204493
297	0.8097310	0.3655380	0.2675389	0.1972496	0.0352160
298	0.6374869	0.2546027	0.1827112	0.1465238	0.0117927
321	0.7824727	0.3680164	0.2542027	0.1687199	0.0124289
322	0.8654740	0.4065663	0.3073651	0.2206344	0.0370946
323	0.7617633	0.3054672	0.2105523	0.1595732	0.0405386
324	0.8839788	0.3923641	0.2613512	0.1878572	0.0107302
325	0.8043512	0.3918570	0.3228369	0.2290791	0.0271517
326	0.7175527	0.2974291	0.2202174	0.1784315	0.0242399
327	0.6654754	0.2787037	0.2036379	0.1579662	0.0193632
328	0.7092611	0.3039774	0.1979196	0.1354185	0.0110742
361	0.6707160	0.2822317	0.1790237	0.1250769	0.0439190
362	0.7625259	0.3835298	0.3160791	0.2299454	0.0240040
363	0.6694208	0.2765464	0.1942008	0.1588508	0.0118217
364	0.9312350	0.4494773	0.3395120	0.2943105	0.2958198

365	0.9221870	0.4564471	0.4375784	0.3527091	0.0406329
366	0.7767179	0.3499180	0.2641670	0.1840824	0.0182961
367	0.7481808	0.3241480	0.1722951	0.1083954	0.0090829
393	0.7737974	0.3121208	0.2081625	0.1567266	0.0247974
407	-	-	-	-	-
408	0.8839289	0.4344801	0.4167760	0.3415265	0.1415457
409	0.7029922	0.3206691	0.2222928	0.1655147	0.0207988
410	0.9537549	0.4673364	0.4112245	0.3051598	0.0392172
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8663089	0.4207095	0.3342810	0.2480301	0.0265907
La moyenne	**0.7782022**	**0.3554771**	**0.2706433**	**0.1999808**	**0.0371298**

Formule F4 (avec propagation)

289	0.6673025	0.2655943	0.1841606	0.1440081	0.0129634
290	-	-	-	-	-
292	0.7827638	0.3940975	0.3365988	0.2338006	0.02281239
293	0.6987072	0.2939789	0.1784045	0.0998042	0.0075289
294	0.8105606	0.41785939	0.34187466	0.25165472	0.0199267
295	0.9133436	0.4624909	0.3210333	0.2238481	0.0201183
296	0.6924511	0.3229960	0.2730933	0.2082505	0.0224439
297	0.8097310	0.3776410	0.3030495	0.2203905	0.0455690
298	0.6428880	0.2594439	0.1915022	0.1513797	0.0147958
321	0.7361164	0.3015474	0.1809944	0.1117534	0.0067396
322	0.8369954	0.3884209	0.2939974	0.2123653	0.0355935
323	0.7617631	0.3054671	0.2105522	0.1595731	0.0405385
324	0.8728281	0.3895482	0.2612209	0.1878619	0.0117734
325	0.8043509	0.3918569	0.3228368	0.2290790	0.0271517
326	0.7175528	0.2974292	0.2202174	0.1784315	0.0269427
327	0.7332779	0.3230092	0.2490600	0.1878037	0.0142720
328	0.7503035	0.3482587	0.2012736	0.1367581	0.0098058
361	0.6707160	0.2822317	0.1790237	0.1250769	0.0439190
362	0.8309415	0.4077506	0.3548339	0.2447560	0.0211265
363	0.6681141	0.2667052	0.1927312	0.1571283	0.0100797
364	0.9312350	0.4494773	0.3395120	0.2943105	0.2958198
365	0.9038562	0.4274475	0.3818872	0.3110488	0.0328807
366	0.7767179	0.3458291	0.2534013	0.1769310	0.0170984
367	0.9729892	0.4595950	0.3497002	0.2597361	0.0216026
393	0.7737945	0.3180940	0.2168972	0.1635394	0.0303017
407	-	-	-	-	-
408	0.8839289	0.4344801	0.4167761	0.3415266	0.1415457
409	0.6842132	0.2876685	0.2120060	0.1651938	0.0172838
410	0.7450641	0.3681466	0.2956934	0.2132842	0.0245726
411	-	-	-	-	-
412	-	-	-	-	-
413	0.9256266	0.4489704	0.3722699	0.2652796	0.0262915
La moyenne	0.7856476	0.3584298	0.2726643	0.2019491	0.0364821

Formule F5 (XFIRM)					
289	0.3783796	0.1278028	0.0596204	0.0312110	0.0071024
290	-	-	-	-	-
292	0.6560670	0.2463477	0.1396063	0.0775009	0.0158930
293	0.3932682	0.1362002	0.0673760	0.0367747	0.0103675
294	0.6950740	0.2504752	0.1259780	0.0672105	0.0101233
295	0.4755264	0.1836313	0.0912018	0.0485769	0.0092732
296	0.4695059	0.1747034	0.0890031	0.0478030	0.0107933
297	0.7728752	0.3205161	0.1739612	0.1066341	0.0502373
298	0.4519463	0.1656089	0.0816533	0.0427652	0.0099564
321	0.6742108	0.2547821	0.1299894	0.0698771	0.0114638
322	0.6793445	0.2619449	0.1313804	0.0724870	0.0279066
323	0.3350879	0.1122756	0.0520980	0.0272406	0.0088969
324	0.3634665	0.1232768	0.0578663	0.0303027	0.0062731
325	0.6779636	0.2545873	0.1279385	0.0674294	0.0144803
326	0.3456487	0.1187203	0.0574712	0.0307348	0.0106989
327	0.6572636	0.2235800	0.1043589	0.0546105	0.0106989
328	0.3739658	0.1274225	0.0602759	0.0317104	0.0054677
361	0.4006044	0.1561084	0.1069856	0.0709263	0.0712900
362	0.5782145	0.1999735	0.0996585	0.0545647	0.0095694
363	0.3695801	0.1279197	0.0599089	0.0313628	0.0088006
364	0.9045226	0.5165691	0.5191913	0.5218403	0.5245164
365	0.4376561	0.1709397	0.1029611	0.0580290	0.0104871
366	0.4988486	0.18851962	0.0967918	0.0526221	0.0143223
367	0.4858275	0.1692410	0.0829906	0.0439830	0.0068889
393	0.3594713	0.1230114	0.5975998	0.0327459	0.0162129
407	-	-	-	-	-
408	0.7590618	0.3963635	0.3881597	0.2434428	0.2352801
409	0.4984716	0.1854688	0.0893272	0.0470126	0.0101382
410	0.5511648	0.2302948	0.1430707	0.0883109	0.0238131
411	-	-	-	-	-
412	-	-	-	-	-
413	0.6885937	0.2675191	0.1550343	0.0921828	0.0149571
La moyenne	**0.5332722**	**0.2076358**	**0.1233432**	**0.0778533**	**0.0416185**
Formule F5 (avec propagation)					
289	0.6673029	0.2655944	0.1853951	0.1447470	0.0135402
290	-	-	-	-	-
292	0.7781034	0.3634822	0.3050066	0.2133083	0.0229243
293	0.6946860	0.2917551	0.1863998	0.1094680	0.0098208
294	0.7526804	0.3581113	0.2745398	0.2050684	0.0327665
295	0.8855701	0.4404488	0.3303002	0.2297381	0.0213139
296	0.6924512	0.3196869	0.2573524	0.1947205	0.0212060
297	0.8510407	0.4439121	0.3795873	0.2768684	0.0638961
298	0.6575337	0.2724717	0.1995100	0.1560252	0.0141335

321	0.7335415	0.3257232	0.2348370	0.1547621	0.0119096
322	0.8538178	0.3983042	0.3018937	0.2172240	0.0362771
323	0.7601620	0.3043120	0.2097584	0.1591274	0.0406153
324	0.8815656	0.3918369	0.2613317	0.1878111	0.0112113
325	0.8751680	0.4276877	0.3448009	0.2702338	0.0345058
326	0.7175526	0.2974291	0.2202173	0.1784314	0.0252399
327	0.7332779	0.3276881	0.2575798	0.2003708	0.0282185
328	0.7760527	0.3323597	0.2022707	0.1276467	0.0077127
361	0.6709963	0.2834700	0.1849149	0.1346221	0.0496199
362	0.7444268	0.3405761	0.2716581	0.1932511	0.0173274
363	0.6686782	0.2670683	0.1933012	0.1577724	0.0094406
364	0.9312350	0.4494773	0.3395120	0.2943105	0.2958198
365	0.8763651	0.4267047	0.3906972	0.3104710	0.0310259
366	0.7767180	0.3473884	0.2586523	0.1804192	0.0176755
367	0.8118644	0.3651254	0.2670758	0.2098739	0.0206673
393	0.7737947	0.3121208	0.2095207	0.1588383	0.0272406
407	-	-	-	-	-
408	0.8839290	0.4344801	0.4167761	0.3415266	0.1415457
409	0.7746190	0.3730032	0.2713916	0.1988859	0.0242396
410	0.8701794	0.4262408	0.3753922	0.2835652	0.0377490
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7130072	0.2852788	0.1997036	0.1597995	0.0117963
La moyenne	0.7787971	0.3525621	0.2689063	0.2017460	0.0385514
Formule F6 (XFIRM)					
289	0.4034804	0.1366272	0.0638003	0.0334215	0.0076209
290	-	-	-	-	-
292	0.5553641	0.2075794	0.1142727	0.0630040	0.0132526
293	0.4512729	0.1593086	0.0807912	0.0452479	0.0133432
294	0.7663563	0.2696424	0.1318036	0.0700941	0.0117694
295	0.4507000	0.1706691	0.0848534	0.0451567	0.0087027
296	0.4736426	0.1766325	0.8944413	0.0480430	0.0109411
297	0.6526765	0.2555353	0.1455937	0.0890970	0.0386187
298	0.4942141	0.1853467	0.0937933	0.0493014	0.0081142
321	0.5556042	0.1997708	0.0986985	0.0524976	0.0089906
322	0.6518468	0.2501214	0.1257253	0.0702014	0.0274090
323	0.3350377	0.1122483	0.0520761	0.0272223	0.0091229
324	0.3660041	0.1239598	0.0581533	0.0304489	0.0063029
325	0.6784404	0.0254389	0.1295720	0.0683061	0.0154498
326	0.3461233	0.1181646	0.0572948	0.0307179	0.0108904
327	0.6176877	0.2099267	0.0980657	0.0513510	0.0105957
328	0.3575202	0.1210691	0.0567649	0.0297776	0.0060591
361	0.3872636	0.1472462	0.0958050	0.0600656	0.0603736
362	0.6494118	0.2252495	0.1135498	0.1135498	0.0628996
363	0.4143533	0.1449913	0.0681400	0.0356820	0.0100273
364	0.8679762	0.4808443	0.4832851	0.4857508	0.4882419

365	0.4622259	0.1760347	0.1016929	0.0570572	0.0098488
366	0.6420889	0.2358239	0.1184073	0.0640838	0.0174811
367	0.3843686	0.1308655	0.0622150	0.0327671	0.0053719
393	0.3596851	0.1246689	0.0605051	0.0332741	0.0164209
407	-	-	-	-	-
408	0.9040001	0.4934771	0.4740957	0.2846225	0.2750789
409	0.4644980	0.1699668	0.0827006	0.0434840	0.0090608
410	0.5870602	0.2443127	0.1498310	0.0913590	0.0243615
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7270052	0.2785735	0.1595241	0.0941159	0.0148270
La moyenne	**0.5359261**	**0.2072516**	**0.1232306**	**0.0767518**	**0.0410395**

Formule F6 (avec propagation)

289	0.6626418	0.2856921	0.1990439	0.1521326	0.0157398
290	-	-	-	-	-
292	0.6414376	0.2602871	0.1571586	0.0979347	0.0084405
293	0.5766793	0.2326942	0.1326619	0.0794133	0.0073997
294	0.8028918	0.4132275	0.3274787	0.2071759	0.0165404
295	0.8837881	0.4610881	0.3966459	0.2689211	0.0272377
296	0.7018628	0.3240198	0.2576929	0.1904557	0.0205830
297	0.7719758	0.3649376	0.2812826	0.2051497	0.03665170
298	0.6431087	0.2576143	0.1836609	0.1474113	0.0112241
321	0.7541724	0.3613724	0.2578350	0.1706515	0.0131933
322	0.8299133	0.3874587	0.2918842	0.2123548	0.0372205
323	0.7628439	0.3062696	0.2111288	0.1598856	0.0405441
324	0.8671155	0.3862288	0.2590033	0.1869310	0.0107074
325	0.8118214	0.3986060	0.3453832	0.2447769	0.0275683
326	0.7181249	0.2962698	0.2202707	0.1788159	0.0250101
327	0.6701313	0.2775798	0.2015083	0.1576085	0.0193908
328	0.7106803	0.3024769	0.1975507	0.1352948	0.0108902
361	0.6702722	0.2802723	0.1774637	0.1228833	0.0414096
362	0.7673017	0.3601705	0.2655426	0.1947816	0.0228441
363	0.6693725	0.2700044	0.1966220	0.1605701	0.0131932
364	0.8905256	0.4358390	0.3201778	0.2756167	0.2770302
365	0.8259119	0.4222586	0.3894008	0.3065205	0.0350305
366	0.7791018	0.3536939	0.2693262	0.1868575	0.0184269
367	0.7481027	0.3238825	0.1721058	0.1082442	0.0068141
393	0.8921807	0.4415236	0.2966946	0.2052880	0.0307807
407	-	-	-	-	-
408	0.9545751	0.5042719	0.5030044	0.4161229	0.1647335
409	0.7324547	0.3407121	0.2261882	0.1663633	0.0193498
410	0.9342441	0.4499273	0.3975683	0.2960509	0.0383183
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7375593	0.3338358	0.2391066	0.1696359	0.0168641
La moyenne	0.7646711	0.3511505	0.2633354	0.1929946	0.0361834

Formule F7 (XFIRM)					
289	0.3943344	0.1333372	0.0621766	0.0325690	0.0074189
290	-	-	-	-	-
292	0.4899181	0.1749975	0.0832864	0.0508847	0.0103802
293	0.4960541	0.1762703	0.0903940	0.0511871	0.0151674
294	0.8329003	0.2916923	0.1413016	0.0748962	0.0129876
295	0.3961905	0.1451109	0.0717866	0.0380896	0.0071899
296	0.4570470	0.1684518	0.0849477	0.0453432	0.0102216
297	0.6280262	0.2459342	0.1396369	0.0846074	0.0356957
298	0.4804618	0.1885577	0.0855342	0.0501179	0.0190963
321	0.4597888	0.1618589	0.0783535	0.0414740	0.0073547
322	0.5369181	0.2029128	0.1031995	0.0583391	0.0225658
323	0.3350223	0.112241	0.0520705	0.0272172	0.0091201
324	0.3707518	0.1254404	0.0587111	0.0307432	0.0064154
325	0.6844933	0.2554693	0.1299837	0.0683838	0.0163129
326	0.3458145	0.1186301	0.0581073	0.0313516	0.0110542
327	0.6624258	0.2254536	0.1053924	0.0552648	0.0123064
328	0.3530935	0.1191790	0.0556320	0.0291320	0.0071907
361	0.3903886	0.1463395	0.0873883	0.0516609	0.0519258
362	0.5839872	0.2028280	0.1023200	0.0563127	0.0111821
363	0.4666642	0.1622708	0.0764874	0.0400665	0.0112558
364	0.8284630	0.4480648	0.4503392	0.4526369	0.4549581
365	0.4622326	0.1759247	0.1010917	0.0553350	0.0113193
366	0.5617578	0.2035771	0.1016098	0.0548231	0.0148084
367	0.3865183	0.1315675	0.0622616	0.0327216	0.0104272
393	0.3634764	0.1241826	0.0597292	0.0324152	0.0158452
407	-	-	-	-	-
408	0.9597459	0.5184484	0.4830709	0.2903836	0.2806469
409	0.4311234	0.1570372	0.0752410	0.0394485	0.0118146
410	0.6158703	0.2418630	0.1386511	0.0814428	0.0208566
411	-	-	-	-	-
412	-	-	-	-	-
413	0.7110238	0.2708908	0.1499849	0.0870267	0.0157691
La moyenne	**0.5244461**	**0.2010190**	**0.1181675**	**0.0729955**	**0.0400460**

Formule F7 (avec propagation)					
289	**0.6626417**	**0.2696984**	**0.1884253**	**0.1465827**	**0.0138020**
290	-	-	-	-	-
292	0.6853841	0.2761240	0.1716290	0.1225074	0.0102633
293	0.8660361	0.4239418	0.3220534	0.2274994	0.0286966
294	0.8028643	0.3882224	0.2377847	0.1557459	0.0104254
295	0.8697464	0.3753643	0.2658070	0.1902701	0.0160620
296	0.6913691	0.3052783	0.2376806	0.1767387	0.0170254

297	0.7719758	0.3912492	0.3207033	0.2323498	0.0434508
298	0.8468161	0.3748844	0.2497087	0.1828181	0.0411208
321	0.7735323	0.3680421	0.2460449	0.1649719	0.0107986
322	0.6572365	0.2812632	0.2089974	0.1633320	0.0271935
323	0.7628439	0.30626961	0.21112885	0.15988561	0.0414219
324	0.8674107	0.3965654	0.2673666	0.1913072	0.0106275
325	0.7932540	0.4075288	0.3514414	0.2411359	0.0282375
326	0.7181248	0.2976527	0.2264682	0.1832137	0.0294743
327	0.7348402	0.3206077	0.2343284	0.1749662	0.0268023
328	0.7076015	0.2984406	0.1938086	0.1333861	0.0138741
361	0.6702721	0.2802723	0.1754867	0.1176035	0.0345040
362	0.7617923	0.3447701	0.2522191	0.1805929	0.0157201
363	0.7858126	0.3315942	0.2302291	0.1785746	0.0195801
364	0.8591002	0.3978908	0.2810653	0.2336429	0.2348411
365	0.9170717	0.4582179	0.3879319	0.3008783	0.0342560
366	0.7791016	0.3474164	0.2307031	0.1547499	0.0135167
367	0.7471909	0.3232229	0.1716802	0.1079311	0.0245440
393	0.8081496	0.3880583	0.2599308	0.1753984	0.0229185
407	-	-	-	-	-
408	0.8970701	0.4493354	0.4338152	0.3519334	0.1426076
409	0.6810600	0.2992884	0.2128723	0.1586931	0.0313605
410	0.8334207	0.3925886	0.2710216	0.1787726	0.0182711
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8656332	0.4196039	0.3198981	0.2229624	0.0234634
La moyenne	0.7791912	0.3540497	0.2557225	0.1860159	0.0351736

Formule F8 (XFIRM)					
289	0.7129972	0.2923692	0.2221239	0.1758844	0.0178391
290	-	-	-	-	-
292	0.6871553	0.3095750	0.2093035	0.1439282	0.0151260
293	0.7384966	0.3299219	0.2350036	0.1654979	0.0218667
294	0.7258345	0.3185832	0.2172130	0.1613409	0.0125783
295	0.8907501	0.4413289	0.3394328	0.2377685	0.0223693
296	0.6829655	0.3076234	0.2375202	0.1760728	0.0170052
297	0.7719758	0.3649376	0.2770115	0.1952134	0.0325224
298	0.6380402	0.2543452	0.1819518	0.1457341	0.0133799
321	0.7177940	0.3374712	0.2584977	0.1864203	0.0162135
322	0.8108181	0.3687060	0.2652022	0.1954106	0.0314112
323	0.7592354	0.3036632	0.2093146	0.1588798	0.0405635
324	0.8725647	0.3911769	0.2626646	0.1887854	0.0107369

325	0.7868373	0.3764857	0.2752159	0.1944546	0.0219682
326	0.7113303	0.2965282	0.2176666	0.1757538	0.0232498
327	0.7541491	0.3349789	0.2405666	0.1769829	0.0276300
328	0.8557480	0.4291113	0.3541681	0.2420277	0.0195725
361	0.6720063	0.2879295	0.1871767	0.1321681	0.0464004
362	0.7511820	0.3443069	0.2340486	0.1700061	0.0146640
363	0.6707154	0.2682030	0.1946661	0.1592397	0.1162908
364	0.9211786	0.4447017	0.3380126	0.2935060	0.2950112
365	0.7558236	0.3439174	0.2975489	0.2367533	0.0269594
366	0.7731241	0.3436840	0.2417562	0.1685115	0.0162012
367	0.6864118	0.2917051	0.2234166	0.1713690	0.0175938
393	0.7596027	0.3040640	0.2032666	0.1539310	0.0218295
407	-	-	-	-	-
408	0.8737932	0.4136061	0.3745737	0.3062780	0.1364149
409	0.6827711	0.3024754	0.2098629	0.1574601	0.0250545
410	0.7584329	0.3592625	0.2984737	0.2252765	0.0294128
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8239254	0.4046485	0.3484531	0.2474514	0.0263836
La moyenne	**0.7587735**	**0.3416182**	**0.2555040**	**0.1907895**	**0.0361281**

Formule F8 (avec propagation)

289	0.6722997	0.2692277	0.1886027	0.1464061	0.0136469
290	-	-	-	-	-
292	0.6853842	0.2891565	0.1841211	0.1291857	0.0117843
293	0.8042958	0.3796054	0.2997963	0.2190091	0.0308673
294	0.8466681	0.4307363	0.3135657	0.2060424	0.0170817
295	0.8839394	0.3683130	0.2604154	0.1851592	0.0158880
296	0.6829656	0.3067069	0.2320446	0.1721950	0.0160483
297	0.7719758	0.3649376	0.2880225	0.2101290	0.0391660
298	0.6382893	0.2544670	0.1820548	0.1458147	0.0149750
321	0.8074573	0.3798185	0.2649864	0.1754356	0.0148347
322	0.6938105	0.2994443	0.2212668	0.1700320	0.0261334
323	0.7592360	0.3036635	0.2093149	0.1588801	0.0414402
324	0.8733312	0.3963775	0.2667851	0.1909516	0.0106479
325	0.8054335	0.3962841	0.2985852	0.2078793	0.0265611
326	0.7113303	0.2965282	0.2176666	0.1757539	0.0253701
327	0.8426856	0.4011372	0.2796073	0.1976117	0.0376516
328	0.7071071	0.2987116	0.1940481	0.1334635	0.0155937
361	0.6720063	0.2879295	0.1828887	0.1269183	0.0419214
362	0.7425092	0.3514775	0.2267254	0.1654937	0.0136186
363	0.6705181	0.2683038	0.1953821	0.1596369	0.0126164
364	0.8591002	0.4158030	0.2989935	0.2526380	0.2539336
365	0.7460104	0.3322327	0.2694036	0.2073664	0.0241216
366	0.7731242	0.3436840	0.2306311	0.1557510	0.0140644
367	0.9134729	0.4624156	0.3518269	0.2375446	0.0234787

393	0.7656910	0.3696809	0.2480282	0.1793928	0.0239714
407	-	-	-	-	-
408	0.9650763	0.5105221	0.4928834	0.4085869	0.1885755
409	0.6667759	0.2921088	0.2084964	0.1564688	0.0330396
410	0.8019665	0.3841423	0.2904481	0.2127743	0.0252173
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8019665	0.3841423	0.2904481	0.2127743	0.0252173
La moyenne	0.7696098	0.3510402	0.2564085	0.1888839	0.0369949
Formule F9 (XFIRM)					
289	0.4307575	0.1466676	0.0686461	0.0359970	0.0081537
290	-	-	-	-	-
292	0.7596092	0.3146144	0.1908922	0.1087971	0.0227050
293	0.4434664	0.1554205	0.0777007	0.0435414	0.0133515
294	0.6620916	0.2323349	0.1117142	0.0594549	0.0098055
295	0.5460976	0.2184505	0.1089407	0.0581381	0.0108813
296	0.5098468	0.1955495	0.1010111	0.0550325	0.0124492
297	0.5397705	0.2002737	0.1175109	0.0723562	0.0306296
298	0.4932034	0.1761605	0.0824895	0.0432104	0.0066316
321	0.7258930	0.2631196	0.1324938	0.0712384	0.0121139
322	0.6782401	0.2711083	0.1380161	0.0756299	0.0272759
323	0.3356222	0.1125545	0.0523166	0.0274174	0.0088844
324	0.3523023	0.1194733	0.0560062	0.0293295	0.0060212
325	0.6900714	0.2632785	0.1334772	0.0704077	0.0128780
326	0.3426138	0.1168092	0.0555821	0.0294697	0.0098079
327	0.6760867	0.2305131	0.1078171	0.0566281	0.0096401
328	0.4403053	0.1657808	0.0924361	0.0532388	0.0096965
361	0.5856834	0.2716112	0.2299648	0.1571292	0.1467000
362	0.6733650	0.2365477	0.1204781	0.0657631	0.0100750
363	0.3633936	0.1245271	0.0582003	0.0304732	0.0085495
364	0.8959966	0.5718666	0.5747694	0.5777019	0.5806645
365	0.4398062	0.1602518	0.0863363	0.0463804	0.0073558
366	0.5567105	0.2082509	0.1068643	0.0587022	0.0162559
367	0.4086150	0.1521843	0.0812920	0.0436105	0.0068108
393	0.3831675	0.1351647	0.0658379	0.0362632	0.0173389
407	-	-	-	-	-
408	0.6084039	0.2869593	0.2643717	0.1869633	0.1806943
409	0.6504671	0.2553857	0.1252729	0.0665132	0.0133034
410	0.7240613	0.2990719	0.1928426	0.1235413	0.0360160
411	-	-	-	-	-
412	-	-	-	-	-
413	0.6734001	0.2771216	0.1707005	0.1055741	0.0186004
La moyenne	**0.5567517**	**0.2200376**	**0.1322851**	**0.0853037**	**0.0447604**

Formule F9 (avec propagation)					
289	**0.6802314**	**0.2921872**	**0.2012288**	**0.1527347**	**0.015764**
290	-	-	-	-	-
292	0.8792013	0.4300721	0.3174478	0.2337298	0.0266607
293	0.7517791	0.3377314	0.2364780	0.1624011	0.0206043
294	0.8423400	0.4149754	0.2762251	0.1933011	0.0142028
295	0.8570993	0.4261946	0.3357477	0.2270330	0.0216304
296	0.7339061	0.3335141	0.2723965	0.2086552	0.0239494
297	0.7719758	0.3649376	0.2770115	0.1934370	0.0303270
298	0.6448384	0.2558721	0.1824686	0.1455997	0.010383
321	0.8759294	0.4362414	0.3179534	0.2216524	0.017338
322	0.8841989	0.4309095	0.3038412	0.2169438	0.0343992
323	0.7583914	0.3030865	0.2089213	0.1586609	0.0420645
324	0.8807087	0.3900423	0.2545551	0.1842473	0.0096949
325	0.8887324	0.4388155	0.3311289	0.3311289	0.0292441
326	0.7079923	0.2957341	0.2162574	0.1743682	0.0192010
327	0.6518826	0.6518826	0.1897002	0.1493544	0.0227443
328	0.7236999	0.3502916	0.3357903	0.2797821	0.0369262
361	0.7550689	0.3543034	0.2858195	0.20457338	0.0785434
362	0.6573925	0.2636700	0.1948411	0.1582576	0.0138649
363	0.6705330	0.2680506	0.1941440	0.1585853	0.0090265
364	0.9466419	0.4537903	0.3369927	0.2904639	0.2919535
365	0.8239554	0.3887973	0.2941531	0.2111989	0.0203031
366	0.8222168	0.4058717	0.3080123	0.2134487	0.0249012
367	0.9219590	0.4674997	0.3958783	0.2697060	0.02735265
393	0.7168718	0.3222638	0.2083862	0.1348772	0.0187915
407	-	-	-	-	-
408	0.8686434	0.3952260	0.3634052	0.3004345	0.3004345
409	0.8411582	0.3998588	0.2695937	0.1947307	0.0200704
410	0.8496352	0.4196108	0.3718855	0.2856948	0.0416856
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8469320	0.4340806	0.4245132	0.3465300	0.0403259
La moyenne	0.7947827	0.3694546	0.2823134	0.2106422	0.0393879

Formule F10 (XFIRM)					
289	0.4140980	0.1404329	0.0656214	0.0343814	0.0077740
290	-	-	-	-	-
292	0.7616766	0.2999474	0.1765327	0.0998946	0.0208534
293	0.4538237	0.1608503	0.0809189	0.0453026	0.0139737
294	0.6138715	0.2185381	0.1065961	0.0569831	0.0092472
295	0.4682087	0.1817315	0.0907860	0.0484559	0.0092809
296	0.4863407	0.1834105	0.0937495	0.0505422	0.0113308
297	0.5397705	0.2002737	0.1178553	0.0732452	0.0311461
298	0.4847541	0.1742489	0.0817346	0.0427999	0.0064493

321	0.6645040	0.2375675	0.1172623	0.0630724	0.0104165
322	0.7530744	0.2950003	0.1495277	0.0817257	0.0307519
323	0.3353515	0.1124093	0.0522054	0.0273289	0.0089515
324	0.3577053	0.1212537	0.0568128	0.0297464	0.0061068
325	0.6869222	0.2603628	0.1301258	0.0684627	0.0145619
326	0.3434255	0.1174885	0.0561994	0.0298223	0.0099720
327	0.7037449	0.2399217	0.1119600	0.0587162	0.0103396
328	0.4185919	0.1520193	0.0817229	0.0460404	0.0083299
361	0.4695697	0.1961052	0.1534633	0.1115196	0.1042992
362	0.7821276	0.2733521	0.1382558	0.0757464	0.0123695
363	0.3630237	0.1253103	0.0586568	0.0307084	0.0086210
364	0.9468061	0.5702205	0.5731151	0.5760391	0.5789932
365	0.4220776	0.1555379	0.0823935	0.0446692	0.0072618
366	0.4704662	0.1724922	0.0872728	0.0473965	0.0129181
367	0.4038963	0.1496426	0.0785604	0.0421577	0.0067077
393	0.3689847	0.1278145	0.0613492	0.0333596	0.0160666
407	-	-	-	-	-
408	0.6279291	0.3071227	0.2955035	0.2089910	0.2019834
409	0.6936860	0.2698233	0.1315997	0.0696782	0.0145691
410	0.5975115	0.2490032	0.1550921	0.1550921	0.0974798
411	-	-	-	-	-
412	-	-	-	-	-
413	0.6684421	0.2649174	0.1528558	0.0919600	0.0155627
La moyenne	**0.5464423**	**0.2127428**	**0.1263475**	**0.0816510**	**0.0434145**

Formule F10 (avec propagation)

289	0.6774439	0.2996748	0.2047856	0.1546923	0.0164797
290	-	-	-	-	-
292	0.8738983	0.4348363	0.3593126	0.2656990	0.0300871
293	0.7189557	0.3229587	0.2235974	0.1501242	0.0183452
294	0.9179715	0.4395824	0.2889735	0.2062098	0.0157846
295	0.9082838	0.4586505	0.3729258	0.2555258	0.0256810
296	0.7251325	0.3288583	0.2661222	0.2031503	0.0228332
297	0.7719758	0.3649376	0.2770115	0.1934370	0.0288778
298	0.6414291	0.2557683	0.1824422	0.1458458	0.0100693
321	0.8578180	0.4276660	0.3420489	0.2371961	0.0189966
322	0.8240166	0.3843323	0.2860764	0.2035596	0.0317032
323	0.7584589	0.3031321	0.2089524	0.1586782	0.0411250
324	0.8761358	0.3872966	0.2546437	0.1843989	0.0104550
325	0.8530102	0.4223404	0.3191152	0.2203396	0.0270763
326	0.7098931	0.2961955	0.2170605	0.1751548	0.0200975
327	0.6552318	0.2706888	0.1908944	0.1502132	0.0175484
328	0.7252446	0.3494132	0.3318583	0.2753134	0.0370929
361	0.6737251	0.3105305	0.2363435	0.1687414	0.0631779
362	0.7767399	0.3556277	0.2660427	0.1970531	0.0214760

363	0.6705692	0.2680786	0.1943374	0.1588320	0.0094041
364	0.9046708	0.4376188	0.3196176	0.2736034	0.2750065
365	0.7374279	0.3237256	0.2337761	0.1774675	0.0168670
366	0.7691327	0.3580703	0.2631018	0.1832425	0.0193132
367	0.7472441	0.3384810	0.2464330	0.1824769	0.0191069
393	0.8317568	0.3993767	0.2666548	0.1823826	0.0251034
407	-	-	-	-	-
408	0.8716093	0.4253609	0.4103253	0.3422260	0.1609658
409	0.7596352	0.3661480	0.2636651	0.1918928	0.0198584
410	0.7274227	0.3465874	0.2928451	0.2260178	0.0321914
411	-	-	-	-	-
412	-	-	-	-	-
413	0.8918247	0.4504284	0.4498617	0.3762052	0.0436921
La moyenne	0.7805949	0.3616559	0.2774580	0.2085600	0.0385149

Formule F11 (XFIRM)					
289	0.4028741	0.1369260	0.0640492	0.0335706	0.0075917
290	-	-	-	-	-
292	0.7875528	0.3227115	0.2066602	0.1198704	0.0592229
293	0.3807100	0.1327557	0.0651752	0.0358986	0.0100257
294	0.3963082	0.1393011	0.0710452	0.0386023	0.0065921
295	0.5828147	0.2163553	0.1064028	0.0570780	0.0106491
296	0.5942418	0.2330943	0.1346088	0.0762320	0.0184323
297	0.6210553	0.2525425	0.1484227	0.0977584	0.0458268
298	0.3427277	0.1154989	0.0539726	0.0282944	0.0042280
321	0.6108044	0.2435395	0.1278115	0.0699508	0.0123487
322	0.7219350	0.2901968	0.1524674	0.0844901	0.0295939
323	0.3389424	0.1146815	0.0542917	0.0288052	0.0094617
324	0.3486291	0.1186559	0.0558928	0.0293320	0.0061263
325	0.6939744	0.2694662	0.1399398	0.0748322	0.0130400
326	0.3427067	0.1164187	0.0559105	0.0300275	0.0099282
327	0.5923717	0.2037143	0.0955772	0.0501017	0.0075939
328	0.7855926	0.3205510	0.1899970	0.1146807	0.0219410
361	0.8710217	0.4219464	0.2877146	0.1618237	0.1506811
362	0.4216732	0.1526325	0.0794161	0.0436331	0.0072364
363	0.3570398	0.1209674	0.0564833	0.0296001	0.0082907
364	0.7956448	0.4595687	0.4619016	0.4642582	0.4666390
365	0.8005867	0.3848001	0.2885385	0.1667719	0.0029065
366	0.8202892	0.3224115	0.1690479	0.0952767	0.0267212
367	0.7706498	0.3309041	0.1936138	0.1139956	0.0205893
393	0.4447856	0.1665049	0.0943698	0.0552233	0.0267476
407	-	-	-	-	-
408	0.6259586	0.2909343	0.2348025	0.1556194	0.1504014
409	0.5870202	0.2040004	0.1007888	0.5431066	0.0114857
410	0.7808093	0.3620816	0.2802610	0.1901807	0.0544658

411	-	-	-	-	-
412	-	-	-	-	-
413	0.9964057	0.2495067	0.4159105	0.2676391	0.0613837
La moyenne	**0.6005402**	**0.2450678**	**0.1566097**	**0.0988520**	**0.0447503**

Formule F11 (avec propagation)

289	0.6823261	0.2956486	0.2100094	0.1584541	0.0190948
290	-	-	-	-	-
292	0.8253800	0.4130184	0.3554386	0.2596472	0.0238981
293	0.6914225	0.2917114	0.1837197	0.1085932	0.0092255
294	0.8032637	0.3920085	0.3192659	0.2397440	0.0442374
295	0.6934533	0.2815038	0.2043702	0.1504650	0.0115260
296	0.8475819	0.4408012	0.4159697	0.3300442	0.0376292
297	0.8801057	0.4381720	0.3805207	0.2880807	0.0568156
298	0.7964144	0.3565903	0.2391305	0.1677782	0.0113751
321	0.8984640	0.4516720	0.3906211	0.2906813	0.0245624
322	0.9286811	0.4416165	0.3328672	0.2383759	0.0309539
323	0.7583630	0.3030674	0.2089083	0.1586537	0.0463333
324	0.8906452	0.3955302	0.2586588	0.1861061	0.0112335
325	0.8751680	0.4276877	0.3448009	0.2702338	0.0345058
326	0.7109166	0.2964342	0.2174923	0.1755810	0.0197262
327	0.7199559	0.3183759	0.2609701	0.2108058	0.0298083
328	0.8619314	0.3715432	0.2679733	0.1646064	0.0093151
361	0.9288173	0.5013183	0.4600193	0.3635816	0.0928472
362	0.4216732	0.1526325	0.0794161	0.0436331	0.0072364
363	0.3570398	0.1209674	0.0564833	0.0296001	0.0082907
364	0.7956448	0.4595687	0.4619016	0.4642582	0.4666390
365	0.8005867	0.3848001	0.2885385	0.1667719	0.0290657
366	0.8202892	0.3224115	0.1690479	0.0952767	0.0267212
367	0.7706498	0.3309041	0.1936138	0.1139956	0.0205893
393	0.4447856	0.1665049	0.0943698	0.0552233	0.0267476
407	-	-	-	-	-
408	0.6259586	0.2909343	0.2348025	0.1556194	0.1504014
409	0.5870202	0.2040004	0.1007888	0.0543106	0.0114857
410	0.7808093	0.3620816	0.2802610	0.1901807	0.0544658
411	-	-	-	-	-
412	-	-	-	-	-
413	0.9964057	0.5430274	0.4159105	0.2676391	0.0613837
La moyenne	0.7569198	0.3483762	0.2652096	0.1927836	0.0491469

Tableau VI-5 : Les résultats des nxCG pour les formules pour toutes les requêtes

Durant les tests effectués, certaines requêtes n'ont pas donné des résultats et sont représentées dans le tableau ci-dessus par des tirets (-). Ces requêtes correspondent aux numéros suivants : 290, 407, 411 et 413.

Et voici un tableau comparatif de la moyenne des nxCG pour toutes les formules avec et sans propagation :

A : signifie avec propagation.

S : signifie sans propagation.

Tableau VI-6 : Les moyennes des nxCG à 5 niveaux pour toutes les formules.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
nxCG à 5 S	0.5240234	0.5257032	0.7862230	0.7782022	0.5332722	0.5359261	0.5244461	0.7587735	0.5567517	0.5464423	0.6005402
A	0.7800814	0.7651794	0.7904571	0.7856476	0.7787971	0.7646711	0.7791912	0.7696098	0.7947827	0.7805949	0.7569198
nxCG à 10 S	0.2026136	0.2036111	0.3635502	0.3554771	0.2076358	0.2072516	0.2010190	0.3416182	0.2200376	0.2127428	0.2450678
A	0.3594330	0.3488459	0.3621441	0.3584298	0.3525621	0.3511505	0.3540497	0.3510402	0.3694546	0.3616559	0.3483762
nxCG à 25 S	0.1189335	0.1192220	0.2772172	0.2706433	0.1233432	0.1232306	0.1181675	0.2555040	0.1322851	0.1263475	0.1566097
A	0.2719076	0.2606428	0.2656813	0.2726643	0.2689063	0.2633354	0.2557225	0.2564085	0.2823134	0.2774580	0.2652096
nxCG à 50 S	0.1174549	0.0793409	0.2033062	0.1999808	0.0778533	0.0767518	0.0729955	0.1907895	0.0853037	0.0816510	0.0988520
A	0.1995742	0.1913696	0.1947163	0.2019491	0.2017460	0.1929946	0.1860159	0.1888839	0.2106422	0.2085600	0.1927836
nxCG pour la totalité	0.0445257	0.0445327	0.0377356	0.0371298	0.0416185	0.0410395	0.0400460	0.0361281	0.0447604	0.0434145	0.0447503
A S	0.0368672	0.0361204	0.0369804	0.0364821	0.0385514	0.0361834	0.0351736	0.0369949	0.0393879	0.0385149	0.0491469

Dans ce qui suit, des histogrammes illustratifs du tableau comparatif des gains des onze formules testées avec la méthode de propagation et autres sans la méthode, ainsi que des interprétations des résultats : (le S signifie sans propagation le A avec propagation)

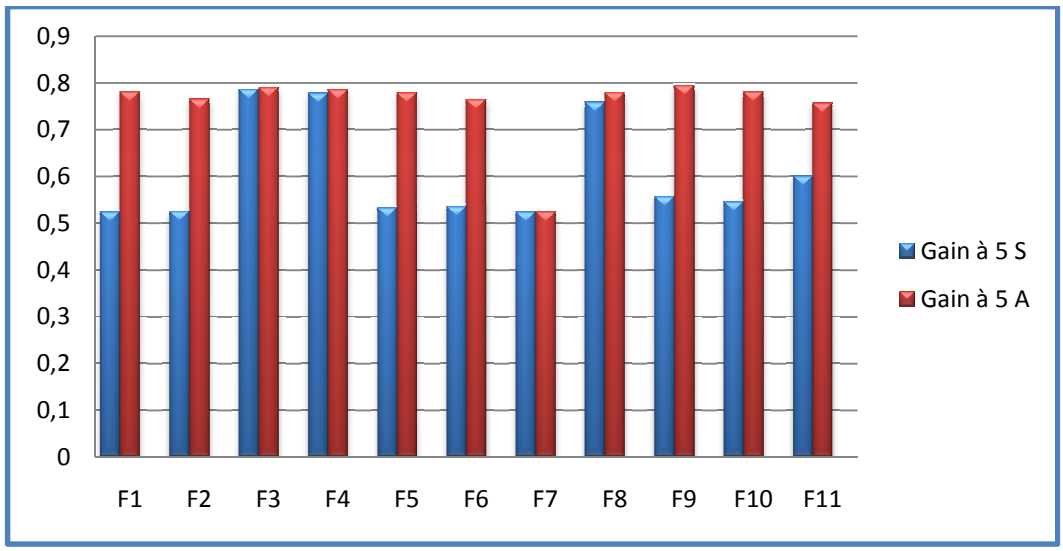


Figure VI-5 : Histogramme des moyennes des gains correspondants aux cinq (5) premiers résultats des onze formules différentes.

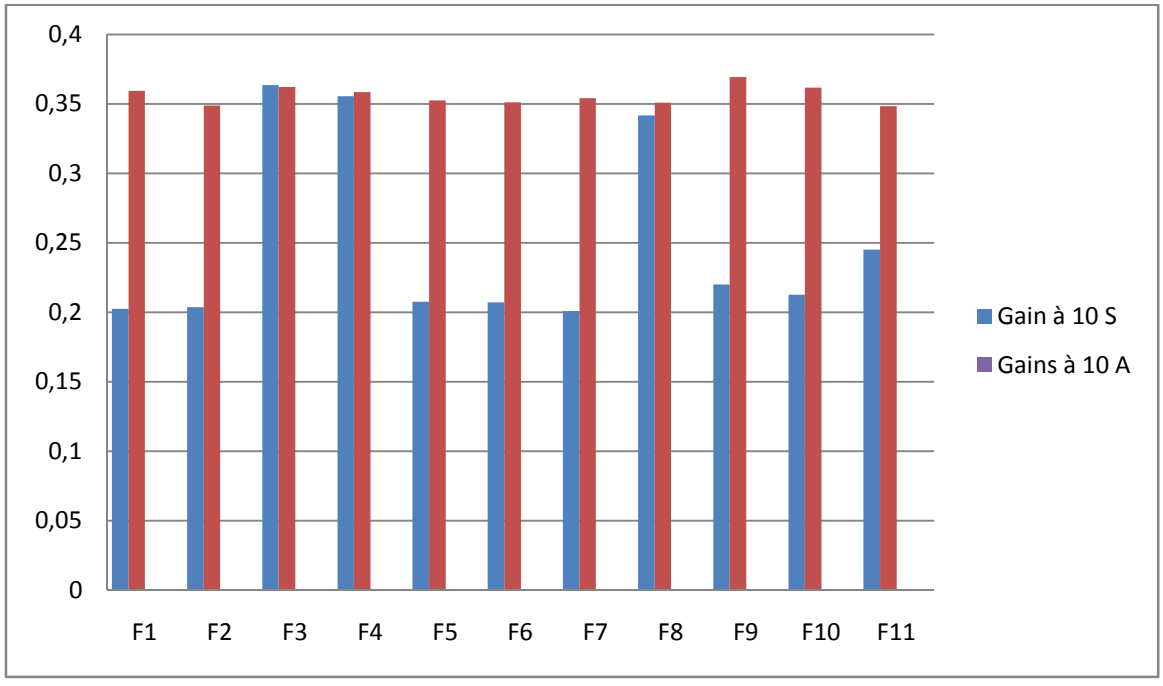


Figure VI-6 : Histogramme des moyennes des gains correspondants aux dix (10) premiers résultats des onze formules différentes.

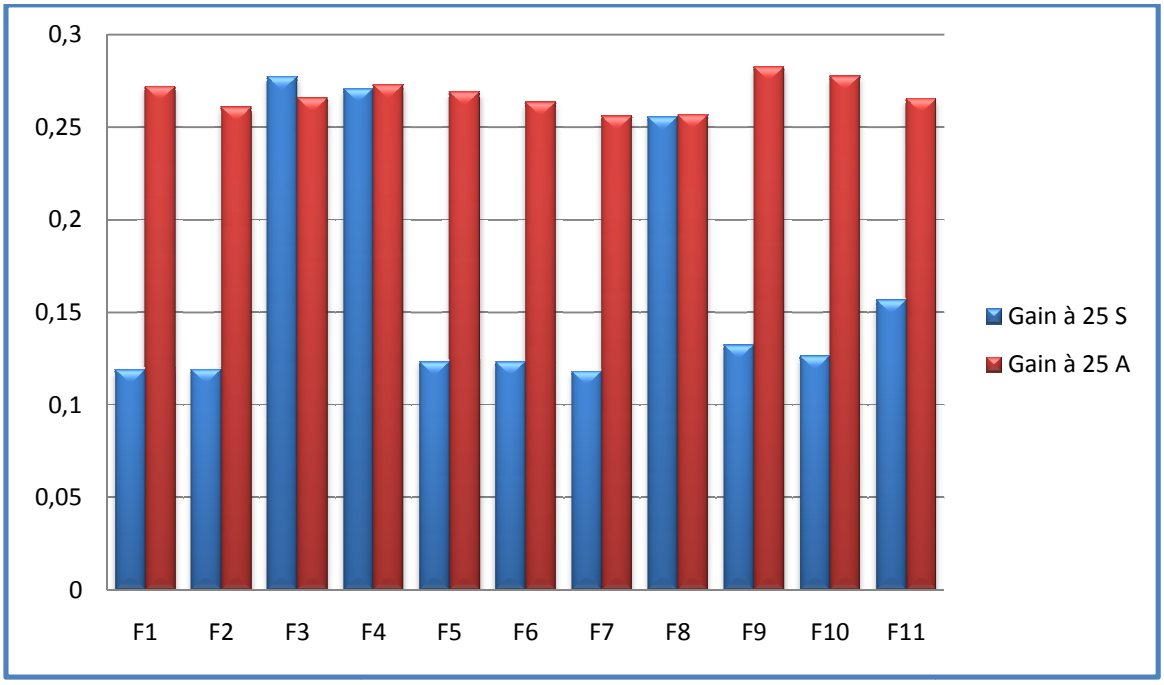


Figure VI-7 : Histogramme des moyennes des gains correspondants aux vingt-et-cinq (25) premiers résultats des onze formules différentes.

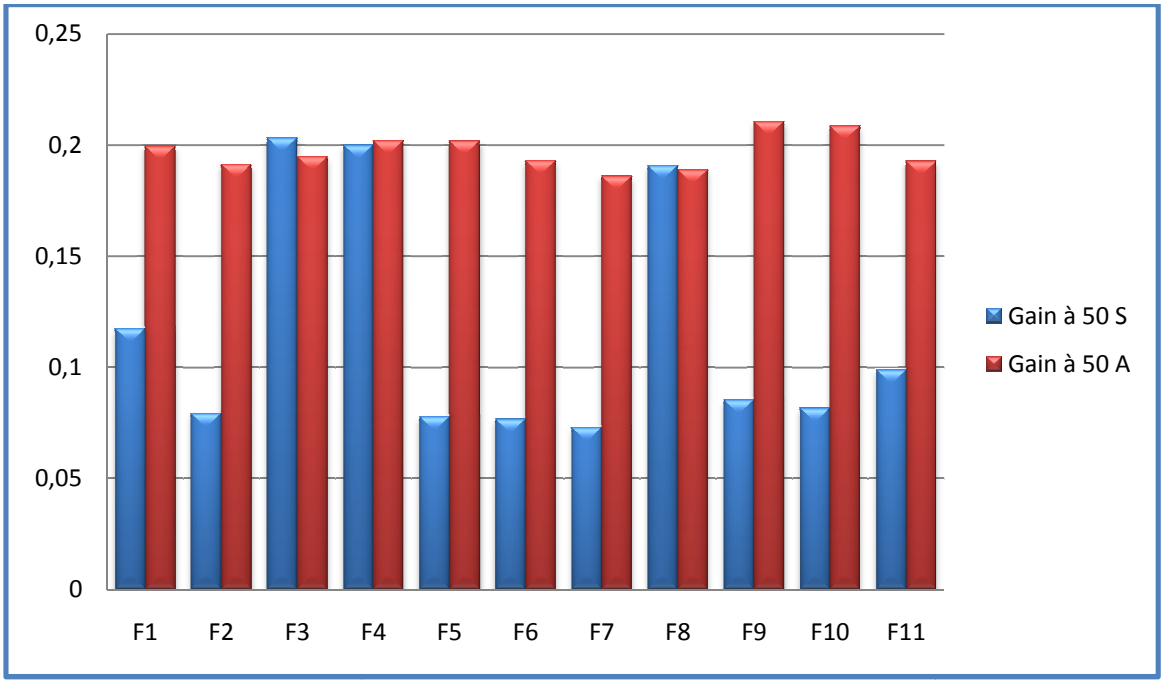


Figure VI-8 : Histogramme des moyennes des gains correspondants aux cinquante (50) premiers résultats des onze formules différentes.

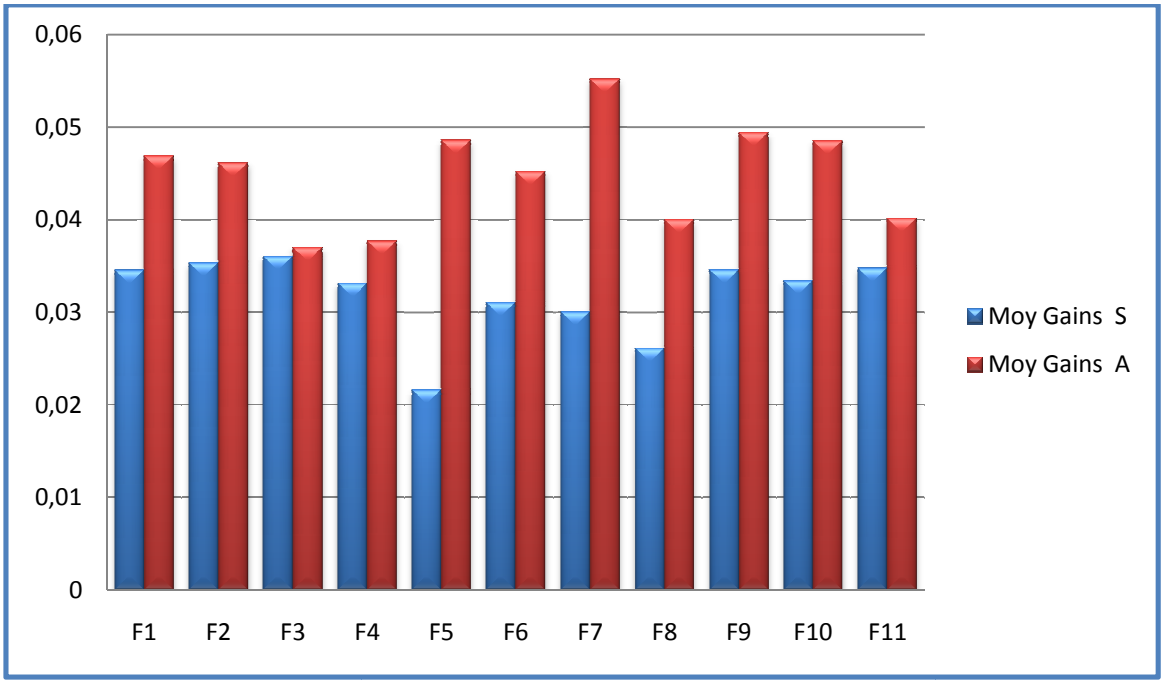


Figure VI-7: Histogramme comparatif des moyennes des gains correspondants aux différents niveaux du calcul pour le jeu des onze formules.

VI.2.4.3 Interprétation des résultats :

Lors de la comparaison entre les résultats obtenus des tests des formules, nous avons constaté que la méthode de propagation des termes que nous avons programmé a pu augmenter le gain ainsi elle a donné de très bons et meilleurs résultats avec les formules de pondération (F1, F2, F5, F6, F7, F9, F10, F11)

Voici un tableau qui montre le pourcentage de différence entre les résultats obtenus avec et sans propagation des termes :

Formules	Gains à 5	Gain à 10	Gain à 25	Gain à 50	Moyenne
F1	25 ,6 %	15,6%	15,2%	8,2%	16,15%
F2	23,9 %	14,5%	14,1%	11,2%	16,00%
F3	4,2 %	-0,1%	-0,1%	-0,8%	0,8%
F4	7,4 %	0,2%	0,2%	0,1%	1,97%
F5	24,5%	14,4%	14,5%	12,3%	16,42%
F6	22,8%	14,3%	14,01%	11,6%	15,67%
F7	25,4%	15,3%	13,7%	13,3%	17,00%
F8	10,8%	0,9%	0,9%	-0,1%	3,12%
F9	23,8%	14,9%	15%	12,5%	16,55%
F10	23,4%	14,8%	15,1%	12,6%	16,47%
F11	15,6%	10,3%	10,8%	9,3%	11,5%
POURCENTAGE DE TOUTES LES MOYENNES : +12 %					

VI-7- Tableau des pourcentages des gains obtenus.

Dans le tableau des pourcentages des gains obtenus nous constatons que pour la majorité des formules de pondération nous avons obtenus des valeurs positives. On note que nous avons obtenu jusqu'à plus de **25%** d'amélioration dans les gains obtenus avec la propagation des termes par rapport a ceux obtenu sans propagation.

Dans les moyennes des gains nous avons pu avoir un excellent pourcentage d'amélioration qui est de **17%** pour la formule de pondération **F7** et plus de **16%** pour les formules **(F1, F2, F5, F9, F10), 11.5%** pour la formule **F11.**

Pour les formules **(F3, F4, F8)** dont les pourcentages sont respectivement **(0.8%, 1,97%, 3 ,12%)** l'amélioration des gains est considéré faible par rapport aux résultats des autres formules.

Au total nous avons pu avoir une amélioration de gain de **+12%** en implémentant notre méthode de propagation des termes.

En plus détaillé :

- Nous constatons que la formule F7, a donné de meilleurs résultats dont l'explication est comme suit :
  - ➢ sachant que :
- $|e_j|$ est la taille de l'élément e_j en nombre de termes et $\Delta|e_j|$ est la taille moyenne d'un élément ;
- tf_{ji} : la fréquence de ti dans e_j ;
- N : le nombre d'éléments dans la collection ;
- df_i : le nombre d'éléments qui contiennent le terme t_i dans le document;
- ndl : le ratio entre la taille de e_j et la taille moyenne des éléments (en nombre de termes) ;
- k_1 = 1.2 et b = 0.75
- *h1= 0,3 ; h2 =0,5 ; h3=0,2.*

  - ➢ *Nous avons :*

- F7 $=\frac{tf_{ij}*(K_1+1)}{K_1*((1-b)+(b*ndl))+tf_{ij}} * log\frac{Ne-df_i+0.5}{df_i+0.5}$ où *Ne* est le nombre de nœuds feuilles dans le document ;
  - ➢ Nous remarquons que F7 utilise le nombre de nœuds feuilles (Ne) ; ce qui veut dire qu'elle se base sur les termes eux même, ce qui fait qu'elle soit bien meilleure que toutes.
- F9 = F7 dont le $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel}$
- F10 = F7 dont le $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dcel}{\Delta cel} + h_3 * \frac{dct}{\Delta ct}$
  - ➢ Dans la formule F10, le paramètre $\frac{dct}{\Delta ct}$ a été rajouté en plus on lui a attribué un poids qui est h3=0,2 ; donc nous prenons en considération le ratio d'une taille d'un document et sa taille moyenne, contrairement à F9 qui utilise seulement le ratio de la taille d'un élément et sa taille moyenne (en nombre de termes) ainsi la taille d'un document et sa moyenne (en nombre d'élément) .

- Nous avons donc déduit que pour F10 le paramètre rajouté n'a pas aidé à donner de meilleurs résultats que F9 et F7 car il prend en considération la taille des documents.
- Nous remarquons aussi que pour F5 Tel que :

$$F5 = tf * idf * ief * \frac{1}{h_1 + h_2 * \frac{det}{\Delta et} + h_3 * \frac{dct}{\Delta ct}}$$ Où :

tf : $fréquence\ local$ ($frequence\ d'un\ terme\ dans\ un\ document$).

idf : $fréquence\ global$ ($fréquence\ d'un\ terme\ dans\ la\ collection$).

ief : $frequence\ inverse\ d'element$.

  - Ici F5 a donné de meilleurs résultats par rapport aux restes : car le paramètre $\frac{det}{\Delta et}$ a un certain poids après lui avoir attribué la constante h2=0,5.

- La formule F1 a donné de meilleurs résultats que F2 et :

$$F1 = tf * idf$$

$$F2 = tf * ief$$

  - Ici F2 n'utilise pas la fréquence d'un terme dans toute la collection mais elle prend en considération la fréquence inverse d'élément et n'utilise pas la fréquence d'un terme dans un document.
  - F2 donné à son tour de meilleur résultat que F4 qui est égale à $tf * ief * \frac{1}{h_1 + h_2 * \frac{det}{\Delta et}}$ car $\frac{det}{\Delta et}$ est un paramètre qui calcule la taille d'un élément \ la taille moyenne de se dernier (en nombre de termes) et en lui donnant un certain poids car il est multiplié par h2 = 0,5.

- La formule F6 qui est égale à $\frac{tf_{ij} * (K_1 + 1)}{K_1 * ((1-b) + (b * ndl)) + tf_{ij}} * log \frac{N - df_i + 0.5}{df_i + 0.5}$ prend en considération le nombre d'élément dans la collection ainsi son

$ndl = \frac{|e_j|}{\Delta |e_j|} = \frac{det}{\Delta et}$ Ce qui fait qu'elle donne de moins bon résultats.

- $F11 = tf * ief * \frac{1}{h1+h2*\frac{det}{\Delta et}+h3*\frac{dct}{\Delta ct}+h4*\frac{dcel}{\Delta cel}}$ consiste en une normalisation de la (*F5*) par la taille du document en nombre de termes et sa taille en nombre d'éléments avec pondération des paramètres d'arrangement : $\frac{1}{h1+h2*\frac{det}{\Delta et}+h3*\frac{dct}{\Delta ct}+h4*\frac{dcel}{\Delta cel}}$
- F8 = F7 dont le $ndl = h_1 * \frac{det}{\Delta et} + h_2 * \frac{dct}{\Delta ct}$ donc il se base sur la taille du document \ moyenne de cette taille (en nombre de terme).
- La formule $F4 = tf * ief * \frac{1}{h_1+h_2*\frac{det}{\Delta et}}$ prend en considération la taille d'un élément ainsi la fréquence inverse d'élément alors que notre but dans la propagation des termes est la focalisation sur les termes eux même .Mais $F3 = tf * idf * ief$ a donné de moins bon résultats que toutes les autres formules qui est 0,8 d'augmentation.

VI. 3.Synthèse :

Les expérimentations que nous avons effectuées ont montré que notre modèle de propagations des termes offre de meilleurs résultats par rapport au modèle XFIRM sur tous les résultats des formules (F1, F2, F5, F6, F7, F9, F10, F11) et des résultats moins bons pour les formules (F3, F4, F8) mais de manière générale nous concluons que la méthode implémentée a pu améliorer les résultats de la recherche dans **XFIRM**.

VI. 4.Conclusion :

Dans ce chapitre, nous avons d'abord décrit notre modèle de RIS implémentant une méthode de propagation des termes. Ensuite nous avons présenté les formules testées et les résultats d'expérimentation obtenus après application de ce modèle et d'autres modèles de RI sur les collections INEX 2006. Notre modèle donne aussi de meilleurs résultats dans quelques formules, mais perd en gain dans d'autres. Les expérimentations que nous avons effectuées ont montré que notre modèle offre de meilleurs résultats par rapport au modèle XFIRM sur tous les résultats des formules (F1, F2, F5, F6, F7, F9, F10, F11).

Conclusion

Générale

Les travaux présentés dans ce mémoire se situent dans le contexte général de la RI et plus précisément dans le cadre de la RI structurée sur des documents XML. L'objectif d'un SRI structurée est de renvoyer à l'utilisateur suite à la formulation d'une requête, non plus des documents entiers, mais des fragments de documents, c'est à dire des sous arbres de documents XML, répondant à son besoin en information. Dans ce but, plusieurs solutions doivent être appréhendées. Ces solutions doivent porter essentiellement sur le stockage des documents, leur interrogation ainsi que sur le tri des unités d'information résultats.

▪ Perspectives

Nous envisageons plusieurs suites à nos travaux.

Notre première perspective serait de proposer et de tester d'autres conditions pour propager un terme au sein du document ou il se situe et cela en se basant sur le principe d'augmenter la moyenne des poids des termes en attribuant un coefficient au poids du terme qu'on veut propager.

Notre deuxième perspective serait que notre système traite aussi les requêtes portantes sur le contenu et la structure CAS. En effet Il serait judicieux de les inclure, et vu le stockage des données proposé ceci est parfaitement envisageable. Dans les requêtes de type CAS, l'utilisateur peut exprimer son besoin en donnant des conditions sur la structure des documents, en indiquant un nom de balise, ou en ajoutant des contraintes sur le contenu des éléments correspondants ou des attributs. Dans notre système la structure du document est représentée par un sous-arbre issu de l'arbre initial. Pour résoudre ça il faudrait garder la structure initiale, et ceci est tout-a-fait possible dans notre cas.

Bibliographie

Bibliographie

[1] : Cours « Problématique Générale de la Recherche d'Information », URFIST Bretagne-Pays de Loire, Alexandre Serres, 2002.
http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html

[2]: G. Salton, The Smart Retrieval System : Experiments in Automatic Document Processing, G. Salton Editor, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971.

[3] : MAMMERI Karima, « Recherche d'information par croisement de média texte et image ». Thèse Magister, Université M'hamed BOUGARA de BOUMERDES, 2009.

[4]: S.E. Robertson, S. Walker, M. Beaulieu: OKAPI at TREC 7: Automatic Adhoc Filtering, VLC and Interactive Track, In Proceedings of the 7th Text Retrieval Conference TREC7, Juillet 1999.

[5] : C. Flurh & F. Debili : Interrogation en Langue Naturelle de Données Textuelles et Factuelles, Intelligent Multimadia Information Systems and Management (RIAO), Grenoble (France), 1985.

[6]: C. Bourne, B.Anderson : DIALOG LabWorkbook, second edition, Lockheed Information Systems, PaloAlto, Californie (USA), 1979.

[7] : A. Lelu & C. François : Information Retrieval Based on Neural Unsupervised Extraction of Thematic Fuzzy Clusters, Les Réseaux Neuromimétiques et leurs Applications (Neuronîmes), 1992.

[8]: Frakes W. B., Stemming Algorithms, pages 131–160. Frakes W B, Baeza-Yates R (eds).

[9]: Porter M.F., An algorithm for suffix stripping. Program 14, 1980.

[10]: Craven Timothy C., HTML Tags as Extraction Cues for Web Page Description Construction, The University of Western Ontario, London, Ontario, Canada.

[11] : G. Zipf, « Human Behaviour and the Principal of Least ». Addison-Wesley, 1949.

[12] : Karen SAVAGNAT, « Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés ». Thèse doctorat, Université Paul Sabatier de Toulouse, 2005.

[13]: S.E.Robertson, and S.Walker, "Some somple effective approximations to the 2-Poisson model for probabilistic weighted retrieval", In proceedings of SIGIR 1994.

[14]: A.Singhal, G.Salton, M.Mitra, and C.Buckley, "Document length normalization", Information Processing and management, 32(5): 619-633, 1996.

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

[16]: Broder Audrei, A Toxonomy of Web Search, IBM Research.

[17], [25] : FELLAG Samia, « Recherche d'Information dans des documents semi-structurés XML ». Thèse Magister, Université Mouloud Mammeri de Tizi-Ouzou, 2006.

[18] : Abderrazak MKADMI : « Cours XML »,
http://h2ptm.hymedia.univ.paris8.fr/mkadmi/cours XML.htm

[19] : Ed Tittel : « XML », Edi Science 2003, Schaum's, Traduit de l'americain par Patric Fabre.

[20] : M. Abolhassani and N. Fuhr, 2002.

[21] M. Abolhassani and N. Fuhr, 2002 : K.Sauvagnat, 2005.

[22] ; [23] : Luk, 2002.

[24] : Torsten Grust "Accelerating XPath Location Steps",In M.J.Franklin, B.Moon, and A.Ailmaki,editors,Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data,Madison,Winconsin,USA,pages 109-120. ACM, 2002.

[26] :K. Sauvgnat. Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés. Thèse de doctorat. Toulouse: Université Paul Sabatier; 2005.

[27] :D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. IEEE Data Engineering Bulletin, 22(3) : pages 27–34, 1999.

[28]: Hatano K, Kinutani H, Yoshikawa M, Uemura S. Information Retrieval System for XML Documents. In: Database and Expert Systems Applications. Berlin: Springer; 2002. p. 5-33

[29] :C. J. V. Rijisbergen. Information Retrieval. Dep of computer Science, University of Glasgow, 1979.

[30]:N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst., 15(1):32–66, 1997.

[31] :N. Fuhr, M. Lalmas, and S. Malik. Inex 2003 workshop proceedings. 2003

[32] K. Sauvagnat and M. Boughanem. Xfirm : A flexible information retrieval model for indexing and searching xml documents. European Conference on Information Retrieval, ECIR, pages 17–18, 2004.

[33] K. Sauvagnat, M. Boughanem, and C. Chrisment. Answering content and structure-based queries on xml documents using relevance propagation. Information Systems, 31(7):621–635, 2006

[34]: K. P. Sauvagnat, M. Boughanem, and C. Chrisment. Answering content-and- structure-based queries on xml documents using relevance propagation. Information Systems, 31:621–635,2006

[35]:K. P. Sauvagnat, L. Hlaoua, and M. Boughanem. Xfirm at inex 2005: adhoc and relevance feedback tracks. INitiative for the Evaluation of XML Retrieval (INEX), Dagstuhl, Germany, 28/11/05-30/11/05, pages 88–103, 2005.

[36] : L. Hlaoua, M. Torjmen, K. P. Sauvagnat, and M. Boughanem. Xfirm at inex 2006. ad-hoc, relevance feedback and multimedia tracks. International Workshop of the Initiative for theEvaluation of XML Retrieval (INEX), Dagstuhl, Allemagne, 18/12/2006-20/12/2006, pages 373–386, 2007.

[37]: Wolff JE, Florke H, Cremers AB. Searching and browsing collections. In: IEEE advances in digital libraries. 2000. p. 141-150

[38] : Torsten Grust, "Accelerating XPath Location Steps", In M. J. Franklin, B. Moon, and A. Ailamaki, editors, Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, pages 109-120. ACM, 2002.

[39] : Keddache & Kecili, 2012 « Intégration de formules de pondération au système de recherche dans les documents XML : XFIRM »
Réalisé par : Melle KEDDACHE Fadhila et Melle KECILI Amel 2011/2012.

[40] : Géry Mathias & *al.*, Lyon/Saint-Etienne BM25t : une extension de BM25 pour la Recherche d'Information ciblée Mathias Géry, Christine Largeron, Franck Thollard : Lyon, Saint-Étienne, France (Laboratoire Hubert Curien).

[41]: N. Fuhr, M. Lalmas, and S. Malik. Inex 2003 workshop proceedings. 2003.

[42] : INEX 2007 Evaluation Measures (Draft).

[43] : INEX repport, Repport on INEX 2012.

Annexe A

XML

1. Origine du XML :

XML (*Extensible Markup Language ou langage de balisage extensible*) est le dernier mot à la mode sur Internet ; C'est un sous-ensemble de SGML, possédant les mêmes objectifs (le balisage de tout type de données), débarrassé toute fois de toute complexité superflue. Il a été conçu pour être totalement compatible avec SGML, et tente de se servir du meilleur du HTML et du SGML.

C'est en 1996 que J. Bosak, T Bray, J. Clark et M. Sperberg-McQueen entre autre commencèrent un travail sur une version simplifiée de SGML combinant la flexibilité et la puissance de SGML avec l'acceptation montante de HTML, et en février 1998, les principes de base de XML sont rédigés par le groupe de travail XML formé par le W3C (World Wide Web Constorium) sous l'égide de J. Bosak de Sun Microsystems (avec des spécialistes du SGML Working Group).

2. Syntaxe de base de XML 1.0 :

XML offre une syntaxe de balisage, qui permet de définir un langage , ce qui en fait un outil capable de produire des documents primaires dont la structure conceptuelle d'un document se présente comme une suite d'éléments enchâssés les uns dans les autres.

En conséquence, le document devient un ensemble d'informations structurées, un conteneur d'information plus qu'une unité de base, ce qui permet aux outils de gestion de se concentrer sur le contenu du document plus que sur le document lui-même, de mettre en relation le texte lui-même et sa structure, puis d'exploiter ces relations.

Les règles grammaticales de la spécification XML 1.0, sont présentées comme suit :

a. **Toute balise d'ouverture doit posséder une balise de fermeture correspondante** :

Les balises de fermetures sont obligatoires en XML, et elles doivent correspondre exactement aux balises d'ouverture.

b. **Les balises ne peuvent se chevaucher** :

XML étant strictement hiérarchisé sous forme d'arbre, les éléments (balise ouvrante et balise fermante) doivent être imbriqués, il faut veiller à fermer les éléments enfants avant de fermer leur parent.

c. Un document XML ne peut posséder qu'un seul élément racine :

L'élément racine, est l'élément le plus haut niveau du document XML, dont tous les autres éléments sont des enfants ou des descendants. Un document XML doit posséder un et un seul élément racine, même dépourvu de contenu.

d. Les noms des éléments doivent répondre à la convention de nommage XML :

XML est généreux en ce domaine de création de noms d'éléments, il n'existe aucun mot réservé à éviter en XML, contrairement à la plupart des langages de programmation, ce qui offre une grande souplesse.

Voici cependant quelques règles à suivre :

- Un nom doit commencer par une lettre ou par le caractère « _ », mais ni par un chiffre ni par un autre signe de ponctuation. Après le premier caractère, les nombres sont autorisés ainsi que les caractères « _ » et «. ».
- Un nom ne peut comporter le caractère d'espace au début, il peut cependant y avoir un espace après le nom.
- Un nom ne peut comporter le caractère « : ». Ce caractère est autorisé, mais la spécification XML stipule qu'il est réservé avec l'utilisation des espaces de noms.

e. XML est sensible à la casse.

Un autre point important à garder à l'esprit est que les balises XML sont sensibles à la casse. Cela signifie que <prenom> est différent de <PRENOM>, tous deux se distinguent de <Prenom>.

3. Les objectifs de conception de XML :

Les objectifs de conception soulignés par la communauté de la toile, chapoté par le W3C, pour mener la technologie XML à son succès sont en dix commandements suivants :

- XML devrait pouvoir être utilisé sans difficulté sur Internet ;
- XML devrait soutenir une grande variété d'applications ;
- XML devra être compatible avec SGML ;
- Il devrait être facile d'écrire des programmes traitant les documents XML ;
- Le nombre de fonctions optionnelles de XML doit être réduite au minimum;
- Les documents XML devraient être lisibles par l'homme et raisonnablement clairs ;

- La conception de XML devrait être préparée rapidement ;
- La conception de XML sera formelle et concise ;
- Il devrait être facile de créer des documents XML ;
- La concision dans le balisage de XML est de peu d'importance.

4. Les constituants d'une DTD

4.1. Types de données :

Les deux types les plus fréquemment utilisés dans les DTD sont : PCDATA et CDATA :

· PCDATA signifie Parsed Character Data (ou chaîne de caractères parsée). C'est le texte contenu dans un élément. Dans l'élément suivant :
<element>texte</element>, « texte » est du type PCDATA.

· CDATA signifie Character Data ou chaîne de caractères. Il s'agit de chaînes de caractères qui ne seront pas analysées lors de la validation. On les utilise dans les attributs : par exemple dans l'élément : <element attribut='hello'>, hello est du type CDATA et sera donc traité comme une chaîne de caractère.

· ID et IDREF : permettent de lier différentes parties d'un document XML. Nous y reviendrons.

4.2. Les éléments :

Les éléments XML sont définis dans une balise <!ELEMENT>.

La syntaxe de base est <!ELEMENT nom (contenu)>. Un élément peut être vide, contenir une chaîne de caractères, ou contenir des sous éléments.

TYPE	DTD	ELEMENT
Elément vide	<!ELEMENT elt EMPTY	<elt/>
Elément contenant du texte	<ElEMENT elt (#PCDATA)>	<elt>texte </elt>
Elément avec sous éléments	<!ELEMENT elt (s1, s2)> <!ELEMENT s1 EMPTY> <!ELEMENT s2 EMPTY>	<elt> <s1/> <s2/> </elt>
Elément avec contenu variabl	<!ELEMENT elt (#PCDATA|s1)> <!ELEMENT s1 EMPTY>	<elt>texte</elt> ou <elt><s1/></elt>
Elément à contenu non défini	<!ELEMENT elt ANY> <!ELEMENT s1 EMPTY>	<elt>texte</elt> ou <elt><s1/></elt>

On peut déclarer la cardinalité des sous éléments :

<!ELEMENT élément (s1)> signifie que le sous élément s1 peut avoir une et une seule occurrence.

<!ELEMENT élément (s1*)> signifie que le sous élément s1 peut avoir 0 à n occurrences.

<!ELEMENT élément (s1?)> signifie que le sous élément s1 peut avoir 0 ou 1 occurrence.

<!ELEMENT élément (s1+)> signifie que le sous élément s1 doit avoir au moins 1 occurrence.

4.3. Les Attributs :

Les attributs sont tous décrits dans une balise <!ATTLIST>. La syntaxe de base est :

<!ATTLIST

nom-element nom-attribut1 type valeur-par-defaut

nom-element nom-attribut2 type valeur-par-defaut

...>

Il existe de nombreux types possibles pour l'attribut. On utilise généralement CDATA (Character Data, ou chaîne de caractères), ou une énumération de valeurs possibles :

DTD	XML
< !ELEMENT elt EMPTY> <!ATTLIST elt attribut CDATA 'defaut'>	<elt attribut='valeur '/>. Et par défaut <elt attribut='defaut '/>
<!ELEMENT elt EMPTY> <!ATTLIST elt attribut ('v1'|'v2') 'v1'> 2 valeur possible : 'v1' et 'v2'	<elt attribut='v2'/> ou par defaut <elt attribut='v1'/>
<!ELEMENT elt EMPTY> <!ATTLIST elt att1 CDATA 'd1' elt att2 CDATA 'd2'>	<elt att1='a' att2='b'/> ou par défaut: <elt att1='d1' att2='d2'/>

La valeur par défaut permet de spécifier la valeur que prendra l'attribut s'il n'est pas renseigné par le constructeur du document XML qui respecte cette DTD. Cette valeur permet aussi de préciser des options :

TYPE	DTD	XML
Attribut obligatoire	< !ELEMENT elt EMPTY> <!ATTLIST elt attribut CDATA #REQUIRED >	<elt attribut='a'/>
Attribut non obligatoire	< !ELEMENT elt EMPTY> <!ATTLIST elt attribut CDATA #IMPLIED >	<elt attribut='a'/> <elt/>
Attribut à valeur fixe	< !ELEMENT elt EMPTY> <!ATTLIST elt attribut CDATA #FIXED 'a' >	<elt attribut='a'/> :ok <elt attribut='b'/> : non

4.4. Les entités :

Les entités peuvent être considérées comme des variables. Elles sont déclarées et non mutables, et peuvent être utilisées dans tout document XML respectant cette DTD. Elles peuvent être internes ou externes (c'est à dire faire référence à un autre document par le biais d'une URI). Leur déclaration est contenue dans l'élément <!ENTITY>.

TYPE	DTD	XML
Interne	<!ENTITY nom 'valeur'>	<elt> &nom; </elt> .Qui donne : <elt> valeur </elt>
Externe	<!ENTITY nom SYSTEM 'www.w3c.org'>	<elt> &nom; </elt>.Qui donne <elt>www.w3c.org</elt>

4.5. Les identifiants :

Un attribut de type ID permet d'attribuer un identifiant à un élément d'un document XML.Cet élément peut être ensuite référencé dans une autre partie du même document en utilisant un attribut de type IDREF. Les identifiants permettent donc de lier différentes parties d'un document XML.Par exemple, la DTD suivante décrit une grammaire permettant de définir des personnes et leurs relations de parenté :

<!ELEMENT personne (#PCDATA)>

<!ELEMENT parent (fils*)>

<!ELEMENT fils EMPTY>

<!ATTLIST personne identifiant ID #REQUIRED>

<!ATTLIST parent identifiant IDREF #REQUIRED>

<!ATTLIST fils identifiant IDREF #REQUIRED>

Exemple de document XML respectant cette DTD. On y définit deux personnes (John et Salton) à qui on attribue un identifiant (respectivement « A » et « B »), et on précise que A est parent de B:

<personne identifiant='A'>John</personne>

<personne identifiant='B'>Salton</personne>

<parent identifiant='A'>

  <fils identifiant='B'/>

</parent>

Du point de vue de l'échange de données, l'utilisation des DTD n'est pas pleinement satisfaisante pour des raisons suivantes :

- Une DTD, ne permet pas de préciser le type des données (entier, date, valeur booléenne, etc.) que représente le contenu textuel d'un élément ou attribut.
- Une DTD ne permet pas la description fine des cardinalités que celles permises par les indicateurs ?, +, et *.
- La DTD est exprimée dans un formalisme non XML, ce qui signifie qu'elle est non extensible, ce qui rend sa gestion plus délicate.

Conscient de ces grandes limitations, la recommandation **XML-Schema**, adopté par le W3C en 2001, permet de pallier ces inconvénients.

5. Les constituants d'un XML-Schéma

5.1. La Déclaration :

L'élément <xsd:schema> permet de déclarer un document XML-Schema. Son attribut targetNamespace permet de préciser l'espace de nommage de ce type de documents. L'attribut elementFormDefault précise si les documents XML respectant cette grammaire doivent référer à cet espace de nommage.

```
<xsd:schema
        xmlns:xsd='http://www.w3.org/2000/10/XMLSchema'
   targetNamespace='http://www.annuaire.org'
           xmlns='http://www.annuaire.org'
 elementFormDefault='qualified'/>
```

5.2. Les types de données :

XML-Schema définit plus de 40 types de données, et fournit un mécanisme de définition de types de données complexes. Parmi ces types définis par XML-Schema on retrouve string, integer, date, year, CDATA, float, double, binary, ENTITIES, byte, etc.

Pour définir ses propres types de données, il est possible de créer un type de données totalement nouveau, de restreindre ou d'étendre un type de données existant. Une déclaration de type est visible par tous les descendants du nœud dans lequel il a été déclaré.

Par exemple, un type qui a été déclaré sous le nœud <xsd:schema> sera par voie de conséquence visible dans tout le document.

Pour créer un nouveau type de données, il faut utiliser la balise <xsd:complexType>. Créons par exemple un type de données 'entree' :

<entree>
<nom>string</nom>
<telephone>decimal</telephone> </entree>
On utilise la syntaxe suivante :

```
<xsd:complexType name='typeEntree'>
  <xsd:sequence>
     <xsd:element name='nom' type='xsd:string'/>
     <xsd:element name='telephone' type='xsd:decimal'/>
  </xsd:sequence>
</xsd:complexType>
```

Pour l'utiliser dans une déclaration d'élément on utilise la syntaxe suivante :

<xsd:element name='entree' type='typeEntree'>

1. Restriction de type existant : utilisation de <xsd:simpleType>

Le type de données string comprend six attributs optionnels : pattern, enumeration, length, minLength, maxlength, whitespace. Si on désire définir un type de string représentant un choix (oui/non) :

```
<xsd:simpleType name='choixOuiNon'>
   <xsd:restriction base='xsd:string'>
      <xsd:enumeration value='oui'/>
      <xsd:enumeration value='non'/>
   </xsd:restriction>
</xsd:simpleType>
```

Pour utiliser ce type de données dans une déclaration d'élément, on utilise l'attribut type :

<xsd:element name='choix' type='choixOuiNon'/>

Une instance de cet élément : <choix>oui</choix>

Il est possible d'empêcher la restriction d'un type lors de sa déclaration :

<xsd:complexType name='choix' final='restriction'>

2. Extension d'un type de données existant *:* utilisation de <xsd:complexContent>

Il est aussi possible de créer des types dérivés, ajoutant des éléments aux structures déjà créées. Reprenons le type 'entree' défini précédemment. Il est possible de créer un type dérivé entreeAvecAdresse contenant en plus du nom et du numéro de téléphone une adresse sous la forme d'un string :

```
<xsd:complexType name='entreeAvecAdresse'>
 <xsd:complexContent>
   <xsd:extension base='entree' >
```

```
      <xsd:sequence>
        <xsd:element name=’adresse’ type=’xsd:string’/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Il est possible d’empêcher la dérivation d’un type lors de sa déclaration (type final):

<xsd:complexType name=’choix’ final=’extension’>

Pour empêcher l’extension et la dérivation d’un type :

<xsd:complexType name=’choix’ final=’#all’>

5.3. Les éléments :

La définition d’éléments se fait dans une balise <xsd:element>. Il existe deux possibilités pour définir un élément :

- Définir un type de données et l’utiliser dans la définition de l’élément. Par exemple, on modélise l’élément <entree> d’un annuaire :

XML :

```
<entree>
   <nom>Harry Cover</nom>
   <telephone>0102030405</telephone>
</entree>
```

Modélisation en XML-Schema :

```
<xsd:complexType name=’entrée’>
      <xsd:sequence>
        <xsd:element name=’nom’ type=’xsd:string’/>
        <xsd:element name=’telephone’ type=’xsd:decimal’/>
      </xsd:sequence>
</xsd:complexType>
```

L’avantage de cette solution est que le type « entree » est réutilisable dans le reste du document XML-Schema où la déclaration du type est visible.

- Définir le type de données à l’intérieur de l’élément.

```
<xsd:element name=’entrée’>
    <xsd:complexType>
      <xsd:sequence>
         <xsd:element name=’nom’  type=’xsd:string’/>
         <xsd:element name=’telephone’  type=’xsd:decimal’/>
      </xsd:sequence>
```

```
    </xsd:complexType>
</xsd:element>
```

XML-Schema permet de gérer la cardinalité des éléments beaucoup mieux que les DTD, mais de manière plus verbeuse. L'élément <xsd:element> possède deux attributs optionnels minoccurs et maxoccurs qui précisent respectivement le nombre minimal et maximal d'occurrences d'un élément. Par exemple, on désire modéliser que l'élément <element> possède un certain nombre de sous éléments <s1/> :

XML-Schema	**Equivalent en DTD**
<xsd:element name='element'> <xsd:complexType> <xsd:sequence> <xsd:element name='s1' minoccurs='0' maxoccurs='unbounded'/> </xsd:sequence> </xsd:complexType> </xsd:element>	<!ELEMENT element (s1*)>
<xsd:element name='element'> <xsd:complexType> <xsd:sequence> <xsd:element name='s1' minoccurs='0' maxoccurs='1'/> </xsd:sequence> </xsd:complexType> </xsd:element>	<!ELEMENT element (s1?)>
<xsd:element name='element'> <xsd:complexType> <xsd:sequence> <xsd:element name='s1' minoccurs='1' maxoccurs='unbounded'/> </xsd:sequence> </xsd:complexType> </xsd:element>	<!ELEMENT element (s1+)>
<xsd:element name='element'> <xsd:complexType> <xsd:sequence> <xsd:element name='s1' minoccurs='1' maxoccurs='11'/> </xsd:sequence> </xsd:complexType> </xsd:element>	<!ELEMENT element (s1, s1?, s1? ,s1?, s1?, s1?, s1?, s1?, s1? ,s1?, s1?)>

XML-Schema permet aussi de modéliser qu'un élément a un contenu mixte :

Type	XML-Schema	Equivalent en DTD
Elément avec contenu variable	<xsd:element name='element'> <xsd:complexType> <xsd:choice> <xsd:element name='s1'/> <xsd:element name='s2'/> </xsd:choice> </xsd:complexType> </xsd:element>	<!ELEMENT elt (s1|s2)>
Elément à contenu non défini	<xsd:element name='element'> <xsd:complexType> <xsd:sequence> <xsd:any/> </xsd:sequence> </xsd:complexType> </xsd:element>	<!ELEMENT elt ANY>

5.4. Les attributs :

La définition d'attributs associés à un élément se fait dans un élément <xsd:attribute>

Chaque élément <xsd:attribute> possède les attributs suivants :

- name : nom de l'attribut.
- type : type de l'attribut. Par exemple xsd:string, xsd:boolean,...
- use : permet de préciser si l'attribut est obligatoire ou optionnel. Valeurs possibles : required (obligatoire), implied (optionnel), fixed (valeur fixe) ou default.
- value : valeur par défaut de l'attribut.

Exemple: <element att='hello' at2='true'/>. On souhaite modéliser cet élément dans un XML-Schema. L'attribut att est optionnel et a comme valeur par défaut « a ».

L'attribut at2 est obligatoire et a comme valeur par défaut « true ».

```
<xsd:element name='element'>
  <xsd:complexType>
    <xsd:attribute name='att' type='xsd:string' use='implied' value='a'/>
    <xsd:attribute name='at2' type='xsd:boolean' use='required' value='true'/>
  </xsd:complexType>
</xsd:element>
```

Il est aussi possible de définir des attributs plus complexes, en utilisant les possibilités qu'offre XML-Schema pour la définition de types.

```
<xsd:attribute name='choix' use='required'>
   <xsd:simpleType>
     <xsd:restriction base='xsd:string'>
       <xsd:enumeration value='oui'/>
     <xsd:enumeration value='non'/>
    <xsd:enumeration value='ne sait pas'/>
  </xsd:restriction>
 </xsd:simpleType>
</xsd:attribute>
```

L'attribut choix peut alors uniquement prendre 3 valeurs : « oui », « non », et « ne sait pas ».

Notons qu'il est possible de définir des attributs globaux pouvant être utilisés dans la déclaration de plusieurs éléments :

Déclaration de l'attribut :

```
<xsd:attribute name='attribut'>
  <xsd:complexType>
     <xsd:any/>
  </xsd:complexType>
</xsd:attribute>
```

On utilise ensuite cet attribut dans la déclaration d'un élément :

```
<xsd:element name='element'>
   <xsd:complexType>
     <xsd:attribute ref='attribut' use='required'/>
   </xsd:complexType>
</xsd:element>
```

5.5. Les identifiants :

Comme pour les DTD, il est possible d'attribuer un identifiant aux éléments (attribut de type ID) pour les référencer dans la suite du document XML (attribut de type IDREF).

Exemple de définition d'élément possédant un attribut identifiant :

```
<xsd:element name='element1'>
  <xsd:complexType>
   <xsd:attribute name='id' type='ID' use='required'/>
  </xsd:complexType>
</xsd:element>
```

Exemple de définition d'élément référençant un autre élément possédant un identifiant :

```
<xsd:element name='element2'>
  <xsd:complexType>
    <xsd:attribute name='ref' type='IDREF' use='required'/>
  </xsd:complexType>
</xsd:element>
```

Exemple dans un document XML, où un élément <element2> fait référence à un élément <element1>:

```
<element1 id='A'/>
<element2 ref='A'/>
```

6. Les espaces de noms (name space) :

Les espaces de noms est équivalent à des répertoires d'un système d'exploitation, permet de structurer les données pour l'amélioration de la gestion. Ils nous offrent la possibilité de fusionner les jeux de balises dans un même document pour éviter le conflit des noms. Un espace de noms est spécifie avec l'attribut « xmlns » dont sa valeur est un URL qui référence en général le document de définition des balises de l'espace.

Un espace de nom est valide pour l'élément dont lequel il est déclaré, et tous les éléments descendants.

Un espace de nom est déclaré dans l'élément racine.

Exemple :

```
< ?xml vertion="1.0" encoding="iso-8859-1" ?>
<organisation xmlns:per="personne.org">
<xmlns:ent="entreprise.org">
<per :personne>
<per :nom>paul</per :nom>
<per : prenom>martin</per :prenom>
</per : personne>
<ent :entreprise>
<ent :nom>filip</ent :nom>
<ent :adresse>rue des quatre frère, alger </ent :adresse>
</ent :entreprise>
</organisation>
```

</xml>

Avec l'utilisation des deux espaces de noms « personne, entreprise » on n'aura pas de conflit entre le nom de la personne et le nom de l'entreprise.

7. Le langage XSL :

XSL (extensible styling language ou langage de type extensible) est un langage de feuille de style du XML, permettant l‟affichage des documents XML compréhensible par l‟utilisateur. Il est un sous ensemble de XML dont les noms appartiennent à l‟espace des noms http//www.w3.org/1999/XSL/transform.

Sa spécification est divisée en deux :

- **XSLT (XSL transformation) :** publié comme une recommandation de W3C en 16 novembre 1999, elle permet le transfert des documents XML possédant un certain format, en un autre prenant un autre format (en particulier un document HTML).
- **XSLFO (XSL formating objects) :** publié comme une recommandation de W3C en 15 octobre 2001, utilisé pour la description de pages imprimables en haute qualité typographique (se charge de la pagination, les marges, les notes de bas de pages, le positionnement des objets sur la page, les polices de caractères, l'affichage de tableaux, …).

Une feuille de style XSL commence par le prologue :

< ?xml version="1.0" ?>

<xsl : stylesheet version="1.0" xmlns : xsl="http://www.w3.org/1999/xsl/transform">

Et devra se terminer par </xsl : stylesheet>

Prenons un exemple simple : un fichier de noms d‟animaux qui peut avoir la structure suivante :

<animaux>

<chien> Medor </chien>

<chat> Mandarine </chat>

<chien> Brutus </chien>

<chat> Clementine> </chat>

<oiseau> Titi </oiseau>

<chat> Prunelle </chat>

<animaux>

Si on le soumet à un processeur XSL avec le fichier XSL suivant :

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="animaux">
<AnimauxTries><chats>
<xsl:for-each select="chat">
<xsl:sort select="."/>
<xsl:value-of select="."/>
</xsl:for-each>
</chats>
<chiens>
<xsl:for-each select="chien">
<xsl:sort select="."/>
<xsl:value-of select="."/>
</xsl:for-each>
</chiens>
<oiseaux>
<xsl:for-each select="oiseau">
<xsl:sort select="."/>
<xsl:value-of select="."/>
</xsl:for-each>
</oiseaux>
</AnimauxTries>
</xsl:template>
</xsl:stylesheet>
```

On obtient le nouveau fichier XML suivant, ou les animaux sont classés par race, leurs noms étant triés par ordre alphabétique

```
<?xml version="1.0" encoding="UTF-8"?>
<AnimauxTries>
<chats> Clementine Mandarine Prunelle </chats>
<chiens> Brutus Medor </chiens>
```

<oiseaux> Titi </oiseaux>

</AnimauxTries>

8. Les langages de requêtes

Les langages de requêtes permettent de donner un moyen d'expression du besoin d'un utilisateur. L'aspect structurel des documents XML permet d'étendre les possibilités d'expression, ce qui a donné naissance à une nouvelle forme d'interrogation.

Nous allons présenter Dans ce qui suit l'un de ses langages : le langage XPath.

8.1. Le langage de requête XPath:

XPath est un langage permettant d'adresser des parties ou des sous arbres d'un document XML, afin de pouvoir en extraire avec précision les informations requises. XPath 1.0 est devenu une recommandation du W3C depuis le 16 novembre 1999.

C'est un langage permettant de désigner un ensemble de nœuds dans l'arbre DOM d'un document XML.

- ***Les expressions XPath :***

Une expression XPath exprime **un ensemble de chemins** dans un arbre XML. Ces chemins débutent à partir du **nœud contexte** dans lequel s'évalue l'expression XPath.

Remarque : une même expression peut renvoyer des résultats différents selon le contexte dans lequel elle est évaluée.

Le résultat d'une expression XPath est un ensemble de nœuds : c'est l'ensemble de nœuds qui se trouvent au bout des chemins spécifiés par l'expression XPath.

- ***Les chemins XPath :***

Un chemin XPath est ***une suite d'étapes*** séparées par le caractère « / ».

On distingue deux types de chemins :

1. **Absolu :** le nœud contexte pour la premiére étape est la racine du document.
2. **Relatif :** le nœud contexte pour la premiére étape est un nœud dans le document (pas forcément la racine).

- ***L'étape Xpath:***

Une étape XPath comporte trois composantes, elle est de la forme suivante:

axe::filtre[prédicat 1] [prédicat 2]… [prédicat n]

- **L'axe:** désigne le sens de parcours des noeuds.
- **Le filtre:** désigne les noeuds qui seront retenus d'prés leur type.
- **Le(s) prédicat(s):** désignent les conditions que doivent satisfaire les noeuds pour être retenus.

a) ***Les axes XPath:***

Parmi les axes XPath nous avons:

- **Child :** désigne l'ensemble de tous les fils du noeud contexte à l'exception des noeuds de type attribut qui ne sont pas considérés vraiment comme des fils de leur noeud parent. On peut donc trouver sur cet axe des noeuds de type élément, mais aussi texte, commentaire, etc...

Remarque: la spécification de cet axe est souvent omise car il s'agit de l'axe par défaut. Donc on pourra écrire **/A/B/C** au lieu de **/child::A/child::B/child::C**

- **Parent :** désigne un seul nœud qui est le noeud parent du noeud courant.
- **Descendant** : désigne tous les noueds qui sont descendants du nœud contexte à l'exception des attributs.

Remarque: le noeud contexte lui-même ne fait pas patie des descendants.

- **Ancestor :** tous les ancestres du noeud contextuel (axe réciproque de l'axe descendant).
- **Self :** il ne cmporte qu'un seul noeud qui est le noeud contexte lui-même.
- **Attribute:** désigne les attributs du noeud contexte. C'est le seul axe à s'appliquer à l'arbre des attributs, il a donc toujours un parent de type *element* et est constitué d'un ensemble non ordonné de noeuds de type *attribut* identifiés par leurs noms.Cet axe sera toujours la derniére étape d'une expression XPath.

Remarque: l'écriture attribute::noeud est souvent abrégée par @noeud.

- **Preceding-sibling:** désigne tous les frères gauches du noeud contexte.
- **Following-sibling:** désigne tous les frères droits du noeud contexte.

- **Preciding:** désigne les noeuds précédant le noeud contexte dans l'ordre de parcours du document.
- **Following:** désigne les noeuds suivant le noeud contexte dans l'ordre de parcours du document.
- **Descendant-or-self:** désigne tous les descendants du noeud contexte et le noeud contexte lui-même.
- **Ancestor-or-self:** désigne tous les ancêtres du noeud contexte et le noeud contexte lui-même

b) Les prédicats:

Un prédicat est une expression booléenne mise entre crochets permettant de sélectionner un ou plusieurs noeuds en fonction du critère de sélection utilisé.

Syntaxe: **[fonction(noeud) = valeur]**

XPath utilise donc un certain nombre de fonctions dans ces expressions, parmi ces fonctions on a:

- **Name():** permet de renvoyer le nom d'un noeud.

- **Text():** renvoie le contenu PCDATA d'un nœud.

- **Position()** : permet d'obtenir la position d'un nœud dans un ensemble de nœuds, selon l'ordre du document.

❖ *Evaluation de l'expession XPath:*

A chaque étape, l'axe définit un ensemble de noeuds a partir du noeud contexte, puis le filtre elemine (filtre) certains de ces noeuds et en conserve d'autres, enfin les prédicats un vont s'appliquer individuellement à chacun des noeudsretenus par l'axe et le filtre pour en éléminer certains. Lensemble des noeuds qui restent formeront le résultat de l'évaluation de l'expression XPath.

Exemple :

Soit le document XML suivant :

```
<?xml version='1.0' standalone='yes'?>
<stock>
   <produit>
        <nom>livre</nom>
        <prix monnaie="dinars">100</prix>
   </produit>
   <produit>
        <nom>cd</nom>
        <prix monnaie="euros">5</prix>
   </produit>
</stock>
```

Pour retrouver les noms des produits qui existent en stock, on peut utiliser l'une des expressions XPath suivantes:

- /stock/produit/nom
- /child::stock/child::produit/child::nom