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

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

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

DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**

Filière : **Informatique**

Spécialité : **Système Informatique**

Présenté par

FERRAH Chahines

Thème

Implémentation et évaluation d'une méthode de sélection de termes d'expansion basée sur la position des termes dans les documents pertinents.

Mémoire soutenu publiquement le 13/10/ 2016 devant le jury composé de :

Président : Mr SADOU S

Encadreur : Mr HAMMACHE A

Co-Encadreur: Mr

Examinateur : Mr YACINE Y

Examinateur : Mr RADJA H

Remerciements

Je remercie tout d'abord Dieu qui m'a donné la foi et le courage pour accomplir ce projet.

je tiens aussi à exprimer ma reconnaissance et profonde gratitude à mon promoteur, Mr Hammache pour m'avoir encadré durant cette année, pour sa forte présence et sa disponibilité, pour son exigence scientifique et ses précieux orientations méthodologiques, pour son encouragement et sa patience.

Que les membres du jury trouvent ici nos plus vifs remerciements pour avoir accepter d'honorer par leur jugement mon travail. Aussi, j'adresse mes remerciements à tous nos enseignants de l'UMMTO pour m'avoir appris le goût de l'effort et du travail.

Un grand merci aussi à toute personne qui m'aidée et contribuée de près ou de loin à la réalisation de ce projet.

Dédicace

J'ai le plaisir de dédie ce travail à :

- *Deux âmes les plus chères au monde mon père et ma mère qui m'ont toujours soutenu et accompagné durant ma vie. C'est grâce à leurs encouragements que je suis arrivé à ce stade, que Dieu vos offre la paix et le bonheur et vous garde en santé.*
- *Mon frère : Lamara.*
- *Mes chères sœurs : Celia , Dehbia*

- *A tous mes ami(e)s, notamment ceux du département d'Informatique.*
- *Toute la promotion 2016.*

FERRAH Chahines

Liste des tableaux

Listes des figures

Sommaire

Chapitre II : Expansion de requêtes

Chapitre III : Evaluation et Expérimentation

INTRODUCTION GENERALE

INTRODUCTION GENERALE

L'objectif fondamental de la RI consiste à mettre en œuvre un mécanisme d'appariement entre une requête utilisateur et les documents d'une base documentaire, afin de restituer l'information pertinente, l'accès à l'information peut être effectué à travers d'un système de recherche d'information (SRI).

Il est souvent difficile, pour l'utilisateur, de formuler son besoin exact en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent pas toujours. Retrouver des informations pertinentes en utilisant la requête initiale seule de l'utilisateur est très difficile, et ce à cause du volume croissant des bases documentaires. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence du système, une étape de reformulation de la requête est souvent utilisée. La requête initiale est traitée comme un essai pour retrouver de l'information. Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'objectif de retrouver plus de documents pertinents.

L'approche proposée dans ce mémoire rentre dans le cadre de l'expansion de requête. Elle consiste à intégrer le facteur de la position des termes dans les documents pertinents, dans l'objectif de classer les termes d'expansion. Cette intégration est réalisée en utilisant le modèle KL Divergence. L'implémentation et l'évaluation de l'approche est réalisée sous la plateforme de RI Terrier (détaillée en annexe).

L'organisation retenue pour la présentation de notre travail et le domaine dans lequel il s'inscrit, s'articule en trois chapitres :

Le premier chapitre présente les concepts et notions de base du domaine de la recherche d'information (RI) d'une manière générale. Ensuite décrit en détail les différents modèles de la RI existants, ainsi que l'évaluation des systèmes de recherche d'information (SRI).

Le second chapitre traite l'expansion de requêtes et ses différentes techniques, ainsi que les stratégies qui sont développées dans les différents modèles.

Le troisième chapitre présente notre approche, les outils utilisés pour son implémentation, ainsi que les résultats d'évaluation obtenus sur la collection TREC (AP88).

CHAPITRE I

La Recherche d’information

I.1. Introduction

La recherche d'information (RI) est une discipline ancienne, elle remonte aux années 50 de siècle dernier. Sa problématique peut être vue comme la satisfaction d'un besoin en information d'un utilisateur, qui est exprimé par une requête, sur un ensemble de documents appelé collection ou corpus **[1] [2].**

Les systèmes de recherche d'information (SRI) permettent d'automatiser la tâche de la RI. L'évaluation de tels systèmes apparaît comme une nécessité. Cette évaluation s'articule autour de la notion de pertinence.

Dans ce chapitre, nous décrivons la recherche d'information en commençant dans la première section à savoir, la définition de la recherche d'information(RI), puis nous présentons les concepts de base de la RI, nous décrivons également son processus global connu sous le nom du processus en **U** en donnant l'utilité et l'importance de ses fonctions. Dans la deuxième section, nous décrivons les modèles de RI à savoir, le modèle booléen, le modèle vectoriel et le modèle probabiliste. Enfin, nous présentons les mesures d'évaluation et les collections de test utilisées pour évaluer les SRI.

I.2. Définition de la recherche d'information (RI)

La recherche d'information selon **Salton [3] :** est l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre au besoin de l'utilisateur exprimé via une requête.

D'après **l'Afnor [4] :** la 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.

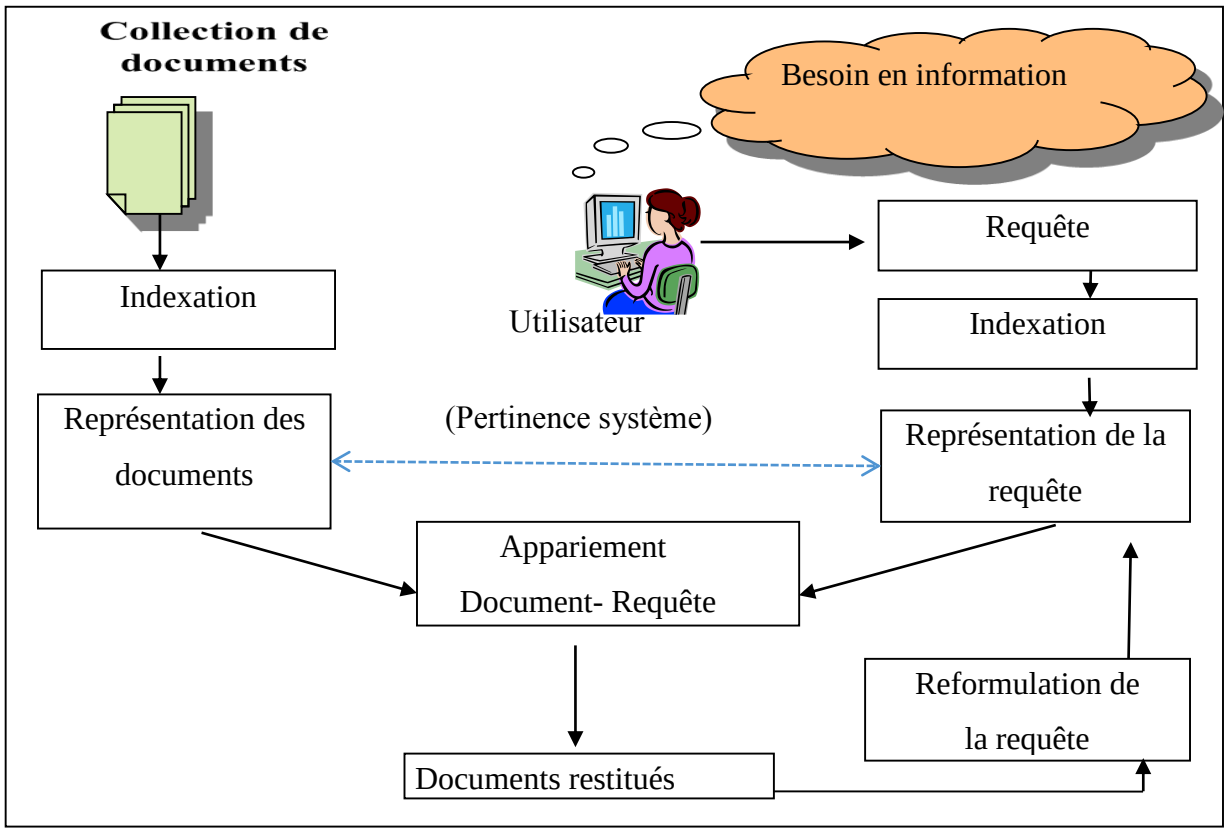
La recherche d'information est donc une discipline scientifique qui a pour objectif la production des solutions permettant de sélectionner à partir d'un corpus d'informations, celles qui sont dites pertinentes pour un utilisateur ayant exprimé une requête.

I.3. Définition d'un Système de Recherche d'Information(SRI)

Un système de Recherche d'Information (SRI) est l'interface entre l'utilisateur qui exprime son besoin par une requête et une grande collection de documents. Il se charge du traitement de ces derniers pour retourner à l'utilisateur ceux qui correspondent le mieux à sa requête. Pour ce faire, il intègre un ensemble de fonctions qui sont :

Le stockage, l'organisation, la sélection d'information répondant aux besoins des utilisateurs et enfin la restitution des informations jugées pertinentes.

Un SRI peut être également défini comme un logiciel qui a pour but de sélectionner des informations pertinentes répondant aux besoins des utilisateurs, exprimés sous forme de requêtes **[5]**. Le fonctionnement général d'un SRI est donné au travers du processus de recherche communément appelé processus en **U [6],** présenté en **Figure I.1**. Ce processus fait ressortir trois mécanismes de base : le processus d'indexation (quelques fois dit processus d'interprétation pour les requêtes), le processus de recherche et le processus de reformulation des requêtes.



FigureI.1 : Architecture générale d'un Système de Recherche d'information (SRI)

I.4. Concepts de base de la recherche d'information

Un système de recherche d'information a pour rôle de sélectionner à partir d'une collection, des documents qui peuvent intéresser l'utilisateur, c'est-à-dire ceux qui peuvent être pertinents à son besoin en information. Cette définition fait apparaître trois éléments de base qu'il convient de préciser : document et collection, besoin en information et requête, pertinence.

I.4.1. Les éléments de base

I.4.1.1. Document et collection de documents

Un document est une unité d'information qui peut constituer une réponse à un besoin en information/requête d'un utilisateur. Un document peut être un texte, un morceau de texte, une image, une bande vidéo, etc... **[7].** La collection de documents (ou fond documentaire) constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents. Dans le cas général et pour un souci d'optimalité, la base constitue des représentations simplifiées mais suffisantes pour ces documents. Ces représentations sont étudiées de telles sortes que la gestion (ajout suppression d'un document) ou l'interrogation (recherche) de la base se font dans les meilleures conditions de coût.

I.4.1.2. Besoin en information et Requête

La requête est une expression approximative du besoin en information de l'utilisateur. Ce dernier est une expression mentale des informations que l'utilisateur recherche. Pour une recherche documentaire, l'utilisateur doit soumettre une requête au moteur de recherche dont laquelle il spécifie les mots clés représentant sont besoin en information. Trois types de besoins utilisateur ont été définis **:**

a) **Besoin vérificatif :** l'utilisateur recherche une donnée particulière, et sait souvent comment y accéder (par exemple chercher un document ayant une adresse connue sur le web). Un besoin de ce type est dit stable car il ne change pas au cours de la recherche.
b) **Besoin thématique connu :** l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et domaine connus. Un besoin de ce type peut être stable ou variable ; il est possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche.
c) **Besoin thématique inconnu :** l'utilisateur cherche de nouveaux concepts ou de nouvelles relations hors des sujets et domaines qui lui sont familiers. Un besoin de ce type est intrinsèquement variable et est toujours exprimé de façon incomplète [**8**].

I.4.1.3. Pertinence

Selon les premières définitions de la pertinence, la pertinence est la correspondance entre un document et une requête ; une mesure de l'informativité du document à la requête. La notion de pertinence est le critère primaire pour l'évaluation des systèmes de recherche d'informations. Le processus de jugement de la pertinence de l'information est basé sur le degré de similitude de la représentation de la requête avec le contenu du document retrouvé par le système **[7].**

Essentiellement deux types de pertinence sont définis, la pertinence système et la pertinence utilisateur :

a) **La Pertinence système :** est déterministe, objective et elle est définie à travers les modèles de RI. Elle est souvent traduite par un score évaluant l'adéquation du contenu des documents vis-à-vis de celui de la requête **[9].**

Selon Denos**[10]**, la pertinence système est l'ensemble des principes qui sous-tendent la fonction de correspondance dans un système de recherche d'information.

b) **La Pertinence utilisateur :** cette pertinence quant à elle est liée à la perception de l'utilisateur sur l'information renvoyée par le système. Elle est subjective, deux utilisateurs peuvent juger différemment un même document renvoyé pour une même requête. Elle peut évoluer dans le temps d'une recherche. Par exemple, une information jugée non pertinente à l'instant *t* pour une requête peut être jugée pertinente à t+1car la connaissance de l'utilisateur sur le sujet a évolué **[11] [12] [13]**.

I.4.2. Processus de recherche d'information

Un processus de recherche d'information comme le montre la **Figure I.1** passe par trois étapes principales :

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

I.4.2.1. Indexation des documents et des requêtes

Dans un processus de recherche d'information, la requête et les documents du corpus sont difficilement exploitables à l'état brut. Une représentation de ces documents ainsi que la requête s'avère indispensable. Afin d'aboutir à ces représentations des techniques et des

modèles sont mis en œuvre. Ces techniques permettent de décrire les documents et la requête par un ensemble de descripteurs. Ce processus de représentation est appelé le processus d'indexation ou tout simplement l'indexation. L'indexation consiste à analyser les documents et la requête afin d'extraire un ensemble de descripteurs **[14] [1]**. Ces descripteurs sont des unités textuelles significatives dans le document (mots clés), il existe trois types d'indexations :

a)Manuelle : chaque document du corpus est examiné par un documentaliste spécialisé dans le domaine afin d'identifier les descripteurs **[15] [16]**. A la fin de cette étape d'analyse des documents, une liste de descripteurs est établie. Ce type d'indexation est fiable et donne des bons résultats. Par conséquent, les documents retournés par le SRI en réponse à une requête utilisateur sont précis **[17]**. Mais, avec l'augmentation incessante de nombre de documents, l'indexation manuelle s'avère difficile. En effet, l'indexation est une tâche lourde et coûteuse en temps.

b) Automatique : c'est un processus complètement automatisé qui se charge d'extraire les termes caractéristiques du document. L'intérêt d'une telle approche réside dans sa capacité à traiter les textes nettement plus rapide que l'approche précédente, et de ce fait, elle est particulièrement adaptée aux corpus volumineux **[21] [22] [23]**. L'indexation automatique classique est fondée sur l'analyse des documents en vue de l'extraction des termes (mots-clés simples ou composés) représentatifs de leur contenu informationnel. Elle repose sur les étapes suivantes : l'extraction des termes d'indexation, élimination des mots vides, la normalisation (lemmatisation ou radicalisation), le choix des descripteurs, et enfin la création de l'index. Nous détaillons ces différentes étapes ci-dessous.

c) Semi-automatique : appelée aussi indexation supervisée, est une combinaison des deux approches d'indexation précédentes **[18] [19] [20]**. Dans ce cas, les indexeurs utilisent un vocabulaire contrôlé sous forme de thésaurus ou de base terminologique. Le choix final des termes d'indexation à partir du vocabulaire fourni, est laissé ainsi à l'indexeur humain (généralement spécialiste du domaine).

I.4.2.1.1. L'analyse lexicale(Tokenisation)

C'est l'opération de transformer un document textuel en un ensemble de termes ou unité lexicale en reconnaissant les espaces de séparation des mots, des caractères spéciaux, des chiffres, les ponctuations ou une liste de séparateurs **[24]**. Il y a des cas particuliers où les mots sont composés ou contiennent des séparateurs par exemple : aujourd'hui, pomme de terre, etc.

I.4.2.1.2.L'élimination des mots vides

C'est la suppression des mots de fort fréquence ou à faible contenu informatif ces mots appelé mots vides (tels que les pronoms personnels, les articles, les mots de liaison, ou les prépositions), pour ne garder que les termes importants. Plusieurs techniques peuvent être mises en œuvre parmi celles-ci, l'utilisation des stops liste ou des anti-dictionnaires (anti-lexiques) et l'utilisation des mesures statistiques. Le traitement lié à un anti-dictionnaire est très simple si un mot apparait dans l'anti-dictionnaire et dans le texte a indexé, il n'est pas considéré comme un index, il peut cependant induire des effets de silence (par exemple, en éliminant le mot **a** de vitamine **a [25].**

I.4.2.1.3.La normalisation (lemmatisation ou radicalisation)

C'est une étape qui a pour but de regrouper les différentes variantes d'un mot à sa forme canonique ou lemme. Par exemple : il peut être utile de retrouver des documents contenant les mots « transmission », « transmis », « transmet », « transmettra », «transmetteur» à partir d'une requête comportant le mot « transmettre ». Pour cela il est possible d'éliminer les différences non significatives et de garder la partie commune. Sur l'exemple, les mots ont la même racine (le lemme) et une terminaison différente. Pour les verbes conjugués il suffit de rendre le verbe à l'infinitif pour les conjugués et le singulier pour les noms. Des fois le passage à la forme canonique supprime le sens du mot comme le verbe portera et le nom porte sera indexé de la même façon. Plusieurs stratégies de normalisation sont utilisées : La table de consultation (dictionnaire), l'élimination des affixes (l'algorithme de Porter), la troncature, les variétés de successeurs, la méthode des n-grammes, l'utilisation des étiqueteurs grammaticaux **[26].**

I.4.2.1.4. Création de l'index

Au terme du processus d'indexation, un ensemble de structure de données sont créés. Ces dernières permettent un accès efficace à la représentation des documents. Le fichier inverse est la structure de données la plus utilisée, il enregistre pour chaque descripteur les identificateurs des documents qui le contiennent et sa fréquence dans chacun de ces documents. Généralement, les structures de données sont compressées avant d'être enregistrées sur le disque, ce qui permet de réduire la taille de l'index. Parmi les méthodes de compression utilisées on peut citer la méthode Elias Gamma qui opère au niveau bit requérant ainsi beaucoup d'opérations pour la compression et la décompression. D'autres méthodes plus efficaces, opérant au niveau octet ont été proposées **[8]**.

I.4.2.2. Appariement document-requête

Dans un SRI, une fois les documents transformés (processus d'indexation), l'utilisateur peut interroger la collection de documents à travers la formulation d'une requête qui exprime son besoin informationnel, cette requête est représentée sous une forme interne compréhensible par le système (processus d'indexation) **[27]**. Le processus d'appariement est le résultat de la comparaison entre les documents et la requête qui donne une liste de documents, ces documents sont ordonnés selon un score de similarité qui est donné par une fonction nommée **Retrieval Status Value**. Elle est notée **RSV** (*d, q*), où "*d* "est un document et "*q* "est une requête **[28]**.Ce score s'appuie sur des approches mathématiques On en distingue : le modèle booléen ou ensembliste, le modèle vectoriel, le modèle probabiliste. Dans ce qui suit (section I.5) nous détaillons les modèles de la RI.

I.4.2.3. Reformulation de la requête

La qualité d'un SRI dépend de sa capacité à retrouver des documents pertinents pour l'utilisateur, pour garantir cet objectif on reformule généralement les requêtes. La reformulation de requête consiste à créer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur, et par conséquent le SRI augmentera son rappel et sa précision [**27**]**.** Concrètement la reformulation de la requête consiste à ajouter, supprimer et/ou pondérer les termes (enrichir et affiner la requête utilisateur), reliés à ceux de la requête initiale. La reformulation est donc liée au choix des termes par le système (index) et par l'utilisateur (requête). On peut distinguer trois types de reformulation **[28]**.

I.4.2.3.1. La reformulation manuelle

Ce type reformulation est généralement utilisé dans les systèmes de recherche booléens. Donc pour reformuler la requête initiale pour faire une nouvelle recherche des documents pertinents on a besoin d'utiliser un vocabulaire contrôlé (thésaurus ou classification), pour trouver les bons termes pour compléter la requête **[27]**.

I.4.2.3.2. La reformulation semi-automatique (interactive)

C'est la reformulation la plus populaire qui appelée aussi réinjection de la pertinence (relevance feedback). Ce mode d'indexation combine entre l'indexation manuelle et l'indexation automatique, toute fois le choix final des index reste tout de même aux indexeurs humains.

I.4.2.3.3. La reformulation automatique

Appelée aussi le pseudo-réinjection de la pertinence, dans ce cas la reformulation est faite d'une manière automatique sans l'intervention de l'utilisateur, soit par l'utilisation d'une ressource externe qui peut être un thesaurus qui regroupe plusieurs informations de type linguistique (équivalence, association, hiérarchie) et statistique (pondération des termes), une ontologie, etc. Soit par l'exploitation des "n premiers" documents renvoyés par le système comme pertinent en réponse à la requête initiale (blind feedback)**.** Le problème avec la reformulation automatique est l'estimation des « bons »termes qui peuvent conduire effectivement à une amélioration du processus de recherche car l'introduction des termes inappropriés peut entraîner un silence ou au contraire augmenter le bruit **[28]**.

I.5. Les modèles de recherche d'information

Un modèle détermine le comportement clé d'un SRI, on distingue trois principaux modèles qui sont : les modèles booléens (ensemblistes), vectoriels et probabilistes. Nous allons décrire dans ce qui suit chacun d'eux.

I.5.1. Le modèle booléen

Les premiers SRI développés sont basés sur le modèle booléen, même aujourd'hui beaucoup de systèmes commerciaux (moteur de recherche) utilisent le modèle booléen. Cela est dû à la simplicité et à la rapidité de sa mise en œuvre. Ce modèle est basé sur la théorie des ensembles et l'algèbre de Boole. Les documents et les requêtes sont représentés par des ensembles de mots clés pris de leurs contenus. Dans ce modèle, la représentation de la requête doit utiliser l'un des opérateurs logiques : ET ($\wedge$), OU ($\vee$), et NON ($\neg$). Chaque document *"d*" est représenté par un ensemble de termes non pondérés, l'index des documents sera constitué par la conjonction des termes t_i.

L'appariement (RSV) entre une requête et un document est un appariement exact autrement dit si un document implique au sens logique la requête alors le document est

pertinent, sinon il est considéré non pertinent. La correspondance entre document et requête est déterminée comme suit :

$$\text{RSV}(q_i, d_j) = \begin{cases} 1 \text{ si } q_i \in d_j \\ 0 \text{ sinon} \end{cases} \quad \text{(I.1)}$$

Malgré la large utilisation de ce modèle, il présente un certains nombres de problèmes :

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

Afin de remédier à ces problèmes, une extension du modèle booléen a été proposée : le modèle booléen étendu **[8]**. Ce modèle a été proposé par **Salton [G.E.A]**, son objectif est de tenir compte de la pondération des termes dans le corpus. Cela permet de résoudre les problèmes cités précédemment en ordonnant les documents retournés par le SRI selon leurs similarités à la requête. Dans ce modèle, la requête demeure une expression booléenne classique, tandis que les termes d'un document sont maintenant pondérés **[29]**.

I.5.2. Le Modèle vectoriel

Ce modèle est introduit au début des années 70 par Gérard Salton et son équipe dans le système de recherche d'information SMART. En se basant sur une formalisation géométrique, le modèle vectoriel représente les documents et les requêtes sous la forme d'un vecteur de termes à "t"dimensions telles que $"t"$est le nombre de termes d'indexation définis par le système. Un document d'une collection $"d_i"$est représenté par un vecteur d_i (W_{i1}, W_{i2},…W_{ij},…W_{in}).

Avec wd_{ij} : la pondération associée au terme d'indexation t_j dans le document d_i.
La requête $"q"$est représentée suivant le même formalisme q (w_{q1}, w_{q2},…, w_{qj},…w_{qn}).
Avec wq_i : poids du terme $"t_i"$dans la requête $"q"$. Ce poids peut être soit une forme de *tf*idf*, soit un poids attribué manuellement par l'utilisateur.

Dans la littérature plusieurs schémas de pondération ont été proposés. La majorité de ces schémas prennent en compte la pondération locale et la pondération globale [**8**].

La pondération locale permet de mesurer l'importance du terme dans le document. Elle prend en compte les informations locales du terme qui ne dépendent que du document .elle correspond en général à une fonction de la fréquence d'occurrence du terme dans le document (noté ***tf*** pour (term frequency), exprimée ainsi :

$$tf_{ij} = 1 + log\left(f(t_i, d_j)\right) \quad \text{(I.2)}$$

Où : f (t_i, d_j) est la fréquence du terme t_i dans le document d_j.

Quant à la pondération globale, elle prend en compte les informations concernant le terme de la collection. Un poids plus important doit être assigné aux termes qui apparaissent moins fréquemment dans la collection. Car les termes qui apparaissent dans de nombreux documents de la collection ne permettent pas de distinguer les documents pertinents des documents non pertinents (i.e. peu utile pour la discrimination). Un facteur de pondération globale est alors introduit. Ce facteur nommé *idf* (*i*nverted *d*ocument *f*requency), dépend d'une manière inverse de la fréquence en document du terme et exprimé comme suit :

$$Idf(ti) = log(\frac{N}{ni}) \quad \text{(I.3)}$$

Où :

n_i : est la fréquence en document du terme considéré, et N est le nombre total de documents dans la collection.

Les fonctions de pondération combinant la pondération locale et globale sont référencées sous le nom de la mesure *tf*idf*. Cette mesure donne une bonne approximation de l'importance du terme dans les collections de documents de taille homogène. Cependant, un facteur important est ignoré, la taille du document. En effet, la mesure *(tf*idf*) ainsi définie favorise les documents longs, car ils ont tendance à répéter le même terme, ce qui accroit leur fréquence par conséquent augmentent la similarité de ces document vis-à-vis de la requête. Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération comme facteur de normalisation [**8**].

L'appariement document requête dans le modèle vectoriel, consiste à trouver les vecteurs documents qui s'approchent le plus de vecteur de la requête. Cet appariement est

obtenu par l'évaluation de la distance entre les deux vecteurs. Plusieurs mesures de similarité ont été définies, dont les plus courantes sont décrites dans le **TableauI.1** ci-dessous.

Mesures	**Formules**
Le Produit scalaire	$\mathbf{RSV}(d_j, q_k) = \sum_{i=1}^{t} (w_{i,j} * w_{i,k})$
La mesure de cosinus	$\mathbf{RSV}(d_j, q_k) = \frac{\sum_{i=1}^{N}(Wd_{ij} * Wq_{ik})}{\sqrt{\sum_{i=1}^{N} W d^2_{ij} * \sum_{ij=1}^{N} W q^2_{ik}}}$
La mesure de Dice	$\mathbf{RSV}(d_j, q_k) = 2 * \frac{\sum_{i=1}^{N}(Wd_{ij} * Wq_{ik})}{\sum_{i=1}^{N} (W d^2_{ij} + Wq^2_{ik})}$
La mesure de Jaccard	$\mathbf{RSV}(d_j, q_k) = \frac{\sum_{i=1}^{N}(Wd_{ij} * Wq_{ik})}{\sum_{i=1}^{N} W d^2_{ij} + \sum_{i=1}^{N} W q^2_{ik} - \sum_{i=1}^{N}(Wd_{ij} * Wq_{ik})}$

Tableau I .1 Les mesures de similarités utilisées dans le modèle vectoriel

Le modèle vectoriel caractérisé par sa prise en compte du poids des termes dans les documents par conséquent, il permet de retrouver des documents qui répondent partiellement à une requête de plus, ce modèle offre un moyen facile pour classer les résultats d'une recherche, qui est basée sur la similarité potentielle entre document et requête.

L'inconvénient majeur de modèle vectoriel est qu'il repose sur l'hypothèse de l'indépendance des termes d'indexation, or ces termes dans les documents sont souvent sémantiquement liés. Plusieurs variantes du modèle vectoriel ont été proposées, pour remédier à cette limitation .C'est-à-dire prendre en compte la dépendance entre termes d'indexation parmi elles, on trouve, le modèle vectoriel généralisé (Generalized Vector Space Model), le modèle LSI (Latent Semantic Indexing) et le modèle connexionniste [**8**].

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

I.5.3. Le modèle probabiliste

Le modèle probabiliste aborde le problème de la recherché d'information dans un cadre probabiliste. Le premier modèle probabiliste a été proposé par Maron et Kuhns [**22]** au début des années 1960.

I.5.3.1. Le modèle probabiliste de base

Le modèle probabiliste est fondé sur la théorie des probabilistes, il trie les documents selon leur probabilité de pertinence vis-à-vis d'une requête.la fonction de classement (tri) de ce modèle est exprimée ainsi [**8**] :

$$RSV(q,d) = \frac{P(Per|q,d_i)}{P(NPer|q,\ d_i)} \tag{I.4}$$

L'idée de base de cette fonction est de retrouver les documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents à la requête.
Où : $P(Per|q,d_i)$ et $P(NPer|q,\ d_i)$: la probabilité qu'un document d_i soit pertinent (per)vis-à-vis de la requête q (respectivement non pertinent$(NPer)$).
Plus la valeur de RSV (q, d) est importante, plus le document n'obtient un meilleur classement.
En appliquant le théorème de Bayes pour les deux probabilités, on obtient :

$$P(Per|q,d_i) = \frac{P(Per|q).P(d_i|Per,q)}{P(d_i)} \tag{I.5}$$

$$P(NPer|q,d_i)\ = \frac{P(NPer|q).P(d_i|NPer,q)}{P(d_i)} \tag{I.6}$$

Où :

$P(d_i)$: est la probabilité de choisir le documentd_i , on considère qu'elle est constante.

$P(d_i|Per,q)$: indique la probabilité que d_i fait partie des documents pertinents pour la requête q.

$P(d_i|NPer,q)$: indique la probabilité que d_i fait partie des documents non pertinents pour la requête q.

$P(Per|q)$ et $P(NPer|q)$ indiquent respectivement la probabilité de pertinence et de non pertinence d'un document quelconque (avec$P(Per|q) + P(NPer|q) = 1$) qui sont fixes.
Après remplacement dans la fonction de tri, on aura la formule suivante :

$$RSV(q,d) = \frac{P(d_i|Per,q)}{P(d_i|NPer,q)} \quad \text{(I.7)}$$

Si on suppose que les termes d'indexation sont indépendants, alors on peut estimer les deux probabilités ainsi :

$$P(d_i|Per,q) = \prod_{t_j \in d_i} P(t_j|Per,q) \times \prod_{t_j \notin d_i} 1 - P(t_j|Per,q) \quad \text{(I.8)}$$

$$P(d_i|NPer,q) = \prod_{t_j \in d_i} P(t_j|NPer,q) \times \prod_{t_j \notin d_i} 1 - P(t_j|NPer,q) \quad \text{(I.9)}$$

Où :

$P(t_j|Per,q)$: indique la probabilité d'apparition du terme t_j sachant que le document appartient à l'ensemble des documents pertinents.

$P(t_j|NPer,q)$: indique la probabilité d'apparition du terme t_j sachant que le document appartient à l'ensemble des documents non pertinents.

En posant$p_i = P(t_j|Per,q)$, $q_i = P(t_j|NPer,q)$ et $p_i = q_i$ pour les termes qui n'apparaissent pas dans la requête, et après simplification, le calcul du score de correspondance entre un document et une requête peut être exprimé ainsi :

$$RSV(d_i,q) = \sum_{t_i \in q} \log\left[\frac{p_i(1-q_i)}{q_i(1-p_i)}\right] \quad \text{(I.10)}$$

Ce modèle présente plusieurs avantages, entre autres :

- De meilleurs résultats sont retournés en utilisant les modèles probabilistes.
- Les documents retrouvés sont classés selon l'ordre de pertinence décroissant.

Cependant, il présente les inconvénients suivants :

- Difficulté de calcul de probabilités conditionnelles.
- Les jugements de pertinence étant propres à un utilisateur particulier, l'apprentissage effectué à partir de ses données sont valables que pour l'utilisateur.

I.5.3.2. Le modèle de langue

Le modèle de langue est emprunté de la linguistique informatique. Son objectif est de capter les régularités linguistiques d'une langue, en observant la distribution des mots et leur succession. Le modèle de langue désigne une fonction de probabilité qui assigne à chaque séquence de mots une probabilité. Les SRI utilisant les modèles de langue, suivent une approche différente des autres modèles. En effet, dans la plupart des modèles, on cherche à comparer une représentation de la requête de l'utilisateur avec une représentation du document recherché pour évaluer la pertinence.

Dans ce modèle, on part de l'observation que l'utilisateur crée la requête à partir d'une représentation hypothétique qu'il se fait du document recherché. La requête est donc générée à partir des documents voulus. La pertinence d'un document est ainsi estimée en calculant la probabilité que la requête utilisateur soit inférée à celui-ci. Elle est formalisée comme suit :

$$score(q_i, d) = \prod_{i=1}^{n} P(q_i|d) \quad \text{(I.11)}$$

Avec :q_i sont les termes de la requête.

Ce modèle a l'inconvénient des probabilités nulles, en effet, un n-grammes qui n'apparaît pas dans le document a une probabilité nulle, donc toute séquence qui le contient a une probabilité nulle **[30]**.

I.6. L'évaluation des systèmes de recherché d'information

L'évaluation de performance des **SRI** (modèles et méthodes) a toujours été un centre d'intérêt très important dans le domaine de la **RI**, tout cela pour atteindre un SRI idéal qui retourne tous les documents pertinents de la collection qui répond au besoin en information de l'utilisateur, et rejette les documents non pertinents. La qualité des **SRI** est reliée avec les réponses du système qu'a été souhaitée par l'utilisateur. Plus les réponses du système correspondent à celles que l'utilisateur espère, meilleur est le système **[31]**.

I.6.1.Les mesure d'évaluations

Pour évaluer la performance d'un SRI, deux principales mesures statistiques sont utilisées : la précision et le rappel. La précision détermine l'aptitude d'un SRI à rejeter les documents non

Pertinents vis à vis à d'une requête utilisateur. Le rappel exprime la capacité d'un SRI à sélectionner tous les documents pertinents vis à vis de cette requête. Ces deux mesures sont données par les formules suivantes **[32]** :

$$\text{Précision} = \frac{|\text{DPR}|}{|\text{DR}|} \tag{I.12}$$

$$\text{Rappel} = \frac{|\text{DPR}|}{|\text{DP}|} \tag{I.13}$$

Avec :

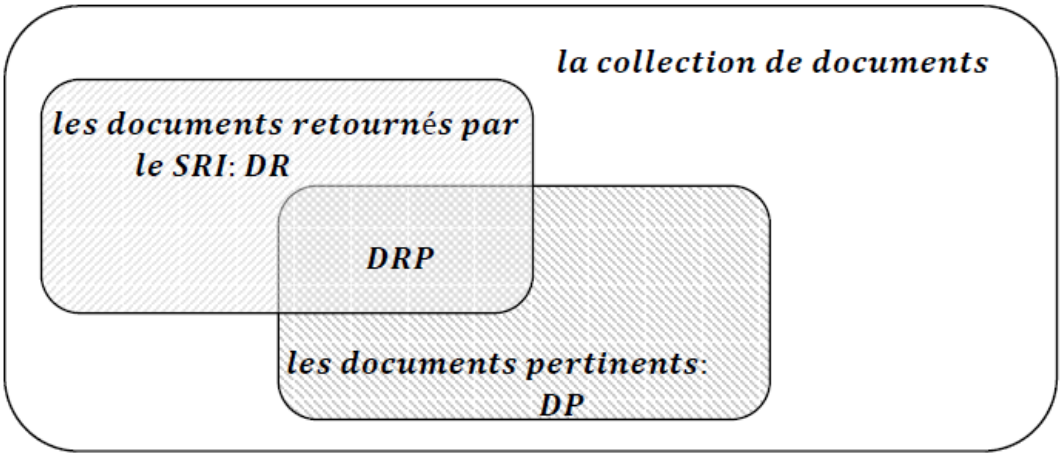
DRP : l'ensemble des documents pertinents vis-à-vis de la requête et qui sont retournés par le SRI,

| DR| : l'ensemble des documents retournés par le SRI,

|DP| : l'ensemble des documents de la collection qui sont pertinents vis-à-vis à la requête,

|DRP|, |DR|, |DP| : la cardinalité des ensembles considérés (le nombre des documents).

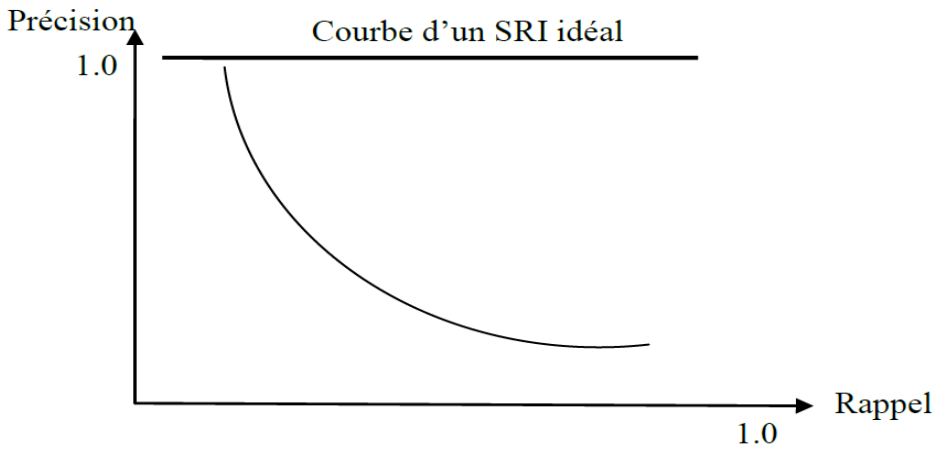
La **Figure I.2** illustre la répartition des documents suite à une interrogation utilisateur. A partir de ces ensembles de documents les deux mesures précision et rappel sont calculées.



FigureI.2 : Répartition des documents d'une collection suite à une interrogation**.**

L'idéal serait d'avoir une précision et un rappel égaux à 1, signifiant que tous les documents pertinents ont été retrouvés et qu'aucun document non pertinent n'a été sélectionné. En pratique cela n'a jamais été atteint. En effet comme indiqué dans la **FigureI.3**, ces deux mesures évoluent en sens inverse. Intuitivement si nous augmentons le rappel pour retrouver plus de documents pertinents, la précision du système diminuera engendrant ainsi trop de

documents non pertinents sélectionnés. Inversement, une plus grande précision risque de rejeter des documents pertinents diminuant ainsi le rappel.



FigureI.3 : courbe d'évaluation de la pertinence d'un SRI rappel/précision

Les mesures complémentaires du rappel et de la *précision* sont respectivement : le silence et le *bruit*.

On parle du silence lorsque les documents pertinents ne sont pas sélectionnés par le système de recherche, alors qu'ils existent dans la collection. Les causes du silence peuvent être multiples .on peut citer par exemple l'imprécision du langage de la requête formués par l'utilisateur.

Le silence est calculé selon la formule suivante :

$$\text{Silence} = 1 - R = \frac{|DPNS|}{|DP|} \tag{I.14}$$

On parle du bruit lorsque des documents non pertinents sont proposés à l'utilisateur par le système de recherche. Ceci est causé par l'ambiguïté des termes de la requête. Le bruit est définit par la formule suivante :

$$\text{Bruit} = 1 - P = \frac{|DNPS|}{|DPS \cup DNPS|} \tag{I.15}$$

En général, la performance d'un SRI est évaluée sur un ensemble de requêtes et non pas sur une seule requête. Pour permettre cette évaluation sur toutes les requêtes de l'ensemble on utilise la précision moyenne appelée MAP (Mean Average Precision).

- **La précision moyenne (Mean Average Precision)**

L'évaluation des performances d'un modèle de RI ou un système de recherche SRI est réalisé sur la base des jugements des résultats de la recherche pour un ensemble de la requête et non par ceux obtenus pour une seule requête. Pour se faire des précisions sont calculées à chaque document pertinent retrouvé en réponse à chaque requête donnée q. La moyenne de ces précisions représente la précision moyenne (AP : AveragePrecision) de la requête q, qui désigne sa performance. La MAP (Mean Average Precision) ou moyenne arithmétique des précisions moyennes AP_q de toutes les requêtes q, permet de mesurer les performances de modèle de RI utilisé. Cette mesure est formellement définie par **[33]** :

$$MAP = \frac{1}{|Q|}\sum_{q=1}^{Q} \mathrm{AP}q \qquad \text{(I.16)}$$

Où :

Q : est l'ensemble des requêtes utilisées dans la recherche.

$|Q|$: est le nombre de requêtes.

AP_q : est la moyenne des précisions calculées aux différents rangs k où un document pertinent est restitué par le système pour une requête q.

I.6.2. Collections de tests

Une collection de test est composée d'un ensemble de documents (corpus documentaire), d'un ensemble de requêtes (topics) et des jugements de pertinence (l'ensemble de documents jugé pertinent pour chaque requête) établis manuellement par des assesseurs humains.

Il existe plusieurs collection de test tels que : la collection CACM, CISI, qui sont utilisées comme moyen d'évaluation des SRI qui se base sur le principe d'envoyer les requêtes test au SRI, puis comparer les réponses retournées par ce dernier avec les réponses idéales. L'écart entre ces deux réponses permet de mesurer la performance d'un système **[33]**.

I.6.3. Les compagnes d'évaluations

Sont des systèmes d'évaluations basées sur les collections de test. Il existe plusieurs compagnes d'évaluation tel que : TREC, CLEF (*Cross-language evaluation forum*) pour l'évaluation des systèmes multilingue et la recherche inter-langues spécifique pour les langues européennes, INEX (Initiative for the Evaluation of XML Retrieval) pour l'évaluation des SRI

dans des collections semi-structurés XML. Ces derniers sont considérés comme étant les compagnes d'évaluation les plus expérimentées. Dans ce qui suit, nous présentons la compagne d'évaluation TREC **[32]**.

I.6.3.1. La compagne TREC

Les compagnes TREC sont les plus utilisée dans le domaine d'évaluation des systèmes de recherche d'information. TREC (Text Retrieval Conference) est un projet lancé en 1992 par le NIST(le National Institute of Standards and Technology) et le soutien financier de DARPA (l'Advanced Research and Development Activity) Centre du Département de la Défense des États-Unis. TREC organise des conférences annuelles, afin d'encourager les travaux dans le domaine de la recherche d'information en fournissant l'infrastructure nécessaire (collection des tests) et en proposant des méthodologies d'évaluation des SRI.

L'évaluation d'un SRI dans le cadre de la compagne d'évaluation TREC consiste à examiner les 1000 premiers documents qu'il retourne pour une requête (thème). Une précision moyenne est calculée qui permet d'obtenir une mesure de la performance globale du système.

Au départ, les recherches dans TREC étaient centrées sur deux tâches principales : la tâche de filtrage et la tâche ad hoc. Par la suite, d'autres tâches ont apparu, tel que : la tâche Terabyte (des évaluations sur des très grands corpus), la tâche QA (Question-Réponse), la tâche multimédia et la tâche web.

I.7.Conclusion

Dans ce chapitre nous avons passé en revue les principaux concepts de la recherche d'information (RI). Nous avons, particulièrement, introduit des éléments de base, telles que le besoin en information et requête, le document et la pertinence. Nous avons aussi décrit les processus de base de la recherche d'information (RI), à savoir l'indexation, correspondance, et la reformulation de la requête. Ensuite, nous avons étudié les différents modèles de la RI. Enfin, nous avons présenté quelques mesures d'évaluations et corpus de tests pour l'évaluation des systèmes de recherche d'information (SRI).Une notion très importante doit être encore définie et étudiée, la reformulation de requête qui est la clé de voute de la plupart des SRI. Dans le chapitre suivant nous parlerons en détails sur l'expansion de requête à savoir les étapes de l'expansion de requête et les différents paramètres de performances d'expansion et l'importance du poids dans le processus de recherche.

CHAPITRE II

Expansion de Requête

II.1. Introduction

Les performances d'un SRI, mesurées en général par la double mesure rappel-précision, dépendent d'une part de l'efficacité du modèle de recherche mis en œuvre pour l'appariement entre requête-documents, et d'autre part des requêtes formulées par l'utilisateur.
La requête initiale de l'utilisateur est souvent représentée par une liste de termes très réduite. Cette liste manque souvent de termes intéressants pouvant exprimer effectivement le besoin en information de l'utilisateur. Ceci a plusieurs raisons, la plus importante vient de la diversité du vocabulaire de la collection de documents. Pour pallier à ce problème le système de recherche d'information proposent des techniques, appelées expansion de la requête, pour affiner et améliorer automatiquement la requête initiale de l'utilisateur.

Ce chapitre est consacré à traiter l'expansion de la requête. Particulièrement nous énumérons le critère de classification des méthodes de reformulation, puis nous citons le processus d'expansion automatique de la requête, et enfin nous allons présenter les paramètres de performances de cette technique.

II.2.Définition

L'expansion de requête est une méthode conçue pour améliorer le processus de recherche d'information. Elle est utilisée dans le domaine de conception de SRI adaptatifs aux besoins des utilisateurs. C'est un processus qui permet de construire une nouvelle requête plus adéquate à la recherche d'information dans l'environnement du SRI, comparée à cette formulée initialement par l'utilisateur. Son principe consiste à élargir le champ de recherche pour une requête et à contenir plus de termes reliés. Plusieurs autres définitions ont été attribuées à l'expansion de requête, Abberley **[34]** définit l'expansion de requêtes comme un moyen qui permet de reformuler les requêtes et d'améliorer le processus de recherche d'informations. Elle est considérée selon **[31]** comme un processus qui a pour but de préciser et d'éclaircir les résultats en permettant à l'utilisateur de modifier sa requête afin d'améliorer la pertinence de ses résultats.

II.3. Classification des techniques de reformulation de requêtes

Les techniques de reformulation de requêtes peuvent être classées selon trois principaux paramètres **[35].**

II.3.1. classification selon la source des termes utilisés

Différentes sources de données sont utilisées pour reformuler la requête initiale. Elles peuvent être **[36]** : des ressources externes, telles que les ontologies, les thésaurus et la relation de cooccurrence entre terme dans la collection. Les méthodes basées sur ces ressources sont dites méthode globales, ou des documents résultats de la première recherche ; les méthodes basées sur ces ressources sont dites méthodes locales. Ces méthodes sont également connues sous le nom de réinjection de pertinence.

La réinjection de pertinence a montré son efficacités avec différents modèles de la RI et a affiché de meilleures résultats que les méthodes globales **[37].**

II.3.2. classification selon la méthode de sélection des termes

Une méthode permet de sélectionner les termes à ajouter à la requête initiale, plusieurs méthodes existent, on trouve : la relation de concurrence, les mesures d'information, les techniques de classification, etc.

II.3.3. classification selon le rôle de l'utilisateur

La reformulation de la requête peut être réalisée par l'utilisateur (manuelle), ou par le système (automatique) comme elle peut être réalisée conjointement par l'utilisateur et le système, dans ce cas elle est dite semi-automatique ou interactive **[38].** Cependant, le rôle de l'utilisateur dans ce processus peut être actif passif, il est actif dans la reformulation manuelle et reformulation interactive de la requête, et passif dans la reformulation automatique de la requête.

II.4. Le processus d'expansion automatique de la requête

Le processus d'expansion de requête contient quatre principales étapes présentées dans la figure **(Figure II.1)** : traitement de données, génération et classement des termes, sélection des termes et reformulation de la requête. La requête initial de l'utilisateur et la source de données, sont considérées comme l'entrée du processus et la requête reformulé comme sortie.

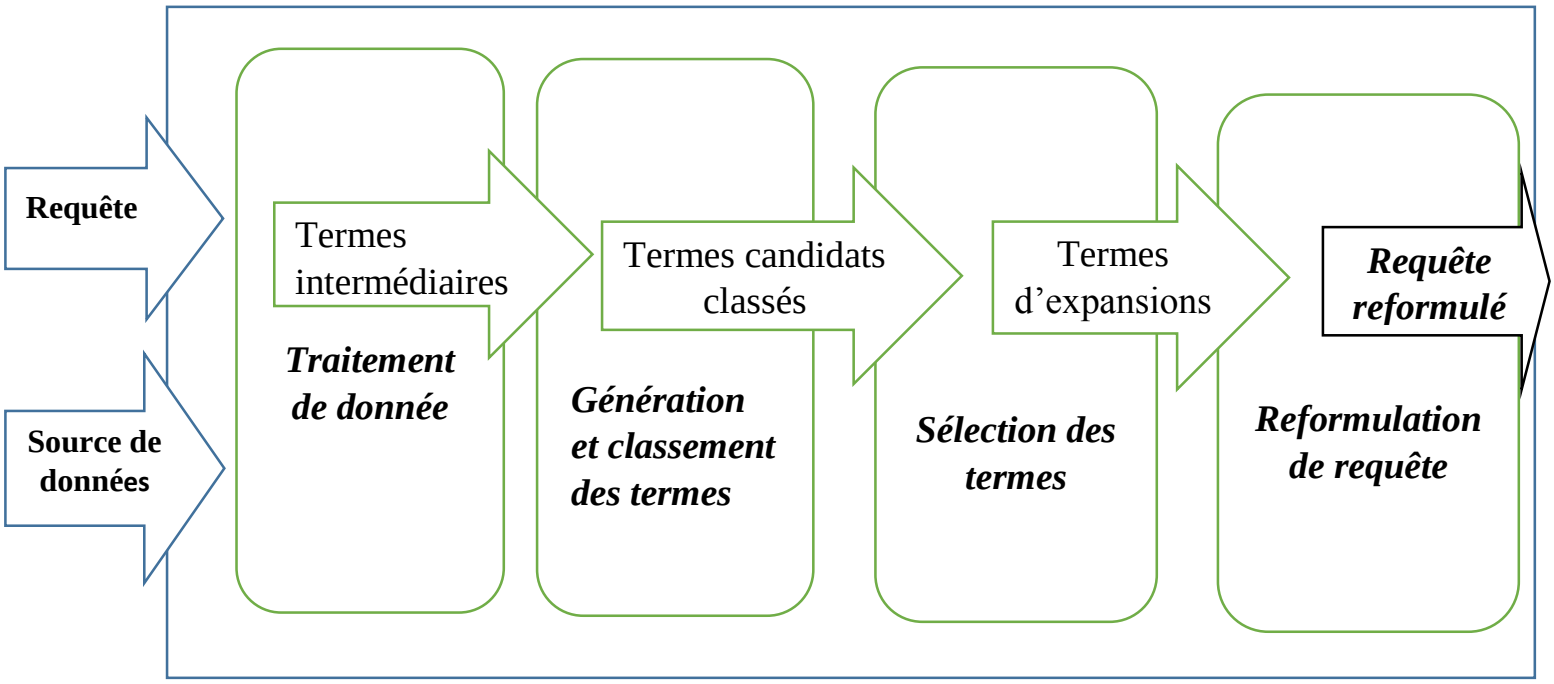


Figure II.1 : Le processus d'expansion automatique de la requête

II.4.1. Traitement de données

La première étape du processus d'expansion automatique des requêtes consiste à transformer la source de données pour rendre la requête de l'utilisateur plus large dans un format qui sera traité avec plus d'efficacité par les étapes suivantes. Cette étapes est souvent une phase d'extraction des termes afin de faciliter l'axées ainsi que la manipulation des fonctions de traitement.

Le prétraitement de la source de données (documents ou autres) est spécifique à son type, mais il est généralement indépendant de la requête utilisateur qui doit être étendue.
Beaucoup de technique d'expansion se basent sur des informations extraites lors d'une réponse à une requête initiale de l'utilisateur à partir d'une collection de documents, dans cette étape de traitement de donnée, il est indispensable d'indexer la collection et d'exécuter la requête.
En plus d'être représenté comme un ensemble de termes pondérés, chaque document est lui attribué un fichier complémentaire inversé de l'indice qui associe les termes du document aux termes de la requête. Le système d'indexation est aussi capable de stocker les positions des termes pour fournir la recherche basée sur la proximité ; quand un corpus externe est utilisé (des données Web pour la recherche sur intranet ou des données de bureau personnel pour la recherche sur Web).

II.4.2. génération et classement des termes candidats d'expansion

La deuxième étape de l'expansion automatique de la requête consiste à générer et à classer les termes candidats d'expansion. La requête initiale et la source de données transformée servent d'entrée pour cette étape, et l'ensemble des termes d'expansion représentent sa sortie.
Le classement est important, en raison des méthodes d'expansion de requêtes qui choisissent un petit nombre de termes candidats d'expansion pour les ajouter à la requête initiale.
Les techniques utilisées pour effectuer la génération et le classement des termes candidats sont classées selon le type de la recherche entre les termes d'expansion générés et les termes de la requête initiale.

II.4.3. Sélection des termes

Après avoir classé les termes candidats. Dans la deuxième étape, les principaux éléments (termes) sont sélectionnés pour l'expansion de la requête.
Les termes avec une probabilité supérieure à un certain niveau peuvent être sélectionnés seulement quand les scores des termes sont considérés comme des probabilités.
Plusieurs techniques pour la sélection des termes ont été proposées, elles utilisent différentes informations et pas seulement que les poids attribués aux termes candidats.
Une de ces techniques utilisé plusieurs fonctions de classement de termes, et sélectionne pour chaque requête les termes les plus courants **[38].**

Une autre stratégie consiste à choisir une quantité variable de terme d'expansion en fonction de la difficulté de la requête. Dans **[39]** les auteurs utilisent un classificateur afin de distinguer entre la pertinence et non pertinence du classement des termes d'expansion. Pour apprendre les paramètres du classificateur, un ensemble d'informations est créés dans lequel les termes simple sont étiquetés comme bons ou mauvais selon leurs influences sur les résultats de la recherche.
La sélection des meilleurs termes d'expansion pour une requête donnée est vue comme un problème d'optimisation. Cette étape de sélection des meilleurs termes sera discutée dans les paramètres de performance d'expansion de la requête.

II.4.4. La reformulation de la requête

C'est la dernière étape du processus d'expansion automatique de la requête. Elle décrit la requête élargie qui sera soumise au système de recherche d'information, ce qui revient

généralement à affecter un poids à chaque terme qui décrit la requête étendue appelé « pondération de la requête ».

La technique de pondération des termes d'une requête la plus populaire est reproduite à partir de la formule de Rocchio pour la réinjection de pertinence **[40].**

La formule générale est donnée comme suit :

$$W'\,\mathrm{t},\mathrm{q}' = (1-\boldsymbol{\lambda}).W'\,\mathrm{t},\mathrm{q} + \boldsymbol{\lambda}.\,\mathrm{score}\,\mathrm{t} \qquad \text{(II.1)}$$

Ou' :

q' : est la requête étendue, q est la requête originale, $\boldsymbol{\lambda}$ représente le paramètre de pondération, et score $_t$ Représente le poids attribué au terme d'expansion t.

Les poids basés sur les documents pour la requête qui n'est pa étendue et les score en fonction de différence de distribution utilisés pour les termes d'expansion possèdent de différentes échelles, alors leurs valeurs doivent être normalisées avant de les additionner dans la formule (II.1).

Lorsque les termes d'expansion sont extraits des documents pseudo-pertinents et leurs score est calculé en utilisant les documents, il est facile de montrer que le vecteur de la requête étendue calculé par l'expression ci-dessus se dirige vers les documents pseudo-pertinents (selon les pondérations de document). Cependant, les avantages de la prise en compte de la différence de la répartition des termes entre les documents pseudo-pertinents et toute la collection pour sélectionner les termes d'expansion peut être réduire si nous pondérons ces termes.

Même un simple schéma de pondération sur la base d'une fonction inverse de classement de terme peut produire de bons résultats. Notez que les poids basés sur des documents utilisés pour la requête non expansée et les scores en fonction de différence de distribution utilisés pour les termes d'expansion ont différentes échelles, leurs valeurs doivent être normalisées avant de les additionner dans l'expression ci-dessus.

Plusieurs techniques de normalisation simple, discutées dans **[41]**, ont été proposées, en général, elles produisent des résultats comparables mais également accroitre son uniformité pourrait être plus efficace**.**

La valeur de $\boldsymbol{\lambda}$ de l'expression ci-dessus peut être ajoutée de façon à optimiser les performances, si les données sont disponibles. Un choix par défaut typique est de donner plus d'importance aux termes de la requête d'origine ; par exemple deux fois plus que les termes d'expansion. Autre possibilité est d'utiliser une formule de pondération de requête sans paramètre tels que proposé dans **[42]**, compte tenu d'un paramètre de réinjection de pertinence,

les auteurs utilisent une approche d'apprentissage pour prédire la valeur optimale de λ pour chaque requête et chaque ensemble de documents de feedbeck, explorer un certain nombre de paramètre liées à la requête et aux documents (tels que la longueur, et la clarté) et à la divergence entre la requête et les documents feedback.

L'expression ci-dessus peut également être utilisée lorsque les termes d'expansion ont été extraits d'un thésaurus ou WordNet. Les pondérations peuvent être fondées sur des critères tels que le nombre de termes, le nombre de cooccurrence, la longueur du document, et le type de relation.

L'étape de pondération de termes est naturellement prise en charge si l'approche de modélisation du langage effectue le classement des documents. Dans Voorhess **[41],** par exemple, le vecteur de requête expansée est composé de sous-secteurs de onze types de concepts différents avec un poids important associé : un pour les termes de la requête d'origine, un pour les synonymes, et pour chacun des autres types de relations contenues dans la partie nom de WordNet.

Si le classement des documents est effectué par le biais d'une approche de modélisation du langage, l'étape de pondération de requête est naturellement prise en charge.

II.5. Les paramètres de performance d'expansion de la requête

Un nombre considérable d'expérimentations ont été effectuées sur les collections de documents pour l'évaluation de l'impact induit par la reformulation de requête sur le processus de recherche d'information. Une étude synthétique de ces différents travaux annonce que l'ordre des performances imputées à l'intégration de l'expansion de requête est variable, et elle dépend de plusieurs paramètres **[35].**

- Le modèle de recherche d'information utilisé
- L'hypothèse de base quant à la distribution des termes dans les documents, concept, phrases et relation sémantique entre eux.
- Les caractéristiques des collections de documents ; nombre, taille, etc.

En faisant abstraction des paramètres caractéristiques inhérents à chacune des techniques de reformulation de requête présentées, les autres de l'étude ont dégagés, les paramètres de performance intrinsèques suivant :

II.5.1. Nombre de termes d'expansion ajoutés à la requête

La pertinence est expérimentée par **[42]** dans l'environnement **TREC**. Ils ont montré que le taux de performance est avantage corrélé avec le nombre de termes ajoutés qu'avec le nombre de documents initialement retrouvés.il ont abouti à la mise au point de l'équation de variation :

$$RP(N) = A.\log(N\ s) + B.\log(X) + C \qquad \text{(II.2)}$$

Ou' :

RP(N) : performance du système pour N documents restitués

N s : Nombre de documents restitués

X : nombre de termes ajoutés à la requête

A, B, C : Constantes

Ils ont conclu que le seuil critique du nombre de terme à la requête dépend des caractéristiques de la collection. En outre, la pondération différenciée des termes ajoutés à la requête accroît la performance du système. On attribue alors un poids :

-moins important aux termes ajoutés.

-plus important aux termes issus des documents pertinents que ceux des documents non pertinents.

II.5.2. Méthode de sélection des termes

La méthode de sélection des termes à ajouter à la requête est aussi importante que le choix de leur seuil. Nous citerons les principales méthodes expérimentées.

*Salton& Buckley **[43]** ont expérimenté séparément, l'ajout de tous les nouveaux termes, tous les termes issus des documents pertinents et les termes les plus fréquents dans les documents restitués à la requête initiale.

L'expansion de la requête avec tous les nouveaux termes offre de meilleurs résultats que les autres méthodes, toutefois l'écart de performance n'est pas très considérable relativement aux exigences de temps et d'espace mémoire.

Robertson **[44]** et Haines**[45]** adoptent une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer pour chaque terme un nombre traduisant sa valeur de pertinence. Les termes sont alors triés puis sélectionnés sur la base d'un seuil.

Robertson propose la formule suivante pour le calcul de la valeur de sélection d'un terme :

$$SelValue(i) = w\ \text{ij} * (Pi - Ui) \quad \text{(II.3)}$$

Où :

$$W = log\frac{pi(1-U\text{i})}{Ui(1-pi)} \quad \text{(II.4)}$$

Avec :

W_{ij} : poids du terme i dans la requête

P_i : probabilité (di = 1/D est pertinent)

U_i : probabilité (di =0/D est non pertinent)

- Harman **[46]** propose les fonctions suivantes :

1)

$$\text{SelValue (i)} = \frac{RT\text{j} * df\text{i}}{N} \quad \text{(II.5)}$$

Où :

RT_j : Nombre total de document retrouvés par la requête

df_i : fréquence d'occurrence du terme ti dans la collection.

N : nombre total de documents dans la collection

2)

$$\text{SelValue (i)} = \frac{r\text{i} * df\text{i}}{R} \quad \text{(II.6)}$$

Où :

r_i : Nombre de documents pertinents contenant t_i

R : Nombre de documents pertinent

3)

$$\text{SelValue(i)} = \log_2\frac{pi(1-qi)}{(1-pi)} \quad \text{(II.7)}$$

Avec : p_i : probabilité que t_i appartienne aux documents pertinents

q_i : probabilité que t_i appartienne aux documents non pertinents

II.5.3. Longueur moyenne de requête

L'accroissement des performances est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite **[47].**

Dans ce sens, des expérimentations intéressantes ont été réalisées sur la base TREC7 et présentées dans **[48]**. Les auteurs montrent en effet que la dérivation automatique de courtes requêtes à partir de documents jugés ou supposés pertinents à la suite d'une recherche initiale, permettent d'atteindre des résultats à la suite d'une recherche initiale, permettent d'atteindre des résultats très performants pour différentes tâches : recherche, filtrage et routing.

II.5.4. Choix des documents d'expansion

Il existe peu de travaux réalisés pour la sélection des documents comparativement à la sélection des termes, la méthode présentée dans **[49]** considère le document pertinent comme mesure où il peut aider efficacement pour trouver les documents pertinents à partir de D rel par l'intermédiaire d'une recherche effectuée sur le corpus.

L'objectif de cette méthode est de sélectionner un ensemble de K documents pertinents représentatifs qui aide à trouver les documents pertinents lors de l'utilisation de retour de pertinence basée sur la recherche pour classer tous le corpus, les performances de recherche qui en résultent seront optimales en ce qui concerne tous les sous-ensembles des K documents dans D rel.

Deux grandes familles de méthodes sont présentées pour la sélection de documents :
La première estime la représentativité d'un document indépendamment des autres documents dansD_{rel}, en sélectionnant les k-tops documents classés, cette méthode assigne au document d (appartient àD_{rel}) un score, score(d) qui reflète la représentativité de d dansD_{rel}.

Les k-tops documents dans D_{rel} sont ensuite utilisés comme entrée à la méthode de sélection des termes d'expansion. Parmi les caractéristiques utilisées pour la sélection des documents :

- La méthode Random qui affecte un score à chaque document selon la fonction suivante :

$$score_{(random)}(d)^{def} = \frac{1}{n} \qquad \text{(II.8)}$$

Puis choisir les k-documents à partir de D_{rel} au hasard.
Où n est le nombre de documents dans la collection.

- La méthode QuerySim qui considère le document d qui a une similarité élevée avec la requête comme un bon représentant et le calcul du score pour chaque document se fait selon la fonction suivante :

$$score_{QuerySim}(d)^{def} = sim(q,d) \quad \text{(II.9)}$$

- La méthode basée sur la longueur d'un document suppose que les documents pertinents courts sont les meilleurs représentants que les documents longs pertinents :

$$score_{length}(d)^{def} = -|d| \quad \text{(II.10)}$$

Où :

|d| : est la taille de document d.

La deuxième est basée sur l'hypothèse suivante : les documents représentatifs sont semblables les uns des autres en exploitant les relations (similitude) entre les documents dansD_{rel}. Cette méthode nécessite une connaissance de tout l'ensemble de documents pertinents. A titre d'exemple :

- La méthode centroïde : en termes de modèle de langage, le centroïde est une probabilité de distribution de tout le vocabulaire.

$$p(w\backslash Cent(D_{rel}))^{def} = \frac{1}{n}\sum_{d\in D_{rel}} p(w\backslash d) \quad \text{(II.11)}$$

Ensuite, la méthode de centroïde permet d'estimer les documents représentatifs en utilisant le KL-divergence du modèle de langage induit à partir du centroïde :

$$score_{centroid}(d)^{def} = -|\mathrm{d}| * (p\ (.\backslash cent\ (D_{rel})\ )||p(.\backslash d)) \quad \text{(II.12)}$$

Les méthodes basées sur le graphe : Certains travaux sur le reclassement de la première liste de documents trouvés en réponse à une requête, ont montré que les documents qui sont très semblables aux autres documents dans la liste ont une forte probabilité de pertinence. L'idée est que ces documents représentent la liste entière, et par la vertu de la façon dont la liste a été créée, c'est-à-dire en réponse à la requête, ils pourraient être pertinents au besoin fondamental de l'information. Toutefois, la liste est composée de plusieurs documents à la fois pertinents et non pertinents **[50].**

II.6. L'expansion de la requête dans le modèle de langue

La modélisation de la fonction de correspondance dans le modèle de langue part d'un principe différent des approches traditionnelles de la recherche d'information. On ne tente pas de modéliser la notion de pertinence dans ce modèle, mais on considère que la pertinence d'un document vis-à-vis d'une requête est en rapport avec la probabilité que la requête puisse être générée par le modèle de langue du document. On suppose en effet, qu'à chaque document vis-à-vis d'une requête sachant le modèle de ce document.

II.6.1. Modèle de LavrenKo et Croft

Au lieu de modéliser la RI comme processus de génération de la requête, LavrenKo et Croft **[51]** ont proposé de modéliser explicitement le modèle de pertinence. Ils ont en effet, proposé d'estimer ce modèle à partir du modèle de la requête sans utiliser les données d'entrainement.

En faisant le parallèle avec la modélisation de la pertinence proposée dans le modèle probabiliste classique. Ils considèrent en effet, que pour chaque requête, il existe un modèle permettant de générer le sujet (thème) abordé par la requête, c'est ce que les auteurs appellent le modèle de pertinence (θ_R).

Le but est alors d'estimer la probabilité p(θ_R), de générer un terme à partir du modèle de pertinence. Comme le modèle de pertinence n'est pas connu, les auteurs ont suggéré d'exploiter les documents retournés les mieux classés (feedback documents) en assurant qu'ils sont générés du modèle de pertinence. Ce modèle est formalisé comme suit

$$p(t|\theta R) = \sum_{d \in R} p(d)p(t|d) \prod_{i=1}^{K} p(qi|d) \qquad \text{(II.13)}$$

Avec : R : l'ensemble des tops documents pertinents.

P(d) : la probabilité de choisir un document d des tops documents pertinents retournés.

Ainsi, le modèle de pertinence obtenu est une combinaison pondéré du modèle individuel de chaque document feedback p (t\d) avec le score de ce document vis-à-vis de la requête p (q_i\d).

Les résultats des expérimentations ont montré que cette approche améliore sensiblement les performances de la recherche d'information, de +10% à +29%d'amélioration de précision moyenne par rapport au modèle de langue de base**[52]**.

II.7. Les facteurs de la pondération

Les modèles probabilistes se basent sur la théorie des probabilités dans le processus de recherche d'information. Le premier modèle a été proposé par Maron et Kuhn **[22]**. Le principe de base de ces modèles est la présentation des résultats de recherche d'information dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête **[53].**

Différents modèles probabilistes ont été proposés **[54] [55]**, nous présentons dans les sous-sections suivantes quelques modèles les plus représentatifs, à savoir : le modèle Robertson et Spark-Jones **[56]** et le modèle okapi**.**

II.7.1. Le modèle *Tf*Idf* (Robertson)

Parmi les formules qui ont introduit la taille de document on trouve la formule *Tf*Idf* de sparck Jones **[56]** qui est la base de toutes les autres formules **[52][53].**La formule *Tf*Idf* de sparck Jones est donnée comme suit :

$$w = \frac{tf * (k_1 + 1) * log(\frac{N}{n})}{k1 * \left((1 - b) + b * \frac{dl}{avg\ dl}\right) + tf} \qquad \text{(II.14)}$$

Où :

b : constante

dl : taille d'un document

Avgdl : taille moyenne des documents.

Les paramètres k_1 et b présent respectivement l'influence du poids par rapport à la fréquence et l'effet de longueur du document.

II.7.2. Le modèle Okapi (BM25)

Le schéma de pondération BM25 (BM pour "Best Match" ou meilleur arrangement) qui a été développé par Robertson **[57]** est un schéma de pondération basé sur le modèle probabiliste. Ils utilisent la distribution de probabilité de 2-Poisson. De plus, ils supposent que la longueur d'un document influe directement sur la proportion de termes qu'il emploie. Il s'agit d'une hypothèse similaire à la mesure cosinus du modèle vectoriel (hypothèse de document monothématique). Le score d'un document "d" par rapport à une requête "q" dans la fonction de pondération BM25 peut être calculé comme suit :

$$BM_{25} = \sum_{t \in q \cap d} \left(\frac{tf}{(tf + k_1 . nb} . log\left(\frac{N - df_t + 0.5}{df_t + 0.5} \right) . qtf \right) \quad \text{(II.15)}$$

Où :

tf: fréquence d'apparition du terme,

N : nombre totale de documents dans la collection,

dft : nombre de documents contenant le terme t,

qtf : fréquence d'apparition du terme t dans la requête,

k_1 : paramètre d'influence de la fréquence des termes,

$$\mathrm{nb} = (1 - \mathrm{b}) + \mathrm{b}.\frac{tl}{tl_{\mathrm{avg}}} \quad \text{(II.16)}$$

Où :

nb : facteur de normalisation est calculé comme suit :

tl: nombre de termes dans le document (longueur de document).

tlavg : nombre moyen de termes dans un document.

b = 0.75 ; l'augmentation de la valeur de *b* augmente la pénalisation des plus longs documents, avec une valeur de 1 étant la limite supérieure.

II.8. Les autres facteurs de pondération

II.8.1. La structure BM25E

En partant de l'hypothèse qu'un élément peut être traité comme un document. Le modèleBM25 a été adapté pour pondérer les termes dans les éléments, sauf qu'il peut hériter des informations d'autres éléments du document. La fonction proposée nommée BM25E est une fonction qui a été dérivée de la fonction BM25 **[58]**.Ainsi, chaque élément pouvant être un noeud feuille qui constitué de l'information textuelle, ou un nœud interne donc représente un sous- arbre xml ou encore un nœud unique racine qui représente tout le document, donc l'élément hérite des informations d'autre : éléments spécialement lorsqu'il s'agit d'un nœud feuille ou interne. Les détails de ce modèle sont les suivants :

- Supposons que nous avons :
- *tf* : la fréquence de terme dans l'élément e.
- *el* : la taille de l'élément.
- $tf_{e,j}$: la fréquence du nième terme dans l'élément e.

- el_e : la taille de l'élément.
- df_j : le nombre d'élément qui contient le terme j.
- *avel* : la taille moyenne de l'élément de la collection.
- La formule proposée et qui utilise les paramètres cités ci-dessus, est définie comme suit :

$$wj(e,d,c) = \frac{(k_1+1)tf_{e,j}}{k1\left((1-b)+b\frac{el_e}{avel}\right)+tf_{e,j}}\log\frac{N-df_j+0.5}{df_j+0.5} \quad \text{(II.17)}$$

Où : $wj(e,d,c)$ est le poids du terme t_j dans l'élément e appartenant au document.

II.8.2. Le facteur de proximité des termes

L'intuition considérée dans les proximités des termes de la requête dans un document est la suivante : « un bon document est celui dans lequel les termes de la requête apparaissent proche les uns des autres ».

II.8.2.1. Le modèle de langue de position (Positional Language Model)

Lv et Zahi **[59]** ont proposé un modèle de langue nommé «Positional Language Model : PLM»

Dans ce dernier, ce modèle de langue pour chaque position du document est créé, il est exprimé ainsi :

$$p(t|d,i) = \frac{c'(t,i)}{\sum_{t'\epsilon v}(t',i)} \quad \text{(II.18)}$$

Où **:**

V est vocabulaire et c' (t, i) est la fréquence virtuelle du terme "t" à la position "i" obtenue par la propagation de l'ensemble des occurrences du terme "t" dans toutes les positions du document. Cette propagation est réalisée en utilisant une fonction de densité (Gaussian Kernel, trangle Kernel, Circle Kernel, Hamming Kernel).

Afin de calculer le score d'un document vis-à-vis d'une requête, ils utilisent les scores obtenus sur les différentes positions du document. Différentes stratégies de combinaison de scores ont été utilisées : la stratégie de la meilleure position, la stratégie multi-position et la stratégie multi-gamma.

Ces approches utilisant la notion de proximité pour intégrer les relations entre termes ont montré que cette source d'information est utile pour la RI.

II.8.3. La position des termes

En 1950 Luhn**[21]** a proposé que « la fréquence du terme dans un document donne une bonne indication sur l'importance du terme et que la position relative du terme dans une phrase détermine l'importance de la phrase dans le document. »

Luhn a donc combiné ses deux propositions pour mesurer l'importance du terme dans la phrase et pour pouvoir ensuite mesurer l'importance de la phrase dans le document. Suite à la proposition de Luhn, de nouveaux travaux sont apparus exploitant la position du terme dans le document. Parmi ces travaux on trouve celui de D.Troy et al **[60]** ; où ils déclarent (proposition) qu'un bon document est celui où les termes de la requête apparaissent dans le premier positon de ce document.

II.9. Conclusion

Ce chapitre, a porté essentiellement sur l'expansion de requête, d'abord nous avons présenté le processus d'expansion automatique de la requête, ensuite, nous avons décrit les principales étapes du processus d'expansion de requête ainsi que ses paramètres de performance, puis nous avons présenté l'expansion de requête dans le modèle langue. Enfin nous avons étudiés les facteurs de la pondération des termes d'expansion. L'un de ces facteurs est la position des termes, qui a montrés des résultats encourageants.

Le but commun à ces techniques est principalement, l'amélioration de la qualité des résultats de recherche lorsque la seule évaluation de la similarité entre les requêtes et les documents, n'est plus suffisante.

Dans le chapitre suivant, nous allons présenter notre approche, qui consiste à étendre le modèle de KL-Divergence utilisé pour la reformulation de la requête, en intégrant la position des termes.

CHAPITRE III

Expérimentation et Evaluation

III.1. Introduction

Le travail présenté dans ce mémoire se situe dans le contexte de la recherche d'information, et plus particulièrement dans le cadre de l'expansion de requête.

Nous avons présenté dans le chapitre précédent les différentes approches d'expansion de requête, dans ce chapitre nous allons présenter une nouvelle approche de pondération de termes d'expansion, basée sur la position des termes dans les documents pertinents.

Dans ce qui suit, nous allons donner les fondements théoriques (architecture) de notre approche qui est implémentée en utilisant la plateforme Terrier, puis les outils de développement utilisés et enfin quelques résultats expérimentaux obtenus sur la collection de test TREC AP88.

III.2. Architecture générale de notre approche

La figure suivante illustre l'architecture de notre approche, plus précisément, notre travail se situe dans la partie représentée en couleur.

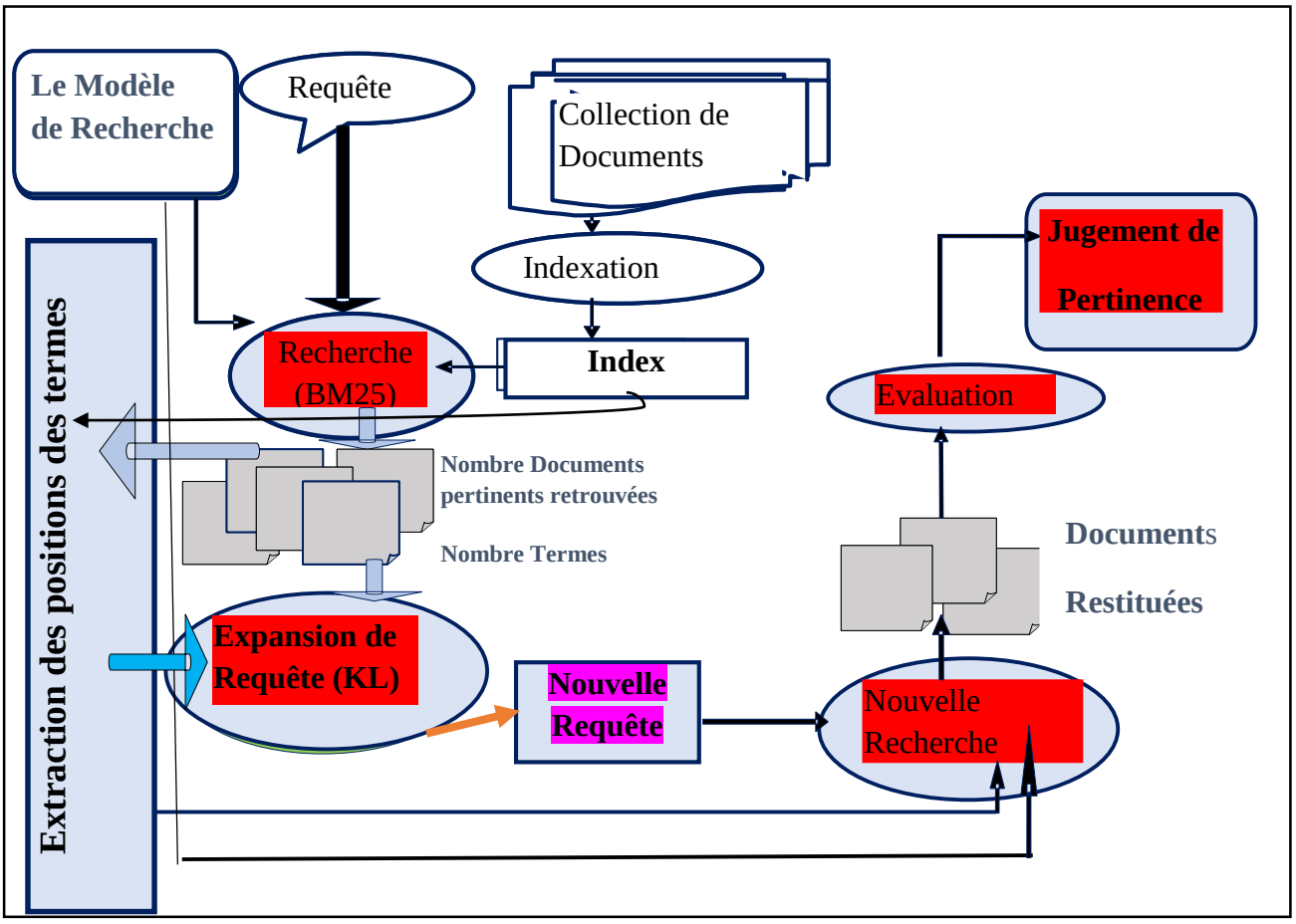


Figure III.1 : Architecture générale de notre approche

III.3. Présentation de notre approche

La reformulation de requête a été proposée comme une méthode élaborée pour la RI, s'inscrivant dans la voie de conception des SRI adaptatifs aux besoins des utilisateurs. C'est un processus permettant de générer une requête plus adéquate à la RI dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est de modifier la requête de l'utilisateur par sélection des documents d'expansion, puis sur la base de ces derniers, sélection des termes d'expansion. Dans cette section nous allons présenter notre travail qui consiste en l'implémentation d'une nouvelle approche de sélection de termes d'expansion de la requête sous la plateforme Terrier, en étendant le modèle de base.

L'objectif de notre travail, est de pouvoir renvoyer à l'utilisateur les documents les plus pertinents répondant à sa requête, en "**introduisant un facteur de position des termes dans les documents pertinents**" dans le modèle Kullback-Leibler (KL) pour classer les termes d'expansion.

Avant de décrire notre approche nous spécifions d'abord certain éléments utilisés.

III.3.1. Éléments Existants utilisés

Nous avons utilisés le modèle de recherche : BM25, et modèle d'expansion : KL-Divergence, présentés ci-dessous :

a) Modèle de recherche utilisé : le modèle de recherche a pour objectif ***d'assigner un score pour chaque document***, puis retourner une liste ordonnée (résultat de la première recherche), Dans notre cas nous avons utilisé le modèle BM25, qui a montrés de bons résultats, ce modèle est présenté dans ce qui suit **[60]** :

$$Score(q,d)\mathrm{BM25} = \sum_{t \in q \cap d} \frac{\mathrm{tf}}{(0.5 + 1.5.. \frac{\mathrm{dl}}{avdl}) + \mathrm{tf}} . \log(\frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5}) \qquad \text{(III.1)}$$

En plus du tf et du $\mathrm{df_t}$ comme défini ci-dessus ;

t : est un terme de la requête ;

N : est le nombre de document dans la collection ;

dl : est la longueur du document ;

Avdl : est la longueur moyenne des documents de la collection.

b) Modèle d'expansion utilisé (KL-D) :

L'objectif de modèle d'expansion de requête est de ***classer les termes d'expansion*.** En ce qui nous concerne nous avons utilisé ***le modèle KL-Divergence.*** Ce choix provient du fait que cette divergence est celle qui a montré son intérêt théorique (compatibilité avec le bouclage de pertinence), et qui expérimentalement est compatible avec des fichiers inverse. Ce modèle est présenté dans ce qui suit :

$$KL(t_i,R) = \sum_{D_j \in R} P(t_i \,|D_j\,) * log \frac{P(t_i\,|D_j)}{P(t_i|C)} \qquad \text{(III.2)}$$

Où : R : ensembles de documents pertinents

$P(t_i\,|D_j)$: est la probabilité d'apparition du terme t_i dans le document D_j.

$P(t_i\,|C_j)$: est la probabilité d'apparition du terme t_i dans la collection$C$.

III.3.2. Approche proposée :

L'approche proposée consiste à étendre le modèle KL-Divergence en intégrant la position des termes dans les documents pertinents. Nous présentons dans ce qui suit l'intuition de l'approche, sa formalisation et sa mise en œuvre.

a) Intuition de l'approche : les termes qui **couvrent** bien les documents pertinents retournés sont des bons candidats pour l'expansion. Par **couverture** nous voulons dire qu'un terme qui apparait dans plusieurs parties d'un document est meilleur qu'un terme qui apparait dans peu de parties d'un document.

Ci- dessous un exemple qui explique cette intuition :

Exemple :

Soit le document D suivant décrit par les termes qu'il contient ainsi que leur position :

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Terme	**t_1**	t_3	t_2	t_2	t_3	t_5	t_4	t_2	t_8	**t_1**	t_9	t_{12}	t_{10}	t_7	t_4	t_{13}	t_8	t_6	t_{11}	**t_1**

Pour ce document D ; la fréquence des termes t_1 et t_2 est : t_f (t_1) =3, t_f(t2) =3.

On remarque dans cet exemple que les termes t_1 et t_2 ont la même fréquence dans le document D ; par contre, le terme t_1 offre une meilleure couverture pour le document D, car il apparait dans beaucoup de parties du document D.

b) La Formalisation de l'approche :

La similarité entre document pertinents et la collection est mesurée par la fonction de divergence de Kullback-Leibler(Kl).

La partie d'extraction des positions des termes dans la Figure III.1 qui montre l'emplacement de notre approche dans la SRI est définit par un facteur position dans la matrice comme suit :

$$P(t_i|D) = \alpha \times P_{ML}(t_i|D) + (1 - \alpha) \times P_{pos}(t_i|D) \quad \text{(III.3)}$$

$$P_{pos}(t_i|D) = \left(\frac{nb\ Fenêtre}{nb\ Fernêtre\ Total}\right) \quad \text{(III.4)}$$

Où :

$nb\ Fenêtre$ est le nombre de fenêtres où le terme t_i apparait dans le document D.

$nb\ Fernêtre\ Total$ est le nombre de fenetres que contient le document D.

α est un facteur d'interpolation entre le modèle de base ($P_{ML}(t_i|D)$) et le modèle intégrant la position des termes ($P_{pos}(t_i|D)$).

c) La mise en œuvre de l'approche :

L'objectif de notre travail est précisément d'estimer le facteur de position, et de voir son impact sur les résultats de la RI. En effet, ce facteur modélise la position du terme dans les documents.

Nous avons modifié et ajouté des classes au système Terrier. Nous expliquons, plus loin, ces classes.

III.4. L'environnement technique

Notre travail consiste à implémenter notre approche sous Terrier qui est une open source écrit en java, donc ce dernier est imposé à l'utilisation. Dans la section suivante nous présentons les outils utilisés.

III.4.1. La plateforme Terrier

Terrier**, Terra byte Retriever** : est un Système de Recherche d'Information de robuste et efficace, utilisé avec succès pour la recherche ad-hoc, la recherche sur le web et la recherche multilingue dans des environnements centralisés et distribués.

Terrier offre une plateforme idéale destinée à l'indexation de volumes importants de documents : jusqu'à 25 millions de documents. Il est développé par le département informatique de l'université Glasgow de Scotland. C'est un logiciel open Source écrit en java.

Selon des études comparatives sur les SRI en open sources effectuées par Ricardo BaezaYates Terrier fait partir des trois meilleurs systèmes dans un environnement java, les deux autres sont MG4 et Lucence.

Comme tous moteurs de recherche, terrier possède les principaux suivants :

- L'indexation : extraction des mots clés des documents appartenant à une collection et les stocker dans un index.
- La recherche des documents pertinents pour répondre aux requêtes formulées par l'utilisateur.
- L'évaluation des résultats de la recherche.

Terrier est livré avec trois types d'applications :

Desktop Terrier : elle permet d'indexer et de rechercher un contenu local d'un utilisateur.

- **Trec (Batch) Terrier** : elle permet l'indexation et la recherche dans les collections, notamment TREC.
- **Interactive Terrier** : elle permet la recherche d'un document depuis un navigateur (la recherche web).

Nous présentons une description détaillée de ce système en annexe. La figure suivante donne un aperçu de l'architecture de Terrie.

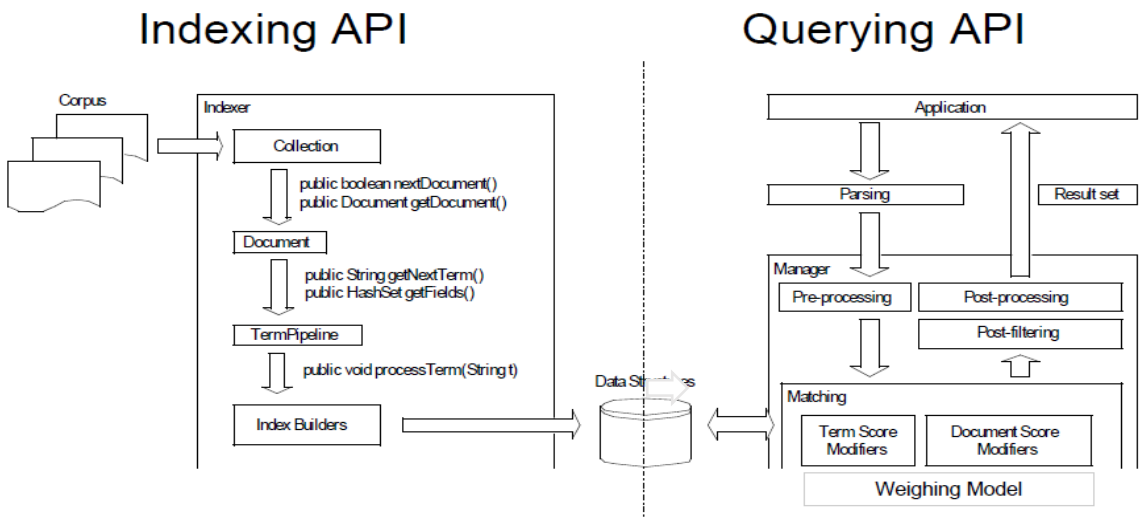


Figure III.2 : Vue d'ensemble d'architecture de terrier

A. API d'indexation : l'indexation dans Terrier est divisée en quatre procédures et à chaque procédure, des classes java peuvent être ajoutées pour la personnalisation du système.

Les quatre procédures sont :

1) Splitter la collection de documents : consiste à parcourir l'ensemble du corpus reçu en entrée par Terrier et envoyer chaque document à l'étape suivante.
2) Extraction des termes (Tokenize Document) : qui consiste à relever chaque document reçu et extraire les différents termes.
3) Traitement des termes extraits avec TermPipeline : consiste en l'élimination des mots vides et la lemmatisation des termes.
4) La construction de l'index.

Toutes ces étapes, les modules en charge de leur exécution ainsi que les fichiers résultants de cette indexation sont présentés de façon plus détaillée dans l'annexe.

La figure ci-dessous donne une vue d'ensemble d'interaction des composants principaux impliqués dans le processus d'indexation.

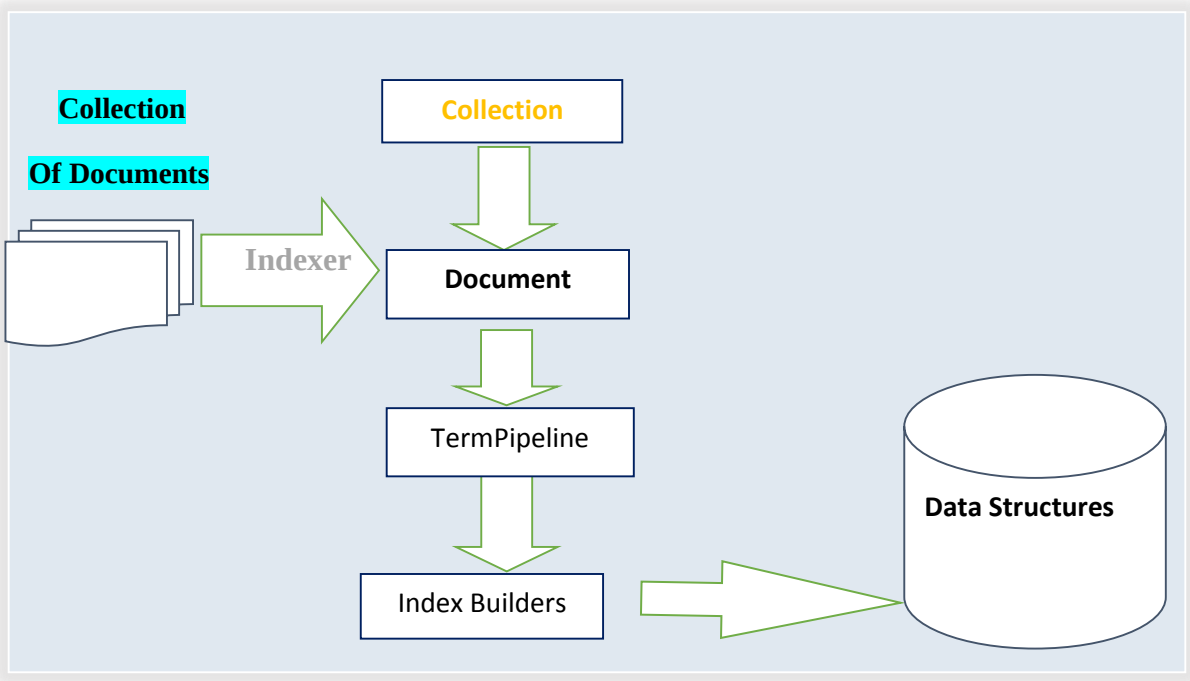


Figure III.3 : Le processus d'indexation dans Terrier.

Les différentes classes associées au processus d'indexation sont organisées dans un ensemble de package dont on trouve :

Org.terrier.indexing : ce package contient les différentes classes permettant de réaliser un ensemble d'opérations sur la collection des documents, dans le but d'extraire les termes de tous les documents de la collection.

Org.terrier.terms : les classes qui se trouvent dans ce package permettent d'effectuer un ensemble de traitements sur les termes extraits. Parmi ces traitements, l'élimination des mots vides, lemmatisation des termes,…etc.

Org.terrier.structures : les classes de ce package permettent la construction d'un ensemble de structures ou un ensemble de données stockées. Parmi ces structures, on a :

- **Lexicon :** contient les informations sur chaque terme de la collection (Terme, Id terme, nombre de documents qui contiennent le terme, fréquence du terme dans la collection, Offset dans le fichier inverse).
- **Direct index :** il enregistre pour un document les termes qui apparaissent dans ce dernier. Il est souvent utilisé pour la reformulation de la requête, la classification et la comparaison des documents.

Index (Id Terme, Id document, Fréquence terme dans le document, #fields).

- **Inverted Index** : contrairement à l'index direct, il enregistre pour un terme les documents dans lesquels il apparait, il contient aussi la position de chaque terme et sa fréquence dans ces documents**.**

Fichier inverse (Id Terme, Id document, Fréquence terme dans le document, #fields).

- **Document Index :** contient des informations sur les différents documents de la collection (Id Terme, Fréquence terme).

B. API de recherche : durant le processus de recherche, chaque requête doit passer par les étapes suivantes :

1) Query : classe abstraite qui représente la requête.

   Terrier supporte trois modèles de requête :

   - ✓ *SingleTermQuery :* désigne la requête qui contient un seul terme.
   - ✓ *MultiTermQuery :* désigne la requête qui contient plusieurs termes.
   - ✓ *FieldQuery :* terme qualifié par un champ (Exp : dans le titre du document).

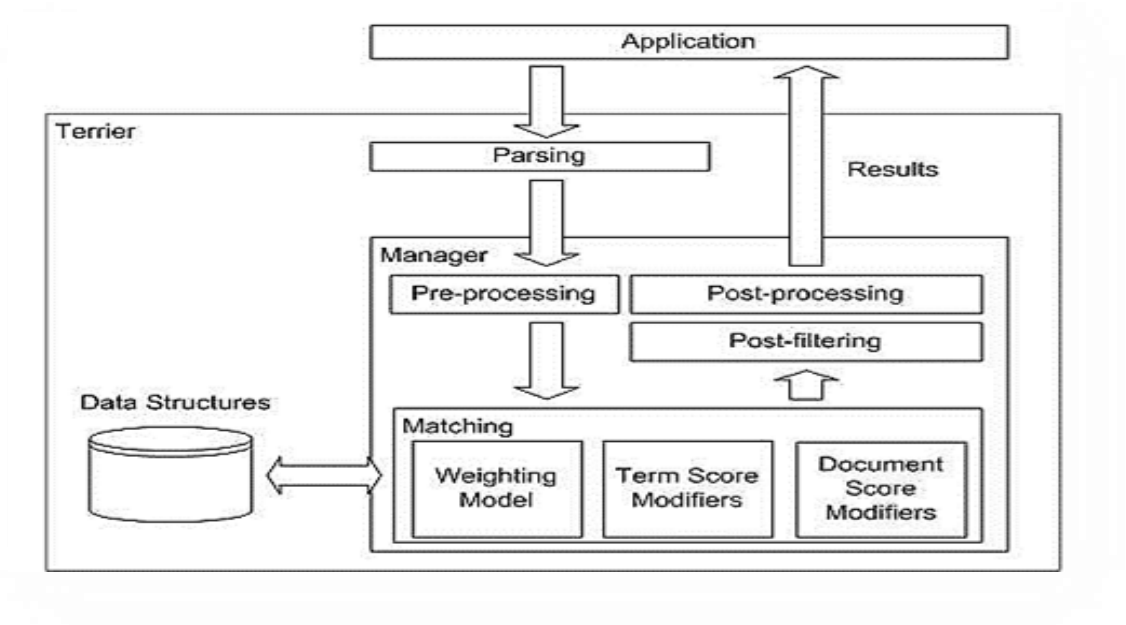
2) Parsing : qui se charge de tokenizer la requête.
3) Pré-processing : qui applique le TermPipeline à la requête. Elimine les mots vides et les lemmatise.
4) Matching : responsable de l'initialisation du Weighting Model et du calcul des scores entre la requête et les documents.
   - **WeightingModels :** assigne un score pour chaque terme de la requête dans le document (Pondération), plusieurs modèles de pondération sont implémentés : TF_IDF, BM25, etc.
   - **DocumentScoreModifiers :** il permet de modifier le score d'un document en fonction du langage de la requête.
5) Post-traitement : peut modifier le ResultSet, par exemple, par un procédé QueryExpansion, afin de générer un meilleur classement de documents. Ce procédé fonctionne en faisant l'extraction des termes informatifs, à partir des tops documents classés (un nombre de documents spécifiés du ResultSet), par attribution du score pour chaque terme en utilisant le modèle de pertinence étendu (notre cas), et ajouter ceux avec les scores les plus élevés à la requête originale. La nouvelle requête est ré-pondérée, et une nouvelle recherche est faite à l'aide du modèle BM25 et en ajoutant la position des termes implémenté par KL. Un ensemble de documents plus pertinents,

qui seront stockés dans le fichier.res (pertinence système) est retourné comme résultat à la nouvelle recherche effectuée.

6) Post-filtering : filtrage des résultats.

Toutes ces étapes et les modules en charge de leur exécution seront détaillés en Annexe.

La figure ci-dessous donne une vue d'ensemble d'interaction des composants de Terrier dans la phase de recherche.



FigureIII.4 : Le processus de recherche dans Terrier

III.4.2. Le processus de reformulation de requête (Query Expansion)

Après l'indexation, les termes sont stockés en structure de données suivantes :

- Lexicon : qui stocke les informations de chaque terme de la collection ;
- Inverted Index : où le fichier inverse est stocké ;
- Document Index : qui contient des informations sur les différents documents de la collection.

- **Index direct** :

L'index direct enregistre pour un document les termes qui apparaissent dans ce dernier. Il est souvent utilisé pour la reformulation de la requête, la classification et comparaison des documents.

- **Index inversé :**
- L'index inversé contrairement à l'index direct enregistre pour un terme les documents dans lesquels il apparait, il contient aussi la position de chaque terme et sa fréquence dans ces documents.

La reformulation de la requête se fait après avoir des résultats de la première recherche. Telle qu'une autre requête générer de la requête précédente.

III.4.3. Le langage java

Java est un langage de programmation moderne, développé par Sun Microsystems (aujourd'hui racheté par Oracle). Une de ses plus grande force est son excellente portabilité : une fois votre programme est créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc. On peut faire de nombreuses sortes de programmes avec java :

- ✓ Des applications, sous forme de fenêtres ou de console ;
- ✓ Des applets, qui sont des programmes java incorporés à des pages web ;
- ✓ Des applications pour appareils mobiles, ave J2ME ;
- ✓ et bien d'autres J2EE, JMF, J3D pour la 3D.

Java se voit attribuer plusieurs qualités dont voici quelques-unes :

- **Orienté objet** : la programmation orientée objets présente l'immense intérêt de faciliter la réutilisabilité des composants développés. Java est un langage orienté objet. Ecrire un programme revient à écrire une classe d'objets avec ses données et les méthodes qui permettent d'y accéder.
- **Simple :** java hérite une grande partie de la syntaxe du langage C++, et dans le but d'écrire des codes facilement et sans erreurs, il en a été dépouillé de tous les mécanismes complexes, redondants ou devenus inutiles, tels que la gestion des pointeurs, la gestion de la mémoire (la libération de la mémoire en particulier n'est plus à la charge du développeur mais est générée de manière automatique par ramasse-miettes intégré), l'héritage multiple,…etc.
- **Robuste :** plusieurs raisons font que le code généré est effectivement plus robuste qu'avec d'autres langages, et risque donc moins de générer des erreurs :
  - Le développeur n'a pas la possibilité d'accéder aux pointeurs, réduisant ainsi le risque d'écraser des données par erreur dans une zone mémoire.
  - Le mécanisme de gestion des exceptions qui permet une meilleure maîtrise des erreurs. Ce mécanisme permet en effet de gérer des évènements non souhaités (ex : division par zéro) en imposant un traitement adapté (ex : arrêt du programme).
  - La déclaration des variables doit obligatoirement être explicite en java. Le code est vérifié (Syntaxe, type) à la compilation et également au moment de l'exécution, ce qui permet de réduire les bugs et les problèmes d'incompatibilité de version.
- **Sécurisé :** au moment de l'exécution d'un programme java, le JRE utilise un processus nommé la ClassLoader qui s'occupe du chargement du *byte code* (ou langage binaire

intermédiaire) contenu dans les classes java. Le byte code est ensuite analysé afin de contrôler qu'il n'a pas fait de création ou de manipulation de pointeurs en mémoire et également qu'il n'y a pas de violation d'accès.

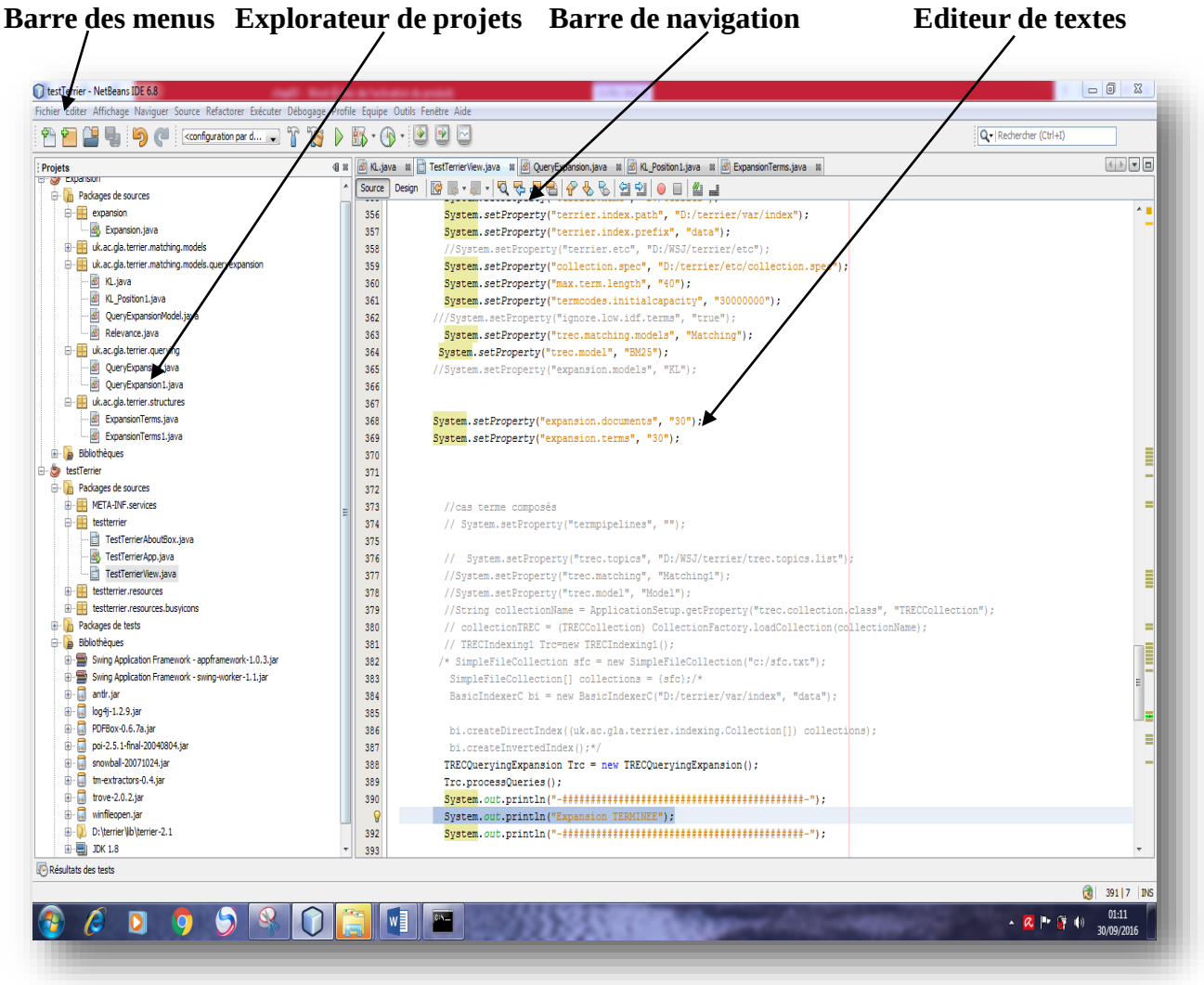
- **Portable :** cette caractéristique est l'une des celles qui ont contribué à leur grande réputation parmi les communautés d'internet et ceci grâce à son indépendance de toute plateforme d'exécution, car un programme java peut tourner sur n'importe quelle machine possédant une JVM.
- **Multi plates-formes :** les programmes tournent sans modification sur tous les environnements (Windows, Unix,…etc.)

III.4.4. NetBeans

NetBeans est à l'origine un EDI (Environnement de Développement Intégré) Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelques parts en 2002, Sun a décidé de rendre NetBeans open-source. Mais NetBeans n'est pas uniquement un EDI java, c'est également une plateforme. Il vous est possible de créer votre propre application Awt ou Swing, basé sur la plateforme NetBeans. Pour celles et ceux d'entre vous qui viennent du monde Eclipse, cela correspond à Eclipse RCP. Sa conception est complètement modulaire : tout est module, même la plateforme. Ce qui fait de NetBeans une boite à outils facilement améliorable ou modifiable. La licence de NetBeans permet de l'utiliser gratuitement à des fins commerciales ou non. Les modules que vous pourriez écrire peuvent être open-sources comme ils peuvent être closed-source, ils peuvent être gratuits, comme ils peuvent être payants. Il présente une interface conviviale **GUI** (Graphical User Interface) qui nous permet d'éditer, compiler et exécuter un programme écrit en langage java.

NetBeans est téléchargeable sur le site officiel www.netbeans.org.

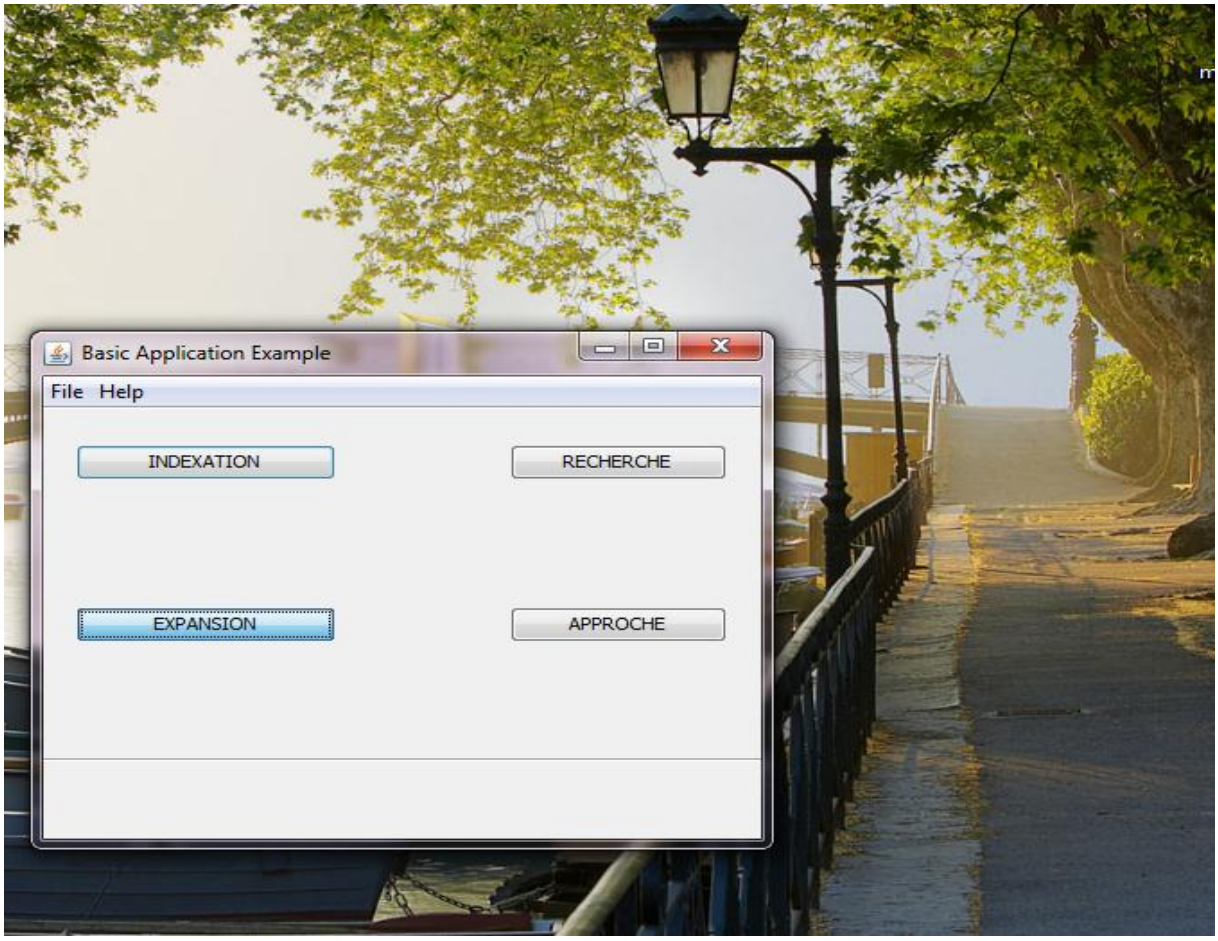
La figure ci-contre représente un aperçu de l'interface de l'IDE NetBeans.



FigureIII.5 : L'environnement de développement NetBeans

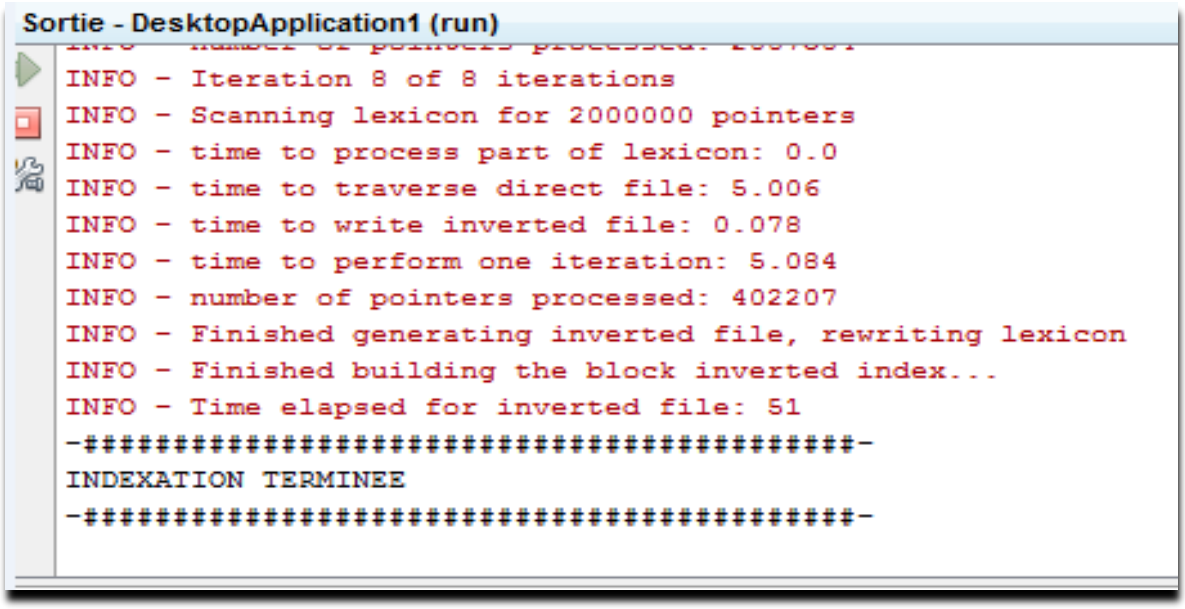
1. Présentation des interfaces

Nous présentons ci-dessous l'interface principale de notre application mettant en œuvre notre approche.

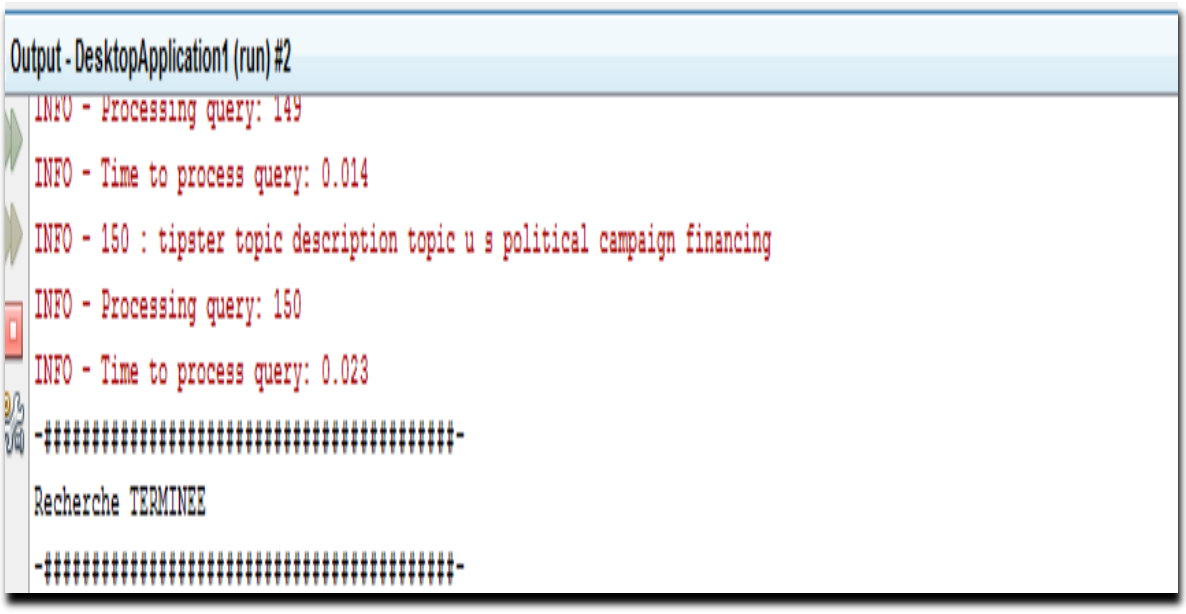


Cette interface permet de choisir le processus à traiter en cliquant sur l'un des boutons Indexation, Recherche ou Expansion

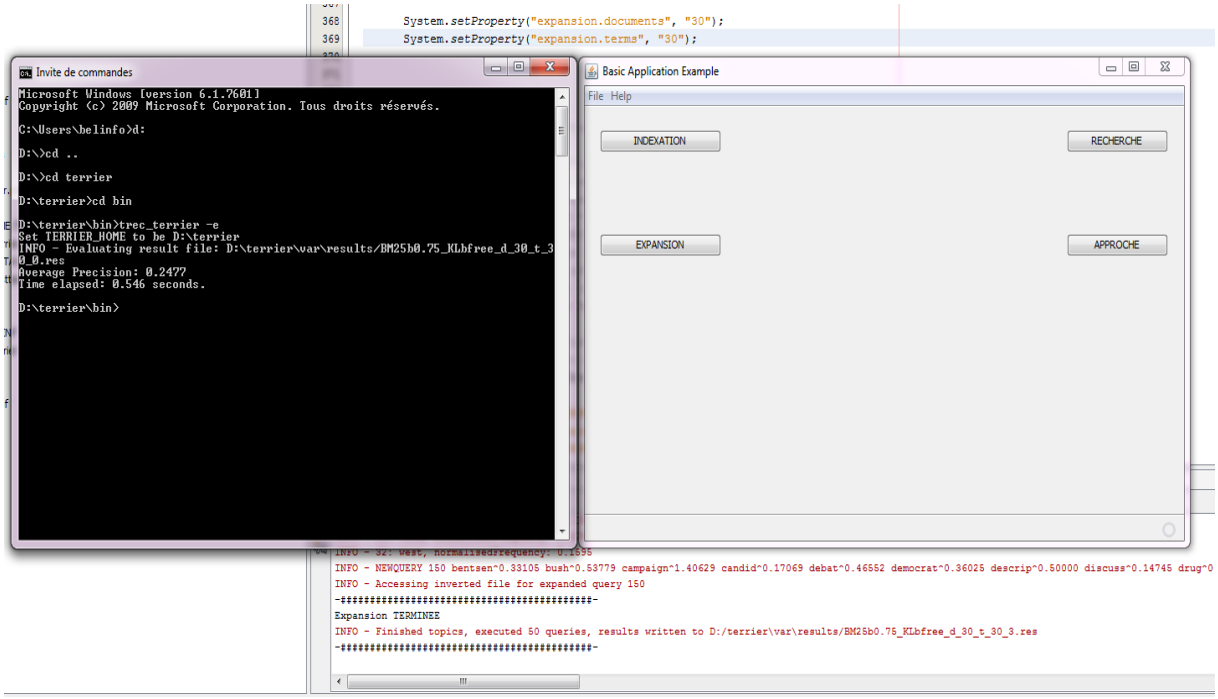
Une fois le processus d'indexation est terminé, les structure de données (fichier direct, fichier inverse, lexicon, … etc.) seront créés et enregistrées dans le répertoire index spécifié.



Via cette interface nous pouvons rechercher dans l'index créés précédemment au niveau de l'indexation. Et ce en spécifiant les requêtes et l'emplacement de l'index ainsi que le modèle de pondération et en cliquant sur le bouton « recherche ». À la fin de ce processus un fichier (.res) contenant le résultat de la recherche sera créé.



L'evaluation des résultats de la recherche se fait via la consonne "MS-DOS" et cela en spécifiant le fichier résultat (.res) à évaluer ainsi que les jugements de pertinence (qrels).



III.5. Résultats et expérimentation

Dans cette section nous allons présenter les expérimentations réalisées ainsi que les résultats obtenus avec notre approche.

III.5.1. Implémentation de notre approche sous terrier

Pour réaliser notre approche nous avons modifié et ajouter quelques classes : Le « Matching », ExpansionTerm »et « QueryExpansion » qui font partie de la plate-forme Terrier. Et la classe étendu dans notre approche est **« QueryExpansion »**.

//Classe Etendue

//Récupération des positions d'un terme dans la classe **KL_position**. Extraire de la classe abstraite QueryExpansionModel

POSITION [ ][ ]

//Classe abstraite

//utilisation d'un tableau **ArrayList** ; puis; créer une matrice de position du terme dans les top documents **Mat_position[][]**ensuite ajouter le terme à la liste **list.add(terms[0][j])** puis Recopier la matrice dans une matrice convenable en terme de **Mat_position_final[s]** enfin entrer dans la fonction qui calcul la couverture spatiale d 'un document selon le nombre de fenêtre (implémentation de KL).

La classe **« Macthing »** : elle s'occupe de la correspondance entre la requête et les documents de l'index via la méthode Macth qui nécessite le numéro de la requête et les termes de la requête et produit résultat qui contient l'identificateur des documents et le score des documents comme résultat.

Pour avoir ce résultat elle traite tous les termes (terme par terme) en consultant l'inverted index pour récupérer tous les documents qui contient le terme et ces derniers sont représentés sous forme de deux tableaux qui représentent les identificateurs des données et la fréquence du terme dans les documents.

Après avoir récupéré ces informations, elle initialise WeightingModels qui assigne un score pour chaque terme de la requête dans le document (pondération), plusieurs modèles de pondération sont implémentés : TF_IDF, BM25…

La classe **« QueryExpansion »** : Terrier inclut la pseudo-relevance feedback automatique, sous forme de QueyrExpansion. Cette méthode fonctionne en faisant l'extraction des tops terme informatives à partir des tops documents classées (un nombre de document spécifié) par attribution du score pour chaque terme en utilisant le modèle d'expansion Bo1 par défaut et les ajouter à la requête. Elle nécessite la requête initiale Set(les identificateurs des tops documents et leurs score), elle modélise le nombre et la taille totale des documents d'expansion.

La classe **« ExpansionTerms »** :néccessite les trois paramètres collection statistic. La taille des documents et le lexicon) et elle modélise les termes d'expansion après insertion des termes avec leurs fréquences de chaque document qui appartient au top document àl'aide de la méthode insert terms (Identificateur du termeet sa fréquence) après récupération des termesd'expansionà partir Expansion.

Nous allons étendre la classe **« QueryExpansion »**.En se basant sur les étapes dans lesquelles nous allons proposer une extension de modèle Kullback leibler divergence (KL)en se basant sur des formules probabilistes que nous allons évaluées et ajoutées à la classe étendue.

III.5.2. Collection de test utilisée

Nous avons mené nos expérimentations en utilisant la collection de test TREC *AP88* (Associated Press news wire, 1988).

Pour la recherche nous avons utilisé **50** requêtes (seulement le titre), issues des topics numérotées **« 101-151 »** de la collection TREC. Cette dernière contient des documents plats, elle est de taille moyenne. Le tableau ci-dessous montre quelques statistiques sur la collection et les topics utilisées.

Collection	Taille	Documents	Topics
AP88	237Mo	79919	101-150

Tableau III.1 : Statistiques sur la collection de test et les topics utilisées

Un document est identifié comme suit :

<DOC>
<DOCNO> AP880212-0001 </DOCNO>
<FILEID> AP-NR-02-12-88 2344EST</FILEID>
<FIRST>u i AM-Vietnam-Amnesty 02-12 0398</FIRST>
<SECOND> AM-Vietnam-Amnesty, 0411</SECOND>
<HEAD>Report Former Saigon Officials Released from RE-education Camp</HEAD>
<DATELINE>BANGKOK, Thailand (AP) </DATELINE>
<TEXT>
MORE than 150former officers of the overthrown South Vietnamese government have been released from a re-education camp after years of detention, the official Vietnam News agency reported Saturday.
The report from Hanoi, moitored in Bangkok, did not give specific figures, but said those freed Friday included an ex-Cabinet minister, a deputy minister, 10 generals, 115 field-grade officers and 25 chaplains.

It quoted Col. Luu Van ham, director of the Nam Ha camp south of Hanoi, as saying all 700 former South Vietnamese officials who had held at the camp now has released……..He said many of former inmates would return to their families in Ho Chi Minh City, formerly the South Vietnamese capital of Saigon.The amnesties apparently are part of efforts by Communist party chief Nguyen Van linh to heal internal divisions and improve Vietnam's image abroad.

_____

Exemple de Requête : Dans Terrier le fichier / **etc**->trec.topics.list, nous spécifier quel fichier contient les requêtes à traiter (#add the topic files to use for querying….**d://topic-101-150.txt**)

<top>
<head> Tipster Topic DescriptionTipster
<num> Number: **101**
<dom> Domain: Science and Technology
<title> Topic: Design of the "Star Wars" Anti-missile Defense System
<desc> Description:
Document will provide information on the proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system…
<def> Definition(s):

</top>

III.5.3. Evaluation et résultats

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base à partir desquels nous allons construire un repère. Ces résultats seront par la suite comparés à ceux obtenus après la reformulation avec notre approche, en appliquant les mêmes paramètres de recherche (nombre de documents d'expansion, nombre de termes d'expansion, modèle de recherche de base (modèle BM25) et le modèle d'expansion (KL-Divergence)). Pour l'évaluation des résultats, nous avons utilisé la MAP (précision moyenne).

III.5.3.1. Résultats obtenus avec la recherche simple

L'objectif de la recherche simple est de déterminer la meilleure valeur de précision (Map) du modèle de recherche BM25. Le tableau suivant montre les résultats obtenus.

Recherche simple	
Modèle de recherche	Map
BM25	0.1105

Tableau III.2 : Précision moyenne obtenue avec la recherche simple.

III.5.3.2. Résultats obtenus avec le modèle de base BM25

Dans l'étape de l'expansion, nous avons remplacé les deux classes ExpansionTerms et QueryExpansion situées dans la librairie de Terrier, avec celles utilisant le modèle de que nous avons développés. Plus précisément, la classe ExpansionTerms s'est mise à l'emplacement *D:\terrier\lib\terrier-2.1\uk\ac\gla\terrier\structures*,et la classe QueryExpansion s'est mise à son tour à l'emplacement *D:\terrier\lib\terrier-2.1\uk\ac\gla\terrier\querying*. Nous avons ensuite testé l'expansion, en faisant varier le nombre de documents d'expansion de**5**, puis de **5** à **30** avec un pas de **5**et celui des termes d'expansion de **5**, puis de **5** à **30** avec un pas de **5**. Les résultats obtenus avec le modèle de pertinence sont montrés dans le tableau ci-dessous.

Documents

Termes

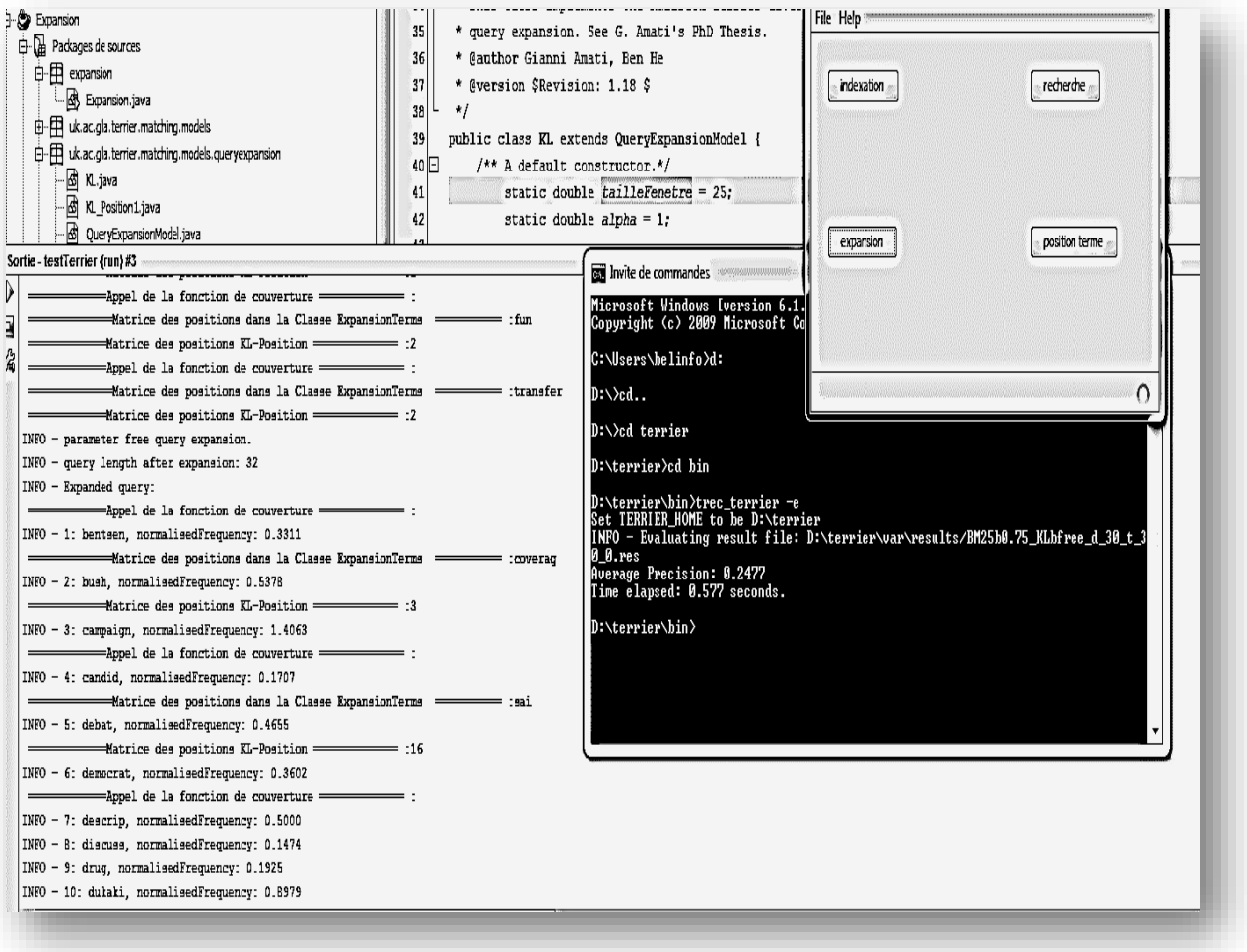
MAP	**1**	**5**	**10**	**15**	**20**	**25**	**30**
1	0,1633	0,1633	0,1633	0,1633	0,1633	0,1633	0,1633
5	0,2135	0,2135	0,212	0,2156	0,2156	0,2168	0,2162
10	0,2197	0,2197	0,2214	0,2236	0,2248	0,2276	0,2282
15	0,2247	0,2247	0,2272	0,2283	0,2317	0,2367	0,2272
20	0,221	0,221	0,2227	0,2244	0,226	0,2284	0,2284
25	0,2305	0,2305	0,2325	0,2374	0,2395	0,2387	0,2403
30	0,2282	0,2282	0,2292	0,2367	0,2386	0,2387	**0,2422**

MAP	**35**	**40**	**45**	**50**
1	0.2358	0.2358	0.2358	0.2358
5	0.2358	0,2340	0,2378	0,2368
10	0.2350	0,2272	0,2267	0,2389
15	0.2335	0.2282	0,2272	0,2389
20	0.2280	0.2267	0,2227	0,2403
25	0.2271	0.2320	0,2325	0,2408
30	0.2271	0,2282	0,2345	0,2412

TableauIII.3 : Précision moyenne en faisant varier le nombre de documents et le nombre de termes d'expansion.

D'après le tableau ci-dessus, nous constatons que le modèle de base utilisé pour l'expansion a apporté des améliorations notables. Le nombre de documents et celui de termes d'expansion jouent un rôle important dans l'amélioration de la précision. Par ailleurs, le meilleur résultat est obtenu avec un nombre de documents d'expansion égal à **30**, et un nombre de termes d'expansion égal à **30**, avec une précision moyenne de **0.2422.**Pour cela, nous avons alors testé notre approche en considérant le même paramètre (**30 documents et termes**).

III.5.3.3. Résultats obtenus avec notre approche



Nous avons remplacé les deux classes KL et KL position situées dans les classes de l'expansion ; avec celles dans la librairie de Terrier, Plus précisément, les classes. De l'emplacementse trouve dans *D:*\Expansion\build\classes\uk\ac\gla\terrier\macthing\models\ *Queryexpansion* ; s'est mise à l'emplacement dans *D:\terrier\lib\terrier-2.1\uk\gla\.terrier \.macthing\model\queryexpansion.* Nous avons ensuite testé l'expansion ; en faisant varier le

nombre de fenêtres de 1 à 200 avec un pas de 25 ; et un constant alpha qui est égale à 1 ; en considérant les mêmes paramètres (**30 documents et 30 termes).**

Le tableau suivant montre les résultats obtenus après l'expansion avec le modèle de base BM25 avec les différentes les paramètres proposées précédemment.

Alpha	Taille de la fenêtre	MAP
1	1	0.2459
	25	**0.2477**
	50	0.2450
	75	0.2424
	100	0.2432
	125	0.2448
	150	0.2451
	175	0.2442
	200	0.2450

Tableau III.4 : Précision moyenne après l'expansion avec le modèle KL.

D'après le tableau ci-dessus et représenté, les résultats obtenus après l'expansion avec le modèle KL : nous montre que le modèle de pertinence étendu avec un nombre de fenêtre qui est égale à**25**, nous apporte une importante amélioration avec une précision moyenne(MAP)de **0.2477**par rapport au modèle de pertinence de base qui a obtenu une précision moyenne (MAP)de **0.2422.**

III.5.3.3.1. Evaluation requête par requête

Afin d'avoir une interprétation plus précise des résultats obtenus précédemment, nous avons évalué les résultats des meilleures précisions requête-par-requête. Ainsi nous avons obtenu les tableaux suivants :

Num Requête	MAP (Recherche Simple)	Num Requête	MAP (Modèle BM25)	Observation des résultats de Requête
101	0,1555	101	0,817	améliorée
102	0,4129	102	0,4671	améliorée
103	0,0537	103	0,0639	améliorée
104	0,1804	104	0,695	améliorée
105	0,0000	105	0	nulle
106	0,0789	106	0,1025	améliorée
107	0,0921	107	0,2687	améliorée
108	0,0391	108	0,1784	améliorée
109	0,0000	109	0	nulle
110	0,2540	110	0,3905	améliorée
111	0,3680	111	0,736	améliorée
112	0,2097	112	0,7106	améliorée
113	0,0291	113	0,0268	dégradée
114	0,1582	114	0,2279	améliorée
115	0,0737	115	0,3335	améliorée
116	0,0000	116	0	nulle
117	0,1029	117	0,5919	améliorée
118	0,0337	118	0,0245	dégradée
119	0,0431	119	0,0967	améliorée
120	0,0029	120	0,0017	dégradée
121	0,0008	121	0,0027	améliorée
122	0,0301	122	0,688	améliorée
123	0,0374	123	0,173	améliorée
124	0,2270	124	0,4583	améliorée
125	0,0375	125	0,2074	améliorée
126	0,0866	126	0,3852	améliorée
127	0,0285	127	0,2264	améliorée
128	0,0047	128	0,0016	dégradée
129	0,0467	129	0,0516	améliorée

130	0,2855	130	0,6205	améliorée
131	0,0590	131	0,044	dégradée
132	0,3872	132	0,8901	améliorée
133	0,7095	133	0,6756	dégradée
134	0,7500	134	0,8667	améliorée
135	0,0854	135	0,625	améliorée
136	0,0000	136	0	nulle
137	0,0112	137	0,0032	dégradée
138	0,0545	138	0,1113	améliorée
139	0,0498	139	0,0521	améliorée
140	0,0239	140	0,0243	améliorée
141	0,0704	141	0,0735	améliorée
142	0,0981	142	0,2447	améliorée
143	0,0021	143	0,0019	dégradée
144	0,0005	144	0,0004	dégradée
145	0,0627	145	0,3078	améliorée
146	0,2367	146	0,6383	améliorée
147	0,0101	147	0,0074	dégradée
148	0,0124	148	0,0024	dégradée
149	0,005	149	0,0012	dégradée
150	0,0162	150	0,0054	dégradée

Tableau III.5 : Résultats obtenus avec l'analyse requête par requête avant et après l'expansion avec le modèle de pertinence

Le tableau ci-dessus montre le modèle de pertinence (KL-Divergence) par rapport à la recherche simple où nous avons obtenus les résultats suivants :

- **33** Requêtes améliorée par rapport au modèle de Recherche simple.
- **13** Requêtes dégradées par rapport au modèle de Recherche simple.
- **4** Requête nulles par rapport au modèle de Recherche simple.

Num de requête	MAP (recherche Simple)	Num de Requête)	MAP (notre Approche)	Observation des résultats de requête
101	0,1555	101	0,067	dégradée
102	0,4129	102	0,3413	dégradée
103	0,0537	103	0,018	dégradée
104	0,1804	104	0,655	améliorée
105	0,0000	105	0	nulle
106	0,0789	106	0,1135	améliorée
107	0,0921	107	0,5011	améliorée
108	0,0391	108	0,251	améliorée
109	0,0000	109	0	nulle
110	0,2540	110	0,4237	améliorée
111	0,3680	111	0,767	améliorée
112	0,2097	112	0,7485	améliorée
113	0,0291	113	0,0243	dégradée
114	0,1582	114	0,2257	améliorée
115	0,0737	115	0,3345	améliorée
116	0,0000	116	0	nulle
117	0,1029	117	0,5875	améliorée
118	0,0337	118	0,207	améliorée
119	0,0431	119	0,0989	améliorée
120	0,0029	120	0,0003	dégradée
121	0,0008	121	0,0027	améliorée
122	0,0301	122	0,0753	améliorée
123	0,0374	123	0,2165	améliorée
124	0,2270	124	0,4534	améliorée
125	0,0375	125	0,2059	améliorée
126	0,0866	126	0,4503	améliorée
127	0,0285	127	0,2943	améliorée
128	0,0047	128	0,002	dégradée

129	0,0467	129	0,0325	dégradée
130	0,2855	130	0,7223	améliorée
131	0,0590	131	0,0452	dégradée
132	0,3872	132	0,8901	améliorée
133	0,7095	133	0,7542	améliorée
134	0,7500	134	0,7	dégradée
135	0,0854	135	0,6415	améliorée
136	0,0000	136	0	nulle
137	0,0112	137	0,0015	dégradée
138	0,0545	138	0,1097	améliorée
139	0,0498	139	0,0473	dégradée
140	0,0239	140	0,0187	dégradée
141	0,0704	141	0,0735	améliorée
142	0,0981	142	0,2498	améliorée
143	0,0021	143	0,0016	dégradée
144	0,0005	144	0,0003	dégradée
145	0,0627	145	0,314	améliorée
146	0,2367	146	0,6454	améliorée
147	0,0101	147	0,0096	dégradée
148	0,0124	148	0,0008	dégradée
149	0,005	149	0,0006	dégradée
150	0,0162	150	0,004	dégradée

Tableau III.6 : résultats d'évaluation requête par requête avant et après l'expansion avec notre approche. Sur la collection AP88

Les résultats obtenus dans ce tableau nous donnent les résultats suivants :

- **28** Requêtes améliorées.
- **4** requêtes nulles.
- **18** requêtes dégradées.

NUM Requête	MAP (Modèle BM25)	MAP (Notre Approche)	NUM Requête	Observation du résultat de requête
101	0,817	101	0,067	Dégradée
102	0,4671	102	0,3413	Dégradée
103	0,0639	103	0,018	Dégradée
104	0,695	104	0,655	Dégradée
105	0	105		Nulle
106	0,1025	106	0,1135	Améliorée
107	0,2687	107	0,5011	Améliorée
108	0,1784	108	0,251	Améliorée
109	0	109	0	Nulle
110	0,3905	110	0,4237	Améliorée
111	0,736	111	0,767	Améliorée
112	0,7106	112	0,7485	Améliorée
113	0,0268	113	0,0243	Améliorée
114	0,2279	114	0,2257	Dégradée
115	0,3335	115	0,3345	Améliorée
116	0	116	0	Nulle
117	0,5919	117	0,5875	Dégradée
118	0,0245	118	0,207	Améliorée
119	0,0967	119	0,0989	Améliorée
120	0,0017	120	0,0003	Dégradée
121	0,0027	121	0,0027	Nulle
122	0,688	122	0,0753	Dégradée
123	0,173	123	0,2165	Améliorée
124	0,4583	124	0,4534	Dégradée
125	0,2074	125	0,2059	Dégradée
126	0,3852	126	0,4503	Améliorée
127	0,2264	127	0,2943	Améliorée
128	0,0016	128	0,002	Améliorée
129	0,0516	129	0,0325	Dégradée

130	0,6205	130	0,7223	Améliorée
131	0,044	131	0,0452	Améliorée
132	0,8901	132	0,8901	Nulle
133	0,6756	133	0,7542	Améliorée
134	0,8667	134	0,7	Dégradée
135	0,625	135	0,6415	Améliorée
136	0	136	0	Nulle
137	0,0032	137	0,0015	Dégradée
138	0,1113	138	0,1097	Dégradée
139	0,0521	139	0,0473	Dégradée
140	0,0243	140	0,0187	Dégradée
141	0,0735	141	0,0735	Nulle
142	0,2447	142	0,2498	Améliorée
143	0,0019	143	0,0016	Dégradée
144	0,0004	144	0,0003	Dégradée
145	0,3078	145	0,314	Améliorée
146	0,6383	146	0,6454	Améliorée
147	0,0074	147	0,0096	Améliorée
148	0,0024	148	0,0008	Dégradée
149	0,0012	149	0,0006	Dégradée
150	0,0054	150	0,004	Dégradée

Tableau III.7 : Comparaison de l'analyse requête par requête dans l'expansion avec notre approche et celle obtenue avec le modèle de pertinence

Le tableau ci-dessus montre l'amélioration de notre approche par rapport au modèle de pertinence où nous avons obtenus les résultats suivants :

- **22** Requêtes améliorées.
- **21** Requêtes dégradée.
- **07** Requêtes nulles

Les résultats de l'évaluation requête-par-requête, nous montre que notre approche obtient des meilleurs résultats en utilisant la meilleure configuration en terme de document et de nombre de termes qui est égal à **30,30** avec une taille de la fenêtre égal à **25**.

III.6. Conclusion

Dans ce chapitre, nous avons d'une part exposé notre approche qui consiste à étendre le modèle d'expansion KL-Divergence avec le facteur de position des termes d'expansion.

Nous avons présenté les détails de l'implémentation ainsi que les expérimentations et les résultats associés à notre approche ; comme nous avons effectué une comparaison des résultats entre le« modèle de KL-Divergence » de base, et celui obtenu de« notre approche » sur une collection de Test TREC AP88. Nous avons constaté que notre modèle améliore les résultats de modèle de base, ce qui indique que la position des termes est un bon facteur pour améliorer la pertinence de la RI.

CONCLUSION GENERALE

Conclusion Générale

La dernière partie de ce mémoire est pour nous l'occasion de revenir sur les éléments essentiels que nous avons présentés, d'en discuter quelques points. En particulièrement au niveau de l'expansion de requête et quelques perspectives pour ce travail.

Notre objectif consistait introduire les positions des termes dans les documents pertinents, dans l'objectif de calculer la couverture d'un terme vis-à-vis d'un document. Une fois que la couverture des termes est calculée, les termes d'expansion sont classés. Pour cela nous avons choisi d'étendre le modèle KL-Divergence.

Les expérimentations réalisées sont des expérimentations de grandeur nature. C'est-à-dire nous avons utilisé les collections utilisés dans le domaine, à savoir les collections TREC. Précisément, nous avons utilisé la collection de test : AP88.

Les résultats obtenus étaient encourageant. En effet des améliorations ont été constatées en introduisant le facteur de position des termes d'expansion.

La réalisation de ce travail nous a permis tout d'abord de nous imprégner des principaux concepts de cette nouvelle discipline qui est la recherche d'information.

Sur le plan théorique, nous avons pu apprendre à développer, à analyser et à formuler nos rapports dans un cadre pédagogique.

Sur le plan conceptuel, nous avons acquis des connaissances sur le langage JAVA lors de la réalisation de l'implémentation notre approche de travail.

A l'issu du travail élaboré dans ce mémoire, et d'après les résultats obtenus, nous sommes dans la bonne voix sur la collection de test AP88. Donc notre approche très prometteuse dans l'amélioration des performances des Systèmes de Recherche d'information (SRI).

Notre perspective est de tester le facteur de position d'un terme d'expansion dans un document sur d'autres collections pour une meilleure interprétation des résultats.

BIBLIOGRAPHIE

[1]: Rijsbergen.,V. :"**Information Retrieval**". Butterworths & Co, Ltd, London, 1979.

[2]: Salton, G. and M. McGill. :"**Introduction to Modern Information Retrieval**".McGraw-Hill, New York, 1983.

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

[4] : 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.

[5]: Salton., G. and M. McGill, 1984. :"**Introduction to Modern Information Retrieval**".McGraw-Hill, Int. Book Co, 1984.

[6]: Belkin et al., Nicholas J. Belkin, Peter Ingwersen, Annelise Mark Pejtersen : "**Proceedings of the 15th Annual International**". ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992.

[7]: Daoud M. :"**Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche**", thèse de doctorat en informatique, Université Paul Sabatier, 2009.

[8]: Arezki HAMMACHE. :"**Recherche d'Information : un modèle de langue combinant mots simples et mots composés**", Thèse de Doctorat , Université Mouloud Mammeri de Tizi-Ouzou, 2013.

[9] : Cleverdon, C. Progress in documentation :"**Evaluation of information retrieval systems**". Journal of Documentation, 1970.

[10] : Denos, N. :"**Modélisation de la pertinence en recherche d'information**" : modèle conceptuel, formalisation et application,Thèse de Doctorat de l'Université Joseph Fourier-Grenoble I, 1997.

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

[12]: Saracevic. T. :"**Relevance reconsidered. Conceptions of Library and Information Science**", pages 201–218, 1996.

[13] : Mizzaro. S. Relevance. :"**the whole (hi) story**". Journal of the American Society for Information Science, 1997.

[14]: Salton. G.:The SMART retrieval system :" **Experiments in automatic document Processing**". Prentice Hall, 1970.

[15] : Mustapha BAZIZ :"**Indexation conceptuelle guidée par ontologie pour la recherche d'information**". Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, 2005.

[16]: Roussey C.:"**Une méthode d'indexation sémantique adaptée aux corpus multilingues informatique**", thèse de l'INSA de Lyon ,150 pages,2001.

[17] : Ren F., Fan L., Nie J-Y. SAAK Approach :"**How to Acquire Knowledge in an Actual Application System**", IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, pp.136-140,1999.

[18]: Salton. G. and M. J. McGill: "**Introduction to Modern Information Retrieval**". McGraw-Hill, Inc., New York, NY, USA, 1986.

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

[20]: Jacquemin, C., Daille, B., Royanté, J., and Polanco, X. 2002. :"**In vitro evaluation of a program for machine-aided indexing**". Inf. Process. Manage. 38, 6, 765-792, 2002.

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

[22]: Maron, M., and Kuhns, J. :"**On relevance, probabilistic indexing and information retrieval**".Journal of the Association for Computing Machinery 7 pages 216–244.

[23] : Mohand BOUGHANEM : Conférence : "**Introduction : présentation du domaine de la RI, modèles et approches classiques, évaluation**".EARIA'06 conférences 2006. http://www.irit.fr/~Mohand.Boughanem.

[24] : Ho Bao-Quac : "**Vers une indexation structurée basée sur des syntagmes nominaux**", thèse de doctorat, Université Joseph Fourier-Grenoble I, 2004.

[25]: Nongdo Désiré Yawbsom Kompaoré : "**Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes vers un processus de RI adaptatif**", thèse de doctorat, Université Paul Sabatier, 2008.

[26] : Yaël Champclaux: "**Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information**", Université Toulouse, thèse de doctorat, 2009.

[27]: Ricardo Baeza Yates ., Berthier R N.: "**Modern information retrieval**", ACM (Association for Computing Machinery), 2001. .

[28]: Boughanem M. and J. Savoy: "**editors. Recherche d'information états des lieux et perspectives**". Hermès Science Publications, 2008. http://www.editions-hermes.fr/.

[29] : Salton., E.A. Fox, and H. Wu.: "**Extended boolean information retrieval. Commun**." ACM, 26(11) :1022–1036, 1983.

[30] : Lynda TAMINE: "**le système de recherche d'information : reformulation de requêtes et apprentissage basés sur les algorithmes génétiques**", thése doctorat, Université Mouloud MAMMERI de Tizi-Ouzou, Institut d'informatique, 1998.

[31]: Gerard & McGILL, Michaek: "**Introduction to modern Information Retrieval**". New York : McGraw-Hill Book Company, 1983.

[32] : Rami Harrathi : "**Recherche d'information conceptuelle dans les documents semi-structurés**", Thèse en vue de l'obtention du grade de docteur Institut National des Sciences Appliquées de Lyon, 2010.

[33] : Wassila Azzoug :"**contribution a la définition d'une approche d'indexation sémantique de document textuels**", mémoire de magister Université M'hamed Bougara, Boumerdes, 2007.

[34]: Jones.K., S.:"**A statistical interpretation of term specificity and its application in retrieval**".Journal of Documentation 28 111–121, 1972.

[35]: Sun & al.:"**Mining dependency relations for query expansion in passage retrieval**".In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, 382–389, 2006**.**

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

[37]: Xu. J. & W.B.Croft :"**Query Expansion Using Local and Global Documents Analysis. In Proc**", ACM SIGIR Annual Conference on Research and Development, Zurich, 1995.

[38] :M'Hamed Mataoui :"**reformulation de requête dans les systèmes de recherche d'information dans les documents XML**", thèse de magister , université M'hmed Bougara de Boumerdes, 2007.

[39] : Salton G., C. S. Yang, and C. T. Yu.:"**A theory of term importance in automatic text analysis**". Journal of the American Society for Information Science and Technology, 26(1):33–44, 1975.

[40] : Folz.P, Laham. D. :"**An introduction to Latent Semantique Analysis**", Discover process", vol, 25, pp 259-284, 1998.

[41] : Voorhees, E."**Using WordNet to Disambiguate Word Senses for Text Retrieval** ", Processings of the Annual Conference on Research and Development in Information Retrieval, SIGIR 93, Pittsburgh, PA, 1993.

[42] : Robertson. S. :"**The probabilistic ranking principle in IR**".Journal of Documentation, 33:294–304, 1977.

[43]: Buckley, C. G. Salton & J. Allan: "**The Effect of Adding Information in a Relevance Feedback Environment**", Conference on Research and Development in Information Retrieval (SIGIR), 1994.

[44]: Robertson. S. and S. Walker.: "**Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval**". Journal of Documentation, 33 :294–304, 1994.

[45]: Haines. D. & W.B Croft :"**Relevance Feedback and Inference Networks, Conference on Research and Development in Information Retrieval**" (SIGIR), pp 2-11, 1993.

[46] : Wong, W,S., Luk, R.W, P:"**examining the effects of relevance information in a relevance feedback environnment information proccessing and Management**" 44, 3, 1086-1116, 2008.

[47] : Amati, G, Carpineto, C, and Romano, G: "**comparing weightingmodèls monological information retrieval in procedings of the WorkShop of the Language Evaluation Forum**", CLEF, Springer, Trondheim, Norway, 310-318, 2003.

[48]: Salton.G. & C. Buckley :"**Improving Retrieval Performance by Relevance Feedback, Journal of the American Society for Information Science**", Vol. 41, N°4, p 288-297, 1990.

[49]: Latiri & al :"**Query expansion using fuzzy association rules between terms**".In Proceedings of the 4th International Conference Journ´ees de l'InformatiqueMessine (JIM'03), 2004 **.**

[50]: Saracevic.T. :"**Relevance reconsidered. Conceptions of Library and Information Science**", pages 201–218, 1996.

[51] : Lavrenko, V. AND ALLAN, J. :"**Realtime query expansion in relevance models. IR 473**" , University of Massachusetts**,** 2006.

[52]: Fiana Raiber, Oren Kurland. :"**On Identifying Representative Relevant Documents**" Faculty of Industrial Engineering and Management, 2000.

[53] : Hiemstra. D. :"**Term-specific smoothing for the language modeling approach to information retrieval: the importance of aquery term.InSIGIR '02**".Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pages 35–41, New York, NY, USA,ACM Press, 2002.

[54]: Cormack. G., C. R. Palme, M. V Biesbrouk & C. L. A. Clarck. :"**Deriving Very Short Queries for High Precision and Recall**". In Proceedings of the 7th Text Retrieval Conference, TREC7, 1999.

[55]: Rana EL CHARIF :"**Analyse des paramètres de pondération dans le cadre de collections volumineuses**" :Coopération dans les sciences de traitement de l'information, université Paul Sabatier–I.R.I.T., 34-35, 2006.

[56]: Robertson. S. and K. Spark Jones. :"**Relevance weighting of search terms.Journal of Americain Society for Information Retrieval**", 27(3) :129–146, 1976.

[57]: Robertson, S., Walker, S., Hancock-Beaulieu, M.et Gatford, M. :"**Okapi at TREC-3**".In proceeding of the Third Text Retrieval Conference, TREC'94, 1994.

[58]: Maron. M. and J. Kuhns. On relevance :"**probabilistic indexing and information retrieval**". Journal of the ACM, 7 :216–243, 1960.

[59]: Yuanhua Lv et ChengXiang Zhai :"**propositional Language Models for Information Retrieva**l".In proceeding of the 32nd Annual International ACM SIGIR Conference on Research and Development in information Retrieval (SIGIR'09) , pages 299-306, 2009.

[60]: Robertson & al. S.E Robertson, S.E. Walker & M.M Hnacock-Beaulieu : "**Large Test Collection Experiments on an Operational Interactive System**" : Okapi at TREC, in IP&M, pp 260-345, 1995.