REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE D'ENSEIGNEMENT ET DE RECHERCHE SCIENTIFIQUE

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

Mémoire

De fin d'études
En vue de l'obtention du diplôme de
Master en Informatique
OPTION : Systèmes Informatiques

Thème

Implémentation et évaluation d'une approche d'estimation automatique du nombre de termes d'expansion

Proposé et dirigé par :

Mr HAMMACHE AREZKI

Présenté par :

Mlle FEKKANE Nora
Mlle ABDI Malha

PROMOTION : 2014/2015

Remerciements

Avant tous nous remercions Dieu tous puissant de nous avoir donné la force, la patience, le courage et surtout la volonté nécessaire pour la réalisation de ce modeste travail.

Nous présentons nos remerciements les plus sincères à notre promoteur ***Monsieur Hammache Arezki****, pour avoir été très disponible, pour son aide, ses conseils et pour nous avoir soutenus et encouragés. Qu'il soit ici assuré de notre très grand respect et de notre profonde reconnaissance.*

Nos remerciements s'adressent aussi aux membres du Jury qui nous font l'honneur d'examiner et de juger notre travail.

Nos remerciements vont également à Monsieur BELKAID Mohammed Said Doyen de notre Faculté, nos chefs de service des personnels et de la scolarité, de nous avoir permis de poursuivre nos études, tous nos collègues ATS et Enseignants de la faculté de Génie Electrique et d'Informatique de l'université Mouloud MAMMERI de Tizi-Ouzou.

Merci à nos familles et nos amis(es) et les anciens DEUA de notre promotion pour leurs soutiens et leurs encouragements.

Nous remercions également toute personne ayant contribué de prés ou de loin à l'aboutissement de cette quête.

Dédicaces

Je dédie ce modeste travail:

A la mémoire de mon père,

A ma très chère mère qui ma toujours soutenue et encouragé,

A mes frères et sœurs et à leurs petites familles,

A toute ma famille et belle famille,

A mon binôme Malha,

A tous mes ami(e)s et collègues,

Et bien sûr a mon époux Cherif que je remercie tendrement pour sa patience, son soutien et ses encouragements qui m'ont permis de continuer mes études, et enfin à mes adorables enfants ***Ramy*** *et* ***Lina****.*

Nora.

Dédicaces

Je dédie ce modeste travail A:

La mémoire de mon père et mon grand père

Ma très chère mère

Mon mari Omar

Mon frères Mahrez et sa femme Fazia

Ma sœur Ghania ainsi que son mari Ismail

Mes nièces Dalia, Lina, Dyna et mes neveux Amine et Ghiles

Toute ma famille et belle famille

Mon binôme Nora

Mes ami(e)s et mes collègues

Malha.

Sommaire

Chapitre II : La reformulation de la requête

Chapitre III : Expérimentation et évaluation

Liste des figures

.

Liste des tableaux

Introduction générale

INTRODUCTION GENERALE

Une quantité toujours croissante d'informations électroniques (Internet, CD ROM, etc) est disponible et mise à la disposition du grand public. Le domaine de la recherche d'information (RI) a pour objectif de fournir des méthodes d'accès rapide et efficace à ces informations.

Précisément, La Recherche d'Information (RI) est un domaine qui s'intéresse à la structure, à l'analyse, à l'organisation, au stockage et à la recherche et à la découverte de l'information. Le défit est de pouvoir, parmi un volume important de documents disponibles, trouver ceux qui correspondent au mieux aux besoins en information de l'utilisateur. L'opérationnalisation de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI), ces systèmes ont pour but de mettre en correspondance une représentation du besoin de l'utilisateur avec une représentation du contenu des documents au moyen d'une fonction de correspondance.

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.

Le nombre de termes d'expansion est un des critères de succès de la reformulation de requêtes.

Le travail réalisé dans ce mémoire consiste à implémenter des méthodes d'estimation automatique de nombre de termes d'expansion. La première méthode est basée sur la taille de la requête, la seconde sur le nombre de document d'expansion et en fin la dernière méthode combine les deux premières méthodes.

Pour la réalisation et l'évaluation de nos expérimentations nous avons utilisé la plateforme de RI Terrier.

L'organisation retenue pour la présentation de notre mémoire, s'articule en trois chapitres :

Le premier chapitre présente les concepts et notions du domaine de la recherche d'information d'une manière générale.

Le second chapitre traite la reformulation (expansion) de requêtes et les différents critères de classification des méthodes de reformulation, ainsi que la présentation des étapes du processus de reformulation automatique de requêtes. Le troisième chapitre présente notre approche, les outils utilisés pour son implémentation, la collection de test TREC (AP88) utilisée, ainsi que les résultats d'évaluation obtenus.

Chapitre I

Recherche d'Information

I.1. INTRODUCTION

La Recherche d'Information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer un ensemble de documents à un utilisateur en fonction de son besoin en informations. Le défi est de pouvoir, parmi un volume important de documents, trouver ceux qui correspondent au mieux au besoin d'information d'un 'utilisateur.

Les Systèmes de Recherche d'Information (SRI) ont pour but d'automatiser la tâche de la RI. cela est réaliser par la mise en correspondance entre la représentation du besoin de l'utilisateur (requête) avec la représentation du contenu des documents (fiche ou enregistrement) au moyen d'une fonction de comparaison (ou de similarité).

Ce chapitre a pour objectif de présenter le domaine de la RI, il est organisé comme suit : en premier lieu nous avons présenté les concepts de base de la RI, puis ses processus tels que l'indexation, la reformulation de requêtes et l'appariement document /requête. Par la suite nous avons présenté les différents modèles de RI, et en fin nous terminons le chapitre par la présentation de l'étape d'évaluation des systèmes de recherche d'information.

I.2. Définition de la Recherche d'Information

La recherche d'information selon **Salton[1] :"**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[2], "** *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 de 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. Les Systèmes de Recherche d'Information

Selon **Alan Smeaton [3]** : « Le but d'un système de recherche d'information est de retrouver des documents en réponse à une requête des utilisateurs, de manière à ce que les contenus des documents soient pertinents au besoin initial d'information de l'utilisateur ».

Un système de recherche d'information est défini par un langage de représentation des documents (qui peut s'appliquer à différents corpus de documents) et des requêtes qui expriment un besoin de l'utilisateur (sous forme de mots-clés par exemple), et une fonction de

mise en correspondance du besoin de l'utilisateur et du corpus de documents en vue de fournir comme résultats des documents pertinents pour l'utilisateur, c'est-à-dire répondant à son besoin d'information **[4]**. La figure I.1, présente l'architecture d'un système de recherche d'information.

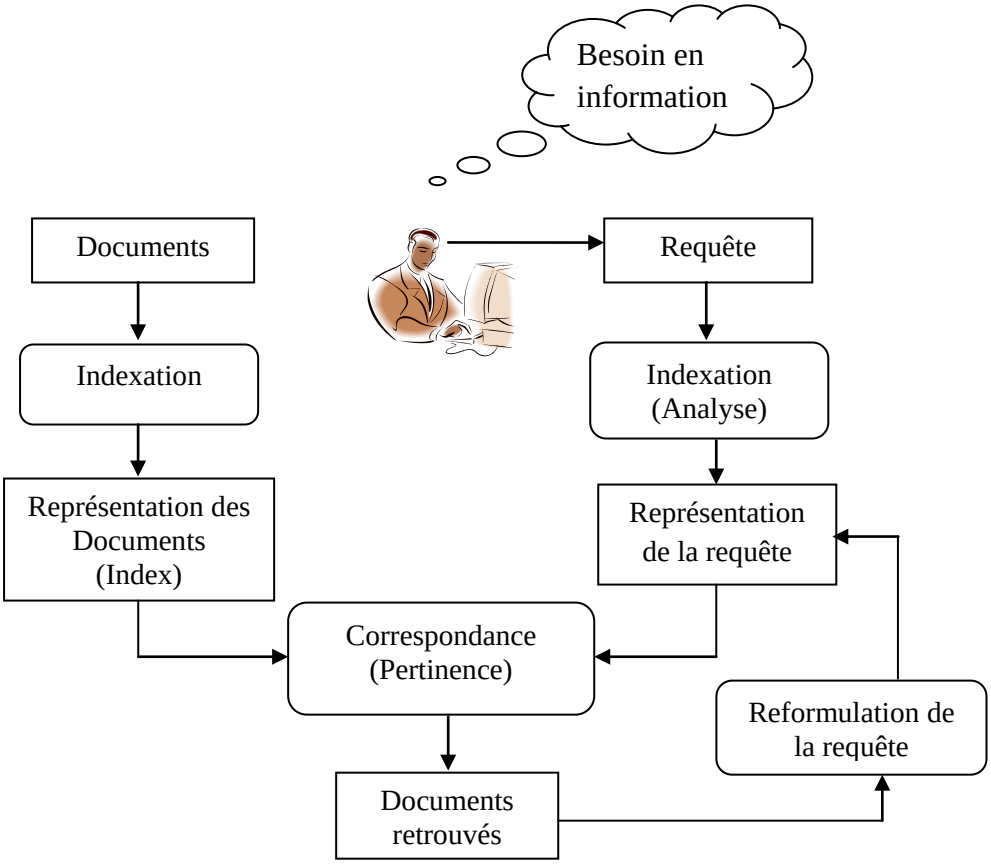

Figure I.1 Architecture générale d'un Système de Recherche d'Information

L'architecture générale d'un SRI illustrée par la figure I.1 fait ressortir des éléments constitutifs tels que : le document, le besoin en information, la requête et la pertinence qui sont les concepts de base de la recherche d'information, ainsi que trois principales fonctionnalités (processus) : l'indexation, la recherche et la reformulation de la requête. Dans ce qui suit, nous détaillons ces éléments et ces processus.

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

I.4.1 Document et collection de documents

Un document est un élément essentiel dans un SRI. Dans son acceptation courante, l'une des définitions possible du terme document est de le considérer comme un support physique de l'information, qui peut être du texte, une page web, une image, une séquence vidéo, etc.

Dans le cas d'un document text on peut le répresenter selon trois vues **[5] [6] :**

La vue sémantique (ou contenu) : elle se concentre sur l'information véhiculée dans le document.

La vue logique : elle définit la structure logique du document (structuration en chapitre, sections)

La vue présentation : elle consiste en la présentation sur un médium à deux dimensions (alignement de paragraphes, indentation, en-têtes et pieds de pages, etc).

l'ensemble des documents manipulés par un SRI se nomme collection de documents (ou base documentaire ou encore corpus).

I.4.2. 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.

Les requêtes soumises au SRI par les utilisateurs peuvent ne pas refléter leurs besoins en information. Cela est dû, d'une part, au fait que l'utilisateur ignore le fonctionnement interne du SRI, et il n'a qu'une vision restreinte des documents disponibles dans la collection.

D'autre part, le SRI n'a souvent aucune connaissance a priori de ses utilisateurs (centres d'intérêt, niveaux, parcours, etc.). Ce biais entre la requête et le besoin en information est une des difficultés majeures de tout système de recherche d'information. Afin de remédier partiellement à ce problème un mécanisme de reformulation de requêtes peut être intégré dans les SRI.

I.4.3. Pertinence

La pertinence est une notion fondamentale dans le domaine de la RI. Elle représente un critère majeur de l'évaluation des performances du système de RI **[7][8].**

Elle peut être définie comme la correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête **[9].**

Elle permet donc d'évaluer jusqu'à quel point les documents restitués traitent le sujet de la requête.

Un document est dit pertinent lorsqu'il satisfait le besoin en information de l'utilisateur, la notion de pertinence est alors fortement subjective, c'est-à-dire dépendante de l'utilisateur. Il existe deux types de pertinence :

***a) Pertinence système* :** cette pertinence 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 **[10].**

b) 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.5. Les processus de la Recherche d'Information

Les SRI se basent sur trois processus afin de resituer une réponse au requête de l'utilisateur, ces processus sont : l'indexation, reformulation de requêtes et l'appariement document/requête.

I.5.1. L'Indexation

L'indexation permet d'extraire à partir d'un document ou d'une requête une représentation paramétrée qui couvre au mieux son contenu sémantique.

Cette étape est adoptée afin que le coût et le temps de réponse de la recherche soient acceptables, c'est-à-dire pour avoir un système de recherche de qualité, il est important que son index reflète au mieux le contenu de la collection originale. Le résultat de l'indexation constitue le descripteur du document ou requête, ce descripteur peut être composé de mots simples, composés ou groupes de mots.

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

I.5.1.1. Indexation manuelle

Cette indexation assure une meilleure pertinence puisqu'elle permet de repérer de façon plus précise les mots clés décrivant un document, mais elle est très coûteuse en terme de temps et subjective du fait qu'elle est fondée sur le jugement d'un être humain. En conséquence, pour un même document, des termes différents peuvent être affectés par des indexeurs différents et il peut même arriver qu'une personne, à des moments différents, indexe différemment le même document **[14].**

I.5.1.2. Indexation semi-automatique

Le processus d'indexation se fait en premier lieu d'une manière automatique, le documentaliste intervient seulement pour ajouter des mots clés qu'il trouve intéressants pour représenter un document **[14].** Dans ce cadre, les indexeurs utilisent un thésaurus ou une base terminologique, qui est une liste organisée de descripteurs (mots clés) obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques.

I.5.1.3. Indexation automatique

Ce type d'indexation est complètement automatisé, il a été mis pour pallier les inconvénients des systèmes manuels. Il comprend un ensemble de traitements automatisés sur les documents qui sont: l'analyse lexicale, l'élimination des mots vides, la normalisation et enfin la création de l'index **[15]**. Ces étapes sont décrites comme suit:

1) **L'analyse lexicale**

Elle permet de convertir un texte du document en une liste de termes. Un terme est un groupe de caractères constituant un mot significatif **[16].** L'analyse lexicale permet de reconnaître et d'éliminer les espaces de séparation des mots, les ponctuations, etc. Certain systèmes permet aussi l'élimination des chiffres.

2) **Elimination des mots vides**

Cette opération consiste à supprimer les termes non significatifs (pronoms personnels, prépositions, etc.) ou les mots athématiques qui peuvent se retrouver dans n'importe quel document, comme par exemple appartenir, contenir, etc. Ce sont des mots qui exposent le sujet d'un document mais n'ayant pas de réels rapports avec le sujet traité**.**

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

La deuxième consiste au comptage du nombre d'apparition d'un mot dans un document de la collection. On supprime les mots ayant une fréquence qui dépasse un certain seuil et qui deviennent alors vraisemblablement des mots vides.

L'élimination des mots vides permet non seulement de réduire le nombre de termes d'indexation, mais aussi peut réduire le taux de rappel, c'est-à-dire le rapport de documents pertinents renvoyés par le système sur l'ensemble des documents pertinents, une notion que nous allons détailler dans la section I.7.

3) Normalisation (lemmatisation ou radicalisation)

On peut trouver dans un texte, différentes formes d'un mot désignant le même sens, On peut par exemple citer économie, économiquement, économétrie, économétrique, etc. Pour cela, on élimine les différences non significatives et on garde la partie commune (radical, lemme), c'est-à-dire un seul mot suffirait pour représenter le concept et c'est lui seul qu'on va indexer.

Plusieurs stratégies de lemmatisation sont utilisées, on cite par exemple selon **Frakes et Baeza-Yates [17] :**

- ***Elimination des affixes* :** On peut par exemple citer l'algorithme de **porter [18]** qui n'est utilisable que pour la langue anglaise. La méthode de radicalisation de Porter permet de construire progressivement le radical en inférant les modifications grammaticales potentielles qui se manifestent le plus souvent par des postfixes ou préfixes. En français [**19**], il n'existe pas d'algorithme fiable pour lemmatisation, par exemple, la complexité des formes prises par les verbes du troisième groupe (le verbe être peut prendre les formes : est, sommes, fûmes..).
- ***Troncature :*** La troncature est utilisée pour éliminer les variations morphologiques ou les variantes orthographiques des mots clés de l'index. Elle consiste à couper le mot à partir d'un rang précis, afin d'obtenir son radical. Pour la langue française, la troncature à 7 caractères est adoptée.
- ***Méthode des n-grammes* :** C'est une représentation originale d'un texte en séquence de N caractères consécutifs, elle est très intéressante pour la radicalisation. Cette modélisation correspond au fait à un modèle de Markov d'ordre n. Prenons l'exemple suivant : Les tri-grammes au niveau de caractères de l'expression «recherche» sont : 'rec', 'her', 'che'**[20].**

4) Création de l'index

Afin d'accélérer la réponse à une requête, des structures de stockages particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants :

- ***Le fichier direct (maître) :*** c'est le fichier de base dans lequel sont stockées les données. Le stockage peut durer quelques secondes sur un fichier direct de quelques centaines d'enregistrements, cependant, il peut se révéler très lent si la base atteint des milliers de documents.

Le tableau ci-dessous présente un exemple du fichier maître.

Document	**Contenu**
d_1	$\underbrace{La}_{1}\ \underbrace{recherche}_{4}\ \underbrace{d'}_{14}\ \underbrace{information}_{16}\ \underbrace{gère}_{28}\ \underbrace{des}_{33}\ \underbrace{textes}_{37}$
d_2	$\underbrace{un}_{1}\ \underbrace{système}_{4}\ \underbrace{de}_{12}\ \underbrace{recherche}_{15}\ \underbrace{d'}_{25}\ \underbrace{information}_{27}\ \underbrace{doit}_{39}$ $\underbrace{restituer}_{44}\ \underbrace{l'}_{54}\ \underbrace{information}_{56}\ \underbrace{pertinente}_{68}\ \underbrace{\text{à}}_{79}\ \underbrace{l'}_{81}\ \underbrace{utilisateur}_{83}$
d_3	$\underbrace{Une}_{1}\ \underbrace{information}_{5}\ \underbrace{est}_{17}\ \underbrace{pertinente}_{21}\ \underbrace{si}_{32}\ \underbrace{elle}_{35}\ \underbrace{satisfait}_{40}\ \underbrace{l'}_{50}\ \underbrace{utilisateur}_{52}$

Tableau 1.1. Exemple de collection (fichier maître)

• ***Le fichier inverse*** **:** Il est créé autour du fichier maître **[21][22].** Ce fichier comme son nom l'indique est le résultat de l'inversion du fichier maître. Plus exactement, au lieu de donner pour chaque document les mots qui le constituent et leurs positions, on donne pour chaque mot les documents qui le contiennent et sa position dans chacun. Le tableau ci-dessous présente un exemple du fichier inverse du fichier maître précédent.

Terme	**d1**	**d2**	**d3**
recherche	4	15	
information	16	27.56	5
gère	28		
textes	37		
système		4	
restituer		44	
pertinente		68	21
utilisateur		83	52
satisfait			40

Tableau 1.2. Exemple de fichier inverse

Exemple d'indexation de document

Nous présentons dans ce qui suit un exemple d'indexation d'un document (pris à partir de la collection TREC que nous présentons dans la section (1.7.3) tout en montrons à chaque étape le résultat obtenu.

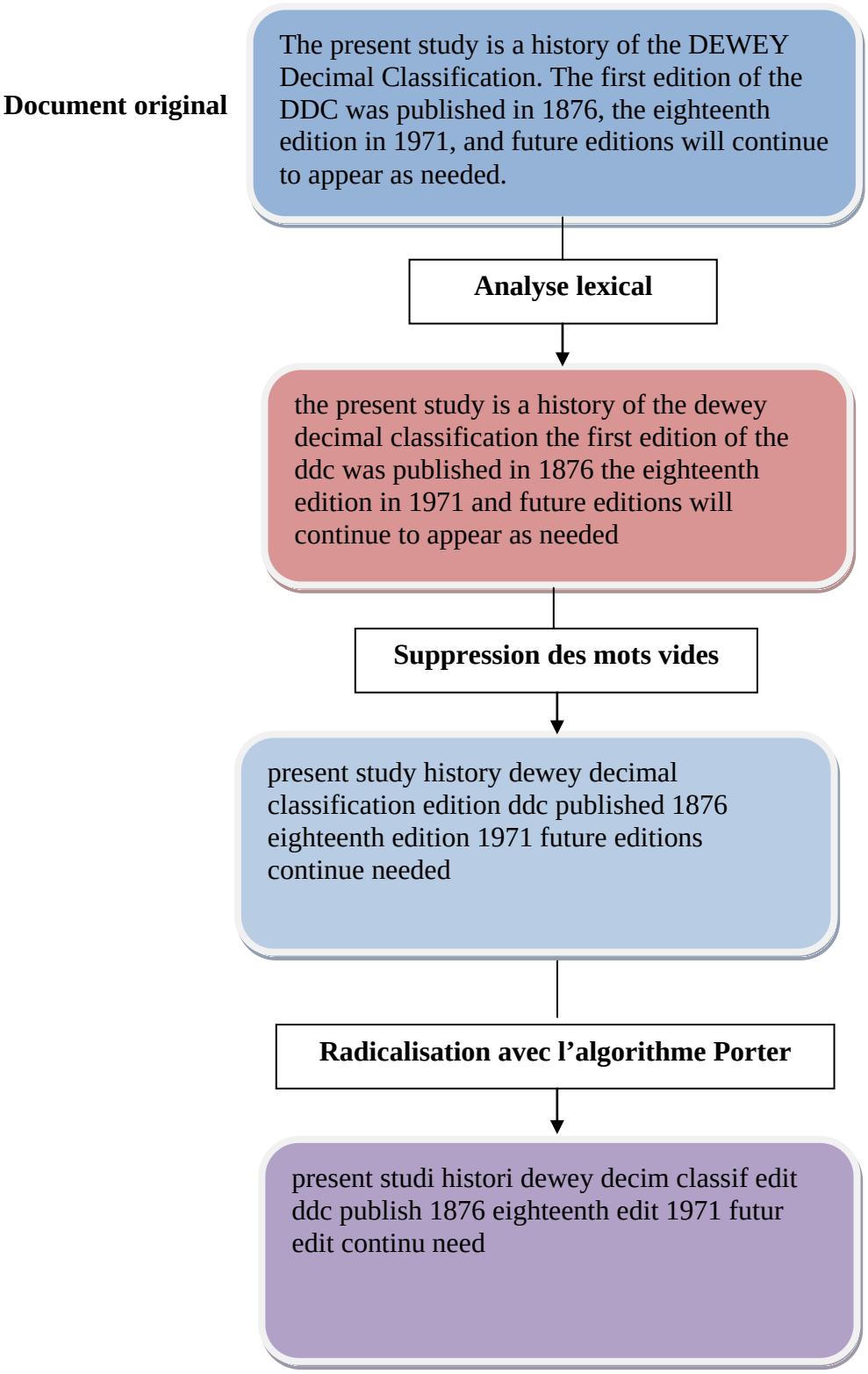

Figure I.2 Exemple d'indexation d'un document

I.5.2. La Reformulation de la Requête :

La reformulation de la requête est un processus qui a pour but d'améliorer la performance du système en offrant un mécanisme de raffinement de la requête utilisateur. Elle est souvent opérée par ajout et/ou réévaluation des poids des termes de la requête initiale.
La reformulation de la requête est détaillée dans le deuxième chapitre.

I.5.3. L'appariement Document/ Requête

L'objectif de tout SRI est de calculer la pertinence d'un document par rapport à une requête. Cette pertinence est basée sur la fonction d'appariement qui consiste à calculer un score représentatif ou la similarité entre chaque document et la requête de l'utilisateur. Ce score de similarité entre le document et la requête est donné par une fonction nommée **R**etrieval **S**tatus **V**alue ; **RSV (Q, D)** où Q est une requête et D un document de la collection.
Un système de RI idéal ne renvoie que les documents pertinents à une requête, tous les documents non pertinents ne sont pas retournés ou doivent être retournés après les documents pertinents, c'est-à-dire la fonction d'appariement permet d'ordonner les documents selon leur degré de pertinence **[97].**

I.6. Les Modèles de Recherche d'Information

Le modèle détermine le comportement clés d'un SRI**.** Son rôle est de créer une représentation interne pour un document ou pour une requête à partir des termes issus de l'indexation et de définir une méthode d'appariement document-requête afin de déterminer leurs degrés de correspondance **[23].**
On distingue trois principaux modèles qui sont: les modèles booléens, vectoriels et probabilistes.

I.6.1. Le modèle booléen

Les premiers SRI développés sont basés sur le modèle booléen, même aujourd'hui beaucoup de systèmes commerciaux (moteurs de recherche) utilisent le modèle booléen. Cela est dû à la simplicité et à la rapidité de sa mise en œuvre.
Le modèle booléen est basé sur la théorie des ensembles et l'algèbre de Boole. Dans ce modèle, un document *d* est représenté par un ensemble de mots-clés (termes) ou encore un vecteur booléen. La requête *q* de l'utilisateur est représentée par une expression logique, composée de termes reliés par des opérateurs logiques : ET ($\wedge$), OU ($\vee$) et SAUF ($\neg$).

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 :

$$RSV\ (d,q) = \begin{cases} 1 & \textit{si d appartient à l'ensemble décrit par q} \\ 0 & \textit{sinon} \end{cases} \qquad (\text{I. } 1)$$

Malgré la large utilisation de ce modèle, il présente un certain nombre de faiblesse :

- Les documents retournés à l'utilisateur ne sont pas ordonnés selon leur pertinence.
- La représentation binaire d'un terme dans un document est peu informative, car elle ne renseigne ni sur la fréquence du terme dans le document ni sur la longueur de document, qui peuvent constituer des informations importantes pour la RI.
- Il est difficile pour les utilisateurs de formuler de bonne requêtes. par conséquent, l'ensemble des documents trouvés est souvent trop grand, pour les requêtes courtes, ou complètement vide dans le cas de requêtes longues.
- Ce modèle ne supporte pas la réinjection de pertinence.
- Les tests effectués sur des collections d'évaluation standards de RI ont montré que les systèmes booléens sont d'une efficacité de recherche inférieure.

A fin de remédier à certains problèmes de ce modèle, des extensions ont été proposées, parmi elles on trouve : le modèle booléen basé sur la théorie des ensembles flous **[23],[24]**, le modèle booléen étendu **[6][97].**

1.6.2. Le modèle vectoriel

Le modèle vectoriel de base a été introduit par **Salton [1],** concrétisé dans le cadre du système SMART. Ce modèle se base sur une formalisation géométrique. En effet, les documents et les requêtes sont représentés dans un même espace, défini par un ensemble de dimensions, chaque dimension représente un terme d'indexation. Les requêtes et les documents sont alors représentés par des vecteurs, dont les composantes représentent le poids du terme d'indexation considéré dans le document (la requête). Formellement, si on a un espace T de termes d'indexation de dimension n, $T = \{t_1, t_2, \ldots, t_j, \ldots t_n\}$. Un document d_i est représenté par un vecteur $d_i(w_{i1}, w_{i2}, \ldots, w_{ij}, \ldots w_{in})$. une requête q par un vecteur $q(w_{q1}, w_{q2}, \ldots, w_{qj}, \ldots w_{qn})$.

Où w_{ij} (resp.w_{qj}) représente le poids du terme t_j dans le document d_i (respectivement dans la requête q).

Le modèle vectoriel offre des moyens pour la prise en compte du poids de terme dans le document. Dans la littérature, plusieurs schémas de pondération ont été proposés. La majorité de ses schémas prennent en compte la pondération locale et la pondération globale **[25].**

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 *t*erm $f$requency), exprimée ainsi :

$$tf_{ij} = 1 + log\ (f(t_i, d_j)) \qquad (I.2)$$

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 dans la collection. Un poids plus importants 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 :

$$idf = log\left(\frac{N}{n_i}\right) \qquad (I.3)$$

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 mesure $tf \times idf$. Cette mesure donne approximation de l'importance du terme dans les collections de documents de taille homogéne. Cependant, un facteur important est ignoré, la taille du document. En effet, la mesure ($tf \times idf$) ainsi définie favorise les documents longs, car ils ont tandance à répéter le même terme, ce qui accroit leur fréquence, par conséquant augmentent la similarité de ces documents vis-avis de la reqête.

Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération, comme facteur de normalisation [**26] [27].**

L'appareiement document-requête dans le modèle vectoriel, consiste à trouver les vecteurs documents qui s'approche le plus de vecteur de la requête. Cet appareiement est obtenu par l'évaluation de la distance entre les deux vecteurs. Plusieurs mesures de similarité ont été définies [**28]**, dont les plus courantes sont décrites dans le tableau 1.3 ci-dessous.

Mesures	**Formules**
Le produit scalaire	$RSV(q,d_i)=\sum_{j=1}^{\lvert T\rvert} w_{qj}.w_{ij}$
La mesure de cosinus	$RSV(q,d_i)=\frac{q.d_i}{\lVert q\rVert.\lVert d_i\rVert}=\frac{\sum_{j=1}^{\lvert T\rvert} w_{qj}.w_{ij}}{\sqrt{\sum_{j=1}^{\lvert T\rvert} w_{qj}^2\ \sum_{j=1}^{\lvert T\rvert} w_{ij}^2}}$
La mesure de Dice	$RSV(q,d_i)=\frac{2\times\sum_{j=1}^{\lvert T\rvert} w_{qj}.w_{ij}}{\sum_{j=1}^{\lvert T\rvert} w_{qj}^2+\sum_{j=1}^{\lvert T\rvert} w_{ij}^2}$
La mesure de Jaccard	$RSV(q,d_i)=\frac{\sum_{j=1}^{\lvert T\rvert} w_{qj}.w_{ij}}{\sum_{j=1}^{\lvert T\rvert} w_{qj}^2+\sum_{j=1}^{\lvert T\rvert} w_{ij}^2-\sum_{j=1}^{\lvert T\rvert} w_{qj}.w_{ij}}$

Tableau 1.3 les mesures de similarité utilisées dans le modèle vectoriel

Le modèle vectoriel caractérisé par sa prise en compte du poids des termes dans les documents, 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 documents 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 **[29]**, le modèle LSI (Latent Semantic Indexing) **[30] [31] [32] [33] [34]** et le modèle connexionniste **[35] [36] [97].**.

I.6.3 Le modèle probabiliste

Ces modèles se basent sur la théorie des probabilités. On distingue principalement le modèle probabiliste de base et le modèle de langue.

I.6.3.1 Le Modèle probabiliste de base

ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête **[37][38]**.

On peut donc classer les documents par ordre de pertinence à l'aide de la fonction RSV(q,d) définie comme suit :

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

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 obtient un meilleur classement.

En appliquant le théorème de Bayes pour les deux probabilités, on obtient :

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

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

Où :

$P(d_i)$: est la probabilité de choisir le document d_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 $(Per|q) + P(NPer|q) = 1$) qui sont fixes.

Après remplacement dans la fonction de tri, on aura la formule suivante :

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

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

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

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

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 :

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

Afin de classer les documents avec cette formule, il faut estimer les valeurs des deux probabilités p_i et q_i. En l'absence de collection (documents) d'apprentissage; on peut attribuer la valeur fixe à p_i comme par exemple 0,5 **[39]**, comme elles peuvent être estimées à l'aide de l'avis de l'utilisateur sur les résultats d'une première recherche (réinjection de pertinence).

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 ne sont pas valables que pour cet utilisateur **[97]**.

I.6.3.2 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:

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

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 **[97]**.

Pour pallier ce problème, des techniques de lissage ont été proposées **[40]**, parmi elles, la technique de lissage de dirichlet**,** le lissage de Laplace, le lissage de Good-Turing ou le lissage de Backoff. Elles consistent à assigner des probabilités non nulles aux termes, qui n'apparaissent pas dans les documents.

I.7. Évaluation de SRI

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

Cette évaluation peut porter sur plusieurs critères : le temps de réponse, la pertinence, la qualité et la présentation des résultats, etc. Le critère le plus important est celui qui mesure la capacité du système à satisfaire le besoin en information de l'utilisateur, c'est à dire la pertinence.

I.7.1 Les Mesures d'évaluation

Le principal objectif d'un système de recherche d'information est de restituer à l'utilisateur tous les documents pertinents et de rejeter tous les documents non pertinents. Cet objectif est évalué à l'aide de différentes mesures d'évaluation **[41].** On présente ci-dessous les plus utilisées.

- **La précision :** La précision mesure la capacité du système à ne retrouver que les documents pertinents pour une requête, c'est-à-dire la proportion de documents pertinents retrouvés parmi tous les documents retrouvés par le système :
  Précision= pertinents retrouvés/documents retrouvés

- **Le rappel:** Le rappel mesure la capacité du système à retrouver tous les documents pertinents pour une requête c'est-à-dire la proportion de documents pertinents retrouvés parmi tous les documents pertinents :
  Rappel= pertinents retrouvés/documents pertinents

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

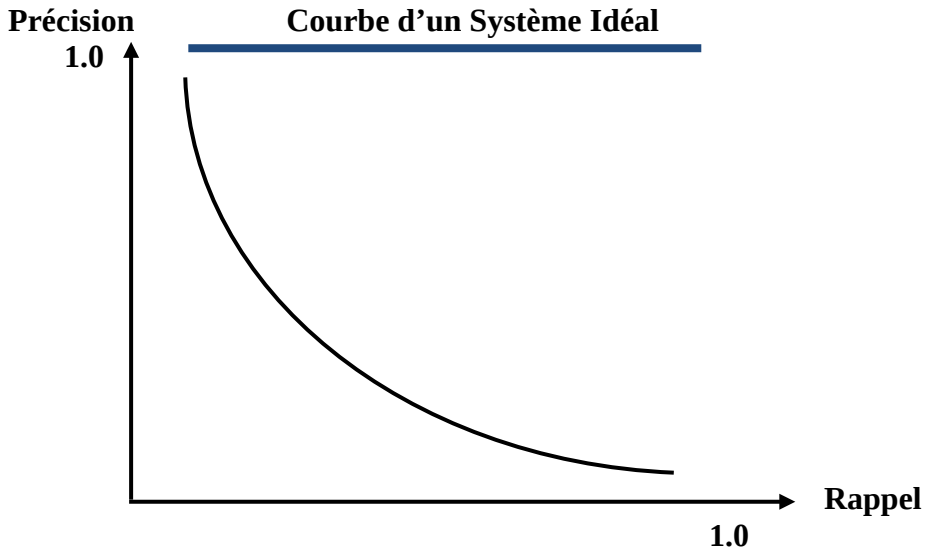

Figure I.3 Allure d'une courbe Rappel et Précision

Cette situation ne se produit pas dans un système réel car le taux de précision et de rappel sont antagoniste. En effet, lorsque la précision augmente, le rappel diminue et inversement. Ainsi, pour mesurer les performances d'un système il faut utiliser les deux mesures conjointement. Cela est réalisé en calculant la paire des mesures (taux de rappel, taux de précision) à chaque document restitué.

Nous considérons par exemple une requête pour laquelle il existe six (6) documents pertinents dans le corpus. Le tableau I.4 illustre le calcul de la précision et de rappel pour les dix (10) premiers documents retournés par un SRI. La lettre (P) précise que le document est pertinent.

Rang du document renvoyé	Pertinence	rappel	Précision
Doc1	P	0.17	1
Doc2		0.17	0.5
Doc3	P	0.33	0.66
Doc4		0.33	0.5
Doc5	P	0.5	0.6
Doc6		0.5	0.5
Doc7	P	0.67	0.57
Doc8		0.67	0.5
Doc9		0.67	0.44
Doc10	P	0.83	0.5

Tableau I.4 Exemple de calcul de rappel et de précision pour une requête

La figure I.4 illustre la courbe de rappel et précision correspondante aux résultats du tableau I.4. Pour rendre la courbe lisible on ne garde que la précision calculé à chaque point de rappel (c'est-à-dire à chaque document pertinent restitué).

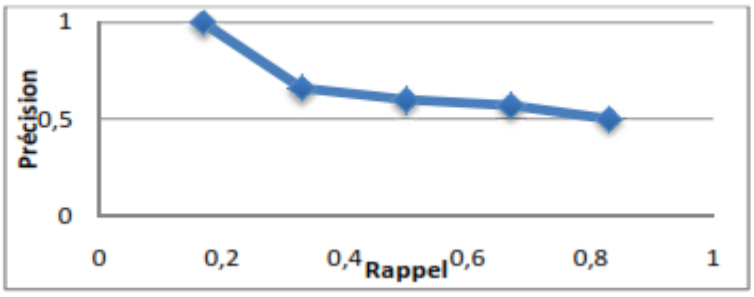

Figure I.4 Courbe de rappel et précision

La courbe ci-dessus permet d'évaluer les performances du système pour la requête considérée. Afin d'évaluer le système pour un ensemble de requêtes, on calcule la moyenne des précisions à chaque niveau de rappel. Comme les niveaux de rappel ne sont pas unifiés pour

l'ensemble des requêtes, on retient généralement 11 points de rappel standards de 0 à 1 avec un pas de 0.1.

Les valeurs de précisions sont calculées par une interpolation linéaire. Pour deux points de rappel, *i* et *j*, *i* < *j*, si la précision au point i est inférieure à celle au point j, on dit que la précision interpolée à i égale la précision à *j*.

Cette interpolation est encore discutable, mais présente un intérêt dans l'évaluation de système de recherche d'information. Elle permet entre autre de construire des courbes décroissantes plus simple à comparer **[42] [97].**

Deux autres facteurs complémentaires au rappel et précision permettent d'évaluer les performances et l'efficacité de la recherche d'information, il s'agit du bruit et du silence.

- **Le bruit :** Le bruit est une notion complémentaire à la précision (Bruit=1- Précision), elle représente les documents retournés, mais non pertinents

  Bruit=documents retournés non pertinents/documents retournés
- **Le silence :** Le silence est une notion complémentaire au rappel (Silence=1-Rappel), elle représente les documents pertinents non retournés :

  Silence=pertinents non retournés/documents pertinents

La précision moyenne non interpolée (MAP) : La précision moyenne non interpolée (Average Mean Precision) est calculée en deux étapes. D'abord on calcule la précision moyenne pour une requête donnée (AP_q), ainsi pour chaque document pertinent retrouvé on calcule sa précision ($pr\ (d_i)$) qui est égale au nombre de documents pertinents retrouvés sur le rang de ce document ; pour les documents retrouvés non pertinents leur précision est égale à zéro**[97]**.

La précision moyenne pour une requête donnée est alors obtenue en calculant la moyenne des précisions des documents pertinents, exprimée ainsi :

$$AP_q = \frac{1}{N} \sum_{i=1}^{N} pr(d_i) \qquad (I.12)$$

Avec

$$pr(d_i) = \begin{cases} \dfrac{r_{ni}}{n_i} & si\ d_i\ est\ retrouvé \\ 0 & sinon \end{cases} \qquad (I.13)$$

Où n_i dénote le rang du document d_i qui a été retrouvé et qui est pertinent pour la requête, r_{ni} est le nombre de documents pertinents retrouvé au rang n_i et N est le nombre total de documents pertinents pour la requête q.

Dans la seconde étape, on calcule la précision moyenne pour un ensemble de requêtes, en effectuant la moyenne des précisions moyennes de chaque requête, elle est exprimée ainsi :

$$MAP = \frac{1}{M}\sum_{j=1}^{M} AP_{qj} \qquad (I.14)$$

Où AP_{qj} dénote la précision moyenne pour la requête « *j* » et M représente le nombre de requêtes considérées.

I.7.2. Collection de tests

Une collection (ou corpus) de test constitue le moyen d'évaluation des SRI. Elle est généralement composée d'un ensemble de documents, d'un ensemble de requêtes et des jugements de pertinence associés à ces requêtes.

Les collections de test sont le résultat de projets d'évaluation qui se sont multipliés depuis les années 1970. Une des collections les plus utilisées actuellement en RI est la collection TREC.

TREC (*Text REtrieval Conference*) est un projet international qui a été lancé en 1992 par le NIST (National Institute of Standards and Technology) aux Etats-Unis. Il est aujourd'hui co-sponsorisé par le NIST et DARPA/ITO (*Defense Advanced Research ProjectsAgency – Information Technology Office*) **[43].**

Il offre une très large collection de documents de source très variées organisées en sous collections, qui évoluent d'année en année. Pour chaque session de TREC, un ensemble de documents et de requêtes est fourni **[44]**.

La taille des collections augmente au fil des années, passant de 2 Go dans TREC 1 à 25 To dans TREC 2011. Chaque collection est composée d'un certain nombre de documents, allant de quelques milliers à plusieurs millions. Les documents sont codés à l'aide de SGML dans un format spécifique TREC.

La figure I.5 illustre un exemple d'un document TREC.

```
<DOC>
<DOCNO> WSJ920324-0113 </DOCNO>
<DOCID> 920324-0113. </DOCID>
<HL> Venture of Kimbaco </HL>
<DATE> 03/24/92 </DATE>
<SO> WALL STREET JOURNAL (J), PAGE C9 </SO>
<CO> H.TSI </CO>
<MS> FINANCIAL (FIN) </MS>
<IN> ALL BANKS, BANKING NEWS AND ISSUES (BNK)
SECURITIES (SCR) </IN>
<NS> JOINT VENTURES (JVN) </NS>
<RE> FAR EAST (FE)
HONG KONG (HK)
PACIFIC RIM (PRM)
SOUTH KOREA (SK)
</RE>
<LP>
NEW YORK -- South Korean merchant banking firm Kimbaco said it joined
with Hong Kong brokerage house Peregrine Securities to form a new
investment firm Kimbaco Peregrine Capital Ltd.
The firm will seek out cross-border transactions and direct investment
opportunities in Asia, with special emphasis on U.S.-Korean ventures.
</LP>
<TEXT>
</TEXT>
</DOC>
```

Figure I.5 Exemple d'un document TREC

Chaque collection TREC a généralement 50 à 100 requêtes correspondantes. Une requête TREC est structurée comme suit: un identifiant de requête unique TREC, un titre, une description plus détaillée du besoin en information et une rubrique qui explique dans quelles circonstances un document doit être jugé pertinent ou non pertinent pour une requête.
Un exemple d'une requête TREC est montré dans la figure I.6.

```
<top>
<num> Number: 562
<title> world population growth
<desc> Description:
What is the outlook for world population growth?
<narr> Narrative:
Relevant documents include projections of and
discussion of world population growth. Growth of
individual nations' populations is relevant, but
data on states within the U.S. is not relevant.
</top>
```

Figure I.6 Exemple d'une requête TREC

Afin d'évaluer les résultats de recherche un ensemble de jugements de pertinence sont associées aux requêtes. Un exemple est montré dans la figure I.7.

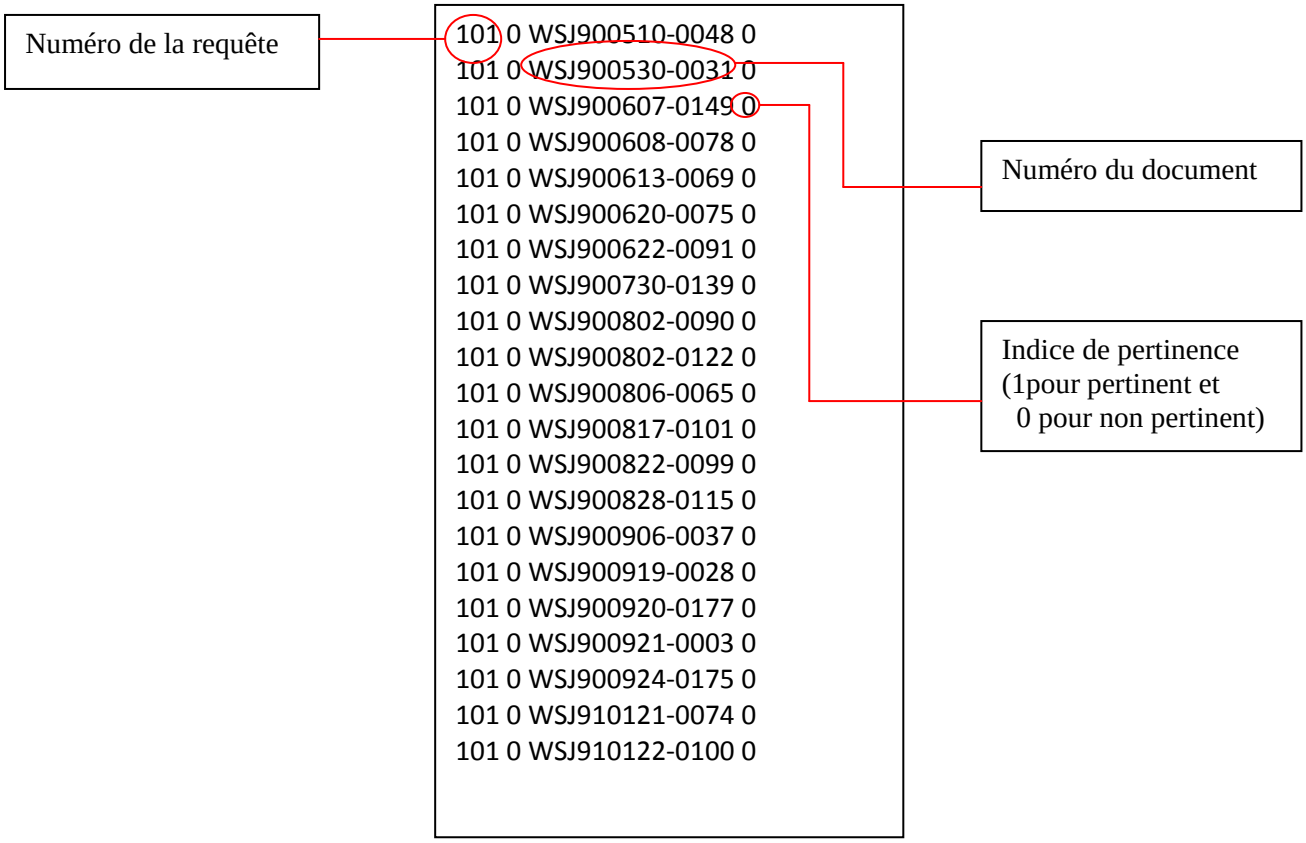

Figure I.7 Extrait d'un fichier « .qrel »

I.8. Conclusion

Ce premier chapitre a porté essentiellement sur l'étude du domaine de la RI de manière générale, nous avons notamment présenté, les concepts de base de la RI (document requête, etc) et les principales étapes d'un processus de recherche d'information soit, l'indexation, l'appariement requête /document et la reformulation de la requête. Cette dernière étape et présentée en détails dans le chapitre suivant, car notre travail rentre dans le cadre de cette étape.

Nous avons aussi présenté, les différents modèles de RI. Et enfin nous avons présenté les métriques d'évaluation d'un SRI, ainsi que les collections de test, et comme exemple de collection nous avons présenté la collection TREC, que nous avons utilisé dans nos expérimentations.

Chapitre II

La Reformulation de la Requête

II.1. Introduction

Blair et Maron [45] avaient montré que la faible performance des systèmes de recherche d'information est due à l'incapacité des utilisateurs de formuler les requêtes adéquates. En effet, la requête initiale de l'utilisateur est souvent exprimée par une liste de termes souvent très réduite qui exprime mal les besoins en information de l'utilisateur. Pour remédier à ce problème, une solution consiste à enrichir (reformuler) la requête afin d'améliorer la qualité des documents retrouvés.

En effet, les techniques de reformulation de requêtes tentent de construire une nouvelle requête plus représentative du besoin en information de l'utilisateur. La reformulation de requêtes passe par plusieurs étapes: le prétraitement des informations, tel que le résultat de la première recherche, le classement et la sélection des termes d'expansion, et en fin la réécriture de la requête.

Dans ce chapitre nous traitons de la reformulation de requêtes particulièrement nous « énumérons » les critères de classification des méthodes de reformulation, puis nous étudions les étapes de l'expansion (reformulation) de requête, l'expansion de requête dans le modèle de langue est en fin traitée.

II.2. Définition de la reformulation de la requête

La reformulation de la requête est un processus permettant la construction d'une nouvelle requête, plus à même de représenter les besoins en information de l'utilisateur. Elle est souvent opérée par ajout/suppression et/ou réévaluation des poids des termes de la requête initiale.

La figure II.1 représente les principales techniques d'amélioration des SRI par reformulation de la requête initiale:

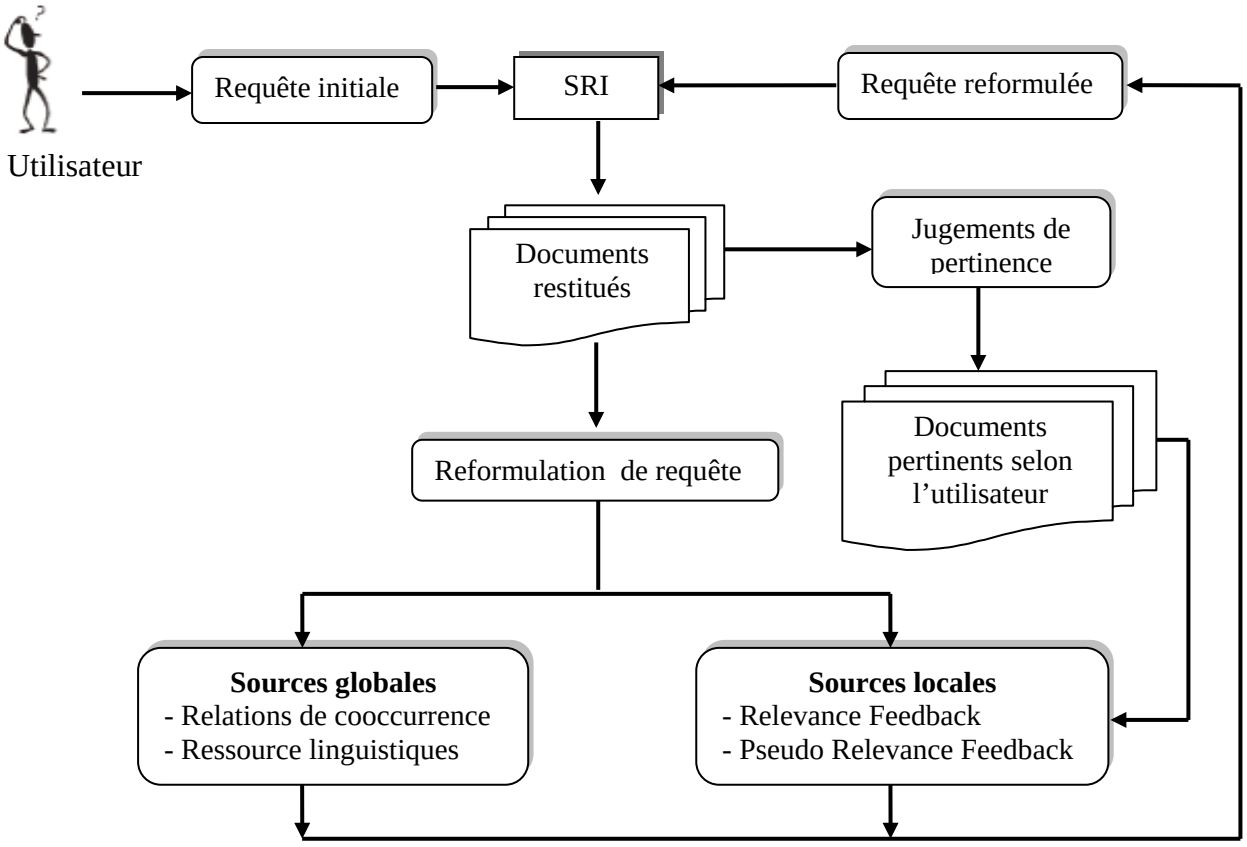

Figure II.1 : Processus d'amélioration des SRI par reformulation de requêtes **[46]**

II.3. Les méthodes de reformulation de requêtes

nous décrivons ci-dessous les critères (paramètres) de classification des méthodes de reformulation de requêtes ainsi que deux principales techniques (méthodes) de reformulation.

II.3.1. Classification des méthodes de reformulation de la requête

Les méthodes de reformulation de la requête peuvent être classées en tenant compte de plusieurs paramètres **[47]** :

- Les sources de données utilisées pour l'expansion de requêtes.
- La méthode de sélection des termes d'expansion : La relation de cooccurrence, les mesures d'information, les techniques de classification, etc.
- La sélection des termes d'expansion en considérant chaque terme de la requête individuellement, ou la requête dans son ensemble.
- La représentation de la requête (document) comme un ensemble de mots simples (sac de mots) ou une représentation prenant en compte les relations de proximité entre termes.
- Le type de terme d'expansion (mot simple ou mot composé).
- L'intervention de l'utilisateur dans le processus d'expansion de la requête (automatique, manuelle, semi-automatique).

II.3.1.1. Classification Selon l'intervention de l'utilisateur

Trois classes de méthodes se distinguent selon ce paramètre (intervention de l'utilisateur):

II.3.1.1.1. La reformulation manuelle

C'est à l'utilisateur de sélectionner à partir d'une source de données (par exemple des documents pertinents) les termes à rajouter à la requête initiale**.** Cette approche est associée aux systèmes de recherche booléens. On peut procéder à la reformulation de requêtes en utilisant un vocabulaire contrôlé (thésaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes pour compléter sa requête **[48]**.

II.3.1.1.2. La reformulation semi-automatique

Dans une reformulation semi-automatique, l'utilisateur joue un rôle actif. A l'inverse de la reformulation automatique, définit ci-après, ici, ce sont le système et l'utilisateur qui sont, ensemble, responsables de la détermination et du choix des termes candidats à la reformulation.

- ✓ Le système joue un grand rôle dans la suggestion des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes.
- ✓ L'utilisateur examine cette liste et décide du choix des termes à ajouter dans la requête. C'est donc l'utilisateur qui prend la décision ultime dans la sélection des termes **[48]**.

II.3.1.1.3. La reformulation automatique

La reformulation automatique de requêtes consiste à ajouter d'autres termes à ceux utilisés par l'utilisateur pour l'interrogation. Les nouveaux termes sont choisis automatiquement, sans l'intervention de l'utilisateur et doivent avoir des sens proches des termes donnés par l'utilisateur. Cette reformulation consiste à ajouter à la requête initiale des termes issus de ressources linguistiques existantes de ressources construites à partir des collections ou à partir des premiers documents pertinents retournés.

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 un bruit **[48].**

II.3.1.2. Classification selon la source de données utilisée

Les techniques d'enrichissement (automatique) de requêtes peuvent être classées en deux catégories : celles basées sur l'analyse du contexte local et celles basées sur l'analyse du contexte global.

II.3.1.2.1. Approche basée sur le contexte local

Les techniques basées sur l'analyse du contexte local permettent d'identifier les relations entre termes afin d'enrichir les requêtes et cela par l'analyse des documents retrouvés (les mieux classés) **[49][50][51]**. **Buckley, Salton, Allan et Singhal [50]** proposent une technique qui suppose que les premiers documents retrouvés (les mieux classés) sont pertinents, ensuite la requête est enrichie suivant la méthode standard de relevance feedback **[52].** Une méthode similaire est utilisée dans **[51],** où les premiers documents retrouvés sont utilisés pour re-estimer les probabilités des termes.

II.3.1.2.2. Approche basée sur le contexte global

Les techniques basées sur l'analyse du contexte global permettent d'identifier les relations entre termes et documents par l'analyse du corpus documentaire. Une des techniques d'analyse du contexte global est la classification des termes [**53],** où les termes d'indexation sont regroupés par classes, en se basant sur leurs cooccurrences. Ensuite ces classes de termes sont utilisées pour l'enrichissement des requêtes. Des méthodes d'enrichissement de requête par un thésaurus de similarité ont été proposés [**54][55].** Un thésaurus de similarité est une matrice terme-terme construite à partir du corpus documentaire, où chaque terme est représenté par un

vecteur de documents dans un espace de documents. Ainsi, **Qiu et Frei [54]** ont proposés une méthode d'enrichissement de requête par un thésaurus de similarité, l'approche consiste à rajouter à la requête des termes issus du thésaurus. Le choix des termes à rajouter se fait par un calcul de similarité entre les termes de la requête et ceux du thésaurus.

Le modèle LSI (Latent Semantic Indexing) **[56][57]** est aussi considéré comme une méthode d'analyse du contexte global **[58].** LSI utilise une décomposition en valeurs singulières sur la matrice représentative du corpus documentaire (*documents unités linguistiques*), cette décomposition permet d'extraire les principales associations entres les unités linguistique d'un document. Enfin, dans **[58]** une méthode combinant analyse local et global a été proposée.

D'autres méthodes de reformulation de requêtes sont basées sur l'exploitation de ressources linguistiques.

Plus précisément, au niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. Ce mécanisme assure la restitution des documents indexés par des variantes des termes composant la requête. Les associations établies manuellement traduisent généralement des relations de synonymie et de hiérarchie.

Les thésaurus construits manuellement sont un moyen efficace pour l'expansion de la requête.

II.3.2. Présentation de quelques méthodes de reformulation de requêtes

Plusieurs méthodes ont été proposées pour améliorer les performances des SRI. Ces méthodes apportent des solutions aux deux principales questions :

1. Comment peut-on retrouver plus de documents pertinents vis-à-vis d'une requête donnée ?
2. Comment peut-on mieux exprimer la requête de l'utilisateur de manière à mieux répondre à son besoin ?

Nous présentons dans ce qui suit deux principales méthodes de reformulation de requêtes.

II.3.2.1. Reformulation par réinjection de la pertinence

Ces méthodes impliquent que l'utilisateur doit sélectionner les documents qu'il considère pertinents à partir des résultats issus de sa requête initiale. Ce jugement de pertinence de l'utilisateur est ensuite exploité pour reformuler la requête initiale en modifiant le poids des termes qu'elle contient et/ou en ajoutant de nouveaux termes considérés utiles pour retrouver des documents pertinents.

La technique de réinjection de pertinence à été mise en place à l'origine dans le modèle vectoriel. **Rocchio[52]** a proposé le modèle de reformulation de requête suivant :

$$Q_{\mathrm{N}} = \alpha . Qo + \beta . \frac{1}{|R|} \sum_{r \epsilon R} r - \frac{1}{|R'|} \sum_{r' \epsilon R'} r' \qquad \text{(II. 1)}$$

Où :

Q_{N} est le vecteur de la nouvelle requête (reformulée) ;

Qo est le vecteur de la requête originale ;

R est l'ensemble des vecteurs r des documents jugés pertinents par l'utilisateur ;

R' est l'ensemble des vecteurs r' des documents jugés non pertinents par l'utilisateur ;

α, β, δ sont les paramètres de la reformulation.

On peut remarquer que cette formule permet d'obtenir une nouvelle requête dont le vecteur se rapproche des vecteurs des documents jugés pertinents et s'éloigne des vecteurs des documents jugés pertinents.

Dans le modèle probabiliste, la réinjection de pertinence est mise en place directement dans la modèle de mesure de pertinence. Elle consiste à revoir les poids des termes de la requête **[59] [97].** comme suit :

$$W_{qj} = \log \left[\frac{r' + 0.5/(r - r' + 0.5)}{(df_{\mathrm{j}} - r' + 0.5)/(n - df_{\mathrm{j}} - r + r' + 0.5)} \right] \qquad \text{(II. 2)}$$

Où :

r représente le nombre de documents pertinents ;

r' est le nombre de documents pertinents contenant le terme q_j ;

df_{j} est le nombre de documents contenants le terme q_j ;

n est le nombre total de documents dans la collection.

II.3.2.2. La réinjection par pseudo Feedback (réinjection aveugle)

Ces méthodes de reformulation nommée aussi, pseudo-réinjection de pertinence (ou blind) sont effectuées de manière automatique. Elles se basent sur l'hypothèse que les documents les mieux classés (les premiers) sont considérés comme pertinents. Le système utilise alors les premiers documents pour reformuler la requête.

La variante de la **formule de Rocchio** (formule II.1) pour la réinjection automatique de la requête est exprimée par la formule suivante :

$$Q_{\mathrm{N}} = \alpha . Qo + \beta . \frac{1}{|R|} \sum_{r \epsilon R} r \qquad \text{(II. 3)}$$

On voit dans cette formule que l'expansion de la requête est uniquement positive, car on ne peut faire aucune hypothèse sur les documents non pertinents, mais rien ne nous empêche de prendre les derniers documents de la liste non pertinents.

Plusieurs travaux **[60] [61]** ont tenté d'évaluer l'impact de la pseudo-réinjection, en variant le nombre de nombre de termes rajouter à la requête. Ils montrent que la performance du système est obtenue lorsque la requête est construite entre 20 et 40 termes**[97].**.

II.4. Les étapes de l'expansion (reformulation) automatique de requêtes

L'expansion de requête comprend quatre étapes principales que nous allons définir par la suite, l'entrée de ce processus c'est la requête initiale de l'utilisateur et en sortie on aura la requête reformulée avec les termes d'expansion. Nous schématisons ce processus avec la figure suivante :

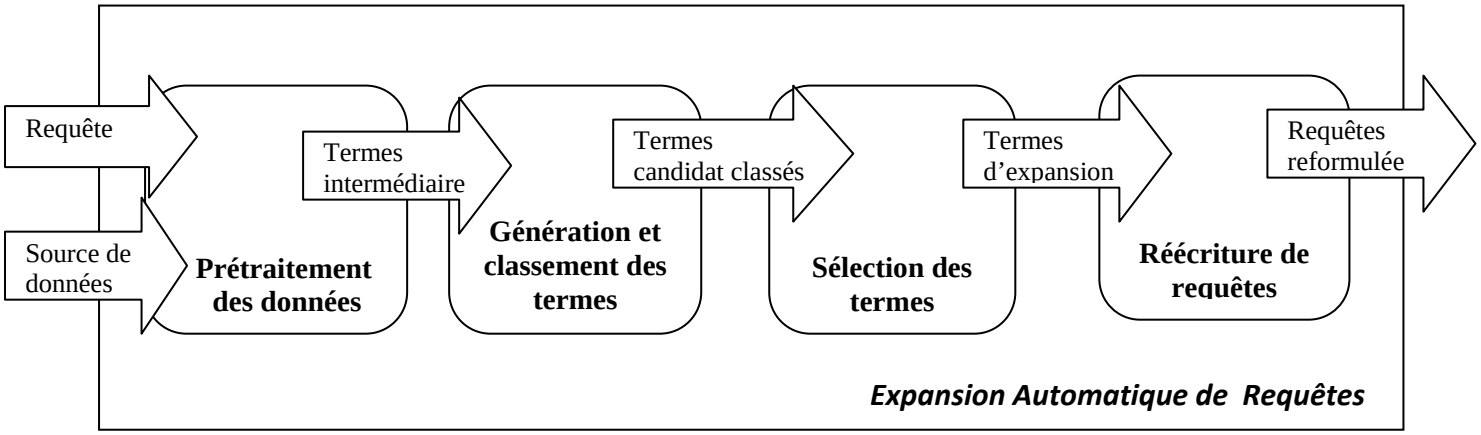

Figure II.2 : Processus d'expansion automatique de requêtes.

II.4.1.Prétraitement des données (requête et autres sources utilisées)

Cette étape transforme la première source de donnée utilisée pour étendre la requête de l'utilisateur dans un format qui sera traité plus efficacement par les étapes suivantes. Elle consiste habituellement d'une phase d'extraction de termes pour faciliter l'accès et la manipulation des fonctions de traitement.

Le prétraitement de la source de données (documents ou autre) est généralement indépendant de la requête de l'utilisateur qui doit être étendue, mais elle est spécifique au type de sources de données.

De nombreuses techniques d'expansion de requêtes sont basées sur les informations contenues dans les tops documents, extraits en réponse à la requête de l'utilisateur initiale à partir d'une collection de documents. Dans ce cas dans l'étape de prétraitement des données, il est nécessaire d'indexer la collection et d'exécuter la requête au préalable.

En conséquence, chaque document est représenté comme un ensemble de termes pondérés, avec un fichier complémentaire inversé de l'indice qui associe les termes du document aux termes de la requête. Le système d'indexation peut également stocker les positions des termes afin de fournir la recherche basée sur la proximité. Lorsque la collection utilisée pour l'expansion de requêtes est la même que celle en cours de recherche, le système de classement à lequel la requête élargie sera soumise est généralement utilisé pour effectuer aussi un premier passage de classement. Si un corpus externe est utilisé (par exemple, des données Web pour les recherches sur l'intranet, ou données de bureau personnel pour recherches sur le Web), comme dans **[58][62][63][64],** en générale un système de recherche d'information différent, est nécessaire; plusieurs options sont disponibles, telles que l'installation et le fonctionnement d'un moteur de recherche de bureau (commercial ou disponible gratuitement), en utilisant les récupérations API Web, ou encore le développement de son propre système pour l'indexation des documents et le classement.

D'autres techniques d'expansion de requêtes, basées sur l'analyse du corpus, nécessitent l'extraction des termes particuliers de la collection manuellement, qui sont généralement différents de ceux utilisés à des fins d'indexation par un système de recherche d'information classique. Une approche bien connue de **Qiu et Frei [54],** où chaque terme est représenté comme un vecteur de documents pondéré en utilisant les statistiques d'une collection non-standard.

Un autre exemple de **Crouch et Yang [65],** qui construit un thésaurus de statistique en regroupant d'abord l'ensemble de la collection de documents via l'algorithme complet de clustering.

Certaines techniques d'expansion de requêtes nécessitent des procédures de prétraitement adaptés aux certaines sources de données. Par exemple, si l'expansion de requêtes utilise des textes d'ancrage, il faut analyser une collection de lien hypertexte pour extraire le contenu du texte de l'ancre balisé, et de traiter ensuite ces textes : les normaliser et / ou de supprimer ceux qui contiennent trop peu ou trop de termes **[66]**. Les requêtes URL extraites des journaux

d'archives des moteurs de recherche sont une autre source de données pour L'expansion de requêtes.

Dans les approches discutées jusqu'ici, le prétraitement est appliqué à une source de données précises. Telle est la situation prédominante, mais il ya des exceptions. La source de données peut être sélectionnée parmi plusieurs choix, comme dans **[67][68].**

Une collection de FAQ (Frequently Asked Questions) est automatiquement construite en utilisant d'abord les requêtes Web tels que "inurl: faq" et appliquer ensuite des techniques d'apprentissage automatique pour extraire les FAQ réels de l'ensemble de pages récupérées.

II.4.2. Génération et Classement des termes candidats d'expansion

Dans la deuxième étape de l'expansion de requêtes, le système génère et classe les termes candidat d'expansion.

La raison pour laquelle le classement est important, c'est que la plus part des méthodes d'expansion de requête, ne pourront choisir qu'un petit nombre de termes candidats d'expansion à ajouter à la requête initiale.

L'entrée de cette phase est la requête d'origine et la source de données transformée; le résultat est un ensemble de termes d'expansion, généralement avec des scores associés. La requête initiale peut être prétraitée pour supprimer des mots communs et / ou extraire des termes importants pertinents.

Nous classons les techniques utilisées pour exécuter la génération et le classement des termes candidats selon le type de relation entre les termes d'expansion générés et les termes de la requête initiale (requête après prétraitement, le cas échéant).

II.4.2.1. Association un à un

La forme la plus simple de génération et de classement des termes candidats est basée sur les associations un-à-un entre les termes d'expansion et les termes de la requête initiale, c'est à dire, chaque terme d'expansion est lié à un terme unique de la requête. Dans la pratique, un ou plusieurs termes d'expansion sont générés et marqués pour chaque terme de la requête à l'aide d'une variété de techniques.

L'une de ces techniques consiste à s'appuyer sur les associations linguistiques, comme l'utilisation d'un algorithme de lemmatisation pour réduire des mots différents pour le même radical.

Une autre technique linguistique commune est de trouver des synonymes et mots connexes d'un mot de la requête à partir d'un thésaurus, le plus souvent utilisé est le WordNet,

ce dernier utilise un groupes de mots anglais dans des ensembles de synonymes appelés **synsets** (*synonym set*), il y'en a diverses relations sémantiques lexicales entre ces ensembles de synonymes.

En particulier, il comprend : l'Hyperonyme / relations hyponyme entre synsets nominaux qui peuvent être interprétées comme des relations de généralisation / spécialisation entre les ensembles correspondant à ces synsets.

La génération des termes d'expansion de WordNet implique de sélectionner un synset pour un terme de la requête, résolvant ainsi le problème de l'ambiguïté. Afin de choisir un synset avec un sens similaire au terme de la requête, les termes de la requête adjacents peuvent être mieux adaptés aux ensembles présents dans chaque synset contenant le terme de la requête. Après avoir sélectionné le synset le plus pertinent, on pourrait envisager pour l'expansion de la requête tous les synonymes du terme de la requête dans le synset.

Les approches statistiques consistent à analyser un document, en évaluant les éléments d'un document par leur fréquence d'occurrence dans ce document. Ces statistiques peuvent être utilisées pour créer des index ou extraire les concepts d'un domaine en vue de sa modélisation.

Par contre l'approche linguistique consiste à calculer automatiquement la similarité terme-à-terme dans une collection de documents.

L'idée générale est que les deux termes sont sémantiquement liés s'ils apparaissent dans les mêmes documents, tout comme deux documents sont considérés comme similaires s'ils contiennent les mêmes termes. Deux simples mesures de similarité sont le coefficient Dice et l'indice de Jaccard. Compte tenu des termes *u* et *v*, le coefficient Dice (D) est défini comme :

$$D = \frac{2.df_{u \wedge v}}{df_u + df_v} \qquad \text{(II. 4)}$$

Où :

$df_{u \wedge v}$ est le nombre de documents qui contiennent à la fois les termes u et v,

df_u, df_v sont les nombres de documents contenants les termes *u* et *v* respectivement.

L'indice de Jaccard (J) est défini comme suit :

$$J = \frac{df_{u \wedge v}}{df_{u \vee v}} \qquad \text{(II. 5)}$$

Où :

$df_{u \vee v}$ est le nombre de documents contenants le terme *u* ou le terme *v*.

Une approche plus générale est la suivante. Considérons une matrice terme-document A,

Où chaque cellule $A_{t,d}$ est un poids $W_{t,d}$ pour le terme *t* et le document *d*.

Si l'on calcule $C = AA^t$ alors C est une matrice de corrélation terme-terme, où chaque élément $C_{u,v}$ est une corrélation (similarité score) entre les termes *u* et *v* donnée comme suit:

$$c_{u,v} = \sum_{dj} W_{u,j} . W_{v,j} \qquad \text{(II. 6)}$$

En utilisant la formule ci-dessus on peut calculer la corrélation entre chaque terme de la requête et chaque terme dans la collection de documents. Pour tenir compte de la fréquence des termes, il est préférable de produire des facteurs de corrélation normalisés, par exemple par la mesure de cosinus donnée comme suit :

$$\frac{c_{u,v}}{\sqrt{\sum_{dj} w_{u,u}^2} . \sum_{dj} w_{v,v}^2} \qquad \text{(II. 7)}$$

Selon la manière dont les documents et la fonction de pondération sont choisis, la formule **(II.6)** peut donner lieu à différentes méthodes de corrélation terme-à-terme. Une technique bien connue, proposée dans **[69],** repose sur l'ensemble des documents retournés en réponse à la requête initiale et utilise la fréquence des termes pondérés.

La cooccurrence des termes dans le document est simple, mais elle a l'inconvénient que la position n'est pas prise en compte, alors que deux termes qui apparaissent dans la même phrase semblent plus corrélés que deux termes qui apparaissent loin l'un de l'autre dans un document. Cependant, la cooccurrence simple, que ce soit dans un contexte grand ou petit, ne signifie pas nécessairement que les termes sont corrélés. Une mesure plus complète pour l'association de mots qui incorpore la dépendance entre termes c'est l'Information mutuelle **[70], [71],** définie comme suit:

$$I_{u,v} = \log_2 \left[\frac{P(u,v)}{P(u).P(v)} + 1 \right] \qquad \text{(II. 8)}$$

Où P(*u*,*v*), est la probabilité conjointe que les termes *u* et *v* apparaissent dans un certain contexte, habituellement un document, et P(*u*) et P(*v*) sont les probabilités d'occurrence des termes *u* et *v* respectivement.

L'information mutuelle est nulle en cas de cooccurrence zéro, égal à un si *u* et *v* sont indépendants, et égal à : $\log_2 \frac{1}{P(u)} + 1$ si *v* est parfaitement associé avec ***u***.

Un de ses inconvénients est qu'elle a tendance à favoriser les termes rares sur les termes communs, parce que ($I(u,v)$ augmentera si P($v|u$) est fixée, mais P (u) diminue.

Ce problème peut devenir plus aigu pour les données éparses. Sinon, nous pourrions envisager la définition classique de la probabilité conditionnelle pour mesurer le degré de l'association

de terme v au terme u, donnée comme suit :

$$P(v,u) = \frac{P(u,v)}{P(u)} \qquad \text{(II. 9)}$$

La probabilité conditionnelle peut être calculée en divisant le nombre de contextes (par exemple phrases) dans laquelle les termes *u* et *v* coexistent par le nombre de contextes dans lesquels le terme *u* apparait.

II.4.2.2. Associations Un à Plusieurs

L'association un-à-un a tendance à ajouter un terme quand il est fortement lié à l'un des termes de la requête. Toutefois, cela peut ne pas refléter exactement les relations des termes d'expansion à la requête dans son ensemble.

Ce problème a été analysé par **Bai & al [72]**, par exemple, alors que le mot «programme» peut ainsi être fortement associé au mot «ordinateur», une expansion automatique de toutes les requêtes contenant «programme» avec «ordinateur» pourrait bien fonctionner pour certaines requêtes (par exemple, «programme Java», «programme d'application»), mais pas pour d'autres (par exemple, «programme de télévision», «programme de gouvernement», «programme spatial»). Ici encore, nous rencontrons la question de l'ambiguïté de la langue.

Le principe de l'approche associations un-à-plusieurs est d'étendre l'association un à un (technique décrite dans la section précédente) pour les autres termes dans la requête. L'idée est que si un terme d'expansion est corrélé à plusieurs termes de la requête, donc il est corrélé à la requête dans son ensemble.

Selon **Voorhees[73]** par exemple, il est nécessaire qu'un nouveau terme extrait du WordNet soit lié à au moins à deux termes de la requête originale avant qu'il soit inclut dans la requête étendue. Si nous utilisons des corrélations terme-à-terme, nous pourrions calculer les facteurs de corrélation d'un terme d'expansion candidat donné *v* pour chaque terme de la requête, puis combiner les scores trouvés pour trouver la corrélation globale de la requête *q*, par exemple, par la formule suivante :

$$c_{q,v} = \frac{1}{|q|}\sum_{u \epsilon q} c_{u,v} \qquad \text{(II. 10)}$$

Une approche similaire a été proposée dans **[54]** et **[74],** et suivie de plusieurs autres travaux de recherche **[75][76].**

Les deux précédents articles sont intéressants non seulement parce qu'ils étendent le paradigme de la corrélation un à un à l'ensemble de la requête, mais aussi en raison de leurs fonctions de pondération particulières et les types de termes d'expansion.

Dans **Qiu et Frei [54],** la formule (II.6) est utilisée pour calculer la corrélation terme-terme dans toute la collection, considérée comme un espace concept-terme où les documents sont utilisés pour extraire les termes d'indexation. Le poids d'un terme dans un document est exprimé comme le produit de la fréquence du terme dans le document par la fréquence inverse du terme associée à ce document.

La fréquence inverse du terme d'un document d_j est donné par $\log\frac{T}{DT_j}c$, où T est le nombre de termes dans la collection et DT_j est le nombre de termes distincts dans le document d_j.

Ce concept est similaire à la fréquence inverse du document utilisé pour le classement des documents.

Un concept est un groupe de noms adjacents dans les documents les plus recherchés, et les concepts candidats sont analysés en utilisant des passages (c'est à dire, une portion de texte de taille fixe) à la place du document complet. La formule (II.6) est appliquée pour calculer la corrélation d'un terme-concept (au lieu d'une corrélation terme-terme), où $w_{u,j}$ est la fréquence du terme de la requête dans le j-ième passage et , $w_{v,j}$ est la fréquence du concept c dans le j-ième passage.

L'approche étendant l'association un-à-un peut être utile pour filtrer les termes d'expansion qui sont faiblement liés à certains termes de la requête, mais il n'est pas garantit que le terme d'expansion qui est fortement connecté à un seul terme sera écarté. Par exemple, si l'association de «ordinateur» avec «programme» est assez forte, «ordinateur» peut rester comme un terme d'expansion, même pour les requêtes «programme de télévision» ou «programme de gouvernement».

Ce problème peut être résolu en ajoutant des mots de contexte à une association terme à terme qui précisent les conditions dans lesquelles l'association est valide. Ces mots de contexte, par exemple, peuvent être dérivés d'une base de connaissances, ou ils peuvent être extraits à partir d'un corpus en utilisant des cooccurrences de termes **[77].**

Considérant à nouveau notre exemple, si nous exigeons que «programme» apparaît avec «application» (ou «Java»), nous limitons l'applicabilité de l'association «programme» - «ordinateur» à des contextes appropriés.

Lorsque l'expansion de requêtes est basée sur WordNet, la nécessité est de relier les termes d'expansion à l'ensemble de la requête, et non à ses termes pris isolément.

Voorhees **[73]** a montré que ces dernières techniques ne sont généralement pas efficaces car ils ne garantissent pas une bonne homonymie.

L'approche, associations un-à-plusieurs est basée aussi sur la combinaison de plusieurs relations entre les paires de termes dans un cadre de la chaîne de Markov **[78].** Pour chaque requête, un réseau d'expression est construit, qui contient des paires de mots reliés par plusieurs types de relations, comme des synonymes, des radicaux, des cooccurrences, ainsi que les probabilités de transition. Ces relations peuvent être obtenues auprès de diverses sources. Ensuite, les mots ayant la plus forte probabilité de pertinence sont sélectionnés comme termes d'expansion, car ils reflètent au mieux les multiples aspects de la requête donnée.

II.4.3. La sélection des termes d'expansion

Après avoir classé les termes candidats, les éléments principaux sont sélectionnés pour l'expansion de la requête. La sélection se fait sur une base individuelle, sans tenir compte des interdépendances entre les termes d'expansion.

Ceci est bien sûr une hypothèse simplificatrice, bien qu'il existe quelques résultats expérimentaux qui semblent suggérer que l'hypothèse d'indépendance peut être justifiée **[79].**

Habituellement, seul un nombre limité de termes est sélectionné pour l'expansion, en partie parce que la requête qui en résulte peut être traitée plus rapidement, en d'autre parce que l'efficacité de recherche avec un petit ensemble de termes n'a pas nécessairement moins de succès qu'en ajoutant tous les termes candidat d'expansion.

Des recherches ont été effectuées sur le nombre optimal de termes à inclure et il ya des suggestions différentes, allant de cinq à dix termes **[80][81]** à quelques centaines **[82][83][84].**

La plupart des études expérimentales s'accordent à dire que le nombre de termes d'expansion est de faible importance, le choix typique est d'utiliser 10 à 30 termes.

Lorsque les scores des termes peuvent être interprétés comme des probabilités, on peut sélectionner uniquement les termes ayant une probabilité supérieure à un certain seuil, par exemple, p = 0,001, comme dans **[85].** Plutôt que de se concentrer sur la recherche d'un nombre optimal de termes d'expansion, il peut-être plus pratique d'adopter des politiques de sélection plus adéquate**.** Il a été montré que les différentes requêtes ont un nombre varié de termes d'expansion optimal, et que de nombreux termes d'expansion **[86][87][37],** environ un tiers, sont nuisibles à la performance du repérage des documents. En fait, si l'on était en mesure de choisir exactement les meilleurs termes pour chaque requête, l'amélioration de la performance serait beaucoup plus élevée que d'habitude.

Pour aller au-delà d'une sélection simple fondée sur les poids attribués aux termes candidats, plusieurs méthodes qui emploient des informations supplémentaires ont été proposées.

Une technique présenté dans **[88]** consiste à utiliser plusieurs fonctions de classement de termes et en sélectionnant pour chaque requête les termes les plus courants.

Une autre stratégie consiste à choisir une quantité variable de termes d'expansion en fonction de la difficulté de la requête. Dans **[64],** le nombre de termes d'expansion est en fonction de l'ambiguïté de la requête originale sur le web (ou dans le répertoire de données personnelles de l'utilisateur), mesurée par le score de clarté **[89].**

Dans **[37],** les auteurs utilisent un classificateur pour faire la distinction entre la pertinence et la non pertinence des termes d'expansion.

Pour apprendre les paramètres du classificateur SVM (Support Vector Machine), un ensemble de mots simples sont étiquetés bon ou mauvais selon qu'ils améliorent ou nuisent aux performances, et chaque terme est décrit par un ensemble de fonctionnalités telles que la cooccurrence et la proximité avec les termes de la requête.

La sélection des meilleurs termes d'expansion pour une requête donnée est explicitement considérée comme un problème d'optimisation. En optimisant par rapport à un ensemble d'incertitudes définies autour des données observées.

II.4.4. La réécriture de la requête

La dernière étape d'expansion de requêtes est la réécriture de la requête, à savoir comment décrire la requête étendue qui sera soumise au système de recherche d'information. Cela revient généralement à l'attribution d'un poids à chaque terme décrivant la requête élargi appelé *pondération de requête.*

La technique de pondération des termes d'une requête la plus populaire est calquée sur la formule de Rocchio pour la réinjection par pseudo Feedback **[52]** et ses améliorations ultérieures **[90],** adaptée à la mise en expansion automatique de la requête. La formule générale est la suivante, où q' est la requête étendue, q est la requête originale, λ est un paramètre de pondération, et le $score_t$ est un poids attribué au terme d'expansion t.

$$w'_{t,q'} = (1 - \lambda).w_{t,q} + \lambda.score_t \qquad \textbf{(}\text{II.11)}$$

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 réduite si nous pondérons ces termes.

Même un simple schéma de pondération sur la base d'une fonction inverse de classement de termes 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 simples, discutées dans **[83],** ont été proposé, en général, elles produisent des résultats comparables, mais également accroître son uniformité pourrait être plus efficace **[91].**

La valeur de λ de l'expression ci-dessus peut être ajustée 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 paramètres tels que proposé dans **[81],** 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 feedback, 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 cooccurrences, la longueur du document, et le type de relation. Dans **Voorhees [73],** 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 un 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. L'expansion de la requête dans le modèle de langue

Nous présentons ci-dessous les méthodes, les plus en vue, d'expansion de la requête dans le cadre du modèle de langue.

II.5.1. Modèle de Lavrenko et Croft

Au lieu de modéliser la RI comme processus de génération de la requête, Lavrenko et Croft **[92]** 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 $p(t|\theta_{\mathrm{R}})$, 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 assumant qu'ils sont générés du modèle de pertinence. Ce modèle est formalisé comme suit :

$$P(t|\theta_{\mathrm{R}}) = \sum_{d \in R} P(d)p(t|d) \prod_{i=1}^{k} P(q_{\mathrm{i}} \,|\mathrm{d}) \qquad \text{(II. 15)}$$

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 pondérée 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**[93][97].**.

II.5.2. Modele de Zhai et Lafferty

Dans la même optique que **[92]**, Zhai et Lafferty **[85**] ont proposé un modèle nommé modèl-based feedback, où le nouveau modèle de la requête est obtenu par l'interpolation du modèle original de la requête avec un modèle de matière θ_T (Topic model), obtenu en utilisant les documents les mieux classés (retour de pertinence). La construction du modèle θ_T consiste en l'extraction d'une partie des documents retournés pertinents qui est distincte de l'ensemble des documents de la collection. Comme les documents les mieux classés sont susceptibles de contenir à la fois des informations pertinentes génériques (ou même non pertinents), ils peuvent être représentés par un modèle génératif mixte qui combine le modèle θ_T (à estimer) et le modèle de langue de la collection. Le logarithme de la probabilité des documents les mieux classées est donnée comme suit :

$$\log P(r|\theta_{\mathrm{T}}) = \sum_{d \epsilon R} \sum_{t} c(t,d) \log((1-\lambda)p(t|\theta_{\mathrm{T}}) + \lambda p(t|C)) \qquad \text{(II. 16)}$$

Où :

R est l'ensemble des documents les mieux classés, $c(t, d)$ est le nombre d'occurrence du terme *t* dans le document *d*,et λ est le poids d'interpolation.
L'algorithme EM (Expectation- Maximisation) est ensuite utilisé pour extraire le modèle θ_T.

II.6. Conclusion

Dans ce chapitre, nous avons présenté les méthodes de la reformulation de la requête et leur classifications selon quelque critères, puis nous avons énuméré les étapes de la reformulation de la requête, à savoir, le prétraitement des données, la génération et le classement des termes candidats d'expansion, la sélection des termes d'expansion et la réécriture de la requête, et en fin nous avons abordé la reformulation de la requête dans le cadre du modèle de langue.

Dans le chapitre suivant, nous allons présenter notre approche proposée et implémentée, qui consiste à la recherche du nombre optimal de termes d'expansion à ajouter à une requête selon ses propriétés et celles des documents retournés par une première recherche.

Chapitre III

Expérimentations & 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 la reformulation de requête.

Nous avons présenté dans le chapitre précédent les différentes étapes de reformulation de requête, dans ce chapitre nous nous focalisons sur l'étape sélection de termes d'expansion, où nous présentons une nouvelle approche dont l'objectif est l'estimation automatique du nombre de termes d'expansion à ajouter à la requête initiale en se basant sur : propriété dépendante de la requête, précisément sa taille, et la propriété concernant les documents pertinents retournés par la première recherche , précisément le nombre de ces documents.

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

III.2. Présentation de l'emplacement de notre approche

La figure suivante illustre l'architecture générale du processus de RI, où la zone grisée représente l'emplacement de notre approche dans le processus.

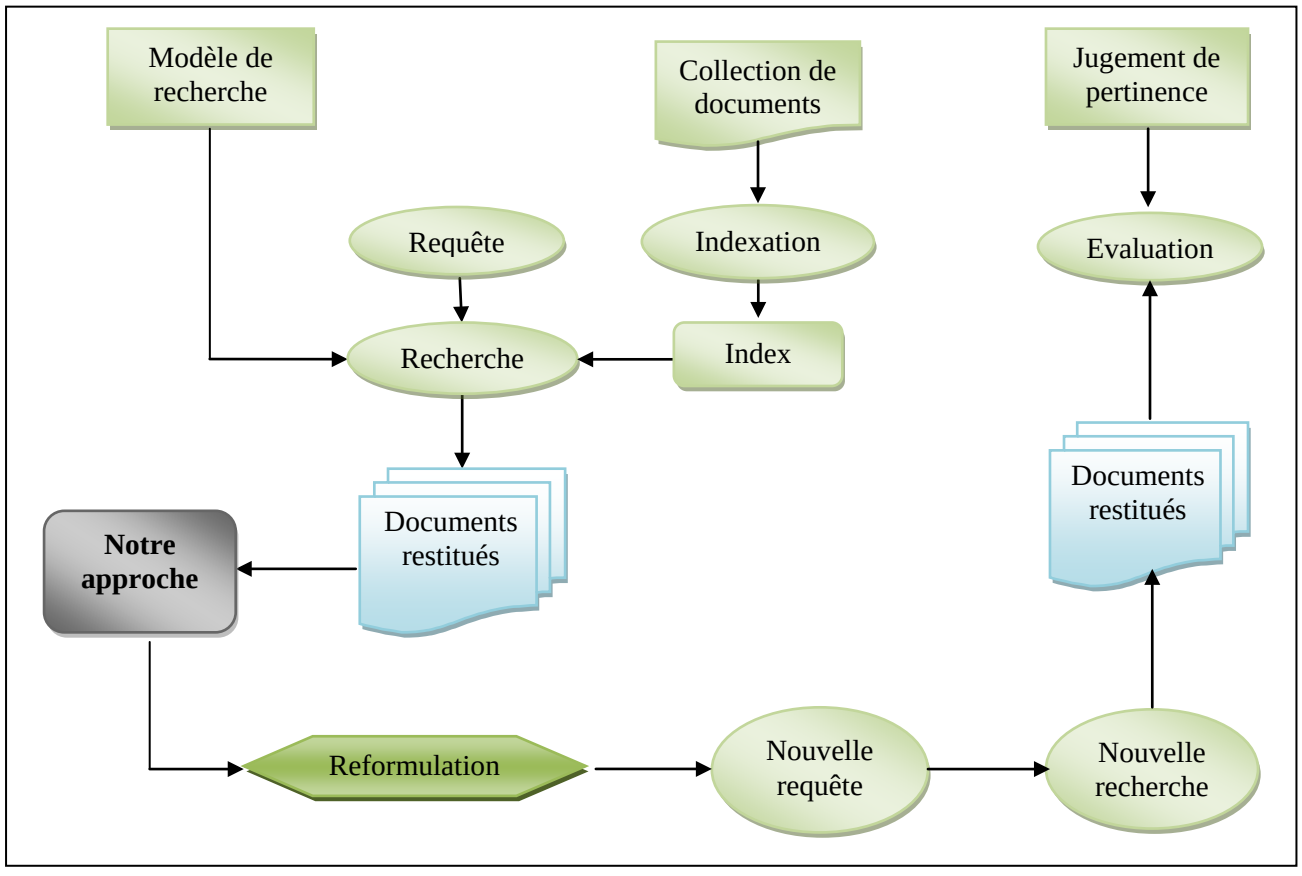

Figure III.1 Présentation de l'emplacement de notre approche

III.3. Présentation de notre approche

La reformulation de requêtes est proposée comme une méthode élaborée pour la RI, s'inscrivant dans la voie de conception des SRI adaptatif 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 ajout d'un nombre optimal de termes d'expansion significatifs.

III.3.1. Estimation de Nombre de termes ajoutés à la requête

L'ajout de termes à la requête accroît la performance du SRI. **Buckley & al [50]** ont expérimenté le retour de pertinence dans l'environnement multi-fond documentaire TREC, ils ont montré que le taux de performance est d'avantage corrélé avec le nombre de termes ajoutés qu'avec le nombre de documents initialement retrouvés.

Ils ont abouti à la mise au point de l'équation de variation :

$$RP(N) = ALog(N) + BLog(X) + C$$

Où :

RP (N): Performance du système pour N documents restitués ;

N : Nombre de documents restitués ;

X : Nombre de termes ajoutés à la requête ;

A, B, C : sont des constantes.

Ils ont conclu que le seuil critique du nombre de termes à ajouter à la requête dépond des caractéristiques de la collection.

En outre, la pondération différenciée des termes ajoutés à la requête accroît la performance du système. On attribue un poids :

- Moins important aux termes ajoutés **[94]**.
- plus important aux termes issus des documents pertinents que ceux issus des documents non pertinents **[90]**.

III.3.2 Facteurs d'estimation automatique de nombre de terme d'expansion

Dans cette section nous allons présenter une nouvelle approche dont l'objectif est l'estimation automatique du nombre de termes d'expansion à ajouter à la requête initiale afin de répondre efficacement aux besoins des utilisateurs en se basant sur trois facteurs :

1) La taille de la requête;
2) Le nombre de documents pertinents utilisés pour la réinjection de pertinence.
3) La combinaison des deux facteurs (1) et (2).

III.3.2.1. Facteur 1 : La taille de la requête

L'accroissement des performances de la reformulation de requête est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite **[50]**. Dans ce sens, des expérimentations intéressantes ont été réalisées sur la base TREC7 et présentées dans **[95]**. 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 très performants pour différentes tâches : recherche, filtrage et routing.

Dans ce sens, dans cette première étape notre objectif est de chercher le nombre optimal de termes à ajouter à une requête selon la taille de cette dernière, à savoir le nombre de termes d'expansion à ajouter à une requête n'est pas le même pour toutes les requêtes, il varie selon la taille de la requête, à savoir qu'une petite requête nécessite plus de termes à ajouter qu'une grande. Car nous supposant qu'une petite requête nécessite plus de termes pour être bien claire qu'une requête longue. Techniquement, le nombre de terme d'expansion est lié inversement à la taille de la requête.

Le calcule du nombre de termes d'expansion à ajouter se fait alors par la formule suivante :

$$\text{nbTerme} = f * \left(\frac{1}{\text{taille}}\right) \qquad \text{(III. 1)}$$

Où :

nbTerme : Nombre optimal de termes d'expansion à ajouter à une requête,

f est un facteur pour la taille,

Taille est la taille de la requête.

Pour illustrer le fonctionnement de cette étape, nous prenons l'exemple suivant :

Supposant qu'on a deux requêtes q1, q2, tels que :

taille q1=3, taille q2=5,

f={ 10 , 15, 20, 25 },

pour la requête q1 :

f=10, nbTerme = 3 ; f=15, nbTerme=5 ; f=20, nbTerme= 6 ; f=25, nbTerme=8.

pour la requête q2 :

f=10, nbTerme = 2 ; f=15, nbTerme=3; f=20, nbTerme= 4 ; f=25, nbTerme=5.

A partir de cette exemple, nous remarquons que le nombre optimal de termes à ajouter à la requête q1 est 8 et celui à ajouter à la requête q2 est 5, donc on déduit qu'une petite requête nécessite plus de termes à ajouter qu'une grande.

III.3.2.2. Facteur 2 : Le nombre de documents pertinents

Lorsqu'une personne interroge une base de données (que ce soit un logiciel documentaire ou un moteur de recherche), elle attend un nombre de réponses (sous forme de documents) supérieur ou égal à un, et pour retourner un réponse optimale à l'utilisateur, on doit choisir le nombre optimal de terme d'expansion à ajouter à sa requête initiale et cela selon le nombre de documents pertinents utilisés.

Tel que, tant que le nombre de documents pertinents utilisés est grand alors le nombre de termes d'expansion à ajouter est grand.

Le calcule du nombre de termes d'expansion à ajouter se fait alors par la formule suivante :

$$\text{nbTerme} = \text{f1} * \text{Nbdoc} \qquad \text{(III. 2)}$$

où :

nbTerme : Nombre optimal de termes d'expansion à ajouter à une requête,

f1 est un facteur pour les documents,

Nbdoc est le nombre de document pertinents utilisés.

Pour illustrer le fonctionnement de cette étape, nous prenons l'exemple suivant :

Supposant qu'on a deux requêtes q1, q2, tels que :

Nbdoc q1 = 4 , Nbdoc q2 = 6,

f1={ 2 , 3, 4, 5 },

pour la requête q1 :

f1=2, nbTerme = 8 ; f1=3, nbTerme=12 ; f1=4, nbTerme= 16 ; f1=5, nbTerme=20.

pour la requête q2 :

f1=2, nbTerme = 12 ; f1=3, nbTerme=18 ; f1=4, nbTerme= 24 ; f1=5, nbTerme=30.

A partir de cet exemple, nous remarquons que le nombre optimal de termes à ajouter à la requête q1 est 20 et celui à ajouter à la requête q2 est 30, donc on déduit que tant que le nombre de documents pertinents utilisés est grand alors le nombre de termes d'expansion à ajouter est grand.

III.3.2.2. Facteur 3 : Combinaison des facteurs

Cette étape est la dernière étape de notre approche, ou on va jumeler les deux étapes précédentes à savoir la propriété de la requête (taille) et la propriété sur les documents pertinents utilisés (nombre).

Le calcule du nombre de termes d'expansion à ajouter se fait par la formule suivante :

$$\text{NbTerme} = (\alpha 1 * (f * 1/\text{taille})) + ((1 - \alpha 1) * (f1 * \text{Nb doc})) \qquad \text{(III. 3)}$$

Où :

nbTerme : Nombre optimal de termes d'expansion à ajouter à une requête,

f est un facteur pour la taille,

Taille est la taille de la requête,

f1 est un facteur pour les documents,

Nbdoc est le nombre de document pertinents utilisés.

III.4 L'environnement technique

Dans ce qui suit, nous allons présenter notre environnement technique et spécifier les différents outils utilisés. Nous commençons par la plate forme Terrier sous laquelle nous avons implémenté notre approche, puis JAVA que nous avons utilisé comme langage de programmation et Netbans qui est l'outil sur le quel nous avons programmé.

III.4.1. La plateforme Terrier

TERRIER**, TERabyteRetrIEveR :** est un moteur de recherche 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.

Comme tous moteurs de recherche, terrier permet :

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

Architecture de Terrier

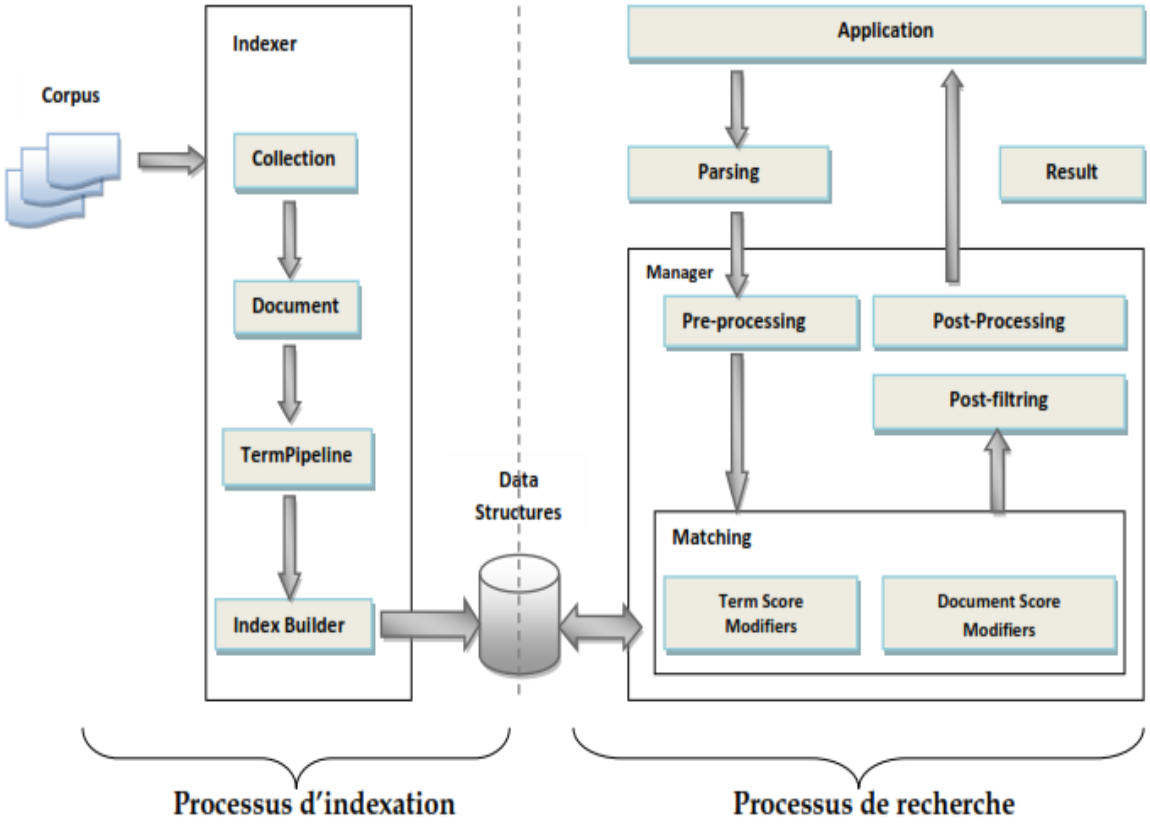

Figure III.2 : Vue d'ensemble d'architecture de Terrier

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 parser chaque document reçu et extraire les différents termes.

3) Traitement des termes extraits avec TermPipeline : consiste à 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.

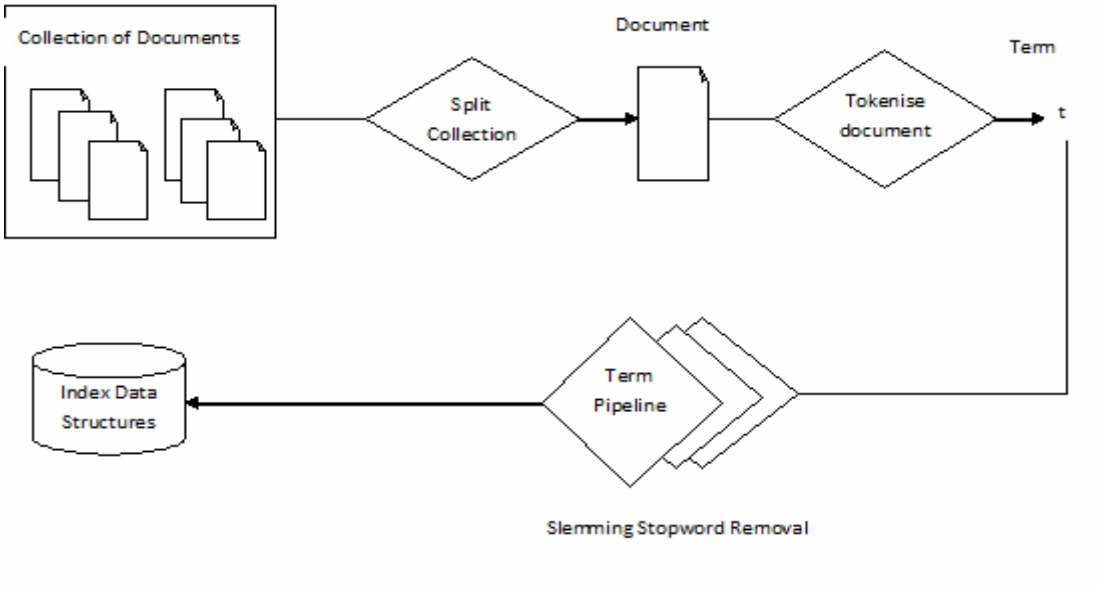

Figure III.3 : Le processus d'indexation dans Terrier

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, #fields).

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 repondérée, et une nouvelle recherche est faite à l'aide du modèle de dirichlet. 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.

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

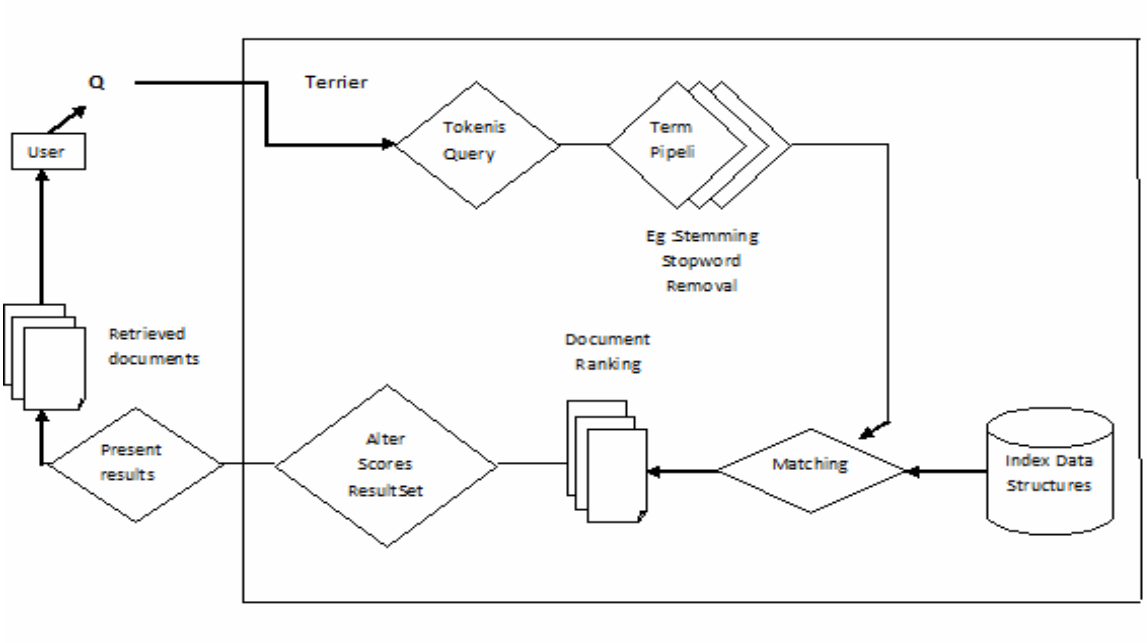

Figure III.4 : Le processus de recherche dans Terrier

III.4.2. Le langage java [96]

Java est un langage de programmation moderne développé par **Sun Microsystems** (aujourd'hui racheté par **Oracle**). Il ne faut pas surtout le confondre avec JavaScript (langage de scripts utilisé principalement sur les sites web), car Java n'a rien à voir. Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc. On peut faire de nombreuses sortes de programmes avec Java :

- des applications, sous forme de fenêtre ou de console ;
- des applets, qui sont des programmes Java incorporés à des pages web ;
- des applications pour appareils mobiles, avec J2ME ;
- et bien d'autres ! J2EE, JMF, J3D pour la 3D...

Java se voit attribuer plusieurs qualités dont voici quelques unes :

- **Java est orienté objet :** Java se veut un pur langage de P.O.O, c'est-à-dire qu'un programme s'y trouvera formé d'une classe ou de la réunion de plusieurs classes et il instanciera des objets.
- **Java est simple :** La syntaxe de Java est, en grande partie, tirée de celle de C++, avec l'avantage de l'élimination des mécanismes complexes, comme la gestion des pointeurs et de la mémoire, rendant ainsi les programmes java plus faciles à écrire et à compiler avec le moins d'erreurs possibles.

- **Java est distribué :** Java implémente les protocoles réseau standards, ce qui permet de développer des applications client/serveur en architecture distribuée, afin d'invoquer des traitements et/ou de récupérer des données sur des machines distantes.
- **Java est interprétée :** Un programme Java n'est pas exécuté, il est interprété par la machine virtuelle ou JVM (*Java Virtual Machine*), ce qui le rend un peu plus lent. Mais cela apporte des avantages, notamment celui de ne pas être obligé de recompiler un programme Java d'un système à un autre car il suffit, pour chacun des systèmes, de posséder sa propre machine virtuelle Java.
- **Java est robuste :** Java est un langage fortement typé et très strict. Par exemple la déclaration des variables doit obligatoirement être explicite en Java. Le code est vérifié (syntaxe, types) à la compilation et également au moment de l'exécution, ce qui permet de réduire les bugs et les problèmes d'incompatibilité de versions.
- **Java est sécurisé :** Au moment de l'exécution d'un programme Java, le JRE utilise un processus nommé le *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.
- **Java est portable :** cette caractéristique est l'une de celles qui ont contribué à sa grande réputation parmi les communautés d'internet et ceci grâce à son indépendance de toute plate forme d'exécution car un programme Java peut tourner sur n'importe quelle machine possédant une JVM (Java Virtual Machine).

III.4.3. 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

III.5. Résultats et expérimentations

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 utilisé :

Le modèle de recherche BM25, le modèle d'expansion Bo1 et les classes « Matching », « ExpansionTerms » et « QueryExpansion » qui font partie de la plate-forme Terrier (présentée dans III.4.1), et la classe étendu dans notre approche est « QueryExpansion ».

- ✓ ***La classe « Matching » :*** elle s'occupe de la correspondance entre la requête et les documents de l'index via la méthode Match qui nécessite le numéro de la requête et les termes de la requête et produit résultset 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 documents et la fréquence du terme dans les documents.

  Après avoir récupéré ces information, 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 *QueryExpansion*. Cette méthode fonctionne en faisant l'extraction des tops termes informatives à partir des tops documents classés (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 et le result set (Les identificateurs des tops documents et leurs score), elle modélise le nombre et la taille totale des documents d'expansion.

- ***La classe « ExpansionTerms »*** : nécessite 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 tops documents à l'aide de la méthode insert terms (Identificateur du terme et sa fréquence). Après récupération des termes d'expansion a partir de Expantion Terms Le classement des ces termes se fait à l'aide de la méthode get Expanded Term en lui spécifiant le modèle de classement et le nombre de termes d'expansion.

- ***Le Modèle « Bo1 »*** **:** Modèle d'expansion de requêtes par défaut.

Nous allons étendre la classe « QueryExpansion » et exactement sa classe « ExpandQuery », en se basant sur trois étapes dans lesquelles nous allons proposer des formules que nous allons évaluées et ajoutées à la classe étendue.

III.5.2. Collection de test utilisée

Pour évaluer les différentes formules proposées dans notre approche, nous avons mené nos expérimentations en utilisant la collection de test TREC AP88 (Associated Press newswire, 1988) décrite dans le tableau III.1.

Pour la recherche nous avons utilisé 50 requêtes issues des topics numérotées «51-100» de La collection TREC, où le champ « *title* » est considéré.

AP88		
Nombre de documents dans la collection	**Nombre de termes dans la collection**	**Taille moyenne d'un document**
79919	**144186**	**235.085**

Tableau III.1 Description de la collection test utilisée

III.5.3. Résultats et évaluation

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base (recherche simple et recherche avec expansion). Ces résultats seront par la suite comparés à ceux obtenus après la reformulation avec notre approche en appliquant les mêmes

paramètres de recherche (le nombre de documents d'expansion, le modèle de recherche BM25 et le modèle d'expansion Bo1).

Pour évaluer notre approche, nous allons faire une comparaison entre les résultats de reformulation obtenus avec le modèle d'expansion Bo1 et ceux obtenus avec notre approche (QueryExpanted_Etendu), et cela pour obtenir le nombre de termes d'expansion optimal. La mesure MAP (Moyenne Average Précision) décrite dans le chapitre I, est utilisée comme mesure d'évaluation.

III.5.2.1. Résultats de la recherche simple

Avant de procéder à la phase de reformulation, nous allons commencer par illustrer les résultats obtenus avec la recherche simple (sans reformulation de la requête).

Le tableau ci-dessous montre la précision (MAP) obtenue avec le modèle de recherche BM25.

Recherche simple	
Modèle de recherche	**MAP**
BM25	**0.1334**

Tableau III.2 Résultats obtenus avec la recherche simple

III.5.2.2 Résultats obtenus après reformulation

Dans cette étape, nous allons présenter les résultats obtenus après reformulation avec le modèle Bo1, nous l'avons testé en faisant varier les deux paramètres essentiels de ce modèle qui sont le nombre de documents d'expansion allant de **1** à **40** avec un pas de **5**, et pour chaque valeur de ce premier paramètre nous avons varié le nombre de termes d'expansion allant de **1** à **40** également avec un pas de **5**.

Les résultats obtenus sont résumés dans le tableau suivant :

Reformulation (Modèle Bo1)					
Nombre de documents	**Nombre de termes**	**MAP**	**Nombre de documents**	**Nombre de termes**	**MAP**
1	1	0.1579	21	1	0.2116
	6	0.1579		6	0.2130
	11	0.1579		11	0.2183
	16	0.1579		16	0.2190
	21	0.1579		21	0.2181
	26	0.1579		26	0.2203
	31	0.1579		31	0.2207
	36	0.1579		36	0.2219
6	1	0.2016	26	1	0.2252
	6	0.1997		6	0.2268
	11	0.2070		11	0.2279
	16	0.2033		16	0.2332
	21	0.2065		21	0.2349
	26	0.2042		26	0.2337
	31	0.2034		31	0.2330
	36	0.2069		36	0.2345
11	1	0.2112	31	1	0.2262
	6	0.2119		6	0.2277
	11	0.2095		11	0.2314
	16	0.2130		16	0.2334
	21	0.2118		21	0.2342
	26	0.2160		26	0.2363
	31	0.2156		31	0.2353
	36	0.2144		36	0.2343
16	1	0.2076	**36**	1	0.2236
	6	0.2111		6	0.2261
	11	0.2103		11	0.2317
	16	0.2115		16	0.2334
	21	0.2126		21	0.2358
	26	0.2124		26	0.2362
	31	0.2140		**31**	**0.2370**
	36	0.2137		36	0.2364

Tableau III.3 Résultats obtenus dans la reformulation avec le modèle d'expansion Bo1

Le tableau ci-dessus résume les résultats obtenus dans la reformulation avec le modèle d'expansion Bo1, nous constatons que la meilleure précision est de **0.2370,** obtenue avec un nombre de documents égal à **36** et un nombre de termes égal à **31**.

Dans l'étape de l'expansion étendue, nous avons remplacé la classe « QueryExpansion » situées dans la librairie de Terrier, avec celle que nous avons étendu avec les trois formules présentées précédemment (QueryExpansion_Etendue).
Nous avons ensuite testé l'expansion, on utilisant les résultats obtenus dans le tableau ci-dessus à savoir le nombre de document = **36** et on faisant varier les facteurs **f, f1 et α** des formules 1, 2 et 3 respectivement.
Les résultats obtenus sont comme suit :

III.5.2.3 Résultats obtenus avec notre approche (Facteur 1)

Dans cette étape, nous allons présenter les résultats obtenus avec le facteur 1(**f**), on le faisant varier tel que **f** allant de **10** à **150** avec un pas de **10**.

Reformulation (Formue 1)			
Valeurs de **f**	MAP	Valeurs de **f**	MAP
10	0.1976	90	0.2335
20	0.2174	100	0.2331
30	0.2198	110	0.2342
40	0.2282	120	0.2339
50	0.2334	130	0.2342
60	0.2342	140	0.2342
70	0.2326	**150**	**0.2356**
80	0.2333		

Tableau III.4 Résultats obtenus dans la reformulation avec notre approche (Facteur 1)

Le tableau ci-dessus résume les résultats obtenus dans notre approche avec la Formule 1(F1), nous constatons que la meilleure précision est de **0.2356,** obtenue avec une valeur de **f=150**.

III.5.2.4 Résultats obtenus avec notre approche (Facteur 2)

Dans cette étape, nous allons présenter les résultats obtenus avec le facteur 2(**f1**), on le faisant varier tel que **f1** allant de **0.5** à **8** avec un pas de **1**.

Reformulation (Formule 2)	
Valeurs de **f1**	MAP
0.5	0.2360
1.5	0.2372
2.5	0.2367
3.5	**0.2405**
4	0.2397
4.5	0.2377
5.5	0.2369
6.5	0.2372
7.5	0.2374

Tableau III.5 Résultats obtenus dans la reformulation avec notre approche (Facteur 2)

Le tableau ci-dessus résume les résultats obtenus dans notre approche avec la formule 2 (F2), nous constatons que la meilleure précision est de **0.2405,** obtenue avec une valeur de **f1=3.5**.

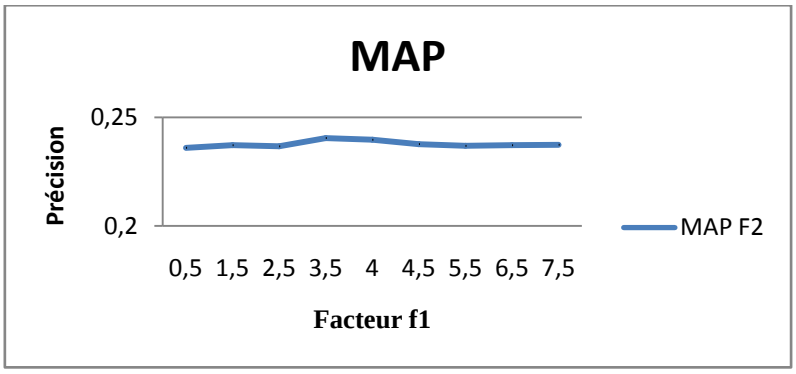

Figure III.5 Résultat de la MAP obtenus avec le facteur f1 de la Formule 2

III.5.2.5 Résultats obtenus avec notre approche (combinaison des Facteurs 1 et 2)

Dans cette étape, nous allons tester notre approche en considérant les résultats obtenus dans le tableau (III.3) à savoir le nombre de document = **36**, les résultats obtenus dans le tableau (III.4) à savoir la valeur de **f=150,** les résultats obtenus dans le tableau (III.5) à savoir la valeur de **f1=3.5,** et on faisant varier le facteur **α1** de la Formule (III.3), tel que **α1** allant de **0.1** à **1** avec un pas de **0.1.**

Reformulation (Formule 3)			
Valeurs de **α1**	MAP	Valeurs de **α1**	MAP
0.1	**0.2381**	0.6	0.2366
0.2	0.2379	0.7	0.2361
0.3	0.2365	0.8	0.2359
0.4	0.2367	0.9	0.2371
0.5	0.2370	1	0.2356

Tableau III.6 Résultats obtenus dans la reformulation avec notre approche (Facteur 3)

Le tableau ci-dessus résume les résultats obtenus dans notre approche avec la Formule 3 (F3), nous constatons que la meilleure précision est de **0.2381,** obtenue avec une valeur de **α1=0.1**.

III.5.3 Comparaison entre la reformulation avec le modèle Bo1 et notre approche

Nous résumons dans la figure III.5 les meilleures précisions obtenues dans les tableaux des deux sections précédentes à savoir, l'expansion Etendue de notre approche avec les formules proposées et la reformulation avec le modèle d'expansion Bo1.

RB = Reformulation avec le modèle d'expansion Bo1.
RF1= Reformulation avec notre approche (Formule 1).
RF2= Reformulation avec notre approche (Formule 2).
RF3= Reformulation avec notre approche (Formule 3).

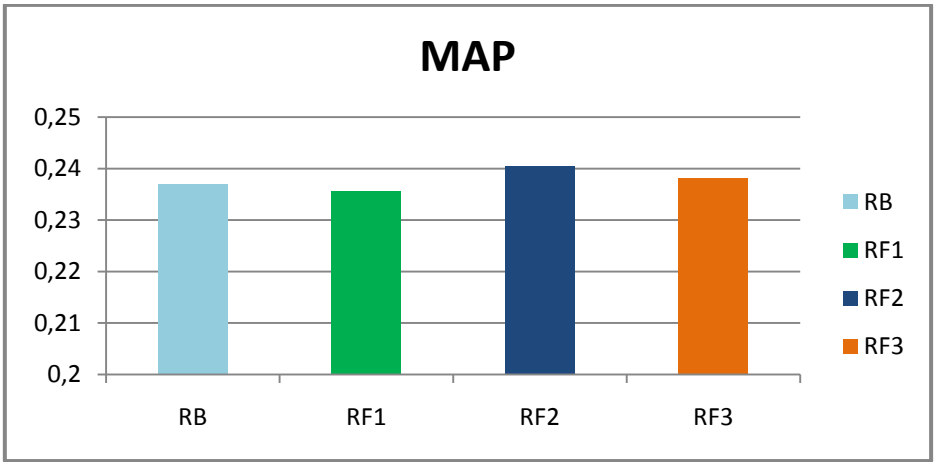

Figure III.6 Comparaison entre les meilleures précisions obtenues notre approche et la reformulation avec Modèle Bo1

Cette figure nous montre que la reformulation avec notre approche a apporté une meilleure précision avec les Formules 2 et 3 par rapport à la reformulation avec le modèle Bo1.

Le tableau ci-dessous nous montre le taux d'amélioration entre la reformulation avec Bo1 (MAP= 0.2370) et la reformulation avec notre approche (F1, F2, F3).
Nous avons pris pour chaque modèle la meilleure précision obtenue dans les deux types de reformulation.

Formules	MAP	Taux d'amélioration
F1	0.2356	**-0,59**
F2	0.2405	**+1,47**
F3	0.2381	**+0,46**

Tableau III.7 Taux d'améliorations obtenues avec notre approche

D'après le tableau ci-dessus, nous constatons que :
La Formule **(F1**), n'a pas apporté d'amélioration par rapport à la reformulation Bo1, contrairement à la formule (**F2**) qui a apporté une amélioration avec une précision de **0.2405,** et un taux d'amélioration de **1,47%.**
Ainsi que la formule (**F3**) qui a apporté une amélioration avec une précision de **0.2381,** et un taux d'amélioration de **0,46%.**

III.5.4 Evaluation requête par requête

Les résultats montrés précédemment donnent une appréciation globale, afin d'avoir une vue précise de ces résultats, nous avons analysé les résultats requête par requête. Nous commençons par évaluer les résultats requête par requête de la recherche simple (sans reformulation) et les comparés avec ceux de la reformulation avec notre approche (Expansion Etendue avec la Formule1), puis on va évaluer les résultats requête par requête de la reformulation avec le modèle d'expansion Bo1 et les comparés avec ceux de la reformulation avec notre approche.

III.5.4.1 Evaluation requête par requête (Recherche simple comparé à notre approche)

Le tableau ci-dessous montre les résultats obtenus avec la recherche simple et les comparés avec ceux de la reformulation avec notre approche (Formule 1), en introduisant la taille de la requête comme facteur pour évaluer le taux d'amélioration par rapport à ce dernier.

RS = Recherche Simple.

RF1= Reformulation avec notre approche (Formule 1).

Requête	Taille requête	MAP (RS)	MAP (RF1)	Taux (%)	Requête	Taille requête	MAP (RS)	MAP (RF1)	Taux (%)
51	2	0.5351	0.8637	**61,40**	76	4	0.0079	0.0024	**-69,62**
52	3	0.5334	0.6893	**29,22**	77	1	0.0455	0.1019	**123,95**
53	2	0.1850	0.5204	**181,29**	78	1	0.1661	0.7355	**342,80**
54	3	0.4443	0.7684	**72.94**	79	4	0.0000	0.0000	**0**
55	2	0.2758	0.4816	**74,62**	80	4	0.0025	0.0334	**1236**
56	5	0.6325	0.8419	**33,10**	81	11	0.1515	0.1251	**-17,42**
57	1	0.1838	0.6900	**275,40**	82	2	0.2092	0.5513	**163,52**
58	2	0.0213	0.0164	**-23,00**	83	5	0.0076	0.0015	**-80,26**
59	3	0.0087	0.0198	**127,58**	84	6	0.0641	0.0280	**-56,31**
60	4	0.0012	0.0012	**0**	85	2	0.0673	0.0426	**-36,70**
61	6	0.5074	0.6829	**34,58**	86	2	0.0683	0.1362	**99,41**
62	3	0.0038	0.0495	**1202,63**	87	8	0.0095	0.0307	**223,15**
63	2	0.2676	0.2215	**-17,22**	88	4	0.0924	0.0344	**-62,77**
64	2	0.0203	0.0055	**-72,90**	89	6	0.0068	0.0095	**39,70**
65	3	0.0000	0.0000	**0**	90	9	0.3968	0.4795	**20,84**
66	3	0.0000	0.0006	**0.06**	91	7	0.0003	0.0000	**-100**
67	4	0.0010	0.0057	**470**	92	4	0.0004	0.0005	**25**
68	6	0.0702	0.3318	**372,65**	93	8	0.0692	0.4794	**592,77**
69	7	0.0660	0.0612	**-7,27**	94	3	0.0091	0.0208	**128,57**
70	2	0.2336	0.8259	**253,55**	95	4	0.0024	0.0045	**87,50**
71	2	0.0488	0.4329	**787,09**	96	4	0.0202	0.0197	**-2,47**
72	5	0.0010	0.0001	**-90**	97	3	0.0919	0.2366	**157,45**
73	5	0.0003	0.0002	**-33,33**	98	4	0.3439	0.3669	**6,68**
74	2	0.0004	0.0005	**25**	99	3	0.2486	0.5574	**124,21**
75	1	0.0139	0.0126	**-9,35**	100	6	0.0094	0.0222	**136,17**

Tableau III.8 Comparaison de l'analyse requête par requête de la recherche simple et de l'expansion avec la Formule F1.

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

- ✓ **32** requêtes améliorées, soit 64%,
- ✓ **15** requêtes dégradées, soit 30%,
- ✓ **03** requêtes nulles, soit 6%.

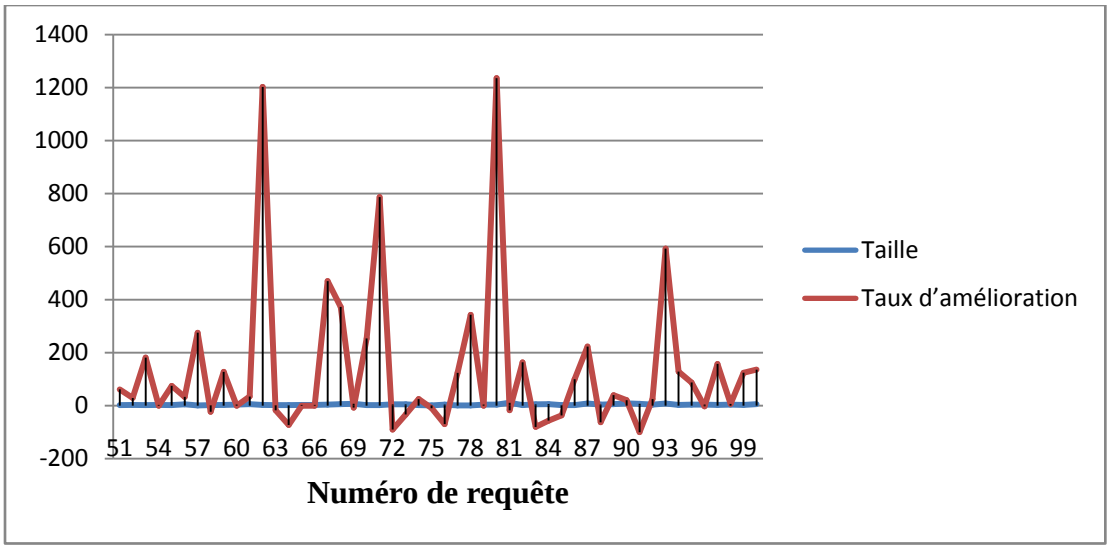

Figure III.7 Comparaison entre le taux d'amélioration de l'analyse requête par requête et la taille de la requête. (Recherche simple)

Les résultats obtenus dans la figure ci-dessus nous donnent les résultats suivants :

- ✓ 3 requêtes améliorées, 1 requête dégradée sur 4 requêtes dont le nombre de termes de la requête (taille) égal à 1 avec un taux d'amélioration de 9,37%.
- ✓ 8 requêtes améliorées, 4 requêtes dégradées sur 12 requêtes dont le nombre de termes égal à 2 avec un taux d'amélioration de 25%
- ✓ 8 requêtes améliorées 1 requête nulle sur 9 requêtes dont le nombre de termes égal à 3 avec un taux d'amélioration de 25%.
- ✓ 5 requêtes améliorées, 3 requêtes dégradées, 2 requêtes nulles sur 10 requêtes dont le nombre de termes égal à 4 avec un taux d'amélioration de 15,62%.
- ✓ 3 requêtes améliorées, 1 requête nulle sur 4 requêtes dont le nombre de termes égal à 5 avec un taux d'amélioration de 9,37%.
- ✓ 4 requêtes améliorées, 1 requête dégradée sur 5 requêtes dont le nombre de termes égal à 6 avec un taux d'amélioration de 12,50%.
- ✓ 2 requêtes dégradées sur 2 requêtes dont le nombre de termes égal à 7.
- ✓ 2 requêtes améliorées sur 2 requêtes dont le nombre de termes égal à 8 avec un taux d'amélioration de 6,25%.
- ✓ 1 requête améliorée sur 1 requête dont le nombre de termes égal à 9 avec un taux d'amélioration de 3,12%.
- ✓ 1 requête dégradée sur 1 requête dont le nombre de termes égal à 11.

Les résultats ont révélé une amélioration considérable pour les requêtes dont la taille est de 2, 3 et 4 termes.

III.5.4.2 Evaluation requête par requête (Reformulation avec le modèle d'expansion Bo1 comparé à notre approche)

Le tableau ci-dessous montre les résultats obtenus avec la reformulation avec le modèle d'expansion Bo1 et les comparés avec ceux de la reformulation avec notre approche (Formule 1), en introduisant la taille de la requête comme facteur pour évaluer le taux d'amélioration par rapport à ce dernier.

RB = Reformulation avec le modèle d'expansion Bo1.

RF1= Reformulation avec notre approche (Formule 1).

Requête	Taille requête	MAP (RB)	MAP (RF1)	Taux (%)	Requête	Taille requête	MAP (RF)	MAP (RF1)	Taux (%)
51	2	0.8588	0.8637	**0,57**	76	4	0.0015	0.0024	**60**
52	3	0.6824	0.6893	**1,01**	77	1	0.0862	0.1019	**18,21**
53	2	0.5193	0.5204	**0,21**	78	1	0.7433	0.7355	**-1,05**
54	3	0.7581	0.7684	**1,35**	79	4	0.0000	0.0000	**0**
55	2	0.4804	0.4816	**0,25**	80	4	0.0276	0.0334	**21,01**
56	5	0.8305	0.8419	**1,37**	81	11	0.1280	0.1251	**-2,26**
57	1	0.6901	0.6900	**-0,01**	82	2	0.5506	0.5513	**0,12**
58	2	0.0163	0.0164	**0,61**	83	5	0.0008	0.0015	**87,5**
59	3	0.0185	0.0198	**7,02**	84	6	0.0200	0.0280	**40**
60	4	0.0025	0.0012	**-52**	85	2	0.0426	0.0426	**0**
61	6	0.6833	0.6829	**-0,05**	86	2	0.1337	0.1362	**1,86**
62	3	0.0396	0.0495	**25**	87	8	0.0770	0.0307	**-60,13**
63	2	0.2215	0.2215	**0**	88	4	0.0327	0.0344	**5,19**
64	2	0.0055	0.0055	**0**	89	6	0.0100	0.0095	**-5**
65	3	0.0000	0.0000	**0**	90	9	0.5038	0.4795	**-4,82**
66	3	0.0009	0.0006	**-33,33**	91	7	0.0000	0.0000	**0**
67	4	0.0050	0.0057	**14**	92	4	0.0008	0.0005	**-37,5**
68	6	0.2882	0.3318	**15,12**	93	8	0.4909	0.4794	**-2,34**
69	7	0.0519	0.0612	**17,92**	94	3	0.0205	0.0208	**1,46**
70	2	0.8259	0.8259	**0**	95	4	0.0045	0.0045	**0**
71	2	0.4314	0.4329	**0,34**	96	4	0.0232	0.0197	**-15,08**
72	5	0.0001	0.0001	**0**	97	3	0.2552	0.2366	**-7,28**
73	5	0.0002	0.0002	**0**	98	4	0.4044	0.3669	**-9,27**
74	2	0.0004	0.0005	**25**	99	3	0.6045	0.5574	**-7,79**
75	1	0.0124	0.0126	**1,61**	100	6	0.0299	0.0222	**-25,75**

Tableau III.9 Comparaison de l'analyse requête par requête obtenue dans la reformulation avec le modèle Bo1 et l'expansion avec la Formule F1.

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

- ✓ **24** requêtes améliorées, soit 48%,
- ✓ **16** requêtes dégradées, soit 32%,
- ✓ **10** requêtes nulles, soit 20%.

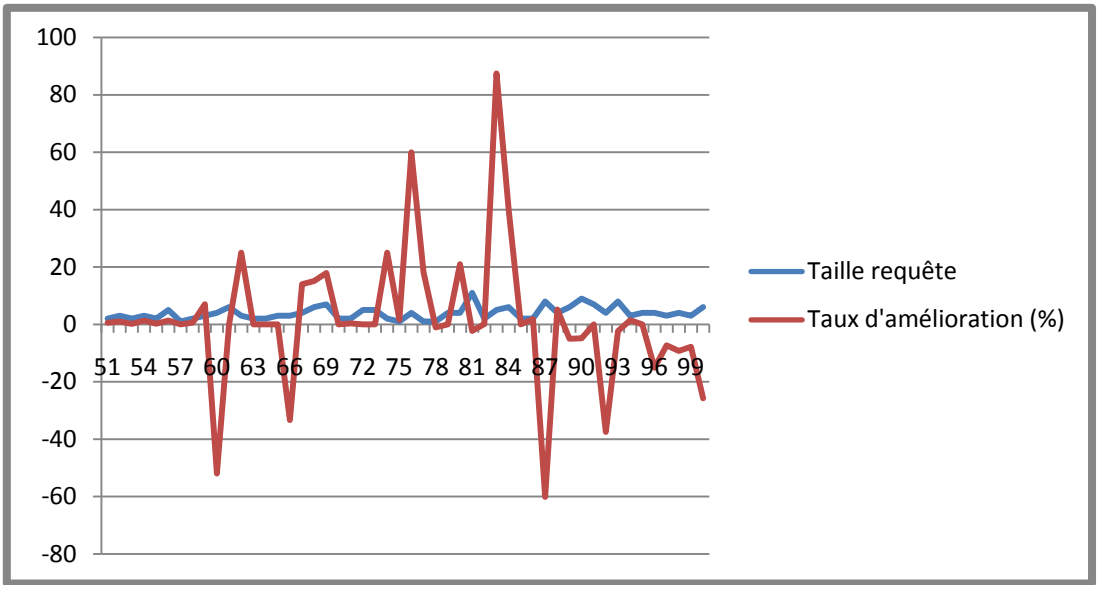

Figure III.8 Comparaison entre le taux d'amélioration de l'analyse requête par requête et la taille de la requête. (Reformulation avec Bo1)

Les résultats obtenus dans la figure ci-dessus nous donnent les résultats suivants :

- ✓ 2 requêtes améliorées, 2 requêtes dégradées sur 4 requêtes dont le nombre de termes de la requête (taille) égal à 1 avec un taux d'amélioration de 8,33%.
- ✓ 8 requêtes améliorées, 4 requêtes nulles sur 12 requêtes dont le nombre de termes égal à 2 avec un taux d'amélioration de 33,33%
- ✓ 5 requêtes améliorées, 3 requêtes dégradées, 1 requête nulle sur 9 requêtes dont le nombre de termes égal à 3 avec un taux d'amélioration de 20,83%.
- ✓ 4 requêtes améliorées, 4 requête dégradées, 2 requêtes nulles sur 10 requêtes dont le nombre de termes égal à 4 avec un taux d'amélioration de 16,66%.
- ✓ 2 requêtes améliorées, 2 requêtes nulles sur 4 requêtes dont le nombre de termes égal à 5 avec un taux d'amélioration de 8,33%.
- ✓ 2 requêtes améliorées, 3 requêtes dégradées sur 5 requêtes dont le nombre de termes égal à 6 avec un taux d'amélioration de 8,33%.
- ✓ 1requête améliorée, 1 requête nulle sur 2 requêtes dont le nombre de termes égal à 7 avec un taux d'amélioration de 4,16%.
- ✓ 2 requêtes dégradées sur 2 requêtes dont le nombre de termes égal à 8.
- ✓ 1 requête dégradée sur 1 requête dont le nombre de termes égal à 9.
- ✓ 1 requête dégradée sur 1 requête dont le nombre de termes égal à 11.

Les résultats ont révélé une amélioration considérable pour les requêtes dont la taille est de 2, 3 et 4 termes.

Nous avons analysé les requêtes par requêtes pour la formule 1 (F1) car elle est basée sur un facteur dépendant de la requête (taille).

D'après les résultats d'analyse des figures (III.7 et III.8), nous constatons que le facteur (Taille de la requête) donne de meilleurs résultats pour les petites requêtes.

III.6. Conclusion

Dans ce chapitre nous avons exposé notre approche qui consiste à estimer d'une manière automatique le nombre de termes d'expansion en utilisant les propriétés suivantes : la taille de requête, nombre de document et la combinaison des deux propriétés et cela en proposant trois formules , puis nous avons testé ces formules et obtenus des résultats que nous avons présentés sous forme de tableaux et de graphes pour les comparer aux résultats obtenus avec la recherche simple et le modèle d'expansion Bo1.

Le test de cette approche en introduisant ces propriétés nous a apporté de meilleures résultats en terme de précision, ainsi que la pertinence retournés à l'utilisateur, dans le cas des dernières formules.

Conclusion générale

Conclusion générale

Le travail développé dans ce mémoire s'inscrit dans le cadre de la reformulation de requêtes, nous nous sommes particulièrement intéressés à la recherche de nombre optimal de termes d'expansion à ajouter à la requête initiale.
En effet, l'expansion de requête ce fait en faisant la sélection, le classement, et le choix des termes d'expansion d'une requête initiale soumise par l'utilisateur jugée courte ou mal exprimée.

Dans notre approche implémentée, nous avons étendu quelques classes de la plateforme de terrier pour la recherche du nombre optimal de termes d'expansion à ajouter à la requête initiale, en se basant sur trois facteurs: la taille de la requête initiale, le nombre de documents d'expansion et la combinaison de ces deux facteurs.

D'après les résultats obtenus nous sommes sur la bonne voix avec la collection de test TREC AP88, et parmi les perspectives envisagées pour notre travail est l'expérimentation de notre approche sur d'autres collections plus volumineuses afin de valider les résultats obtenus. Et examiner l'apport d'autre facteurs, tels que : la clarté de la requête, la taille de la collection, etc.

Références Bibliographiques

Bibliographie

[1] : G. Salton, The Smart Retrieval System : Experiments in Automatic Document Processing, G. Salton Editor, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971.

[2] : 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

[3] A.F. Smeaton. "Information retrieval and natural language processing". In proceedings of a conference jointly sponsored by ASLIB, University of York, page 2, march 1989.

[4] C. Tambellini. "Un système de recherche d'information adapté aux données incertaines: adaptation du modèle de langue". Thèse de doctorat en informatique, Université de Nice-Sophia Antipolis-UFR sciences, 2007.

[5] Fuhr, N.Information Retrieval – From Information Access to Contextueal Retrieval. In M. Eibl, C. Wolf, and C. Womser-Hacker, editors, *Designing Information Systems*. Festschrift fur Jurgen Krause, pp. 47-57. UVK Verlagsgesellschaft, 2005.

[6] G. Salton, E.A. Fox, H. Wu. Extended Boolean information retrieval system. *CACM* 26(11). pp.1022-1036, 1983.

[7]: Rijsbergen, C. (1979). *Information retrieval, Second edition*. Butterworths.

[8]: Salton, G. (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman PublishingCo., Inc., Boston, MA, USA.

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

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

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

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

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

[14]: F. Ren, L. Fan, and J.-Y. Nie. Saak approach: How to acquire knowledge in an actual application system. In IASTED International Conference on Artificial Intelligence and Soft Computing, pages 136–140, 1999.

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

[16]: Fox, C. Lexical analysis and stoplists, Frankes W B, Baeza-Yates R (eds) *Prentice Hall*, New jersey, pp. 102-130. 1992.

[17]: Frakes W. B., Stemming Algorithms, pages 131–160. Frakes W B, Baeza-Yates R (eds).

[18]: M. F. Porter. An algorithm for suffix stripping. Program, 1980.

[19]: G. W. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. Information Storage and Retrieval, 10(7-8):253–260, 1974.

[20]: Craven Timothy C., HTML Tags as Extraction Cues for Web Page Description Construction, The University of Western Ontario, London, Ontario, Canada.

[21]: M. Lesk. Grab - inverted indexes with low storage overhead. Computing Systems, 1(3):207–220, 1988.

[22]: A. Moffat and L. Stuiver. Exploiting clustering in inverted file compression. Data Compression Conference, pages 82–91, 1996.

[23]: Radecki, T. Fuzzy set theoretical approach to document retrieval. Information Processing and Management, 15: pp. 247-259, 1979.

[24]: Kraft, D.H. and Buell, D.A. Fuzzy sets and generalized Boolean retrieval systems, *International journal on Man-Machine studies*, 19: pp.49-56, 1983.

[25]: Lv, Y., Zhai C. Positional Relevance Model for Pseudo-Relevance Feedback. *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp.579-586, 2010.

[26]: Robertson, S.E. , S, Walker. On relevance weights with little relevance information. *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 16-24, 1997.

[27]: Singhal, A., Salton, G., Mitra, M., Buckley, C. Document lengh normalization. *Information Processing and Management*, 32(5). pp. 619-633, 1996.

[28]: Van Rijisbergen, C. J. Information retrieval. London: Butterworth, 1979.

[29]: Wong, S., Ziarko, W., Wong, P. Generalized vector space model in information retrieval. *Proceedings of the 8th ACM SIGIR Conference on Research and Development in information retrieval*, New-York, USA, pp. 18-25, 1985.

[30]: Berry, M.W., Dumais, S.T., O'Brien, G. W. 1995. *Using linear algebra for intelligent information retrieval* . SIAM Rev, 37(4), pp. 573-595, 1995.

[31]: Dumais, S. Latent Semantic Indexing (LSI). (Proceeding of TREC-3, 1994.

[32]: Foltz, P. W. Using Latent Semantic Indexing for information filtering. CACM, pp. 40-47, 1990.

[33]: Furnas, G.W., Landauer, T.K. , Gomez, L.M., Dumais S.T. The Vocabulary Problem in Haman-System Communication, *Communications of the ACM 30*, pp. 964-971, 1987.

[34]: Scott C. Deerwester**,** Susan T. Dumais, Thomas K. Landauer, George W. Furnas and Richard A. Harshman "Indexing by Latent Semantic Analysis". *In Journal of the American Society of Information Science,* Vol. 41:6, pp.391-407, 1990.

[35]: Boughanem, M. Les Systèmes de recherche d'Information: d'un modèle classique à un modèle connexionniste. Thèse de Doctorat de l'Université Paul Sabatier, 1992.

[36]: Kwok, K. L. A neural network for probabilistic information retrieval. *International ACM SIGIR Conference on Research and Developpement in information retrieval*, pp 21-30, 1989.

[37]: Cao, G., Gao, J., Nie, J.Y., Roberston, S.. Selecting good expansion terms for pseudorelevance feedback. *Proceedins of the 31 st Annual International ACMSIGIR Conference on Research and Developpement in information retrieval*. ACM Press, pp 243-250, 2008.

[38]: Cao, G., Nie, J.Y., Bai, J. Integrating word relationships into langage models. *Proceedings of ACMSIGIR Conference on Research and Developpement in information retrieval*. pp 298-305, 2005.

[39]: Croft, W.B., Harper, D. J. Using Probabilistic Models of Document Retrieval without Relevance Information. *Journal of Documentation*, 35 (4): pp. 285-295, 1979.

[40]: Cho, J., Garcia-Molina, H., Page, L. the anatomy of efficient crawling through url ordering. *Computer networks and ISDN Systems*, 30(1-7), pp. 161-172. 1998.

[41]: Sanderson, M. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval* 4, pp. 247-375, 2010.

[42]: Baeza-Yates, R., Ribeiro-Neto, B. A. Modern Information Retrieval. *Pearson Education Ltd*., Harlow, UK, 2nd edn, 2011.

[43]: Nongdo Désiré Kompaoré. Thèse de doctorat «*Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes : vers un processus de RI adaptatif »* l'Université Toulouse III - Paul Sabatier, 2008.

[44]: Van Rijsbergen, C. J.*Information retrieval*. London: Butterworth, 1979.

[45]: Blair D., Maron M., « An Evaluation of Retrieval Effectivness for a Full-Text Document Retrieval System », *Communication of the ACM*, vol. 28, n° 3, 1985, p. 289-299.

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

[47]: Carpineto, C., Romano, G., "A Survey of Automatic Query Expansion in Information retrieval". ACM Computing Surveys, Vol. 44, No. 1, 2012.

[48]: Fatiha Boubekeur-Amirouche, Thèse Doctorat, "*Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets*", 2008.

[49]: Attar, R., Fraenkel, A. S., « Local Feedback in Full-Text Retrieval Systems », *J. ACM*, vol. 24, n° 3, 1977, pp. 397–417, ACM Press.

[50]: Buckley, C., Salton, G., Allan, J., Singhal, A., « Automatic Query Expansion Using SMART : TREC 3 », *Text REtrieval Conference*, 1994.

[51] : Croft W. B., Harper D. J., *Using probabilistic models of document retrieval without relevance information*, Morgan Kaufmann Publishers Inc., 1997.

[52]: Rocchio, J.J. Relevance Feedback in Information Retrieval, in The Smart System Experiments in Automatic Document Processing in Automatic Document Processing. *Editor Prentice-Gall.* Pp. 313-324, 1971.

[53] : Jones K. S., D.M. Jackson, « The use of automatically-obtained keyword classifications for information retrieval », *Information Processing and Management*, vol. 5, 1970, p. 175–201.

[54]: Qiu Y., Frei H.-P., « Concept-based query expansion », *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, Pittsburgh, US, 1993, pp. 160–169.

[55]: Jing Y., Croft W. B., « An association thesaurus for information retrieval », *Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur"*, New York, US, 1994, pp. 146–160.

[56]: Deerwester S., Dumais S., Landauer T., Furnas G., Harsman R., « Indexing by latent semantic analysis », *Journal of the Society for Information Science, vol. 41*, 1990, pp. 391–407.

[57] : Landauer. T., Foltz. P., Laham. D., « An Introduction to Latent Semantique Analysis», *Discourse Process*, vol. 25, 1998, pp. 259–284.

[58]: Xu. J., Croft W. B., « Improving the effectiveness of information retrieval with local context analysis », *ACM Trans. Inf. Syst.*, vol. 18, n° 1, 2000, pp. 79–112, ACM Press.

[59]: Robertson, S.E. , Sparck Jones, K. Relevance Weighting of Search Terms. Journal of the American Society for Information Science 27, pp. 129-146, 1976.

[60]: Boughanem, M., Chrisment, C., Soule-Dupuy, C. Query modification based on relevance back-propagation in adhoc environment. Information Processing and Management, 35, pp. 121-139, 1999.

[61]: Harman, D. Relevance feedback and other query modification techniques. In Information Retrieval : Data Structures and Algorithms, William B. Frakes and Ricardo Baeza-Yates, editors, Prentice Hall, Englewood, Cliffs, NJ, pp. 241-263, 1992.

[62]: Voorhees, E. 2004. Overview of the trec 2004 robust track. *In Proceedings of the 13th Text REtrieval Conference (TREC-7)*, NIST Special Publication 500-261. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA.

[63]: Diaz, F. and Metzler, D. Improving the estimation of relevance models using large external corpora. *In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, Seattle, Washington, USA, pp. 154–161. 2006.

[64]: Chirita, P.-A., Firan, C. S., and Nejdl, W. Personalized query expansion for the web. *In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, Amsterdam, The Netherlands, pp. 7–14. 2007.

[65]: Crouch, C. and Yang, B. Experiments in automatic statistical thesaurus construction. *In Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, Copenhagen, Denmark, pp. 77–88, 1992.

[66]: Kraft, R. and Zien, J. Mining anchor text for query refinement. *In Proceedings of the 13th international conference on World Wide Web*. ACM Press, New York, NY, USA, pp. 666–674, 2004.

[67]: Gauch, S., Wang, J., and Rachakonda, S. M. 1999. A corpus analysis approach for automatic query expansion and its extension to multiple databases. ACM Transactions on Information Systems (TOIS) 17, 3, 250–269.

[68]: He, B. and Ounis, I. 2007. Combining fields for query expansion and adaptive query expansion. *Information Processing and Management* 43, 1294–1307.

[69]: Attar, R. and Fraenkel, A. S. 1977. Local Feedback in Full-Text Retrieval Systems. Journal of ACM 24, 3, 397–417.

[70]: van Rijsbergen, C. J. 1979. Information Retrieval. Butterworths.

[71]: Church, K. and Hanks, P. 1990. Word association norms, mutual information and lexicography. Computational Linguistics 16, 1, 22–29.

[72]: Bai, J., Nie, J.-Y., Cao, G., and Bouchard, H. 2007. Using query contexts in information retrieval. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, Amsterdam, The Netherlands, 15–22.

[73]: Voorhees, E. 1994. Query expansion using lexical-semantic relations. *In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, Dublin, Ireland, 61–69.

[74]: Xu, J. and Croft, W. B. 1996. Query expansion using local and global document analysis. *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, Zurich, Switzerland, 4–11.

[75]: Cui, H., Wen, J.-R., Nie, J.-Y., and Ma, W.-Y. 2003. Query expansion by mining user logs. IEEE Transactions on Knowledge and Data Engineering 15, 4, 829–839.

[76]: Bai, J., Song, D., Bruza, P., Nie, J.-Y., and Cao, G. 2005. Query expansion using term relationships in language models for information retrieval. *In Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM Press, Bremen, Germany, 688–695.

[77]: Bai, J., Nie, J.-Y., and Cao, G. 2006. Context-dependent term relations for information retrieval. *In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, 551–559.

[78]: Collins-Thompson, K. and Callan, J. 2005. Query expansion using random walk models. In Proceedings of the 14th Conference on Information and Knowledge Management (CIKM 2005). ACM Press, Bremen, Germany, 704–711.

[79]: Lin, J. and Murray, G. C. 2005. Assessing the term independence assumption in blind relevance feedback. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. ACM Press, Salvador, Brazil, 635–636.

[80]: Chang, Y., Ounis, I., and Kim, M. 2006. Query reformulation using automatically generated query concepts from a document space. Information Processing and Management 42, 2, 453–468.

[81]: Amati, G., Carpineto, C., and Romano, G. 2003. *Comparing weighting models for monolingual information retrieval*. In Proceedings of the 4th Workshop of the Cross-Language Evaluation Forum, CLEF 2003. Springer, Trondheim, Norway, 310–318.

[82] : Buckley, C., Salton, G., Allan, G., and Singhal, A. 1995. Automatic query expansion using smart: Trec3. In Proceedings of the 3rd Text REtrieval Conference (TREC-3), NIST Special Publication 500-226. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 69–80.

[83]: Wong, W. S., Luk, R. W. P., Leong, H. V., Ho, K. S., and Lee, D. L. 2008. Re-examining the effects of adding relevance information in a relevance feedback environment. Information Processing and Management 44, 3, 1086–1116.

[84]: Bernardini, A. and Carpineto, C. 2008. Fub at trec 2008 relevance feedback track: extending rocchio with distributional term analysis. In Proceedings of TREC-2008. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA.

[85]: Zhai, C. and Lafferty, J. 2001. Model-based feedback in the language modeling approach to information retrieval. In Proceedings of the tenth international conference on Information and knowledge management. ACM Press, Atlanta, Georgia, USA, 403–410.

[86]: Buckley, C. and Harman, D. K. 2003. Reliable information access final workshop report. In Reliable Information Access Workshop (RIA), NRRC. NRRC, Bedford, Massachusetts, USA, 1–30.

[87]: Billerbeck, B. and Zobel, J. 2004a. Questioning query expansion: an examination of behaviour and parameters. In Proceedings of the 15th Australasian database conference - Volume 27. Australian Computer Society, Dunedin, New Zealand, 69–76.

[88]: Carpineto, C., Romano, G., and Giannini, V. 2002. Improving retrieval feedback with multiple term-ranking function combination. ACM Transactions on Information Systems (TOIS) 20, 3, 259–290.

[89]: Cronen-Townsend, S. and Croft, W. B. 2002. Quantifying query ambiguity. *In Proceedings of the 2nd international conference on Human Language Technology Research*. ACM Press, San Diego, California, 104–109.

[90]: Salton, G., Buckley., C. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41, pp. 288-297, 1990.

[91]: Montague, M. and Aslam, J. 2001. Relevance score normalization for metasearch. *In Proceedings of the 10th international conference on Information and knowledge management*. ACM Press, Atlanta, Georgia, USA, 427–433.

[92]: Lavrenko, V., & Croft, W. B. Relevance-based language models. In W.B Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds). *Proceedings of the 24th Annual International ACM-SIGIR Conference on Research and Developpement in information retrieval*, New Orleans, Louisiana, pp. 120-127, 2001.

[93]: Chakrabarti, S. Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., Kleinberg, J. Automatic resource list compilation by analyzing hyperlink structure and associated text. *Proceedings of the 7th international World Wide Web Conference*, pp. 65-74, 1998.

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

[95]: G. Cormack, C.R. Palme, M.V Biesbrouk & C.L.A. Clarck,"*Deriving Very Short Queries for High Precision and Recall*", In Proceedings of the7thText Retrieval Conference TREC7, July 1999.

[96] : Thierry Groussard, " java 6 les fondamentaux du language java", ENI editions, 2009.

[97] : 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.