République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de La Recherche Scientifique
Université Mouloud MAMMERI de Tizi-Ouzou
Faculté de Génie électrique et de l'Informatique
Département Informatique

MEMOIRE

de fin d'études

En vue de l'obtention de diplôme Master en informatique

Option : Systèmes Informatiques

Thème

« Recherche d'information : Reformulation de requêtes basée sur la position des termes»

Réalisé par :

M^elle^ Hamadou Lynda

M^r^ Hadj-said Massinissa

Proposé et dirigé par :

M^r^ A. Hammache

Promotion : 2014/2015

Remerciements

Louange à notre créateur qui nous incité à acquérir le savoir et nous a donné la volonté et le courage pour y arriver. C'est à lui que nous adressons toute notre gratitude en premier lieu.

Nous exprimons notre profonde gratitude à notre promoteur, Monsieur ***A.Hammache*** *pour avoir dirigé ce travail, pour la documentation qu'elle nous a fourni, pour ses critiques, ses conseils et ses orientations durant toute la période de notre travail.*

Nous remercions aussi profondément les membres de jury pour nous avoir fait l'honneur de juger notre travail.

Enfin, nous tenons à remercier nos familles respectives ainsi toute personne ayant contribué de prés ou de loin à l'aboutissement de ce travail…

Dédicace

Je dédie ce modeste travail :

A la lumière de ma vie, mes très chers parents qui m'ont soutenu tout au long de mes études.

A mon frère Ali et ma sœur Dihia.

A toute ma famille.

A tous mes amis. En particulier à

Said, Ali, Madjid, Rahim, Zoufri et Kenza.

Je remercie tous ceux et celles qui m'ont aidé à réaliser ce travail.

MASSINISSA

Dédicace

Je dédie ce modeste travail à :

A mes très chers parents, qui m'ont soutenu tout au long de mes études.

A mes frères et sœur Amar, Ali, Ghiles,Warda.

A tout mes amis. En particulier à Samir, Tinhinane, Nassima, Zina et tiziri

Lynda

Sommaire

Introduction générale....1

Chapitre I : Recherche d'information

I.1 Introduction....3

I.2 Concepts de base de la recherche d'information....3

I.2.1 Système de recherche d'information....3

I.2.1.1 Document et collection de documents....5

I.2.1.2 Besoin en information et requête....5

I.2.1.3 Pertinence....6

I.2.2 Le processus d'indexation....6

I.2.2.1 L'analyse lexicale....7

I.2.2.2 L'élimination des mots vides (Sélection de descripteurs)....8

I.2.2.3 La normalisation (Lemmatisation)....9

I.2.2.4 Construction de l'index....9

I.2.3 Appariement document requête....10

I.2.4 Le reformulation de la requête....10

I.3 Les modèles de recherche d'information....11

I.3.1 Le modèle booléen....12

I.3.2 Le modèle vectoriel....13

I.3.3 Les modèles probabilistes....15

I.3.3.1 Le modèle probabiliste de base....15

I.3.3.2 Le modèle de langue....16

I.4 Evaluation des Systèmes de recherche d'information....18

I.4.1 Mesures d'évaluation....18

I.4.2 Collections de tests....20

I.5 Conclusion....22

Sommaire

Chapitre II : Reformulation de la requête

II.1 Introduction 24

II.2 Techniques de reformulation de requêtes 25

II.2.1 Classifications de techniques de reformulation de requêtes 25

II.2.1.1 Classification selon la source de termes utilisés 25

II.2.1.2 Classification selon la méthode de sélection des termes 25

II.2.1.3 Classification selon le rôle de l'utilisateur 25

II.2.2 Les principales techniques de reformulation de requêtes 27

II.2.2.1 La réinjection de pertinence (Relevance feedback) 27

II.2.2.2 La réinjection de pertinence aveugle (pseudo-feedback) 28

II.3 Etapes de reformulation automatique de la requête 29

II.3.1 Traitement des données 29

II.3.2 Génération et classement des termes candidats d'expansion 30

II.3.2.1 Association un à un 30

II.3.2.2 Association un à plusieurs 33

II.3.3 La sélection des termes 35

II.3.4 La réécriture de la requête 37

II.4 Reformulation de requêtes dans les modèles de RI 38

II.4.1 Reformulation de la requête dans le modèle vectoriel 38

II.4.2 Reformulation de la requête dans le modèle de langue 39

II.5 Conclusion 40

Sommaire

Chapitre III : Expérimentations et évaluation

III.1 Introduction 42

III.2 L'approche implémentée 42

III.2.1 Architecture générale de notre approche 43

III.2.2 Présentation de notre approche 43

III.2.2.1 Le principe de notre approche 44

III.2.2.2 Processus de la reformulation de la requête avec notre approche 44

III.2.2.3 Classes étendues et algorithmes utilisés 45

III.3 L'environnement de travail 48

III.3.1 Présentation de la plate forme Terrier 48

III.3.2 Le langage Java 50

III.3.3 L'environnement de développement (Netbeans) 51

III.4 Résultats et expérimentations 52

III.4.1 Collection de test utilisée 52

III.4.2 Résultats et évaluation 52

III.4.2.1 Résultats obtenus avec la recherche simple 53

III.4.2.2 Résultats obtenus après reformulation (Modèle Bo1) 53

III.4.2.3 Résultats obtenus avec notre approche (Bo1_Etendu) 55

III.4.2.4 Comparaison entre le modèle Bo1 et notre approche (Bo1_Etendu) 56

III.4.2.5 Evaluation requête par requête 58

III.4.2.6 Analyse des résultats basée sur le type de requêtes 60

III.5 Conclusion 64

Conclusion Générale 66

Table de figures

I.1 Architecture générale d'un Système de Recherche d'Information 4

I.2 Les étapes de l'indexation automatique 7

I.3 Exemple de fichier direct 9

I.4 Exemple de fichier inverse 10

I.5 Taxonomie des modèles de recherche d'information 12

I.6 Classification des documents 18

I.7 Allure d'une courbe Rappel et Précision 19

I.8 Exemple d'un document TREC 21

I.9 Exemple d'une requête TREC 21

I.10 Extrait d'un fichier « .qrel » 22

II.1 Les types de reformulation de requêtes selon le rôle de l'utilisateur 26

II.2 Les étapes de la reformulation automatique de la requête 29

III.1 Architecture générale de notre approche 43

III.2 Processus de reformulation de la requête avec notre approche 45

III.3 Architecture de Terrier 49

III.4 L'environnement de développement NetBeans 51

III.5 Comparaison entre les meilleures précisions obtenues dans la reformulation (Modèle Bo1) et la reformulation avec notre approche 57

III.6 Résultats obtenus après évaluation requête par requête (TF_IDF) 59

III.7 Résultats obtenus après évaluation requête par requête (BM25) 60

III.8 Résultats obtenus après évaluation basée sur le type de requêtes (TF_IDF) 63

III.9 Résultats obtenus après évaluation basée sur le type de requêtes (BM25) 63

Liste des tableaux

I.1 Les mesures de similarité utilisées dans le modèle vectoriel 14

III.1 Description de la collection de test utilisée 52

III.2 Résultats obtenus avec la recherche simple 53

III.3 Résultats obtenus dans la reformulation avec le modèle Bo1 (TF_IDF) 54

III.4 Résultats obtenus dans la reformulation modèle Bo1 (BM25) 54

III.5 Résultats obtenus dans la reformulation avec notre approche (TF_IDF) 55

III.6 Résultats obtenus dans la reformulation avec notre approche (BM25) 56

III.7 Taux d'améliorations obtenues avec notre approche 57

III.8 Comparaison entre l'évaluation requête par requête entre la reformulation (Modèle Bo1) et la reformulation avec notre approche (TF_IDF) 58

III.9 Comparaison entre l'évaluation requête par requête entre la reformulation (Modèle Bo1) et la reformulation avec notre approche (BM25) 60

III.10 Taux d'améliorations obtenues entre le modèle Bo1 et notre approche en se basant sur le type de requêtes (TF_IDF) 61

III.11 Taux d'améliorations obtenues entre le modèle Bo1 et notre approche en se basant sur le type de requêtes (BM25) 62

Introduction générale

Aujourd'hui, l'information joue un rôle primordial dans le quotidien des individus et dans l'essor des entreprises. Cependant, le développement de l'Internet et la généralisation de l'informatique dans tous les domaines ont conduit à la production d'un volume d'information sans précédent. En effet, la quantité d'information disponible, particulièrement à travers le web, se mesure en milliards de pages. Il est par conséquent, de plus en plus difficile de localiser précisément ce que l'on recherche dans cette masse d'information. La recherche d'information (RI) est le domaine par excellence qui s'intéresse à répondre à ce type d'attente. En effet, l'objectif principal de la RI est de fournir des modèles, des techniques et des outils pour stocker et organiser des masses d'informations et localiser celles qui seraient pertinentes relativement à un besoin en information d'un utilisateur, souvent, exprimé à travers une requête. Ces outils sont appelés des Systèmes de Recherche d'Information (SRI).

Dans notre projet, nous nous intéresserons à la reformulation de la requête initiale soumise au SRI par l'utilisateur, car celle-ci est souvent courte et n'exprime pas clairement son besoin en information, ce qui à pour conséquence des résultats de recherche non satisfaisants.

La de reformulation de la requête se compose de plusieurs étapes, qui sont le traitement de source de données dans lequel on extrait les termes d'expansion ; le classement et la sélection des termes d'expansion, et enfin la réécriture de la requête.

Dans notre cas nous nous intéressons particulièrement à l'étape de classement des termes d'expansion, qui sont utilisés pour reformuler la requête initiale.

En effet, l'objectif de notre travail est d'introduire un nouveau facteur pour ce classement. Ce facteur est la couverture spatiale d'un terme pour un document, autrement dit : « les positions du terme dans un document ». Ainsi, un terme qui offre une bonne couverture spatiale pour un document a pour effet d'augmenter l'importance de se terme dans ce document.

Le présent mémoire, comporte outre l'introduction, la conclusion, et la bibliographie, les trois chapitres suivants :

- Premier chapitre intitulé « Recherche d'information » : donne une vision globale du domaine de la RI, ainsi que ces concepts fondamentaux à savoir, les étapes principales constituant un système de RI, la notion de pertinence et les principaux modèles de la RI connus à ce jour.
- Deuxième chapitre intitulé « Reformulation de la requête » : offre une vision détaillée de la reformulation de la requête à savoir, les classifications et principales techniques de reformulation de requêtes, les étapes de la reformulation automatique de la requête, et la reformulation de requêtes dans les modèles de RI.
- Troisième chapitre intitulé «Expérimentations et évaluation » : consacré à la présentation, les outils utilisés pour le développement et l'évaluation de notre approche.

Chapitre I. Recherche d'information

I.1 Introduction

La Recherche d'Information (RI) est une branche de l'informatique ayant pour objectif l'acquisition, l'organisation et la sélection de l'information permettant de répondre de façon pertinente aux besoins exprimés par un utilisateur, au travers d'une ou plusieurs requêtes.
La RI peut également être définie comme l'ensemble des méthodes, procédures et techniques permettant, en fonction de critères de recherche propres à l'usager, de sélectionner l'information dans un ou plusieurs fonds de documents **[1]**.

Plusieurs tâches se regroupent sous le vocable de la RI, la plus ancienne est la recherche documentaire, on y trouve également d'autres tâches plus au moins récentes comme : le filtrage d'information, l'extraction de l'information, la recherche d'information multilingue, les questions réponses, la recherche d'information sur le web, etc.

Ce chapitre a pour objectif de présenter le domaine de la RI. Dans la première section, nous allons présenter les concepts de base de la RI à savoir, la définition d'un système de recherche d'information, son architecture ainsi que les notions de document, requête et pertinence. Ensuite, nous décrivons les processus d'indexation, appariement document/requête et reformulation de requêtes. Dans la deuxième section, nous décrivons les modèles de RI à savoir le modèle booléen, le modèle vectoriel et les modèles probabilistes. La dernière section de ce chapitre sera consacrée à l'évaluation des systèmes de recherche d'information.

I.2 Concepts de base de la recherche d'information

Lorsqu'un utilisateur exprime un besoin en information à travers une requête, ce dernier est interprété par un **système de recherche d'information (SRI)** qui se chargera de restituer l'information pertinente satisfaisant l'utilisateur à partir d'une collection de documents.

I.2.1 Système de recherche d'information

Un système de recherche d'information est un système qui permet de mettre en œuvre des techniques et des moyens permettant de retourner des documents pertinents d'une collection en réponse à un besoin en information d'un utilisateur, exprimée par un langage de requêtes qui peut être un langage naturel, une liste de mots clés ou un langage booléen **[2]**.

Afin d'atteindre cet objectif, un processus d'indexation des documents de la collection est effectué. Il permet de construire une représentation synthétique des documents, appelée **index.** Lorsque l'utilisateur formule sa requête, un processus similaire est effectué sur cette dernière. Il consiste à analyser la requête et établir une représentation interne. Puis, le système établi une correspondance entre la représentation de la requête et la représentation des documents (index) pour sélectionner et présenter les

documents qui répondent le mieux au besoin de l'utilisateur (les documents pertinents).
Le SRI s'appuie sur des modèles de RI pour établir cette correspondance entre les documents et la requête.

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. Aussi, il met en œuvre trois processus : le processus d'indexation, le processus d'appariement document/requête et le processus de reformulation de la requête. Ces processus seront présentés d'une manière détaillée dans les sections qui suivent.

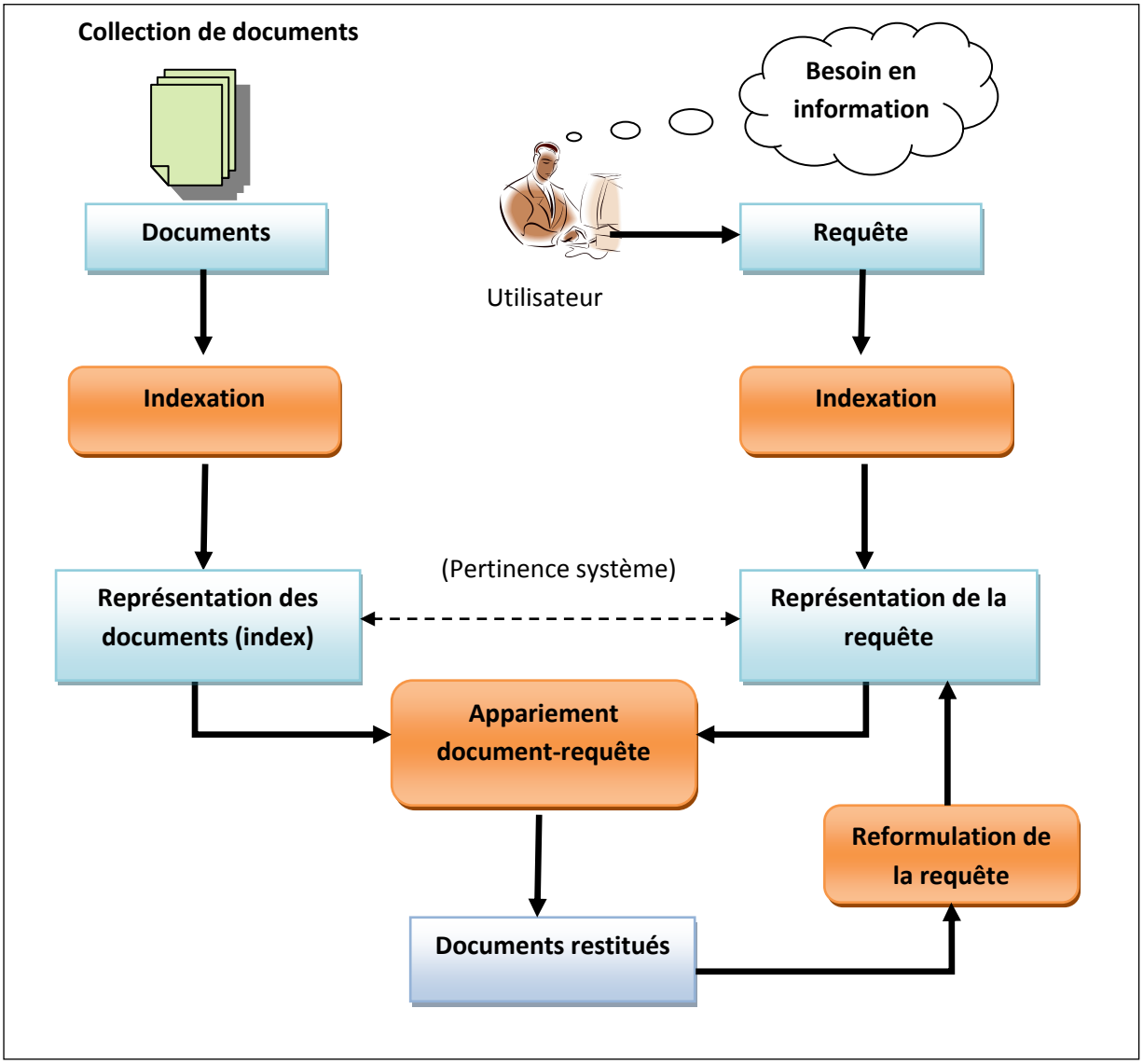

Figure I.1 Architecture générale d'un Système de Recherche d'Information

I.2.1.1 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.

Un document texte peut être représenté selon plusieurs vues **[3]** :

La vue présentation : décrit la représentation d'un document sur un medium à deux dimensions (alignement des paragraphes, entêtes et pieds de pages, etc.).

La vue logique : présente la structure logique d'un document, qui contient des informations sur la structure et la partition du document (structuration de chapitres, sections).

La vue du contenu (sémantique) : se concentre sur le contenu textuel du document, c'est à dire sur l'information elle-même.

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

I.2.1.2 Besoin en information et requête

L'utilisateur est à la fois la source, le déclencheur d'une recherche d'information et le validateur du résultat de cette recherche. On constate pour l'utilisateur de Système de Recherche d'Information un état anormal de connaissances (Anomalous States of Knowlegde) : l'utilisateur déclenche une recherche documentaire lorsqu'il est confronté à un manque dans sa connaissance sur un sujet (besoin en information) **[4]**.

Le besoin de l'utilisateur est l'expression mentale de ce qu'il recherche. Ce besoin est interprété (représenté) au travers d'une **requête**, qui sera ensuite traitée par le SRI. Il s'agit en général d'un ensemble de mots-clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.

Langage naturel (langage libre) : L'utilisateur exprime son besoin de manière naturelle.
Exemple : « chercher les sites de jeux vidéo ».

Langage booléen : Dans se type de langage les requêtes sont exprimées à l'aide de mots clés reliées par des operateurs booléens (ET, OU, NON) et/ou des operateurs linguistiques (PLUS, MOINS, etc.)**.**
Exemple : « recherche ET indexation ».

Langage graphique : La requête est exprimée en utilisant une interface graphique.

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 aucune connaissance à priori des utilisateurs (centre d'intérêt, niveaux, parcours, etc.). Ce biais entre la requête et le besoin en information est l'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.2.1.3 Pertinence

La pertinence constitue l'un des facteurs clés des systèmes de recherche d'information. En effet, elle représente le degré de correspondance entre le contenu sémantique des documents et la requête utilisateur, ce qui a conduit à définir deux types de pertinence: la pertinence système et la pertinence utilisateur **[33]**.

La pertinence Système : correspond à la pertinence que le système a calculée. Elle est souvent présentée par un score attribué par le SRI afin d'évaluer l'adéquation de contenu des documents vis-à-vis de celui de la requête. Ce type de pertinence est objectif et déterministe.

La pertinence utilisateur : correspond à la façon dont l'utilisateur juge les documents retrouvés par le système par rapport au besoin exprimé.

La pertinence utilisateur est subjective, car pour un même document retourné en réponse à une même requête, il peut être jugé différemment par deux utilisateurs distincts (qui ont des centres d'intérêt différents). De plus, cette pertinence est évolutive, un document jugé non pertinent à l'instant *t* pour une requête peut être jugé pertinent à l'instant *t+1,* car la connaissance de l'utilisateur sur le sujet a évolué **[5] [6]**.

I.2.2 Le processus d'indexation

Le SRI intègre un jeu de règles et de notations permettant la traduction de la requête ou du document d'une description brute vers une description structurée. Cette opération est appelée **indexation.**

L'indexation est le processus permettant de créer une représentation des documents et des requêtes facilement manipulable par un système de recherche d'information. Elle consiste à analyser les documents afin d'extraire un ensemble de mots clés servant comme descripteurs des documents **[7]**. L'indexation peut être :

Manuelle : l'extraction et le choix des descripteurs s'effectuent par un documentaliste ou un spécialiste du domaine.

Semi-automatique : l'extraction des descripteurs s'effectue par le système mais le choix final de ces derniers est laissé au spécialiste, qui intervient souvent pour choisir d'autres termes significatifs.

Automatique : l'extraction et le choix des descripteurs s'effectuent d'une façon totalement automatisée.

L'indexation manuelle permet d'assurer une meilleure pertinence dans les réponses apportées par le SRI. Elle présente toutefois plusieurs inconvénients : deux indexeurs différents peuvent présenter des termes différents pour caractériser un même document, et un indexeur à deux moments différents peut présenter deux termes distincts pour représenter le même concept. De plus, le temps nécessaire à sa réalisation est très important.

Dans le cas d'une indexation semi-automatique **[8] [9]**, 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.

Enfin, l'indexation automatique **[10]**, que nous décrivons en détail dans ce qui suit, regroupe un ensemble de traitements automatisés sur un document. On distingue : l'extraction automatique des mots des documents (Analyse lexical), l'élimination des mots vides, la normalisation (radicalisation ou lemmatisation), la pondération des termes et enfin la création de l'index. Ces étapes sont schématisées dans la figure suivante :

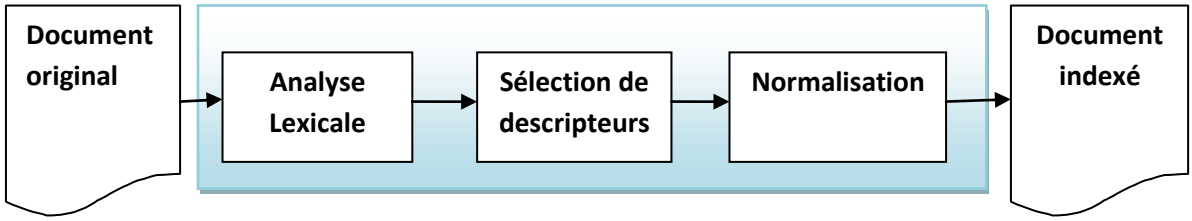

Figure I.2 Les étapes de l'indexation automatique

I.2.2.1 L'analyse lexicale

Elle est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est un groupe de caractères constituant un mot significatif **[10]**. Un terme peut être un mot simple (exemple : pomme) ou un mot composé (exemple : pomme de terre) mais en RI on utilise souvent les mots simples.
L'analyse lexicale permet de reconnaitre les espaces de séparation des mots, des chiffres, les ponctuations, etc.

I.2.2.2 L'élimination des mots vides (Sélection de descripteurs)

Les mots vides sont des mots peu significatifs et porteurs de peu de sens, augmentant ainsi la taille de l'index et rendant la recherche plus lente. De ce fait, leur élimination est impérative.

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

A) L'utilisation d'une liste de mots vides (aussi appelée anti-dictionnaire),
B) L'élimination des mots dépassant un certain nombre d'occurrences dans la collection.

Même si l'élimination des mots vides a l'avantage évident de réduire le nombre de termes d'indexation, elle peut cependant réduire le taux de rappel (décrit dans la section I.4.1).

I.2.2.3 La normalisation (Lemmatisation)

Un mot donné peut avoir différentes formes dans un texte, mais leur sens reste le même ou très similaire. On peut par exemple citer les mots : économie, économiquement, économétrie, etc. Il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept véhiculé.

Cependant, on utilise une opération nommé **normalisation** (ou lemmatisation), qui consiste à représenter les différentes variantes d'un terme par un format unique appelé lemme ou racine, et cela pour but de réduire la taille de l'index.

On distingue plusieurs types stratégiques de lemmatisation **[11]** :

-La table de consultation (dictionnaire).
-L'élimination des affixes (exemple : l'algorithme de Porter **[12]**).
-La troncature.
-Les variétés de successeurs.
-La méthode des n-gramme **[13]**.

L'étape de normalisation permet à l'utilisateur d'éviter de devoir faire introduire les différentes formes d'un mot (les formes de pluriel, singulier, les formes conjuguées des verbes, etc.) lors de sa recherche. Exemple : les mots « camions » et « camion » sont indexés de la même façon (« camion »).

Autre exemple, la forme conjuguée « portera » du verbe « porter » sera indexée sous la forme « **porte** », de la même façon que le mot « portes ». Ainsi, lorsque l'utilisateur formulera une requête avec le verbe « porter », il aura très certainement, parmi la liste des documents résultats, des documents non pertinents relatifs au nom « porte ».

La normalisation permet d'augmenter le taux de rappel (décrit dans la section I.4.1). En revanche, elle diminue en pratique le taux de précision. Ceci est dû à la perte de la sémantique originale du terme lors du passage à la forme finale (canonique), c'est le cas par exemple des termes active/activate normalisé par l'algorithme de porter **[12]**.

I.2.2.4 Construction de l'index

Au terme de toutes les différentes étapes d'indexation, il devient nécessaire de créer et de stocker les informations sélectionnées. Les termes sont stockés dans les structures de données suivantes :

- **Fichier direct** : C'est le fichier de base dans lequel les données sont stockées. Il donne comme information pour chaque document, les mots et les fréquences qui le constituent.

Document	Contenu
d_1	La (1) recherche (4) d' (14) information (16) gère (28) des (33) texte. (37)
d_2	Un (1) système (4) de (12) recherche (15) d' (25) information (27) doit (39) restituer (44) l' (54) information (56) Pertinente (68) à (79) l' (81) utilisateur (83)
d_3	Une (1) information (5) est (17) pertinente (21) si (32) elle (35) satisfait (40) l' (50) utilisateur (52)

Figure I.3 Exemple de fichier direct

- **Fichier inverse** : il est créé autour du fichier direct **[14] [15]**. Ce fichier comme son nom l'indique, est le résultat de l'inversion du fichier direct. Plus exactement, au lieu de donner pour chaque document les mots et les fréquences qui le constituent, on donne pour chaque mot les documents qui le contiennent et sa fréquence dans chacun d'eux. Ainsi l'inversion du fichier donné en figure I.3 donne le fichier illustré dans la figure suivante :

terme	d_1	d_2	d_3
recherche	4	15	3
information	16	27,56	5
gère	28		
textes	37		
système		4	
restituer		44	
pertinente		68	21
utilisateur		83	52
satisfait			40
….			

Figure I.4 Exemple de fichier inverse

Généralement d'autres fichiers index peuvent être crées comme l'index document, qui contient des informations sur le document (taille, etc.) et le fichier lexique qui contient des informations sur les termes de la collection (exemple : fréquence globale du terme).

I.2.3 Appariement document/requête

La comparaison entre le document et la requête permet de calculer une mesure appelée pertinence système, supposée représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction de similarité vectorielle notée RSV (Q, D) (Retrieval Status Value), où **Q** est une requête et **D** un document. Cette mesure tient compte du poids des termes dans les documents. D'une façon générale, l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'information. L'ordre dans lequel les documents susceptibles de répondre à la requête sont retournés est important. En effet, l'utilisateur se contente généralement d'examiner les premiers documents renvoyés (10 ou 20 premiers documents) **[16]**. Si les documents recherchés ne sont pas présents dans cette tranche, l'utilisateur considérera le SRI comme mauvais vis-à-vis de sa requête. La fonction de similarité permet d'ordonner les documents renvoyés à l'utilisateur par ordre de pertinence.
Le but de tout SRI est donc évidemment de rapprocher la pertinence système de la pertinence utilisateur**.**

I.2.4 La reformulation de la requête

Dans les SRI, la requête initiale seule est souvent insuffisante pour permettre la sélection de documents répondant au besoin de l'utilisateur. De ce fait, plusieurs techniques de **reformulation de requêtes** 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?

La reformulation de la requête est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur. Elle est souvent opérée par ajout et/ou réévaluation des poids des termes de la requête initiale, qui est traitée comme un essai (naïf) pour retrouver de l'information.

Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'espoir de retrouver plus de documents pertinents. La reformulation de la requête se fait en deux étapes : trouver des termes d'extension à la requête initiale, et ajouter les termes dans la nouvelle requête.

Le but de cette technique est principalement l'amélioration de la qualité des résultats de la recherche et de retrouver plus précisément les bons éléments dans les documents.

Le chapitre II est entièrement consacré à la reformulation de la requête, car elle constitue le contexte de notre approche.

I.3 Les modèles de recherche d'information

Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence **[17]**. On distingue trois principaux modèles :

- Les modèles ensemblistes
- Les modèles algébriques
- Les modèles probabilistes

Les modèles ensemblistes reposent sur la théorie des ensembles. Dans ces modèles, les termes de la requête sont séparés par des opérateurs logiques : conjonction (ET), disjonction (OU) et négation (NON). Ces opérateurs permettent d'effectuer des opérations d'union « OU », d'intersection « ET » et de différence « NON » entre les ensembles de résultats associés à chaque terme.

Les modèles algébriques se basent sur la théorie algébrique. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance (ou similarité) dans un espace vectoriel.

Enfin, les modèles probabilistes se basent sur la théorie des probabilités. Pour ces modèles, la pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

La figure I.5 présente une taxonomie des principaux modèles de RI.

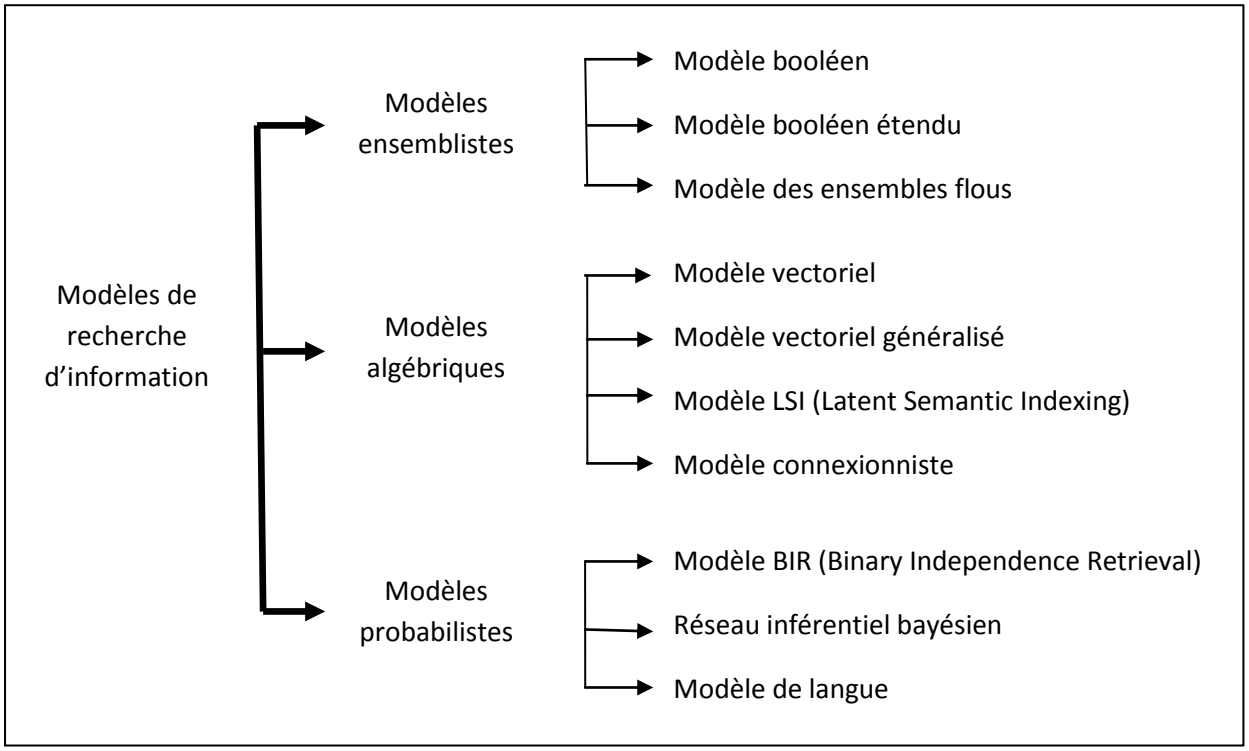

Figure I.5 Taxonomie des modèles de recherche d'information [7]

Dans la suite de cette section, nous nous proposons de décrire tout d'abord les trois modèles les plus représentatifs de chacun de ces courants (à savoir le modèle booléen, le modèle vectoriel et le modèle probabiliste).

I.3.1 Le modèle booléen

Le modèle booléen est le premier modèle qui s'est imposé dans le monde de la recherche d'information. Il se base sur la manipulation des ensembles et l'algèbre de Boole. Dans ce modèle une requête est une expression logique composée de termes séparés par des opérateurs logiques (ET, OU et NON). Les poids des termes dans l'index sont binaires, c'est-à-dire que les termes sont présents ou absents du document ($\mathbf{W}_{ij} \in \{0,1\}$). Le modèle booléen utilise l'appariement exact, c'est-à-dire qu'il ne permet de restituer que les documents appartenant à l'ensemble décrit par la requête. La similarité entre un document et une requête est définie par :

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

Le modèle booléen affirme que chaque document est soit pertinent soit non-pertinent. Il n'y a pas de notion de réponse partielle aux conditions de la requête. Par exemple, considérons un document contenant les trois termes « recherche, information et pertinence ». Ce document ne sera pas pertinent pour la requête « recherche ET information ET pertinence ET modèle ».

Ainsi, Son principal avantage est sa transparence. En effet, pour l'utilisateur, la raison pour laquelle un document a été sélectionné par le système est claire : il répond exactement à la requête qui a été formulée.

Cependant, il est parfois difficile pour l'utilisateur d'exprimer son besoin en information avec des expressions booléennes qui sont généralement très simples, ce qui ne permet pas d'utiliser au mieux les caractéristiques du modèle. De plus, le fait que la pertinence soit basée sur un critère binaire sans notion d'´echelle de gradualité empêche le modèle d'avoir de bonnes performances. Enfin, les résultats de la fonction de similarité (1 ou 0) ne permettent pas de fournir à l'utilisateur une liste ordonnée de résultats.

Aujourd'hui, il est connu qu'une pondération non binaire des termes de l'index peut amener à des améliorations notables des performances. La pondération de ces termes nous amène donc à introduire le modèle vectoriel.

I.3.2 Le modèle véctoriel

Le modèle vectoriel est l'un des modèles qui utilisent l'approche statistique. Il consiste à représenter les documents et les requêtes sous forme de vecteurs de termes pondérés. L'idée de base du modèle vectoriel est d'utiliser une représentation géométrique pour classer les documents par ordre de pertinence par rapport à une requête, c'est-à-dire que les documents et les requêtes sont représentés sous forme de vecteurs dans l'espace vectoriel engendré par les termes extraits de tous les documents de la collection. Cette idée a été développée par Gérard Salton et son équipe **[18]** dans leur projet SMART. Il a proposé un modèle basé sur la mesure de similarité par le produit scalaire.

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 ces schémas prennent en compte la pondération locale et la pondération globale [**19**].

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 :

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

Où $f(t_i\,,\,d_j)$ est la fréquence du terme t_i dans le document d_j .
Quant à la pondération globale, elle prend en compte les informations concernant le terme dans la collection. Un poids plus important doit être assigné aux termes qui apparaissent moins fréquemment dans la collection. Car les termes qui apparaissent dans de nombreux documents de la collection ne permettent pas de distinguer les documents pertinents des documents non pertinents (i.e. peu utile pour la

discrimination). Un facteur de pondération globale est alors introduit. Ce facteur nommé ***idf*** (*i*nverted *d*ocument *f*requency), dépend d'une manière inverse de la fréquence en document du terme et exprimé comme suit :

$$Idf_{(ti)} = log\left(\frac{n}{n_i}\right) \quad \text{(I.3)}$$

Où n_i est la fréquence en document du terme ti considéré, et **N** est le nombre total de documents dans la collection.
Les fonctions de pondération combinant la pondération locale et globale sont référencées sous le nom de la mesure (***TF_IDF)***. Cette mesure donne une bonne approximation de l'importance du terme dans les collections de documents de taille homogène. Cependant, un facteur important est ignoré, la taille du document. En effet, la mesure (***TF_IDF)*** ainsi définie favorise les documents longs, car ils ont tendance à répéter le même terme, ce qui accroit leur fréquence, par conséquent augmentent la similarité de ces documents vis-à-vis de la requête. Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération, comme facteur de normalisation **[20] [21].**

Contrairement au modèle booléen où les termes de la requête doivent être reliés par des connecteurs logiques, le modèle vectoriel permet à l'utilisateur d'exprimer son besoin en information sous forme d'une liste de mots clés ou en langage naturel.

L'appariement document-requête dans le modèle vectoriel, consiste à trouver les vecteurs documents qui s'approchent le plus de vecteur de la requête. Cet appariement est obtenu par l'évaluation de la distance entre les deux vecteurs. Plusieurs mesures de similarité ont été définies **[22],** dont les plus courantes sont décrites dans le tableau I.1 ci-dessous.

Mesures	***Formules***
Le produit scalaire	$\mathrm{RSV}(\mathrm{d_j}, \mathrm{q_k}) = \sum_{i=1}^{t} (\mathrm{w_{i,j}} * \mathrm{w_{i,k}})$
Mesure de Cosinus	$\mathrm{RSV}(\mathrm{d_j}, \mathrm{q_k}) = \frac{\sum_{\mathrm{i=1}}^{\mathrm{N}} (\mathrm{wd_{ij}} * \mathrm{wq_{ik}})}{\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{wd^2_{iJ}} * \sum_{\mathrm{ij=1}}^{\mathrm{N}} \mathrm{wq^2_{ik}}}}$
Mesure de Jaccard	$\mathrm{RSV}(\mathrm{d_j}, \mathrm{q_k}) = \frac{\sum_{i=1}^{N} (wd_{ij} * wq_{ik})}{\sum_{i=1}^{N} wd^2_{\mathrm{ij}} + \sum_{i=1}^{N} \mathrm{wq^2_{ik}} - \sum_{i=1}^{N} (wd_{ij} * wq_{ik})}$
Mesure de Dice	$\mathrm{RSV}(\mathrm{d_j}, \mathrm{q_k}) = 2 * \frac{\sum_{\mathrm{i=1}}^{\mathrm{N}} (\mathrm{wd_{ij}} * \mathrm{wq_{ik}})}{\sum_{\mathrm{i=1}}^{\mathrm{N}} (wd^2_{ij} + wq^2_{ik})}$

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

Les avantages du modèle vectoriel sont nombreux : il permet la pondération des termes, ce qui augmente les performances du système; il permet de renvoyer des documents qui répondent approximativement à la requête de trier les documents répondant à une requête. Les documents sont en effet restitués dans un ordre décroissant de leur degré de similarité avec la requête. Plus le degré de similarité d'un document est élevé, plus le document ressemble à la requête et plus il est susceptible d'être pertinent pour l'utilisateur.

Théoriquement, le modèle vectoriel a l'inconvénient de considérer que les termes de l'index sont tous indépendants. Cependant en pratique, la prise en compte globale de la dépendance des termes peut faire baisser la qualité des réponses d'un système (puisque les dépendances sont généralement locales). Cependant, Wong et al **[23]** ont proposé un modèle vectoriel généralisé *(Generalized Vector Space Model)* qui lève l'hypothèse d'indépendance entre termes.

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

I.3.3 Les modèles probabilistes

I.3.3.1 Le modèle probabiliste de base

Le modèle probabiliste aborde le problème de la recherche d'information dans un cadre probabiliste. Le premier modèle probabiliste a été proposé par Maron et Kuhns **[24]** au début des années 1960. Le principe de base consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête.

En utilisant le théorème de Bayes **[25]** les probabilités de pertinence ($P(R \mid \mathrm{d})$) et de non pertinence ($P(N\,R \mid \mathrm{d})$) d'un document sont calculées par :

$$P(\mathrm{R}|\mathrm{d}) = \frac{\mathrm{p}(\mathrm{d}|\mathrm{R}) * \mathrm{p}(\mathrm{R})}{p(d)} \tag{I.4}$$

$$P(\mathrm{NR}|\mathrm{d}) = \frac{\mathrm{p}(\mathrm{d}|\mathrm{NR}) * \mathrm{p}(\mathrm{NR})}{\mathrm{p}(\mathrm{d})} \tag{I.5}$$

Où :

– P (d | R) est la probabilité que d fasse partie de l'ensemble de documents pertinents.
– P (d | NR) est la probabilité que d fasse partie de l'ensemble de documents non pertinents.
– $P(R)$ est la probabilité de pertinence d'un document quelconque du corpus.
– $P(NR)$ est la probabilité de non pertinence d'un document quelconque du corpus.
– P(d) est la probabilité que le document d soit choisi.

La fonction de correspondance évalue la pertinence d'un document par rapport à la requête :

$$P = \frac{\mathrm{P(R|d)}}{\mathrm{P(NR|d)}} \tag{I.6}$$

$$\approx \frac{\mathrm{P(d|R)}}{\mathrm{P(d|NR)}}$$

Si on considère P(R) et P(NR) sont des constantes.

Cela nous ramène à l'estimation de P (d | R) et P (d | NR). Si on considère que les termes sont indépendants :

$$P(d|R) = \prod_{t_i \in d} P(t_i|R) \tag{I.7}$$

$$P(d|R) = \prod_{t_i \in d} P(t_i|NR) \tag{I.8}$$

Avec $P(t_i | R)$ (respectivement $P(t_i | NR)$ la probabilité de l'apparition du terme t_i dans un document pertinent (respectivement non pertinent).

Pour calculer ces probabilités, on peut procéder par échantillonnage, en choisissant deux ensembles de documents pertinents ou non pertinents. Ils peuvent aussi être calculés pour chaque terme en utilisant la réinjection de pertinence. Les poids sont modifiés par rapport à la distribution du terme dans les documents pertinents ou non pertinents. Dans la pratique, la plupart des expérimentations se fondent sur des méthodes de type TF_IDF.

Parmi les systèmes fondés sur le modèle probabiliste, nous pouvons citer le moteur de recherche OKAPI **[26] [27].**

I.3.3.2 Le modèle de langue

Les modèles de langage sont des modèles probabilistes. Ils désignent une fonction de probabilité qui assigne à chaque séquence de mots une probabilité. Leur objectif est de capter les régularités linguistiques d'une langue, en observant la distribution des mots, successions de mots, dans une langue donnée **[7]**.

L'hypothèse de base dans ces modèles est qu'un document n'est pertinent que s'il ressemble à la requête. Les modèles de langage sont basés sur une hypothèse différente : un utilisateur en interaction avec un système de recherche fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver. La requête est alors inférée par l'utilisateur à partir de ces documents. Un document n'est pertinent que si la requête utilisateur ressemble à celle inférée par le document. On cherche alors à estimer la probabilité que la requête soit inférée par le document **[28]**.

Les modèles de langages calculent cette probabilité et l'utilisent pour ordonner les documents. Etant donné une requête $T_1, T_2, \ldots, T_n$, les documents sont ordonnés selon la mesure suivante :

$$P(T_1, T_2, \ldots\ldots\ldots T_n|D) = \prod_{i=1}^{n} (1 - \boldsymbol{\lambda_i})\, P(T_i) + \boldsymbol{\lambda i}\, P(T_i|D) \quad \text{(I.9)}$$

Cette mesure est une combinaison linéaire du modèle de document $\mathbf{P(T_i|D)}$ et du modèle de contexte du document (la collection) $\mathbf{P(T_i)}$, où : $\boldsymbol{\lambda_i}$ est la probabilité que le terme à la position i soit important, $\mathbf{1 - \lambda_i}$ est la probabilité que le terme ne soit pas important, $P(T_i|D)$ et $P(T_i)$ sont exprimé de la façon suivante :

$$P(T_i|D) = \frac{tf\,(T_i|D)}{\sum_T tf\,(T,D)} \quad \text{, Terme important} \quad \text{(I.10)}$$

$$P(T_i) = \frac{df\,(T_i)}{\sum_T df\,(T)} \quad \text{, Terme sans importance} \quad \text{(I.11)}$$

Où $tf\,(T_i|D)$ est la fréquence du terme T_i dans le document D et $df\,(T)$ est le nombre de documents dans lesquels le terme T apparaît.

I.4 Evaluation des Systèmes de recherche d'information

L'évaluation d'un SRI constitue une étape importante dans l'élaboration d'un modèle de RI, puisqu'elle permet de paramétrer le modèle, et de fournir des éléments de comparaison entre modèles.

Des SRI basés sur des modèles différents donnent des résultats différents **[29]**. Pour mesurer les performances qualitatives des SRI, on procède à la comparaison de combinaisons des ensembles de documents pertinents, de document non pertinents, de document retournés et des documents non retournés sur l'ensemble des requêtes. Il existe à cet effet de nombreuses mesures, chacune mettant en évidence telle ou telle propriété du système. La figure I.6 illustre la classification des documents en réponse a une requête.

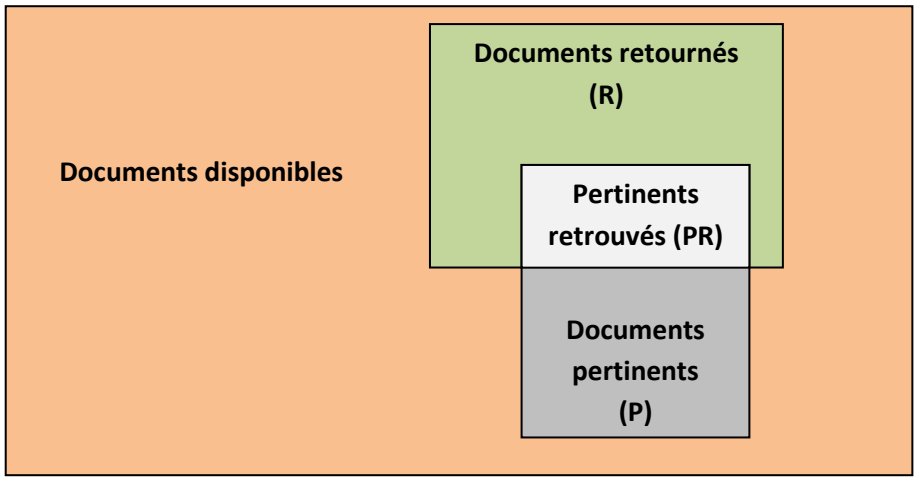

Figure I.6 Classification des documents

I.4.1 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 **[30]**. 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 :

$$\text{Précision} = PR/R$$

- **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 retrouvés parmi tous les documents pertinents :

$$\text{Rappel} = PR/P$$

- **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 précision-rappel qui a en général l'aspect suivant :

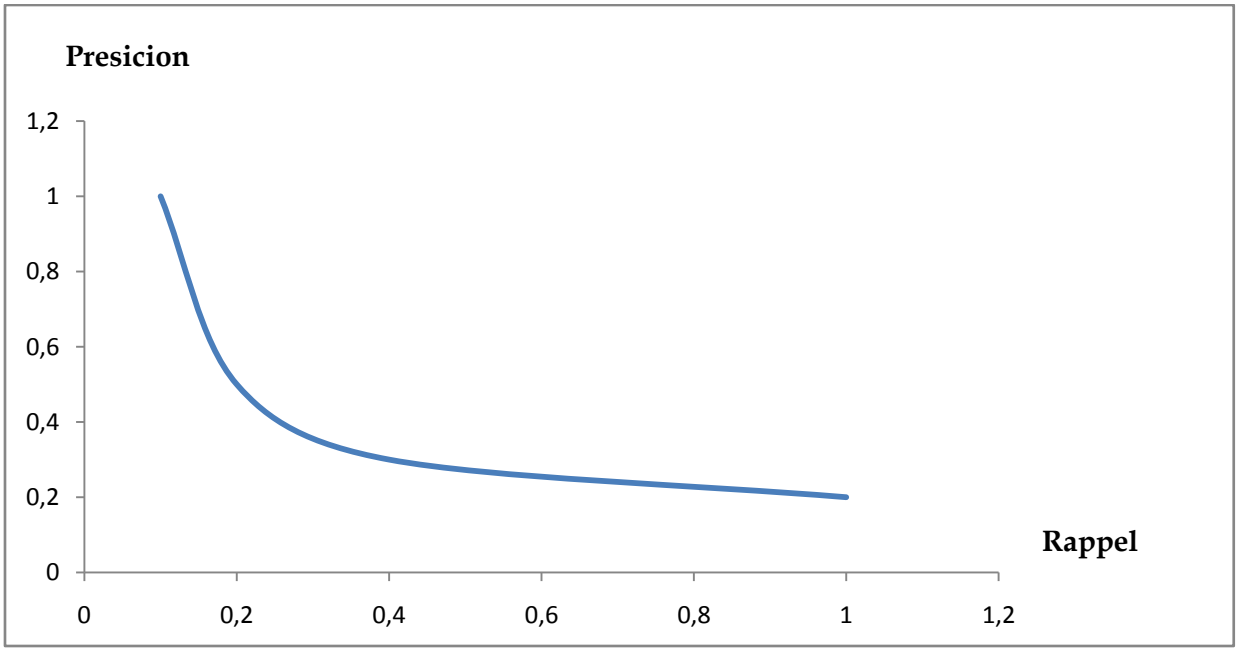

Figure I.7 Allure d'une courbe Rappel et Précision

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 = R (non pertinents) /R

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

Silence = P (non retournés) /P

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

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

Avec :

$$\Pr(d_i) = \begin{cases} \frac{r_{n_i}}{n_i} & \text{si } d_{ij} \text{ est retrouvé} \\ 0 & \text{sinon} \end{cases} \quad \text{(I.13)}$$

Où n_i dénote le rang du document d_i qui a été retrouvé et qui est pertinent pour la requête, r_{n_i} 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 :

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

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

I.4.2 Collections 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 les collections TREC.

TREC (Text REtrieval Conference) est un projet international qui a été lancé en 1992 par NIST (National Institute of Standards and Technology) aux Etats-Unis. Il est aujourd'hui co-sponsorisé par le NIST et DARPA/ITO (Defence Advenced Research Projects Agency/Information Technology Office) **[31]**.

Il offre de très larges collections de documents de source très variées, qui évoluent d'année en année. Pour chaque session de TREC, un ensemble de documents et de requêtes est fourni **[32]**.

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 1.8 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.8 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.9.

```
<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.9 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.10.

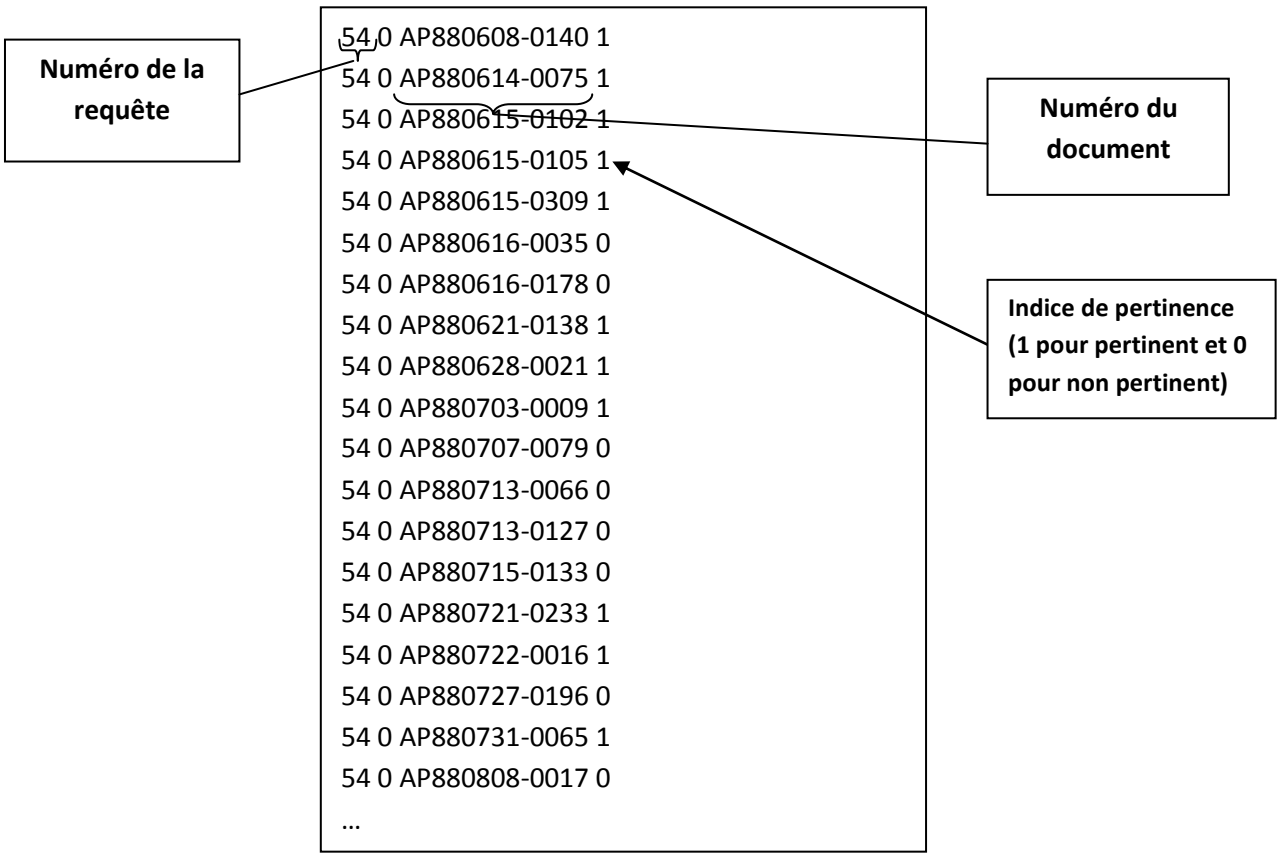

Figure I.10 Extrait d'un fichier « .qrel »

I.5 Conclusion

Dans ce premier chapitre, nous avons présenté les concepts de base de la recherche d'information. Nous nous sommes intéressés entre autre à la définition et à l'architecture générale d'un SRI, les techniques d'indexation utilisées en RI, appariement document requête, La reformulation de la requête, les principaux modèles de recherche, et pour finir, nous avons donné quelques mesures et collections pour l'évaluation des SRI.

Dans le chapitre suivant, nous présentons en détail une phase importante dans le processus de la recherche d'information qui est la reformulation de la requête, car notre travail se situe au niveau de cette étape.

Chapitre II. Reformulation de la requête

II.1 Introduction

Le processus de recherche d'information est incertain. Les utilisateurs peuvent ne pas pouvoir exprimer un besoin en information en termes de requête appropriée. Ils pourraient identifier l'information utile quand elle leurs sera présentée. Les résultats que lui fournit le système de recherche d'information ne lui conviennent parfois pas, retrouver des informations pertinentes en utilisant la requête initiale de l'utilisateur est difficile, Et cela à cause de l'imprécision de la requête.

En effet, Le plus souvent, l'utilisateur formule ses requêtes avec des termes qui lui sont propres, mais qui ne correspondent pas forcément à ceux utilisées pour indexer les documents de collections interrogées. Pour sélectionner le maximum de documents pertinents, il faudrait alors que l'utilisateur puisse choisir les termes utilisés comme index. Cette tâche s'avère difficile dans la mesure où il est impossible de connaître le langage d'indexation utilisé et où le nombre de termes indexés est généralement très grand. De plus, l'indexation et en particulier son exhaustivité, a également une incidence directe sur la qualité des réponses du système de recherche. De ce fait, retrouver les informations pertinentes en utilisant seulement la requête initiale de l'utilisateur est une opération quasi-impossible **[34]**.

La reformulation de requêtes, comme nous l'avons signalé dans le chapitre précédent, est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur. Cette reformulation permet de coordonner le langage de recherche, utilisé par l'utilisateur dans sa requête et le langage d'indexation. Par conséquent, elle limite le bruit et le silence dus à un mauvais choix des termes d'indexation dans l'expression de la requête d'une part, et les lacunes du processus d'indexation d'autre part.

Dans la première partie de ce chapitre, nous allons voir les principales techniques utilisées pour reformuler une requête. Dans la deuxième partie, nous décrivons les étapes de la reformulation automatique de la requête à savoir le traitement des données, la génération et le classement des termes, la sélection des termes, et la réécriture de la requête. Et enfin, nous allons décrire la reformulation de requêtes dans quelques modèles de RI.

II.2 Techniques de reformulation de requêtes

II.2.1 Classifications des techniques de reformulation de requêtes

Les techniques de reformulation de la requête peuvent être classées en tenant compte de trois principaux paramètres **[35]** :

II.2.1.1 Classification selon la source des termes utilisés

Différentes sources de données sont utilisées pour reformuler la requête initiale. Elles peuvent être **[36]** : (1) des ressources externes**,** telles que les ontologies, les thésaurus et la relation de cooccurrence entre termes dans la collection. Les méthodes basées sur ces ressources sont dites **méthodes globales**. (2) Les documents résultats de la première recherche ; les méthodes basées sur ces sources sont dites **méthodes locales**. Ces méthodes sont également connues sous le nom de réinjection de pertinence. La réinjection de pertinence a montré son efficacité avec différents modèles de la RI et a affiché de meilleures résultats que les méthodes globales **[II.55] [II.51][III.5]**.

II.2.1.2 Classification selon la méthode de sélection des termes

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

On trouvera plus de détail sur les méthodes de sélection des termes dans la section II.3.3.

II.2.1.3 Classification selon le rôle de l'utilisateur

La reformulation de la requête peut être réalisée par l'utilisateur (dite manuelle), ou par le système (dite automatique) comme elle peut être réalisée conjointement par l'utilisateur et le système, dans ce cas elle est dite semi-automatique ou interactive **[7]**.

Cependant, Le rôle de l'utilisateur dans ce processus peut être actif ou passif, il est actif dans la reformulation manuelle et reformulation interactive de la requête, et passif dans la reformulation automatique de la requête.

La figure II.1 nous montre les différents types de reformulation de requêtes selon le rôle l'utilisateur.

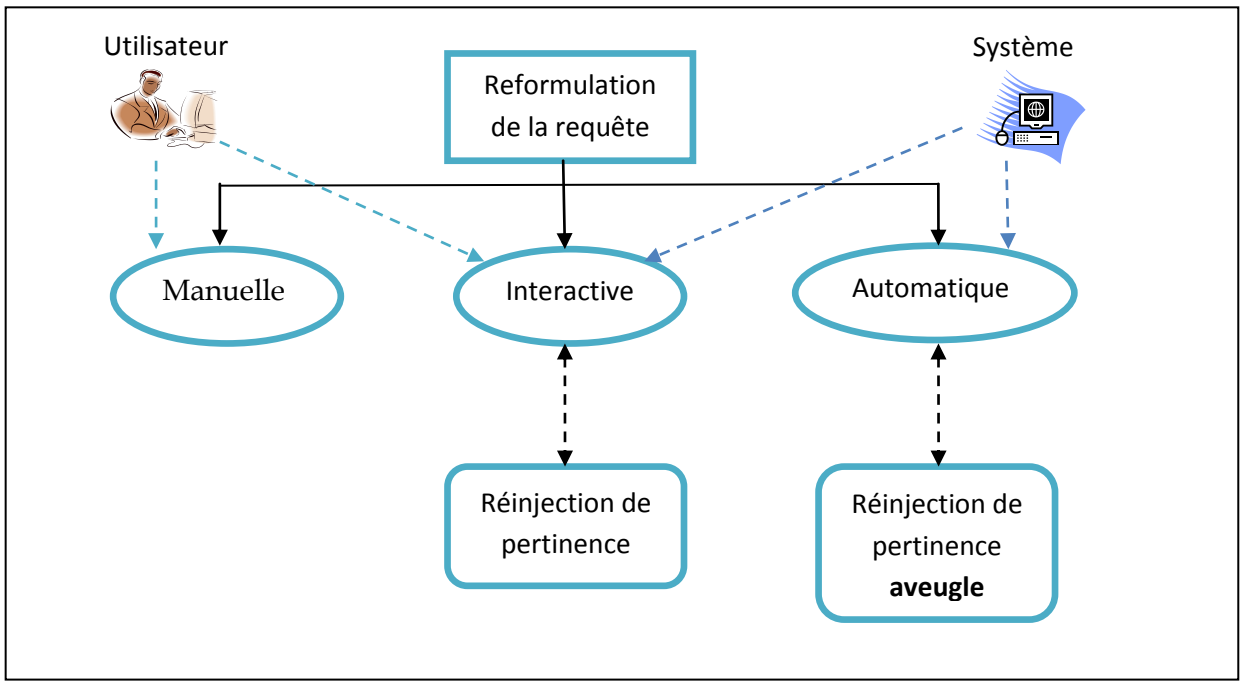

Figure II.1 Les types de reformulation de requêtes selon le rôle de l'utilisateur

A) La reformulation Manuelle

Ce type de reformulation est associé aux systèmes de recherche booléens. On peut procéder à la reformulation de requête en utilisant un vocabulaire contrôlé (thésaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes et ainsi compléter sa requête.

B) La reformulation automatique

Dans ce type de reformulation, l'utilisateur n'intervient pas. La reformulation de la requête peut être effectuée à partir d'un thesaurus, qui définit les relations entre les différents termes de l'index et permet de sélectionner de nouveaux termes à ajouter à la requête initiale. Le thesaurus regroupe plusieurs informations de type linguistique (équivalence, association, hiérarchie) et statistique (pondération des termes). La construction du thesaurus se fait généralement pendant le processus d'indexation, et peut être automatique ou interactive. Parmi les thesaurus construits automatiquement, on peut citer un thesaurus basée sur les similarités **[37]**, un thesaurus statistique **[38]**, ou bien des mini-thesaurus construits seulement d'après la requête et à partir de techniques de clustering **[39]**.

Dans ce cadre de reformulation, on peut citer également la **réinjection de pertinence automatique** : c'est aussi ce qu'on appelle la **réinjection de pertinence aveugle**. Dans ce cas, on applique le même principe de la réinjection de pertinence mais en considérant les ***n*** premiers documents renvoyés par le système comme pertinents **[40]**, **[41]**.

On trouvera plus de détails sur la reformulation automatique de la requête dans la section II.3, et la réinjection de pertinence aveugle sera discutée dans la section II.2.2.

C) La reformulation semi-automatique

La reformulation interactive de la requête est la stratégie de reformulation de la requête la plus populaire **[42] [43]**. On la nomme communément **réinjection de pertinence** ou « relevance feedback » en anglais. Dans un cycle de réinjection de pertinence on présente à l'utilisateur une liste de documents jugés pertinents par le système comme réponse à la requête initiale. Apres les avoir examinés, l'utilisateur indique ceux qu'il considère pertinents. L'idée principale de la réinjection de pertinence est de sélectionner les termes importants appartenant aux documents jugés pertinents par l'utilisateur, et de renforcer l'importance de ces termes dans la nouvelle formulation de la requête.

Cette méthode a pour double avantage une simplicité d'exécution pour l'utilisateur qui s'occupe pas des détails de la reformulation, et un meilleur contrôle du processus de recherche en augmentant le poids des termes importants et en diminuant celui des termes non importants.

On trouvera plus de détails sur la réinjection de pertinence dans la section II.2.2.

II.2.2 Les principales techniques de reformulation de requêtes

Il existe plusieurs techniques de reformulation de requêtes **[37]**, Parmi les plus populaires : on cite la technique nommé « réinjection de pertinence ».

La réinjection de pertinence peut être réalisé d'une manière semi-automatique, c'est-à-dire avec l'intervention du système et de l'utilisateur, dans ce cas on parle de la « relevence feedback ». Comme elle peut être réalisée d'une manière automatique sans l'intervention de l'utilisateur, dans ce cas on la nomme « pseudo-feedback » ou réinjection de pertinence aveugle.

Nous présentons dans ce qui suit ces deux techniques.

II.2.2.1 La Réinjection de pertinence (Relevance feedback)

La réinjection de pertinence (**relevance feedback**) est l'une des techniques de modification de requêtes les plus utilisée dans le domaine de la recherche d'information.

Cette méthode est mise en pratique quand l'utilisateur doit améliorer la requête qu'il a formulée au système de recherche d'information parce que les documents trouvés à l'étape initiale de la recherche ne répondent pas de manière pertinente aux besoins en information de l'utilisateur.

La technique fonctionne comme suit:
- l'utilisateur soumet une requête au SRI, qui produit une liste ordonnée des documents selon leurs degrés correspondants de pertinence à la requête.

- l'utilisateur examine cette liste triée et détermine quels sont les documents pertinents et non pertinents.

- avec cette information, le SRI modifie la requête initiale, en donnant plus d'importance aux termes apparaissant dans les documents pertinents, et affaiblissant la force de ceux qui appartiennent aux non pertinents.

- ce processus est répété jusqu'à ce que l'utilisateur soit complètement satisfait de l'ensemble des documents trouvés.

II.2.2.2 La réinjection de pertinence aveugle (pseudo-feedback)

La technique de réinjection de pertinence, décrite précédemment dépend des jugements de l'utilisateur sur la liste des documents trouvés. Une autre approche alternative, appelée **pseudo-réinjection de pertinence** ou (**blind)** emploie la technique de réinjection de pertinence automatiquement sans utiliser l'information issue des jugements de l'utilisateur.

Dans cette technique le système génère une liste triée de documents pour la première requête initiale. Ensuite, il sélectionne un certain nombre de documents (petit ensemble) à partir des tops documents du classement. Une nouvelle itération RF sera exécutée en supposant que les documents sélectionnés sont pertinents.
La nouvelle requête générée par le processus de reformulation de requêtes est utilisée pour produire une nouvelle liste triée de documents.

Le principe de base du **pseudo RF** est que : « une itération feedback basée sur les premiers documents sélectionnés pendant l'exécution de la requête initiale, va engendrer un meilleur classement pour les documents pertinents ». Selon Croft & Harper **[40]**, cette technique souffre d'un problème majeur qui est appelé: **« query drift »**. Ce problème apparaît lorsque l'ensemble des documents utilisés pour le feedback ne contient que peu de documents pertinents (ou ne contient pas du tout). Donc, cette technique ne fonctionne bien que dans le cas où la requête initiale permet d'extraire de bons résultats (beaucoup de documents pertinents).

II.3 Etapes de reformulation automatique de la requête

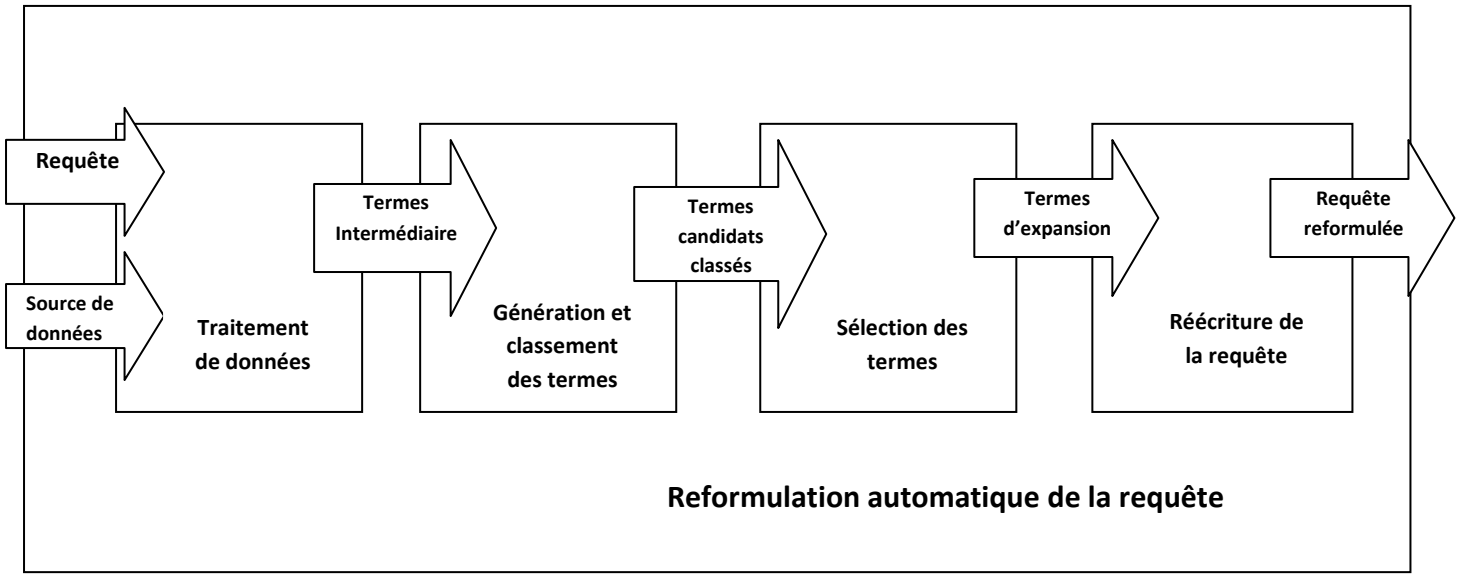

Figure II.2 Les étapes de la reformulation automatique de la requête

II.3.1 Traitement des données

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

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

De nombreuses techniques de reformulation de requêtes sont basées sur les informations contenues dans les tops document, extraits en réponse à la requête de l'utilisateur initiale à partir d'une collection de documents. Dans l'étape de traitement de données, il est nécessaire d'indexer la collection et d'exécuter la requête.

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. C'est ce type d'indexation que nous avons utilisé dans notre cas.

D'autres techniques de reformulation de requêtes, basées sur l'analyse du corpus, nécessitent l'extraction des termes particuliers de la collection manuellement, qui sont généralement différents de celui utilisé à des fins d'indexation par le système de recherche d'information classique. Une approche bien connue est celle développée

dans **[38]**, où chaque terme est représenté comme un vecteur de documents pondéré en utilisant les statistiques de collection non-standard.

Certaines techniques de reformulation de requêtes nécessitent des procédures de prétraitement adaptées aux certaines sources de données. Par exemple, si la reformulation de requête utilise des textes d'ancrage, il faut analyser une collection de lien hypertexte pour extraire le contenu du texte de l'ancre balisé, les normaliser et/ou de supprimer ceux qui contiennent trop peu ou trop de termes **[44]**.Les requêtes URL extraites des journaux d'archives des moteurs de recherche sont une autre source de données pour la reformulation 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 **[38]** et **[45]**.Où 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.

Dans notre cas, la source de données utilisée est les n premiers documents retournés par la première recherche.

II.3.2 Génération et classement des termes candidats d'expansion

Dans la deuxième étape de la reformulation de requêtes, le système génère et classe les termes candidats d'expansion. La raison pour laquelle le classement est important, c'est que la plus part des méthodes de reformulation des requêtes, 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 initiale et la source de données traitées ; le résultat est un ensemble de termes d'expansion, généralement avec des scores associés. Le traitement de la requête initiale et la source de données consiste à supprimer les mots vides 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).
Il existe deux type de relations : Association un à un et un à plusieurs.

II.3.2.1 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, 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 des mots connexes d'un mot de la requête à partir d'un thésaurus, le plus souvent utilisé est le WordNet, groupes de mots en anglais dans des ensembles de synonymes appelés **synsets** sont enregistrés, il y'en a diverses relations sémantiques lexicales entre ces ensembles de synonymes. En particulier, il comprend Hyperonyme (relation 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 à partir de WordNet implique de sélectionner un synset pour un terme de la requête, résolvant ainsi le problème de l'ambiguité, puis de traverser la hiérarchie en suivant ses liens typés. 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 la reformulation de la requête tous les synonymes du terme de la requête dans le synset.

L'approche statistique consiste à calculer automatiquement la similarité terme-à-terme dans une collection de documents.

L'idée générale est que 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 mesures simples de similarité sont utilisées à savoir, le coefficient Dice et l'indice de Jaccard. Compte tenu des termes u et v, le coefficient Dice(D) est défini comme suit :

$$D = \frac{2.df_{u \wedge v}}{df_u + df_v} \tag{II.1}$$

Où $df_{u \wedge v}$ est le nombre de documents qui contiennent à la fois les termes u et v, et df_u, df_v sont les nombres de documents contenant les termes u et v respectivement. L'indice de Jaccard (J) est défini comme suit :

$$J = \frac{df_{u \wedge v}}{df_{u \vee v}} \tag{II.2}$$

Où $df_{u \vee v}$ est le nombre de documents contenants les termes u ou v.

Une approche plus générale est la suivante. Considérons une matrice terme-document A où chaque cellule $A_{t,d}$ est le poids ($W_{t,d}$) du terme t dans 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) entre le terme u et le terme v, Cette similarité est exprimée comme suit :

$$C_{u,v} = \sum_{dj} W_{u,j} \ . W_{v,j} \tag{II.3}$$

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 d'utiliser des facteurs de corrélation normalisés, par exemple, la fonction de cosinus donnée comme suit :

$$\mathrm{C} = \frac{C_{u,v}}{\sqrt{\sum_{dj} W_{u,u}^2 . \sum_{dj} W_{v,v}^2}} \tag{II.4}$$

Selon la manière dont les documents et la fonction de pondération sont choisis, la formule ci-dessus peut donner lieu à différentes méthodes de corrélation terme-à-terme. Une technique bien connue, proposée dans **[39]**, 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 l'ensemble du 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 semble plus corrélés que deux termes qui apparaissent loin l'un de l'autre dans un document. Cet aspect est généralement abordé en considérant la proximité des termes c'est-à-dire, en utilisant des documents textuels restreint tels que des documents de longueur fixe pour mesurer la cooccurrence des termes.

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 a incorporé la dépendance entre termes est l'information mutuelle **[46] [47]**, défini comme suit :

$$\mathrm{I_{u,v}} = \log_2 \left[\frac{P(u,v)}{P(u).P(v)} + 1 \right] \tag{II.5}$$

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 (01), si les termes u et v sont indépendants, et égal à : $\log_2 \frac{1}{P(u)} + 1$ si le terme v est parfaitement associé avec le terme u.

Un des inconvénients de cette méthode est qu'elle a tendance à favoriser les termes rares sur les termes communs parce que, I(u,v) va augmenter si la probabilité 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 :

$$P(v,u) = \frac{P(u,v)}{P(u)} \quad \text{(II.6)}$$

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. Cette approche (par exemple, dans **[48] [49]**) est similaire à la définition des règles d'association dans l'exploration des problèmes de données **[50]**.

En fait, les règles d'association ont été explicitement utilisées pour trouver les termes d'expansion en corrélation avec les termes de la requête **[51] [52]**.

Les termes d'expansion peuvent également être générés par l'utilisateur dans le but d'associer les termes de la requête originale avec des termes dans les dernières requêtes utilisées. Comme les textes extraits de ces données (éventuellement après prétraitement) sont généralement très courts, les techniques de corrélation standard basée sur la fréquence des termes ne peuvent pas être appliquées. En fait, plusieurs termes supplémentaires extraits des journaux de requêtes ont été utilisés pour aider à identifier les associations utiles, telles que l'examen des requêtes qui ont eu lieu dans la même session (les requêtes successives émises par le même utilisateur) ou requêtes qui ont donné des ensembles similaires de documents vraisemblablement pertinents (en déployant le graphe biparti induit par les requêtes et les clics de l'utilisateur **[53]**).

L'approche que nous implémentons rentre dans le cadre de cette catégorie de méthodes.

II.3.2.2 Association 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 dans **[49]**. Par exemple, alors que le mot « programme » peut ainsi être fortement associé au mot « ordinateur », une reformulation automatique de toutes les requêtes contenant « programme » avec « ordinateur » pourrait bien ne pas fonctionner pour certaines requêtes (comme : « 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 association un-à-plusieurs » est d'étendre l'association un à un pour les autres termes de 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.

Dans **[54]**, il est nécessaire qu'un nouveau terme extrait du WordNet soit lié à au moins 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 de ce terme la requête q globale, par la formule suivante :

$$C_{q,v} = \frac{1}{|q|}\sum_{u \in q} C_{u,v} \qquad \text{(II.7)}$$

Une approche similaire a été proposée dans **[37]** et **[55]**, et suivie de plusieurs autres travaux de recherche **[56]**, **[49]**.

Les deux précédentes études 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 des termes d'expansion.

Dans **[37]**, la formule (II.3) est utilisée pour trouver des corrélations 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.

L'inverse de la fréquence du terme d'un document d_j est donné par $\log \frac{T}{DT_j}$ 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 l'inverse de la fréquence du document utilisé pour le classement des documents (idf).

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.3) est appliquée pour calculer la corrélation d'un terme-concept (au lieu d'une corrélation terme-terme), où $W_{v,j}$ est la fréquence du concept C dans le j-ième passage.

La valeur de corrélation terme-concept exacte est déterminée en tenant compte de la fréquence inverse du terme et du concept dans les passages contenus dans l'ensemble de la collection. Les facteurs de corrélation de chaque terme de la requête unique à un concept donné sont ensuite combinés par une fonction de leur produit. Cette méthode est appelée l'analyse du contexte local.

L'approche étendue de l'association un à un peut être utile pour filtrer les termes d'expansion qui sont faiblement liée à certains termes de la requête, 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, peuvent être dérivés comme conséquences logiques d'une base de connaissances, ou ils peuvent être extraits à partir d'un corpus en utilisant des cooccurrences de termes **[49]**. Considérons à nouveau notre exemple, si nous exigeons que « programme » apparait avec « application » (ou « java »), nous limitons l'applicabilité de l'association « programme »-« ordinateur » à des contextes appropriés.

Lorsque la reformulation de la requête est basée sur WordNet, la nécessité est de lier les termes d'expansion à l'ensemble de la requête, et non à ses termes pris isolément. **[54]** ont montré que ces dernière techniques ne sont pas généralement 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 comme c'est le cas dans **[56]**. 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 d'une requête donnée.

Cette approche est plus robuste par rapport à la rareté des données et prend en charge des données complexes impliquant des chaînes de termes. Pour surmonter les limitations des relations entre les termes simples, on peut voir la requête comme une expression et de chercher des phrases qui lui sont liés. Les phrases fournissent généralement un contexte plus riche et une moindre ambiguïté que leurs mots constitutifs, même si une évaluation de similarité de la phrase ne peut pas être simple.

II.3.3 La sélection des termes

Apres avoir classé les termes candidats, les éléments principaux sont sélectionnés pour la reformulation 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 **[57]**. Habituellement, seul un nombre limité de termes est sélectionné pour la reformulation, en partie parce que la requête qui en résulte peut être traitée plus rapidement, en partie parce que l'efficacité de la recherche avec un petit ensemble de termes n'a pas nécessairement moins de succès qu'en ajoutant tous les termes candidats d'expansion, en raison de la réduction du bruit **[58] [59]**.

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 **[60]** à quelques centaines **[61] [62] [63]**. D'autre part, la baisse de performance associée à des valeurs non optimales est généralement modeste, et 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 **[64]**. 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 éclairées. Il a été montré que les différentes requêtes ont un nombre varié de termes d'expansion optimal, et que de nombreux termes d'expansion **[65] [66] [67]** –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 **[68]** consiste à utiliser plusieurs fonctions de classement de termes et en sélectionnant pour chaque requête les termes les plus courants.

Une idée similaire est exploitée dans **[69]**, avec la différence que les multiples modèles feedback sont crées à partir de la même fonction de classement de termes et en générant des variantes de la requête initiale.

Une autre stratégie consiste à choisir une quantité variable de termes d'expansion en fonction de la difficulté de la requête. Dans **[66]**, 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), telle que mesuré par le score de clarté **[42]**.

Dans **[58]**, les auteurs utilisent un classificateur pour faire la distinction entre la pertinence et la non pertinence du classement des termes d'expansion. Pour apprendre les paramètres du classificateur (Support Vector Machine), un ensemble de mots simples sont étiquetés bon ou mauvais selon qu'ils améliorent ou nuisent au rendement, 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.3.4 La réécriture de la requête

La dernière étape de reformulation automatique 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 étendue appelée pondération de la requête.

La technique de pondération des termes d'une requête la plus populaire est calquée sur la formule de Rocchio pour la réinjection de pertinence **[61]** et ses améliorations ultérieures **[66]**. La formule générale est la suivante, où q' est la requête étendue, q est la requête originale, λ est un paramètres de pondération, et le score $w'_{t,q'}$ est un poids attribué au terme d'expansion t.

$$w'_{t,q'} = (1-\lambda).w_{t,q} + \lambda.score_t \qquad \text{(II.8)}$$

Lorsque les termes d'expansion sont extraits des documents pseudo-pertinents et leur score est calculé en utilisant ces documents, il est facile de montrer que le vecteur de la requête étendue calculé par la formule (II.8) se dirige vers les documents pseudo-pertinents (selon les pondérations de documents). 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.

La valeur de $\boldsymbol{\lambda}$ de la formule (II.8) 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 initiale ; 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 proposée dans **[70]**, compte tenu d'un paramètre de réinjection de pertinence, les auteurs utilisent une approche d'apprentissage pour prédire la valeur optimale de $\boldsymbol{\lambda}$ pour chaque requête et chaque ensemble de documents de feedback, explorer un certain nombre de paramètres 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.

La formule (II.8) peut également être utilisée lorsque les termes d'expansion sont extraits d'un thésaurus ou WordNet. Les pondérations peuvent être fondées sur des critères tels que le nombre de termes, le nombre de cooccurrence, la longueur du document, et le type de relation. Dans **[66]**, le vecteur de requête étendue est composé de sous-vecteurs de onze types de concepts différents avec un poids important associé : un pour les termes de la requête initiale, un pour les synonymes, et un pour chacun des autres types de relations contenues dans la partie nom de WordNet.

II.4 La reformulation de requêtes dans les modèles de RI

La reformulation de requêtes est utilisée afin d'améliorer les résultats de la première recherche basée sur la requête soumise par l'utilisateur, pour trouver des documents plus pertinents. La reformulation de requêtes est intégrée dans la plupart des modèles de RI. Dans ce qui suit, nous présentons cette intégration dans les deux modèles (modèle vectoriel et modèle de langue).

II.4.1 La reformulation de la requête dans le modèle vectoriel

La méthode Rocchio **[42]** est largement utilisée en recherche d'information, Rocchio considère que pour restituer les documents pertinents, il faut maximiser la différence entre le vecteur des documents pertinents et celui des documents non pertinents.

Le but de la reformulation est donc de rapprocher le vecteur de la requête initiale du vecteur moyen des documents pertinents, ceci est mis en œuvre en ajoutant de nouveaux termes à la requête initiale. Elle est exprimée comme suit :

$$Q_{nlle} = \alpha Q_{init} + \frac{\beta}{|D_r|}\sum_{D_j \in D_r} D_j - \frac{\gamma}{|D_n|}\sum_{D_j \in D_n} D_j \qquad \text{(II.9)}$$

Où :

$|.|$: Désigne le cardinal d'un ensemble.
$\boldsymbol{Q_{nlle}}$: Le vecteur de la nouvelle requête.
$\boldsymbol{Q_{init}}$: Le vecteur de la requête initiale.
$\boldsymbol{D_r}$: L'ensemble des documents restitués et jugés pertinents par l'utilisateur.
$\boldsymbol{D_n}$: L'ensemble des documents restitués et jugés non pertinents par l'utilisateur.
$\boldsymbol{D_j}$: Le j^{ème} document d'un ensemble.
$\boldsymbol{\alpha, \beta \ et \ \gamma}$: sont des paramètres.

Le jugement de l'utilisateur est ainsi exploité pour :

- Ajouter des termes issus des documents pertinents : leurs coefficients deviennent non nuls dans le nouveau vecteur de la requête.

- Repanderez les termes de la requête : les poids des termes de la requête sont réévalués sur la base de leur fréquence d'occurrence dans les documents pertinents et dans les documents non pertinents.

Ainsi, le nouveau vecteur de la requête est le vecteur de la requête initiale plus les termes qui différencient au mieux les documents pertinents des documents non pertinents.

II.4.2 Reformulation de la requête dans le modèle de langue

Au lieu de modéliser la RI comme processus de génération de la requête, Lavrenko et Croft **[70]** 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 utilisation de données d'entraînement, 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, soit ($\boldsymbol{\theta_R}$).

Le but est alors d'estimer la probabilité, $\mathbf{p(t|\theta_R)}$, 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 à partir du modèle de pertinence. Le modèle de pertinence est alors exprimé comme suit :

$$P(t|\theta_R) = \sum_{d \in R} \frac{P(t|d)P(q|d)P(d)}{P(q)} = \sum_{d \in R} P(t|d)\, P(q|d) \qquad \text{(II.10)}$$

Où R dénote l'ensemble de documents feedback.

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|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 **[30]**.

II.5 Conclusion

Ce chapitre a porté essentiellement sur la reformulation de requêtes qui est une phase importante du processus de recherche d'information. Elle consiste de manière générale à enrichir la requête initiale de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin. Cette technique peut être appliquée automatiquement ou d'une façon interactive, c'est-à-dire avec l'intervention de l'utilisateur.

L'apport de la reformulation de la requête dans l'amélioration des résultats de la RI est conditionné par plusieurs paramètres. Entre autre, la méthode de génération et classement des termes d'expansion avant de procédé a la reformulation de la requête. Dans le chapitre prochain, nous présentons une approche de classement des termes d'expansion, ainsi que sa mise en œuvre.

Chapitre III. Expérimentations et évaluation

III.1 Introduction

Nous avons abordé dans le chapitre précédent les différentes approches pour la reformulation automatique de requêtes, dans ce chapitre nous allons présenter une nouvelle approche pour améliorer le classement des termes qu'on utilise dans le processus de reformulation de la requête initiale, et cela en y ajoutant un nouveau facteur pour la fonction qui calcule le score des termes qui est : « les positions des termes dans les tops documents restitués », afin de donner un résultat plus pertinent pour le besoin en information de l'utilisateur.

Le chapitre suivant sera structuré comme suit :
En premier lieu, nous allons commencer par donner une présentation détaillée de notre approche, puis nous parlerons des outils utilisés pour l'implémentation de notre approche sous la plate forme Terrier, et enfin nous donnerons les résultats expérimentaux obtenus sur la collection de test TREC AP88.

III.2 L'approche implémentée

Dans cette section, nous allons présenter premièrement l'emplacement de notre approche dans le processus de la RI ; Ensuite nous présentons le principe de notre approche en donnant un exemple de fonctionnement. Enfin, nous illustrons les étapes ainsi que les algorithmes développés pour sa mise en œuvre.

III.2.1 Architecture générale de notre approche

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

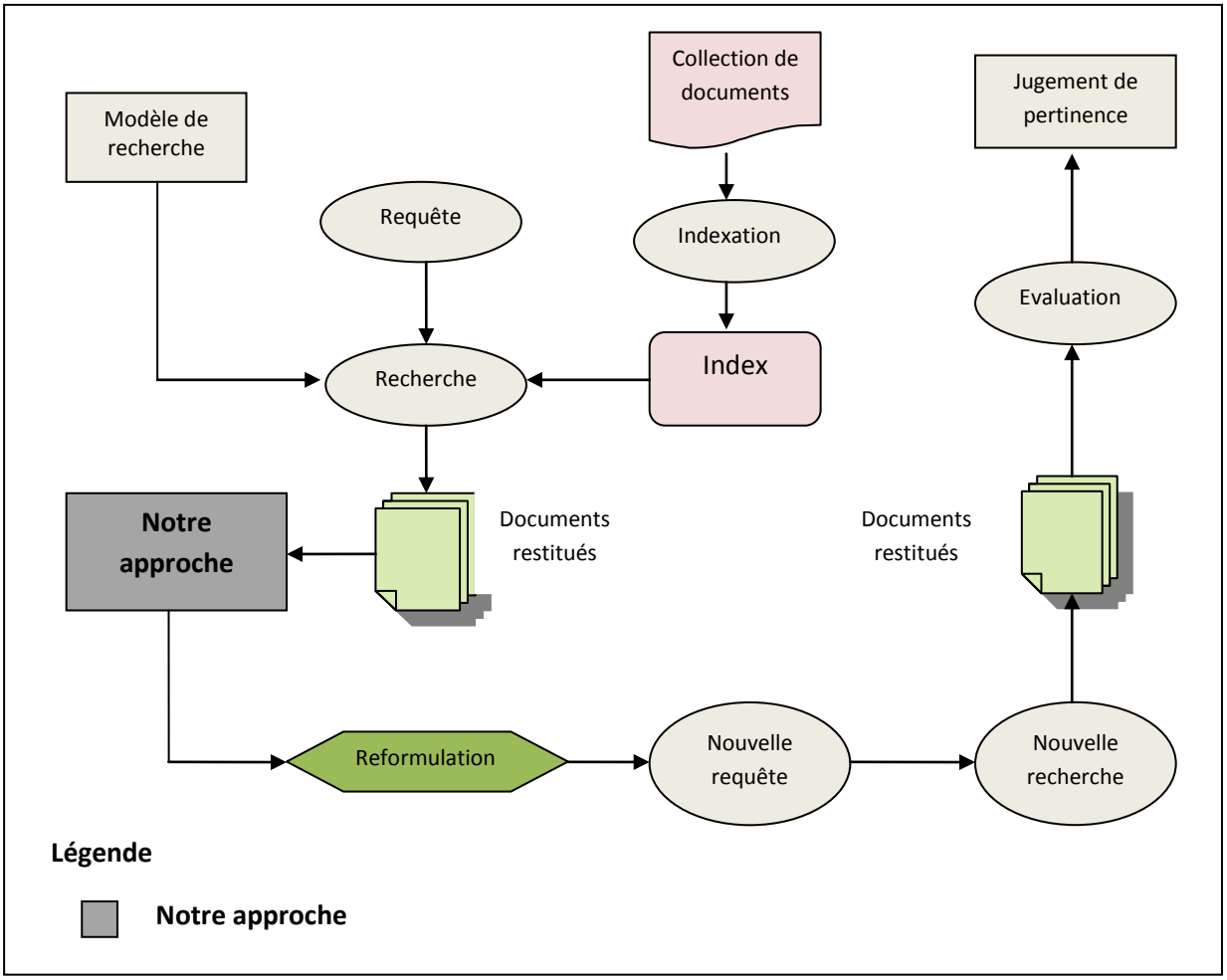

Figure III.1 Architecture générale de notre approche

III.2.2 Présentation de notre approche

Généralement la requête de l'utilisateur est courte et n'exprime pas clairement son besoin en information, ce qui à pour conséquence des résultats de recherche non satisfaisants. Pour y remédier, la reformulation de requête est utilisée, elle consiste à ajouter de nouveaux termes à la requête initiale.

Les techniques de reformulation comme nous l'avons précisé dans le chapitre précédent peuvent être classifiées selon plusieurs critères : la source des termes d'expansion (qui peut être locale ou globale), l'intervention ou pas de l'utilisateur et la méthode de sélection des termes d'expansion : qui peut considérer la requête toute entière ou les termes de la requête individuellement. De plus, ces techniques peuvent être basées sur des règles linguistiques ou statistiques.

Notre approche peut être classée comme suit : c'est une approche automatique (sans l'intervention de l'utilisateur). Elle utilise les premiers documents pseudo-pertinents comme source pour la reformulation de la requête, et enfin elle se base sur une technique statistique considérant les termes de la requête individuellement.

Dans ce qui suit, nous allons donner une étude détaillée sur notre approche et présenter le travail que nous avons réalisé.

III.2.2.1 Le principe de notre approche

La plupart des approches statistiques **[39] [46] [47]** de classement des termes d'expansion utilisent les paramètres suivant sur un terme pondérés : (la fréquence du terme dans les tops documents, et la fréquence du terme dans la collection), et elles se basent sur le principe suivant : « un bon terme d'expansion est terme fréquent dans les tops documents et peu fréquent dans la collection ».

Cependant, ces approches ignorent un facteur important qui est « la couverture spatiale », autrement dit « les positions du terme » dans les tops documents.

Ce que nous voulons exprimer par couverture spatiale d'un document par un terme est qu'un terme qui apparait dans plusieurs parties de document offre une bonne couverture spatiale pour le document par rapport à un terme qui apparait dans peu de parties de document.

Pour illustrer le fonctionnement de notre approche, nous prenons l'exemple suivant.

Supposant qu'on a deux documents d_1 et d_2, tels que :

- d_1 continent les termes : $\mathbf{t_1}$, t_2, t_3, t_5, $\mathbf{t_1}$, t_4, t_7, t_9, $\mathbf{t_1}$
  La fréquence du terme $\mathbf{t_1}$ dans le document $d_1 = 3$.

- d_2 contient les termes : $\mathbf{t_1}$, t_3, $\mathbf{t_1}$, t_7, $\mathbf{t_1}$, t_4, t_9, t_2, t_5
  La fréquence du terme $\mathbf{t_1}$ dans le document $d_2 = 3$.

A partir de cet exemple, nous remarquons que le terme $\mathbf{t_1}$ offre une meilleure couverture spatiale pour le document d_1 que pour le document d_2, et cela même si ça fréquence dans les deux documents est identique (ici 3). Donc on peut voir que la fréquence à elle seule ne peut pas capter l'importance d'un terme dans le document.

III.2.2.2 Processus de reformulation de la requête avec notre approche

Nous schématisons la mise en œuvre de la reformulation de la requête avec notre approche dans la figure suivante :

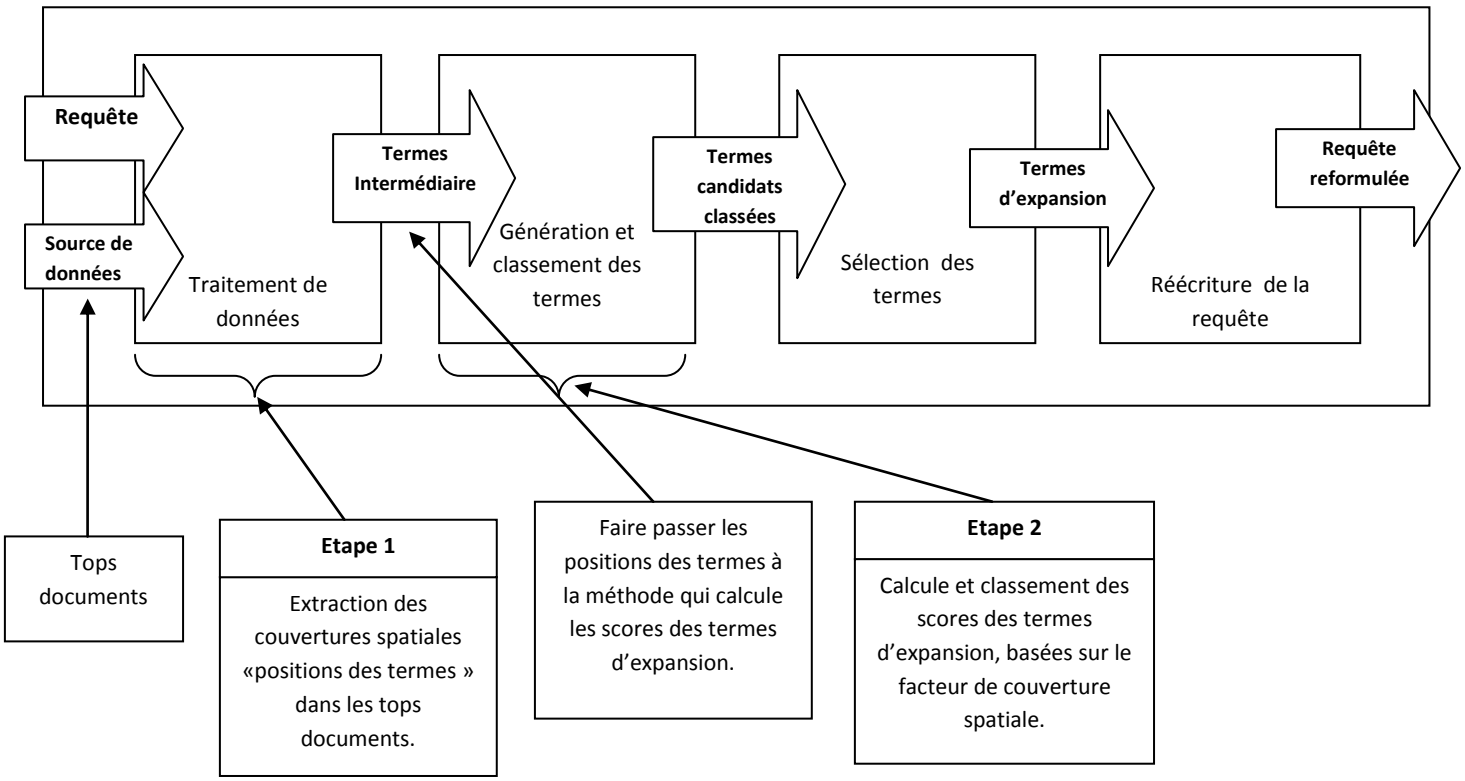

Figure III.2 Processus de reformulation de la requête avec notre approche

Pour les modèles de recherche, nous avons utilisé les deux modèles de recherche TF_IDF **[75]** et BM25 **[76]** dans le but de tester notre approche sur deux modèles différents.

Les classes étendues dans notre approche (« Query-Expansion », « Expansion-terms » et « Bo1 ») font partie de la plate-forme Terrier (présentée dans III.3.1).

Dans ce qui suit, nous allons montrer les algorithmes utilisés pour étendre ces classes.

III.2.2.3 Classes étendues et algorithmes utilisés

Pour réaliser notre approche, nous avons passé par plusieurs étapes pour obtenir les facteurs des positions des termes d'expansion. Pour cela nous avons étendu les trois classes de Terrier qui sont : « QueryExpansion », « ExpansionTerms » et « Bo1 ».

Nous allons maintenant décrire les modifications en utilisant des algorithmes :

Etape 1 : Extraction des positions des termes dans les tops documents :

Le but de cette étape est de récupérer tous les termes d'expansion des tops documents, ensuite associer pour chaque terme une matrice qui contient les documents (taille) où il est apparu, ainsi que ses positions dans ses documents. Cette étape est réalisée au niveau de la classe « QueryExpansion » de Terrier. Nous illustrons ci-dessus l'algorithme utilisé pour la création de ces matrices :

Algorithme de récupération des Matrices de positions :

Entrée : tops_Doc [ ] // Vecteur des tops documents
Mat_term [ ] [ ] // Matrice des termes (Contient Id_term et freq_term de chaque terme d'expansion)

Sortie : Id_term // Identificateurs des termes
Mat_position [ ] [ ] // Matrices de positions de chaque terme

```
Debut
    Parcourir le vecteur tops_Doc [ ] ; // parcourir tous les tops documents
    Récupérer Mat_term [ ] [ ] ;  /* récupérer pour chaque tops documents sa
                                     matrice des termes */
    Liste <Id_term> ;             // Liste des termes récupérés
  Pour (Chaque Mat_term [ ] [ ])
    Recuperer Id_term ;           // récupérer l'identificateur du terme

    Si (Liste contient Id_term) alors
      Récupérer le prochain terme ;

    Sinon
      Ajouter Id_term dans Liste ;
      Debut

        Pour (i=0 jusqu'à taille tops_Docs) ;
          Si id_term est dans tops_Docs (i) alors

            Récupérer la taille du tops_Docs (i) ;
            Mettre la taille du tops_Docs (i) dans Mat_position [i] [0] ;
            Recupérer freq_term du terme dans le tops_Docs (i) ;

            Pour (j=1 j'usqu'à freq_term)
              Récuperer les positions du terme dans le tops_Docs (i) ;
              Mettre chaque position récuperer dans Mat_position [i] [j] ;
              j= j+1 ;
            Fin pour

          Fin si
          i=i+1 ;
        Fin pour
      Fin
      Passer Id_term récupéré et sa Mat_position à la classe «ExpansionTerms » ;
  Fin si
Fin pour
Fin
```

Après avoir récupéré les identificateurs des termes et leurs matrices de positions de la classe « QueryExpansion», nous avons créé dans la classe « ExpansionTerms » une table « hash-map » qui contient (clé, valeur), le premier est l'identificateur du terme (Id_term) et l'autre est la matrice de positions associée à ce terme. Ensuite nous l'avons utilisé pour faire passer chaque matrice associée à un terme à la méthode **SCORE** qui renvoi le score de chaque terme d'expansion.
La méthode **SCORE** s'exécute dans la classe Bo1, la prochaine étape décrira les modifications réalisées sur cette méthode.

Etape 2 : Calcule et classement des scores des termes d'expansion, basée sur le facteur de couverture spatiale « positions des termes » :

Dans cette étape, nous avons modifié la méthode **SCORE** dans la classe « Bo1 » en utilisant les matrices récupérées précédemment pour créer le facteur des positions « FP » qu'on a utilisé dans la fonction qui renvoi les scores des termes. Nous illustrons ci-dessus l'algorithme utilisé pour calculer la valeur du facteur « FP » :

Algorithme de calcul du facteur des positions (FP) :

Entrée : Mat_position [ ][ ] //Matrice des positions du terme
frep //Fréquence du terme dans les tops documents

Sortie : FP //Facteur de positions

Debut

```
FL = 0 ;        //facteur positions du terme dans un tops_Docs
FP = 0 ;        //facteur positions du terme dans tous les tops_Docs

   Pour (i=0  jusqu'à taille lignes Mat_position)
      Doc_taille = Mat_position[i][0] ;        // taille du document
      Fenêtre = Doc_taille / 10 ;              // taille de la fenêtre du tops_Doc(i)
      fact=0 ;
      k=1 ;
      Pour (j=1 jusqu'à j=10)
         Tan que (Mat_position [i][k]<= j * Fenêtre)
            fact= fact+1 ;
            k= k+1 ;
         Fin Tan que ;
       FL = fact/fenêtre ;       // facteur positions du terme dans le tops_Docs(i)

      Fin pour

     FP = FL+FP ;               // facteur positions du terme dans tous les top_Docs

   Fin pour
  Calculer et retourner le score du terme
Fin
```

Après avoir récupéré le facteur FP, nous l'avons ajouté à la fonction qui renvoi le score du terme. Nous définissons d'abord la fonction score de la classe Bo1 avant de décrire l'intégration de ce nouveau facteur dans cette fonction.

La fonction score de (Bo1) utilise deux facteurs :

1) **f** = termFrequecy/numberOfDocuments (fréquece du terme dans la collection / nombre de documents dans la collection)
2) **frep** : frequence du term dans les tops documents.

Le calcule du score d'un terme se fait par la fonction suivante :

$$\textbf{frep} * \text{idf.log}\ (\ (1d + f) / f\ )\ +\ \text{idf.log}\ (\ 1d + f\ ) \qquad \text{(III.1)}$$

La fonction score de (Bo1_Etendue) avec le facteur des positions : on ajoute le facteur FP qu'on a calculé précédemment en le multipliant avec la fréquence du terme frep, et on obtient la formule suivante :

$$\textbf{(frep*FP)} * \text{idf.log}\ (\ (1d + f) / f\ )\ +\ \text{idf.log}\ (\ 1d + f\ ) \qquad \text{(III.2)}$$

III.3 L'environnement de travail

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.3.1 Présentation de la plate forme Terrier

Terrier, TERabyte RetrlEveR est un système de recherche d'information robuste et efficace qui offre une plate-forme idéale pour manipuler de grandes quantités de données « jusqu'à 25 millions de documents ». Il offre en plus un cadre pour l'évaluation des résultats de recherche pour différentes applications. Développé par le département d'informatique de l'université Glasgow de Scotland. Il est utilisé avec succès pour la recherche Ad-hoc, la recherche web et la recherche inter-langages dans des environnements centralisés et distribués **[71]**.

Terrier est un logiciel open source entièrement écrit en java. Selon des études comparatives sur les SRI en open sources effectuées par Ricardo Baeza-Yates **[72]**, Terrier fait partie des trois meilleurs systèmes dans un environnement java, les deux autres sont MG4J et Lucene.
Comme tous les systèmes de RI, Terrier possède les principaux processus suivants :

L'indexation, la recherche et l'évaluation.

La figure III.3 illustre l'architecture de Terrier.

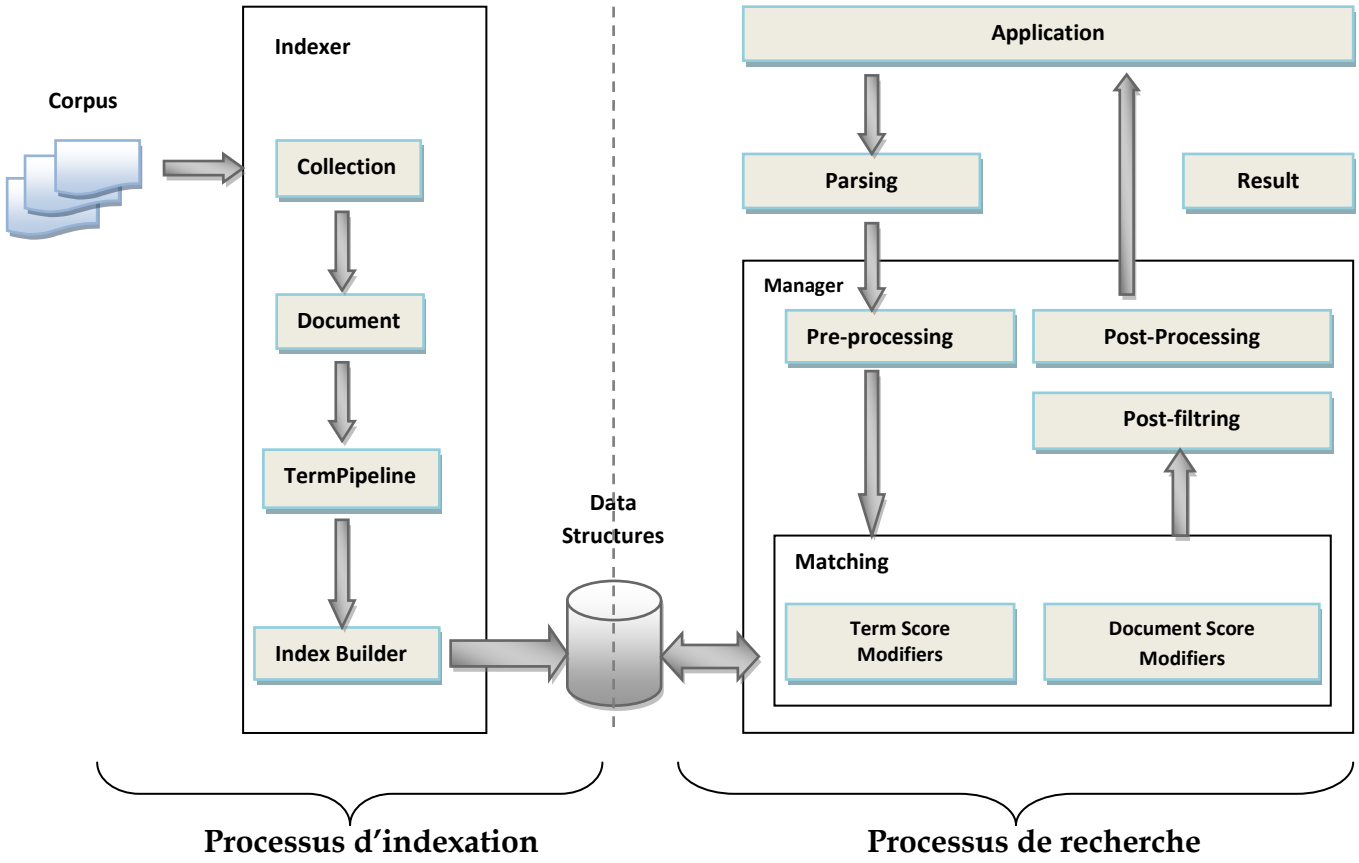

Figure III.3 Architecture de Terrier

A) Le processus d'indexation de Terrier : l'indexation dans Terrier est divisée en quatre procédures :

1) La première procédure consiste à prendre en entrée le corpus à indexer et d'envoyer document à document pour la seconde procédure.
2) Le second traitement consiste à dé-baliser le document reçu (enlever les balises) et le passer au Tokenizer qui extrait les termes du document, et les envois à la troisième procédure
3) Dans la troisième étape, les termes reçus sont traités par le TermPipeline (l'élimination des mots vides et la lemmatisation des termes).
4) La dernière étape consiste à construire les structures de l'index.

B) Le processus de recherche de Terrier : durant le processus de recherche, chaque requête doit passer par les étapes suivantes :

1) **Parsing** : qui se charge d'extraire les tokens de la requête.
2) **Pré-processing:** qui applique le TermPipeline à la requête. Élimine les mots vides et les lemmatise.
3) **Matching** : qui est responsable de l'initialisation du WeightingModel et du calcul des scores entre la requête et les documents.
4) **post-filtrage** : filtre les résultats.
5) **post-traitement** : peut modifier le ResultSet, par exemple, par un procédé QueryExpansion afin de générer un meilleur classement de documents.

III.3.2 Le langage Java

Afin de réaliser notre application, nous avons utilisé le langage java ; qui est un langage de programmation moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle).

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du langage C. Ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est largement utilisé pour le développement d'applications d'entreprises et mobiles. C'est un langage très utilisé, notamment par un grand nombre de programmeurs professionnels, ce qui en fait un langage incontournable actuellement **[73]**.

Le Java est langage très développé doté d'une riche bibliothèque de classes, comprenant la gestion des interfaces graphiques (fenêtres, boites de dialogues, contrôles, menus, graphismes), et la gestion des exceptions.

Une de ses plus grandes forces est son excellente portabilité, java est un langage multi-plate forme, qui permet aux concepteurs d'écrire un code capable de fonctionner dans tous les environnements. Pour cela, il suffit que l'environnement possède une JVM (Java Virtual Machine). En plus, il permet d'interagir aisément avec le Web et d'Accéder et fournir des services simplement au travers du réseau.

Le code source Java est compilé en byte code correspondant aux instructions de la machine virtuelle(JVM). La JVM est un programme source compilé pour chaque type de couple (machine physique + système d'exploitation).

La machine virtuelle fournit un environnement de programmation standard

- Gestion de la mémoire (ramasse-miette)
- Gestion des entrées / sorties (disque, écran, clavier, etc.)
- Interface avec le système d'exploitation

III.3.3 L'environnement de développement (Netbeans)

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun .En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java. NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme **[74]**.

La figure ci-dessus présente un aperçu de l'interface de l'IDE NetBeans 6.8 utilisé.

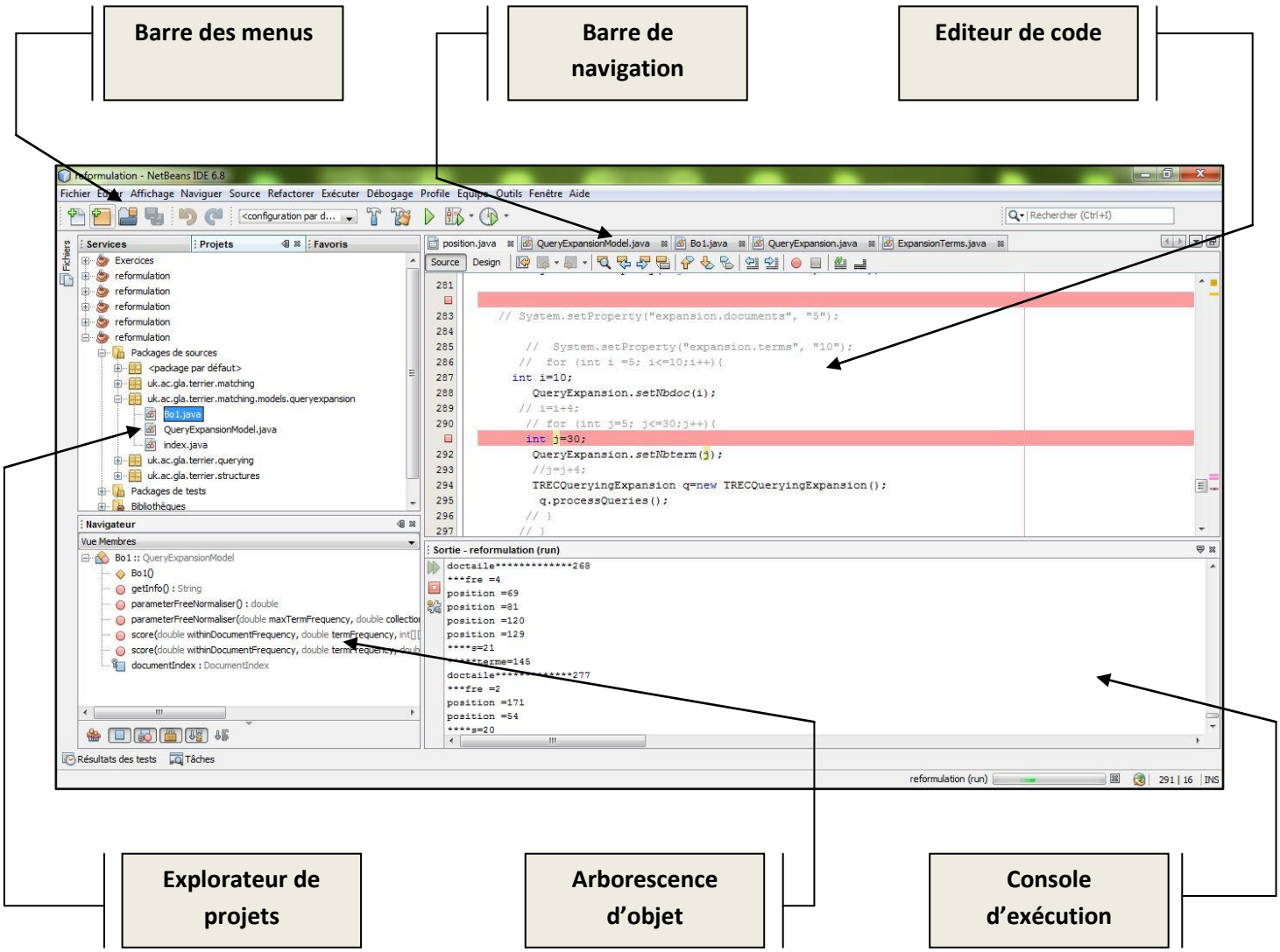

Figure III.4 L'environnement de développement NetBeans

Elle est composée :

1) Du menu et de la barre d'outils qui contient la palette d'outils: permet d'ouvrir, d'enregistrer, de compiler, d'exécuter et de déboguer des projets.
2) L'explorateur de projet : Il montre les classes, les interfaces et les routines de l'unité et il permet de naviguer rapidement.
3) L'arborescence d'objets donne une vue des différents composants de notre application et permet d'afficher et de modifier les relations entre eux.
4) L'inspecteur d'objets permet de modifier les événements et les propriétés d'un composant choisi.
5) L'éditeur de code permet d'introduire le code source.
6) Le concepteur de fiche permet de concevoir les interfaces des applications et offre un aperçu de ces dernières.
7) La console d'exécution : permet de d'afficher le résultat d'exécution du code.

III.4 Résultats et expérimentations

III.4.1 Collection de test utilisée

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 de test utilisée

III.4.2 Résultats et évaluation

Pour évaluer notre approche, nous allons faire une comparaison entre les résultats de reformulation obtenus avec le modèle Bo1 et ceux obtenus avec notre approche (Bo1_Etendu), et cela en appliquant les mêmes paramètres de recherche (nombre de termes et de documents d'expansion). La mesure MAP (Moyenne Average Précision) décrite dans le chapitre I, est utilisée comme mesure d'évaluation.

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).

III.4.2.1 Résultats obtenus avec la recherche simple

Au niveau de la recherche simple, nous avons utilisé les deux modèles de recherche TF_IDF et BM25, pour chacun des deux nous avons fait une évaluation avec la MAP (Moyenne Average Précision) et nous avons obtenu les résultats résumés dans ce tableau :

Recherche simple	
Modèle de recherche	**MAP**
TF_IDF	**0.1245**
BM25	**0.1334**

Tableau III.2 Résultats obtenus avec la recherche simple

III.4.2.2 Résultats obtenus après reformulation (Modèle Bo1)

Dans cette étape, nous allons présenter les résultats obtenus après reformulation avec le modèle Bo1 et cela en utilisant les deux modèles de recherche TF_IDF et BM25, 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 **30** avec un pas de **5**, et en variant aussi le nombre de termes d'expansion allant de **5** a **30** également avec un pas de **5**.

Les résultats obtenus sont résumés dans les deux tableaux suivant :

Reformulation (Modèle Bo1) avec TF_IDF					
Nombre de documents	**Nombre de termes**	**MAP**	**Nombre de documents**	**Nombre de termes**	**MAP**
1	5	0.1518	20	5	0.1572
	10	0.1518		10	0.1572
	15	0.1518		15	0.1571
	20	0.1518		20	0.1571
	25	0.1518		25	0.1571
	30	0.1518		30	0.1571
5	5	0.1521	25	**5**	**0.1573**
	10	0.1522		10	0.1572
	15	0.1522		15	0.1572
	20	0.1522		20	0.1572
	25	0.1522		25	0.1572
	30	0.1522		30	0.1572
10	5	0.1519	30	5	0.1558
	10	0.1518		10	0.1560
	15	0.1518		15	0.1560

	20	0.1518		20	0.1561
	25	0.1518		25	0.1561
	30	0.1518		30	0.1561
15	5	0.1566			
	10	0.1563			
	15	0.1563			
	20	0.1562			
	25	0.1562			
	30	0.1562			

Tableau III.3 Résultats obtenus dans la reformulation avec le modèle Bo1 (TF_IDF)

Le tableau ci-dessus résume les résultats obtenus dans la reformulation (Modèle Bo1) avec le modèle TF_IDF, nous constatons que la meilleure précision est de **0.1573,** obtenue avec un nombre de documents égal à **25** et un nombre de termes égal à **5**.

Reformulation (Modèle Bo1) avec BM25					
Nombre de documents	**Nombre de termes**	**MAP**	**Nombre de documents**	**Nombre de termes**	**MAP**
1	5	0.1533	20	**5**	**0.1592**
	10	0.1533		10	0.1591
	15	0.1533		15	0.1590
	20	0.1533		20	0.1590
	25	0.1533		25	0.1590
	30	0.1533		30	0.1590
5	5	0.1537	25	5	0.1566
	10	0.1537		10	0.1564
	15	0.1537		15	0.1565
	20	0.1537		20	0.1566
	25	0.1537		25	0.1566
	30	0.1537		30	0.1566
10	5	0.1536	30	5	0.1558
	10	0.1535		10	0.1557
	15	0.1535		15	0.1558
	20	0.1535		20	0.1558
	25	0.1535		25	0.1558
	30	0.1535		30	0.1558
15	5	0.1580			
	10	0.1580			
	15	0.1580			
	20	0.1579			
	25	0.1579			
	30	0.1579			

Tableau III.4 Résultats obtenus dans la reformulation avec le modèle Bo1 (BM25)

Le tableau ci-dessus résume les résultats obtenus dans la reformulation (Modèle Bo1) avec le modèle BM25, nous constatons que la meilleure précision est de **0.1592,** obtenue avec un nombre de documents égal à **20** et un nombre de termes égal à **5**.

Dans ce qui suit, nous allons présenter les résultats obtenus dans la reformulation avec notre approche (Modèle Bo1_Etendu).

III.4.2.3 Résultats obtenus avec notre approche (Bo1_Etendu)

Reformulation approche avec TF_IDF					
Nombre de documents	**Nombre de termes**	**MAP**	**Nombre de documents**	**Nombre de termes**	**MAP**
1	5	0.1385	20	**5**	**0.1659**
	10	0.1385		10	0.1656
	15	0.1385		15	0.1656
	20	0.1385		20	0.1656
	25	0.1385		25	0.1656
	30	0.1385		30	0.1656
5	5	0.1533	25	5	0.1630
	10	0.1533		10	0.1620
	15	0.1533		15	0.1620
	20	0.1533		20	0.1620
	25	0.1533		25	0.1620
	30	0.1533		30	0.1620
10	5	0.1592	30	5	0.1621
	10	0.1591		10	0.1615
	15	0.1591		15	0.1615
	20	0.1591		20	0.1616
	25	0.1591		25	0.1616
	30	0.1591		30	0.1616
15	5	0.1622			
	10	0.1621			
	15	0.1621			
	20	0.1621			
	25	0.1621			
	30	0.1621			

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

Le tableau ci-dessus résume les résultats obtenus dans notre approche avec le modèle de recherche TF_IDF, nous constatons que la meilleure précision est de **0.1659,** obtenue avec un nombre de documents égal à **20** et un nombre de termes égal à **5**.

Reformulation approche avec BM25					
Nombre de documents	**Nombre de termes**	**MAP**	**Nombre de documents**	**Nombre de termes**	**MAP**
1	5	0.1455	20	5	0.1650
	10	0.1455		10	0.1653
	15	0.1455		15	0.1653
	20	0.1455		20	0.1653
	25	0.1455		25	0.1653
	30	0.1455		30	0.1653
5	5	0.1603	25	5	0.1652
	10	0.1603		10	0.1652
	15	0.1603		15	0.1652
	20	0.1603		20	0.1652
	25	0.1603		25	0.1652
	30	0.1603		30	0.1652
10	5	0.1623	30	**5**	**0.1662**
	10	0.1623		10	0.1660
	15	0.1623		15	0.1655
	20	0.1623		20	0.1655
	25	0.1623		25	0.1655
	30	0.1623		30	0.1655
15	5	0.1656			
	10	0.1656			
	15	0.1656			
	20	0.1656			
	25	0.1656			
	30	0.1656			

Tableau III.6 Résultats obtenus dans la reformulation avec notre approche (BM25)

Le tableau ci-dessus résume les résultats obtenus dans notre approche avec le modèle BM25, nous constatons que la meilleure précision est de **0.1662,** obtenue avec un nombre de documents égal à **30** et un nombre de termes égal à **5**.

III.4.2.4 Comparaison entre le modèle Bo1 et notre approche (Bo1_Etendu)

Nous résumons dans la figure III.5 les meilleures précisions obtenues dans les tableaux des deux sections précédentes.

RB = Reformulation avec Bo1.

RA= Reformulation avec notre approche (Bo1_Etendu).

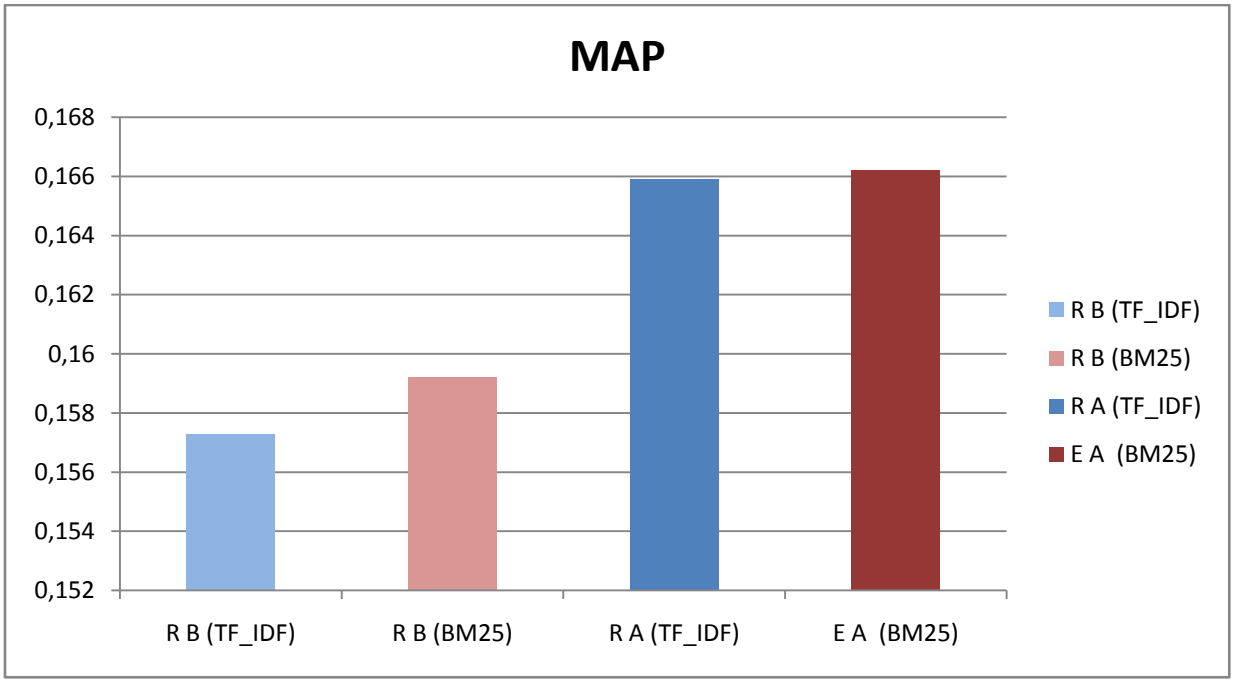

Figure III.5 Comparaison entre les meilleures précisions obtenues dans la reformulation (Modèle Bo1) et la reformulation avec notre approche

Cette figure nous montre que la reformulation avec notre approche a apporté une meilleure précision par rapport à la reformulation avec de le modèle Bo1.

Le tableau ci-dessus nous montre nous le taux d'amélioration entre la reformulation avec Bo1 et la reformulation avec notre approche avec les deux modèles TF_IDF et BM25.

Nous avons pris pour chaque modèle la meilleure précision obtenue dans les deux types de reformulation.

Modèle	**MAP (Bo1)**	**MAP (approche)**	**Taux d'amélioration**
Modèle TF_IDF	**0.1573**	**0.1659**	**+5,47 %**
Modèle BM25	**0.1592**	**0.1662**	**+4.40 %**

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

D'après le tableau ci-dessus nous constatons que le modèle Bo1 étendue avec notre approche a apporté une amélioration de **+ 5.47** % en utilisant le modèle de recherche TF_IDF et un taux de + **4.40** % en utilisant le modèle de recherche BM25.

III.4.2.5 Evaluation requête par requête

Dans les tableaux suivants, afin d'avoir une estimation plus précise des améliorations constatées, nous avons évalué les résultats des meilleures précisions obtenues précédemment, mais cette fois avec une évaluation requête par requête.

R B = Reformulation (Modèle Bo1).
R A = Reformulation avec notre approche (Bo1_Etendu).

Requête	MAP R B (TF_IDF)	MAP R A (TF_IDF)	Taux d'amélioration (%)	Requête	MAP R B (TF_IDF)	MAP R A (TF_IDF)	Taux d'amélioration (%)
51	0.7713	0.8192	6,21	76	0.0069	0.0057	-17,39
52	0.7354	0.7688	4,54	77	0.0003	0.0047	1466,67
53	0.4713	0.4851	2,93	78	0.0040	0.0161	302,5
54	0.5612	0.5492	-2,14	79	0.0000	0.0000	0
55	0.1061	0.1359	28,09	80	0.0774	0.0657	-15,12
56	0.7040	0.8002	13,66	81	0.1927	0.2203	14,32
57	0.0031	0.0105	238,71	82	0.0681	0.1821	167,40
58	0.0716	0.1870	161,17	83	0.0256	0.0063	-75,39
59	0.0000	0.0006	0.06	84	0.0282	0.0352	24,82
60	0.0007	0.0007	0	85	0.0143	0.0176	23,08
61	0.5552	0.6257	12,70	86	0.0031	0.0023	-25,80
62	0.1966	0.2332	18,61	87	0.0147	0.0034	-76,80
63	0.1228	0.0043	-96,50	88	0.0861	0.0923	7,20
64	0.0020	0.0015	-25	89	0.0027	0.0024	-0,11
65	0.0000	0.0000	0	90	0.3602	0.2961	-17,80
66	0.0008	0.0007	-12.5	91	0.0003	0.0000	-0,03
67	0.0008	0.0000	-0.08	92	0.0012	0.0009	-25
68	0.1502	0.1311	-12,71	93	0.0006	0.0031	416,66
69	0.0496	0.0551	11,09	94	0.0007	0.0006	-14,29
70	0.6057	0.6195	2,28	95	0.0047	0.0087	85,11
71	0.0017	0.0144	747,06	96	0.0516	0.0375	-27,33
72	0.0001	0.0007	600	97	0.6057	0.6276	3,62
73	0.0001	0.0000	-0,01	98	0.6606	0.6188	-6,33
74	0.0001	0.0004	300	99	0.3314	0.3743	12,95
75	0.0000	0.0017	0,17	100	0.0544	0.0616	13,24

Tableau III.8 Comparaison entre l'évaluation requête par requête entre la reformulation (Modèle Bo1) et la reformulation avec notre approche (TF_IDF)

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

- 28 requêtes améliorées
- 19 requêtes dégradées
- 3 requêtes nulles

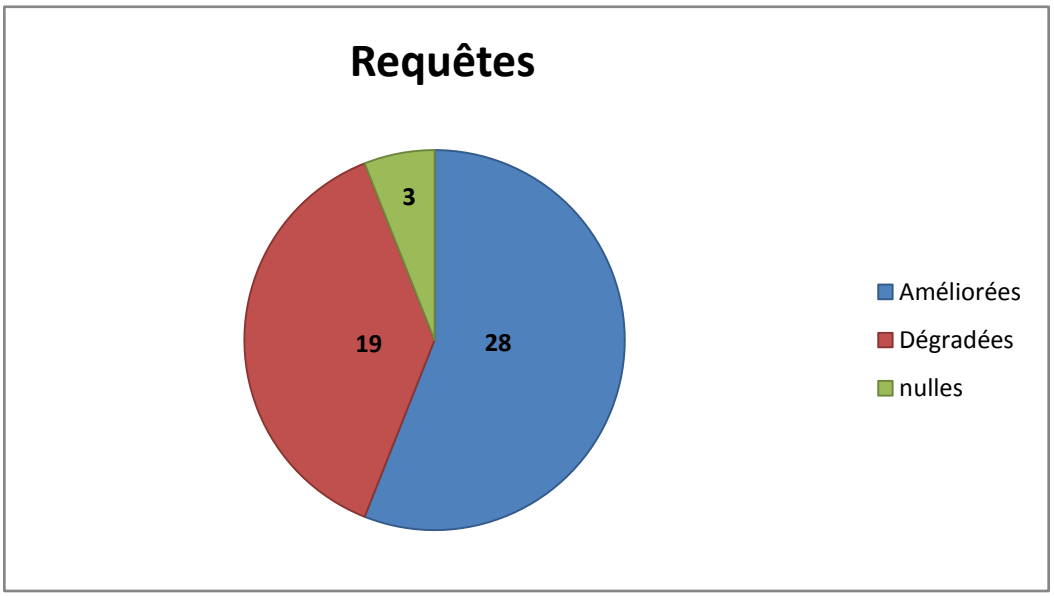

Figure III.6 Résultats obtenus après évaluation requête par requête (TF_IDF)

Requête	MAP R B (BM25)	MAP R A (BM25)	Taux d'amélioration (%)	Requête	MAP R B (BM25)	MAP RA (BM25)	Taux d'amélioration (%)
51	0.7848	0.8194	4,41	76	0.0096	0.0115	19,79
52	0.7324	0.7731	5,56	77	0.0006	0.0123	1950
53	0.4723	0.4872	3,15	78	0.0057	0.0182	219,30
54	0.5591	0.5519	-1,29	79	0.0000	0.0000	0
55	0.1013	0.1133	11,85	80	0.0763	0.0701	-8,13
56	0.6930	0.7965	14,94	81	0.1903	0.2289	20,28
57	0.0424	0.0115	-72,88	82	0.1081	0.2542	135,15
58	0.0318	0.0128	-59,75	83	0.0268	0.0370	38,06
59	0.0000	0.0001	0.01	84	0.0306	0.0345	12,75
60	0.0006	0.0006	0	85	0.0089	0.0079	-11,24
61	0.6017	0.6379	6,02	86	0.0031	0.0023	-25,81
62	0.2014	0.2527	25,47	87	0.0057	0.0167	192 ,98
63	0.1093	0.0081	-92,59	88	0.0863	0.0903	4,63
64	0.0018	0.0008	-55,55	89	0.0028	0.0024	-14,29
65	0.0000	0.0000	0	90	0.3676	0.3390	-7,78
66	0.0009	0.0006	-0,33	91	0.0000	0.0000	0
67	0.0007	0.0001	-58,71	92	0.0011	0.0006	-45,45
68	0.1410	0.1428	1,28	93	0.0004	0.0001	-75
69	0.0570	0.0489	-14,21	94	0.0009	0.0006	-33,33
70	0.6009	0.6142	2,28	95	0.0058	0.0084	44,83
71	0.0263	0.0139	-47,15	96	0.0504	0.0153	-69,64
72	0.0001	0.0011	1000	97	0.6201	0.6312	1,79

73	0.0000	0.0000	**0**	98	0.6609	0.6061	**-8,29**
74	0.0001	0.0004	**300**	99	0.3293	0.4155	**26,18**
75	0.0027	0.0017	**-37,04**	100	0.0483	0.0515	**6 ,62**

Tableau III.9 Comparaison entre l'évaluation requête par requête entre la reformulation (Modèle Bo1) et la reformulation avec notre approche (BM25)

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

- 25 requêtes améliorées
- 20 requêtes dégradées
- 5 requêtes nulles

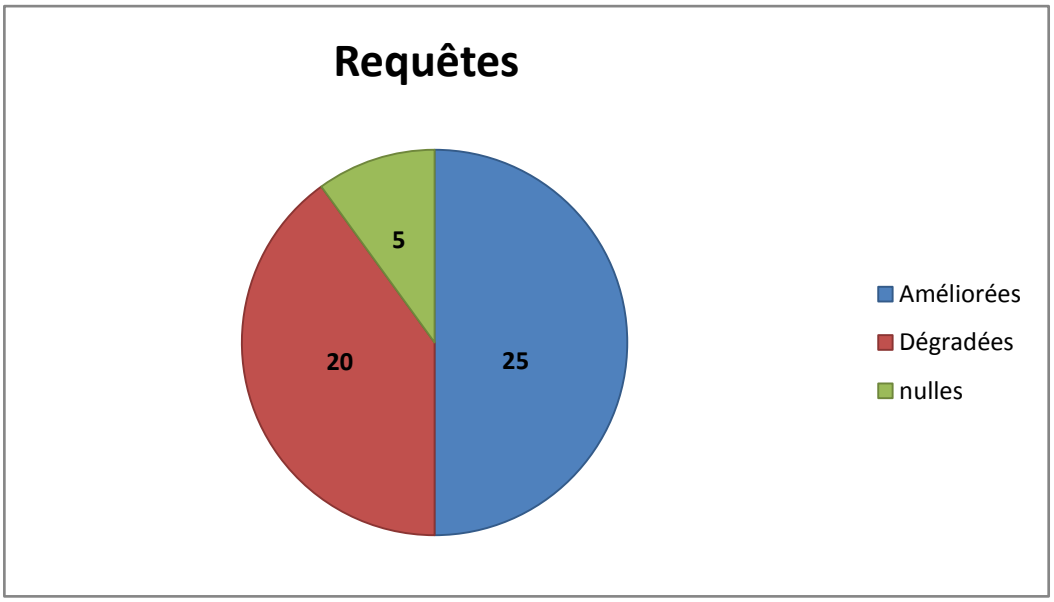

Figure III.7 Résultats obtenus après évaluation requête par requête (BM25)

III.4.2.6 Analyse des résultats basée sur le type de requêtes

Nous présentons dans les tableaux qui suivent le taux d'amélioration des résultats obtenus précédemment selon le type requête, nous avons classé les requêtes en trois types :

- Claires si MAP >0.2
- Moyennes si 0.05<MAP<0.2
- Ambigües si MAP<0.05

R B = Reformulation (Modèle Bo1).
R A = Reformulation avec notre approche.

Requêtes	R B	R A	Taux RA par rapport à RB	Requêtes	R B	R A	Taux RA par rapport à RB
51	Claire	+	**72,72 %**	65	Ambigüe		**48,28 %**
52	Claire	+		66	Ambigüe		
53	Claire	+		67	Ambigüe		
54	Claire			69	Ambigüe	+	
56	Claire	+		71	Ambigüe	+	
61	Claire	+		72	Ambigüe	+	
70	Claire	+		73	Ambigüe		
90	Claire			74	Ambigüe	+	
97	Claire	+		75	Ambigüe	+	
98	Claire			76	Ambigüe		
99	Claire	+		77	Ambigüe	+	
55	Moyenne	+	**60 %**	78	Ambigüe	+	
58	Moyenne	+		79	Ambigüe		
62	Moyenne	+		83	Ambigüe		
63	Moyenne			84	Ambigüe	+	
68	Moyenne			85	Ambigüe	+	
80	Moyenne			86	Ambigüe		
81	Moyenne	+		87	Ambigüe		
82	Moyenne	+		88	Ambigüe	+	
96	Moyenne			89	Ambigüe		
100	Moyenne	+		91	Ambigüe		
57	Ambigüe	+		92	Ambigüe		
59	Ambigüe	+		93	Ambigüe	+	
60	Ambigüe			94	Ambigüe		
64	Ambigüe			95	Ambigüe	+	

Tableau III.10 Taux d'améliorations obtenues entre le modèle Bo1 et notre approche en se basant sur le type de requêtes (TF_IDF)

Requêtes	R B	R A	Taux RA par rapport à RB	Requêtes	R B	R A	Taux RA par rapport à RB
51	Claire	+	75%	60	Ambigüe		37,04%
52	Claire	+		64	Ambigüe		
53	Claire	+		65	Ambigüe		
54	Claire			66	Ambigüe		
56	Claire	+		67	Ambigüe		
61	Claire	+		71	Ambigüe		
62	Claire	+		72	Ambigüe	+	
70	Claire	+		73	Ambigüe		
90	Claire			74	Ambigüe	+	
97	Claire	+		75	Ambigüe		
98	Claire			76	Ambigüe	+	
99	Claire	+		77	Ambigüe	+	
55	Moyenne	+	54,54%	78	Ambigüe	+	
58	Moyenne			79	Ambigüe		
63	Moyenne			83	Ambigüe	+	
68	Moyenne	+		84	Ambigüe	+	
69	Moyenne			85	Ambigüe		
80	Moyenne			86	Ambigüe		
81	Moyenne	+		87	Ambigüe	+	
82	Moyenne	+		89	Ambigüe		
88	Moyenne	+		91	Ambigüe		
96	Moyenne			92	Ambigüe		
100	Moyenne	+		93	Ambigüe		
57	Ambigüe			94	Ambigüe		
59	Ambigüe	+		95	Ambigüe	+	

Tableau III.11 Taux d'améliorations obtenues entre le modèle Bo1 et notre approche en se basant sur le type de requêtes (BM25)

Dans le tableau III.10, la reformulation de notre approche avec le modèle TF_IDF a révélé les résultats suivants :

- 8 requêtes améliorées sur 11 requêtes claires de la reformulation (Modèle bo1) avec un taux de requêtes claires améliorées de 72,72 %.
- 6 requêtes améliorées sur 10 requêtes moyennes de la reformulation (Modèle Bo1) avec un taux de requêtes moyennes améliorées de 60 %.
- 14 requêtes améliorées sur 29 requêtes ambigües de la reformulation (Modèle Bo1) avec un taux de requêtes ambigües améliorées de 48,28 %.

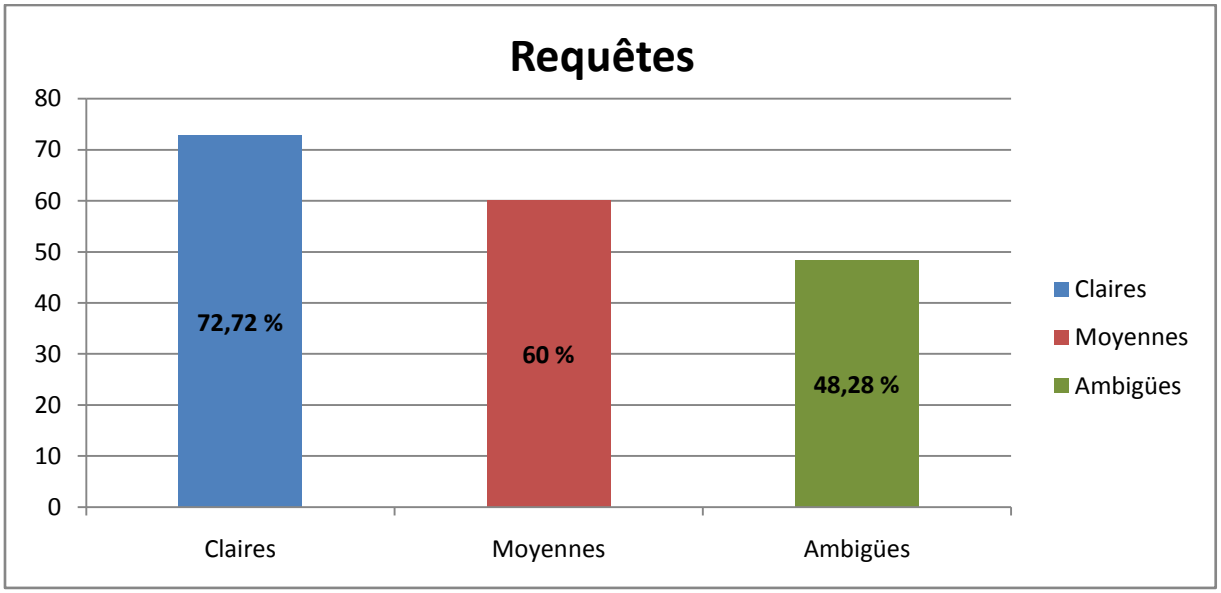

Figure III.8 Résultats obtenus après évaluation basée sur le type de requêtes (TF_IDF)

Dans le tableau III.11, la reformulation de notre approche avec le modèle BM25 a révélé les résultats suivants :

- 9 requêtes améliorées sur 12 requêtes claires de la reformulation (Modèle Bo1) avec un taux de requêtes claires améliorées de 75 %.
- 6 requêtes améliorées sur 11 requêtes moyennes de la reformulation (Modèle Bo1) avec un taux de requêtes moyennes améliorées de 54,54 %.
- 10 requêtes améliorées sur 27 requêtes ambigües de la reformulation (Modèle Bo1) avec un taux de requêtes ambigües améliorées de 37,04 %.

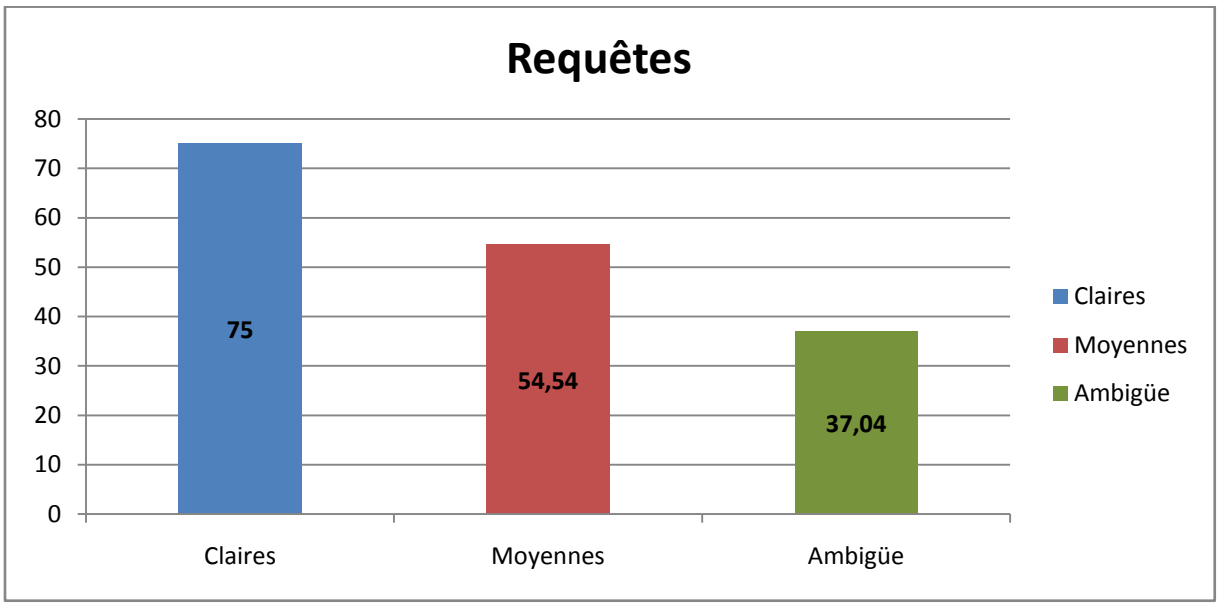

Figure III.9 Résultats obtenus après évaluation basée sur le type de requêtes (BM25)

III.5 Conclusion

Dans ce chapitre nous avons exposé notre approche qui consiste à étendre le modèle Bo1 avec le facteur : couverture spatiale « positions du terme » dans les tops documents restitués, puis nous avons testé cette approche et obtenus des résultats que nous avons présentés sous forme de tableaux et de graphes pour les comparer aux résultats obtenus avec le modèle Bo1.

Le test de cette approche en introduisant ce nouveau facteur nous a apporté de meilleures résultats en terme de précision, ainsi que la pertinence retournés à l'utilisateur.

Conclusion générale

Le travail présenté dans ce mémoire se situe dans le cadre de la recherche d'information, particulièrement au niveau de l'étape de reformulation de requête. Notre objectif consistait à intégrer un nouveau facteur de classement des termes d'expansion, qui est la couverture spatiale d'un document. L'objectif recherché est de retourner à l'utilisateur les bons documents constituants l'information dont il a besoin comme réponse à sa requête.

Notre mémoire s'articule en trois chapitres, le premier chapitre introduit un cadre général ou nous avons présenté les points essentiels du domaine de la RI, dans le second chapitre nous avons décrit en détail le processus de reformulation de requête, les étapes de la reformulation automatique de la requête ainsi que la reformulation de requête dans les modèles de RI. Dans le chapitre trois nous avons présenté notre travail, nous avons décrit les résultats obtenus avec notre approche qur la collection TREC APP88.

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

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

Sur le plan conceptuel, nous avons acquis des connaissances sur le fonctionnement des SRI en particulier le SRI Terrier.

Enfin sur le plan pratique, nous avons amplifié nos connaissances sur le langage Java lors de la réalisation de notre approche.

A l'issu du travail élaboré dans ce mémoire, et d'après les résultats obtenus, nous sommes dans la bonne voix avec la collection de test APP88. Notre perspective est de tester notre approche sur des collections de documents plus volumineuses pour une meilleure interprétation des résultats.

Références bibliographiques

[1] G. Salton and M. McGill. Introduction to modern information retrieval. McGraw-Hill Int. Book Co, 1984.

[2] Baeza-Yates & Ribeiro-Neto, Modern Information Retrieval, pearson education ltd., Harlow, UK, 2nd edn, 2011.

[3] N. Fuhr. Information retrieval. Vorlesung, SommerSemester 2005, 2005.

[4] Mizzaro. S. Relevance, the whole (hi) story. Journal of the American Sciety for Information Science, 48.pp. 810-832, 1997.

[5] Saracevic. T. Relevance reconsidered. Conceptions of Library and Information Science .pp.201-218. 1996.

[6] N. J. Belkin, R. Oddy, and H. Brooks. Ask for information retrieval: Part I background and theory. Journal of Documentation, 38(2) : pages 61–71, 1982.

[7] MATAOUI M'hamed Spécification de logiciels et traitement de l'information, « Reformulation de requêtes dans les systèmes de recherche d'information dans des documents XML », 2007.

[8] J. Maniez and E. de Grolier. A decade of research in classification, 1991

[9] J.-P. Balpe, A. Lelu, and I. Saleh. Hypertextes et hypermédias : Réalisations, outils et méthodes. Paris : Hermès, 1995

[10] C. Fox. Lexical analysis and stoplists, pages 102–130. Frakes WB, Baeza- Yates R (eds) Prentice Hall, New jersey, 1992.

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

[12] M. F. Porter. An algorithm for suffix stripping. Program 14, 1980.

[13] G. 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 :pages 253–60, 1974.

[14] A. Moffat and L. Stuiver. Exploiting clustering in inverted file compression. Data compression Conference, pages 82-91, 1996.

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

[16] Karen SAUVAGNAT, thèse Doctorat, Université Paul Sabatier . « Reformulation de Requêtes par Réinjection de Pertinence dans les documents Semi-Structurés ».2005.

[17] Salton, G., Fox, E., and Wu, H. Extended boolean information retrieval. Communications of the ACM, 26(11) :1022–1036. 1983.

[18] G. Salton. The SMART retrieval system : Experiments in automatic document processing. Prentice Hall. 1970.

Références bibliographiques

[19] 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.

[20] 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.

[21] Singhal, A., Salton, G., Mitra, M., Buckley, C. Document length normalization. *Information Processing and Management*, 32(5), pp. 619–633, 1996.

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

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

[24] M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. Journal of the Association for Computing Machinery, 7 :pages 216–244, 1960.

[25] C. J. VanRijsbergen. *Information retrieval.* London : Butterworths, 1979. 20, 29.

[26] S. Robertson, S. Walker, S. Jones, and M. H.-B. andM. Gatford. Okapi at TREC 3. In Proceedings of the 3rd Text REtrieval Conference (TREC-3), pages 109–126, 1994.

[27] S. Walker, S. Robertson, M. Boughanem, G. Jones, and K. S. Jones. Okapi at TREC-6 automatic and ad hoc, VLC, routing, filtering and QSDR. In Proceedings of TREC-6, pages 125–136, 1997.

[28] J. Ponte and W. Croft. A language modeling approach to information retrieval. In Proceedings of the 21st ACM conference on research and development in information retrieval (SIGIR 98), 1998.

[29] Jean-Martinet. Thèse de doctorat *« Un modèle vectoriel relationnel de recherche d'information adapté aux images »* Université Joseph Fourier – Grenoble I, 2004.

[30] Hammache Arezki. Thèse de doctorat « *Recherche d'information : un modèle de langue combinant mots simples et mots composés* » Université Mouloud Mammeri de Tizi-Ouzou, 2013.

[31] TREC (Text REtrieval Conference). http ://trec.nist.gov.

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

[33] Borlund, P. Measures of relative relevance and ranked hal□ife: performance indicators for interactive ir. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 24– 28, 1998.

[34] HLAOUA, thèse Doctorat l'Université de Paul Sabatier de Toulouse *« Reformulation de requêtes par Réinjection de Pertinence dans les Documents Semi Structurés* ». 2007.

[35] Gauch S., Smith J.B., An expert system for automatic query reformulation, Technical report, University of north california, 1992.

Références bibliographiques

[36] Manning, D., Raghavan, p. And Schute, H. Introduction to Information Retrieval. Cambridge University, Press, 2008.

[37] Y. Qiu and H.P. Frei. Concept based query expansion. In Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PAA, USA, pages 160-169, 1993.

[38] C. J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In Proceedings of the ACM-SIGIR Conference on Research and Development in Information Retrieval, Copenhage, Denmark, pages 77-88, 1992.

[39] R. Attar and A.S. Fraenkel. Local feedback in full-text retrieval systems. Journal of the ACM, 24(3) : pages 397-417, 1977.

[40] B. Croft and D. Harper. Using probabilistic models of information wi-thout relevance information. Journal of Documentation., 35(4) :285,295, 1979.

[41] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrie-val., pages 206-214. Melbourne, 1998.

[42] J.J. Rocchio. Relevance feedback in information retrieval. In The SMART retrieval system-experiments in automatic document processing, pages 313,323. Prentice Hall Inc, 1971.

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

[44] Kraft,R.and zien,J.2004. Mining anchor text for query refinement. In proceeding of the 13th international coference on word wid web. ACM Press, New York, NY ; USA, 666-674

[45] Macdonald,C.and ounis,I.2007.expertise drift and query expansion in expert search. In proceeding of the 16th conference on information and knowledge Management (CIKM 2007).ACM Press, Lisboa, Portugal.

[46] Harper, G. W. and van Rijsbergen, C. J. 1978. An Evaluation of Feedback in Document Retrieval Using Co-Occurrence Data. Journal of Documentation 34, 3, 189-216.

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

[48] Schutze, H. and Pedersen, 0. 1997. A Co-occurrence based Thesaurus and Two Applications to Information Retrieval. Information Processing and Management 33, 3, 307-318.

[49] 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.

Références bibliographiques

[50] Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association mies between sets of itemsin large databases. In ACM SIGMOD International Conference on Management of Data. ACMPress, Washington, D.C., USA, 207 216.

[51] Latiri, C. C., Yahia, S. B., Chevallet, J. P., and Jaoua, A. 2004. Query expansion using fuzzy association mies between terms. In Proceedings of JIM'2003. Metz , France.

[52] Song, M., Song, I.-Y., Hu, X., and Allen, R. B. 2007. Integration of association mies and ontologies for semantic query expansion. Data & Knowledge Engineering 63, 1, 63-75.

[53] Beeferman, D. and Berger, A. 2000. Agglomerative clustering of a search engine query log. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, Boston, MA, USA, 407-416.

[54] E ;Voorhees, (using wordnet to disambiguate word senses for text retrieval), processing of the annual conference on research and development in information retrieval, SIGIR 93, Pittsburgh, PA , 1993.

[55] 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.

[56] 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.

[57] Lin, J.and Murray,G.C.2005. assessing the term independence assumption in blind relevance feedback. In proceeding of the 28th annual international ACM SIGIR conference on research and development in information retreival. ACM Press, Salvador, Brazil, 635-636.

[58] Salton,G. and Buckley,C.1990. Improving retreival Performance by Relevance Feedback.journal of the american Society for information Sciencens 41,4,288-297.

[59] Harman, D.K.1992. relevance feedback and other query modificationtechniques. In information retreival- data Structures and algorithms, W.B.Frakes and R.Baeza-Yates, Eds. Pubprenticehall, Englewood Cliffs,N.J.,USA, 241-263.

[60] Amati, G.2003. probabilistic models for information retreival based on divergence from randomness. Ph.D. thesis, Departement of computing Science, University of Glasgow,UK.

[61] 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 environement information processing and Management 44,3,1086-1116.

[62] 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.

[63] Amati ,G.,Carpineto,C., and Romano,G. 2001 .FUB at trec-10 Web track :A probabilistic Framwork for topic relevance term Weighting. In proceeding of the 10thy text retrieval.

Références bibliographiques

[64] 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.

[65] Billerbeck, B. and Zobel, J.2004a. Questoning query expansion : an examination of behavior and parameters. In proceedings of the 15th Australasian database conference volume 27. Australian computer Society, Dunedin, New Zealand, 69-76.

[66] Cao, G.,J.,Nie, J.-Y, and Robertson, S.2008. Selecting good expansion terms for pseudo-relevance feedback. In proceedings of the 31st annual international ACM SIGIR conference on reseach and Development in Information Retrieval. ACM Press, Singapore, 243-250.

[67] Carpineto,C., Romano, G.,and Giannini,V.2002. Improving retrieval feedback with multipleterm-ranking function combination.ACM Transaction on information systems (TOIS) 20,3,259-290.

[68] Collins-Thompson, K. and callan, j.2007. Estimation and use of uncertainty in pseudo relevance feedback. In Proceeding of the 30th annual international ACM SIGIR conference on research and developpment in Information Retrieval.ACL Press,Amsterdam, The Netherlands, 303-310.

[69] Cronen-Townsend,S. and Xcroft,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

[70] Lavrenko,V. and Croft, W.B.2001. Relevance based language models. In Proceeding of the 24th Annual International ACM SIGIR Conference on research and Development in Information Retrieval. ACM Press, New Orleans, Louisiana, USA, 120-127

[71] Ounis I., Amati G., Plachouras V., He B., Macdonald C., Lioma C., "Terrier : A high Performance and Scalable Information Retrieval Platform", Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006), 2006.

[72] Middleton C., Baeza-yates R. "A Comparison of Open Source Search Engines", Technical report, October, 2007.

[73] JAVA2, le guide du développeur. Pierre-Yves Saumont. Antoine Mirecourt.

[74] http://www.netbeans.org.

[75] Sparck Jones, K., Van Rijsbergen, C.J. Progress in documentation. *Journal of Documentation*, Vol. 32, Num. 1, pp. 59-75, 1976.

[76] Robertson, S.E., Walker. S., Jones, S., Hancock-Beaulieu, M. and Gatford, M. Okapi at trec-3. *TREC*, pp. 109-126, 1994.