REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mouloud Mammeri de Tizi-Ouzou

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire

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

Thème

Implémentation et évaluation d'une méthode de sélection des documents pour l'expansion de requête sous la plateforme Terrier

Proposé et dirigé par :

Mr HAMMACHE AREZKI

Présenté par :

M^{elle} AIT HAMMI OUIZA

PROMOTION : 2013/2014

Remerciements

Mes vifs remerciements pour mon promoteur Mr HAMMACHE Arezki de m'avoir proposé ce thème, comme je tiens à le remercier pour sa gentillesse, sa disponibilité et son aide pendant la durée de ce travail.

Je tiens également à remercier l'ensemble des enseignants qui m'ont suivi durant mon cursus, sans oublier de remercier ma famille et mes amis qui m'ont toujours soutenu et encouragé.

Je tiens à exprimer ma gratitude aux membres du jury qui me font l'honneur d'examiner et de juger mon travail.

Mes remerciements vont également à toute personne ayant contribuée de près ou de loin à l'aboutissement de ce modeste travail.

Dédicaces

Je dédie ce modeste travail à :

Mes chers parents

Mes chères sœurs

Mon cher frère

Ma chère petite et adorable nièce Anaroze

Mes amis (es)

Qu'ils trouvent ici toute ma gratitude

Ouiza

Sommaire

Chapitre I : La recherche d'information

Chapitre II : La reformulation de requête

Chapitre III : Présentation de l'approche et Expérimentation

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

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

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

L'idée de l'approche proposée et implémentée est d'étendre le modèle de pertinence, en introduisant un facteur de classement a priori des documents pertinents retournés. Ce facteur est basé d'une part, sur la clarté du document vis-à-vis des documents pertinents et d'autre part, sur la taille de document.

Pour la réalisation et l'évaluation de nos expérimentations, nous utilisons la plateforme de RI Terrier (détaillée en annexe).

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

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

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

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

CHAPITRE I

Recherche d'information

I.1. Introduction

La croissance intense du volume de l'informationqu'il soit dans une base de données, dans une base documentaire, ou sur le web a engendré des problèmes d'accès à l'information. Pour résoudre ces problèmes, les informations ont besoin d'être stockées, organisées et indexées.De ce fait, des approches de RI, regroupant entre autres des techniques d'indexation ainsi que des mécanismes d'appariement ont été développés afin de mieux répondre aux besoins del'utilisateur.La recherche d'information est donc apparue comme réponse au besoin de gérer l'accès à cette masse d'information.

Dans ce chapitre, nous décrivons la recherche d'information en commençant par un bref historique de la RI et des Systèmes de Recherche d'Information(SRI), puis nous présentons les concepts de base de la RI, nous décrivons également son processus global connu sous le nom du processus en **U** en donnant l'utilité et l'importance de ses fonctions.
Enfin, nous présentons les mesures d'évaluation et les collections de test utilisées pour évaluer les SRI.

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

La recherche d'informationselon **Salton[1],**est l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre au besoin de l'utilisateur exprimé via une requête.
D'après **l'AFNOR[2],***« la RI est un ensemble de méthodes et procédures ayant pour objet d'extraire d'un ensemble de documents, les informations voulues. Dans un sens plus large, la RI est toute opération (ou ensemble d'opérations) ayant pour objet la recherche, la collecte et l'exploitation d'informations en réponse à une question sur un sujet précis »*

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

I.3. Définition d'un système de recherche d'information (SRI)

Les SRI sont nés de la nécessité d'automatiser la gestion de la recherche d'information.Un SRI est l'interface entre l'utilisateur qui exprime son besoin par une requête et une grande collection de documents. Il se charge du traitement de ces derniers pour retourner à l'utilisateur ceux qui correspondent le mieux à sa requête. Pour ce faire, il intègre un ensemble de fonctions qui sont : le stockage, l'organisation, la sélection d'information répondant aux besoins des utilisateurs et enfin la restitution des informations jugées pertinentes. Un SRI peut être également défini comme un logiciel qui a pour but de

sélectionner des informations pertinentes répondant aux besoins des utilisateurs, exprimés sous forme de requêtes**[3]**.

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

Un SRI a pour rôle de sélectionner à partir d'une collection, des documents qui peuvent intéresser l'utilisateur, c'est-à-dire ceux qui peuvent être pertinents à son besoin en information. Cette définition fait apparaître quatre notions clés qu'il convient de préciser: document, besoin en informationet pertinence.

I.4.1. Document

Un document constitue l'information élémentaire d'une collection de documents. L'information élémentaire appelée aussi granule de document, peut représenter tout ou une partie d'un document**[4].**

I.4.2. Besoin en information

Cette notion est souvent assimilée au besoin de l'utilisateur. Trois types de besoins utilisateur ont été définis**[4] :**

a) **Besoin vérificatif :** l'utilisateur cherche à vérifier le texte avec les données connues. Il recherche une donnée particulière, et sait souvent comment y accéder (par exemple chercher un document ayant une adresse connue sur le web). Un besoin de ce type est dit stable car il ne change pas au cours de la recherche.

b) **Besoin thématique connu :** l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et domaine connus. Un besoin de ce type peut être stable ou variable; il est possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche.

c) **Besoin thématique inconnu :** l'utilisateur cherche de nouveaux concepts ou de nouvelles relations hors des sujets et domaines qui lui sont familiers. Un besoin de ce type est intrinsèquement variable et est toujours exprimé de façon incomplète.

I.4.3. Requête

La requête est l'expression du besoin en information de l'utilisateur. Elle est l'interface entre le SRI et l'utilisateur.

Une requête est un ensemble de mots clés comme par exemple dans le système SMART**[5]** et Okapi**[6],** ou exprimée en langage naturel comme dans le système SMART**[5]** et SPIRIT[**7]**, booléen comme dans le système DIALOG**[8]**ou graphique comme dans le système issu du projet NEURODOC**[9].**

I.4.4. Pertinence

La pertinence est une notion fondamentale dans le domaine de la RI. Elle représente un critère majeur de l'évaluation des performances du système de RI**[10][11].***Elle peut être définie comme la correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête***[12].**Elle permet donc d'évaluer jusqu'à quel point les documents réstitués traitent le sujet de la requête.

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

Il existe deux types de pertinence :

a) **Pertinence système :** cette pertinence est déterministe, objective et elle est définie à travers les modèles de RI. Elle est souvent traduite par un score évaluant l'adéquation du contenu des documents vis-à-vis de celui de la requête**[13].**

   Selon Denos**[67]**, la pertinence système est l'ensemble des principes qui sous-tendent la fonction de correspondance dans un système de recherche d'information.

b) **Pertinence utilisateur :** cette pertinence quant à elle est liée à la perception de l'utilisateur sur l'information renvoyée par le système. Elle est subjective, deux utilisateurs peuvent juger différemment un même document renvoyé pour une même requête. Elle peut évoluer dans le temps d'une recherche. Par exemple, une information jugée non pertinente à l'instant *t* pour une requête peut être jugée pertinente à *t+1*car la connaissance de l'utilisateur sur le sujet a évolué**[14][15][16]**.

I.5. Processus de recherche d'information

Pour établir la mise en correspondance des besoins en information des utilisateurs d'une part et des informations contenues dans les fonds documentaires d'autre part, les systèmes de RI adoptent une démarche simple appelée processus en U de recherche d'information.Ce processus cherche alors à mettre en relation les besoins utilisateurs et les informations, afin que le système de recherche d'information ne retourne à l'utilisateur que les documents pertinents par rapport à son besoin**[17][18].**

La figure **I.1** illustre les principales tâches d'un SRI qui sont :

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

Les trois tâches citées précédemment seront détaillées dans les sections suivantes.

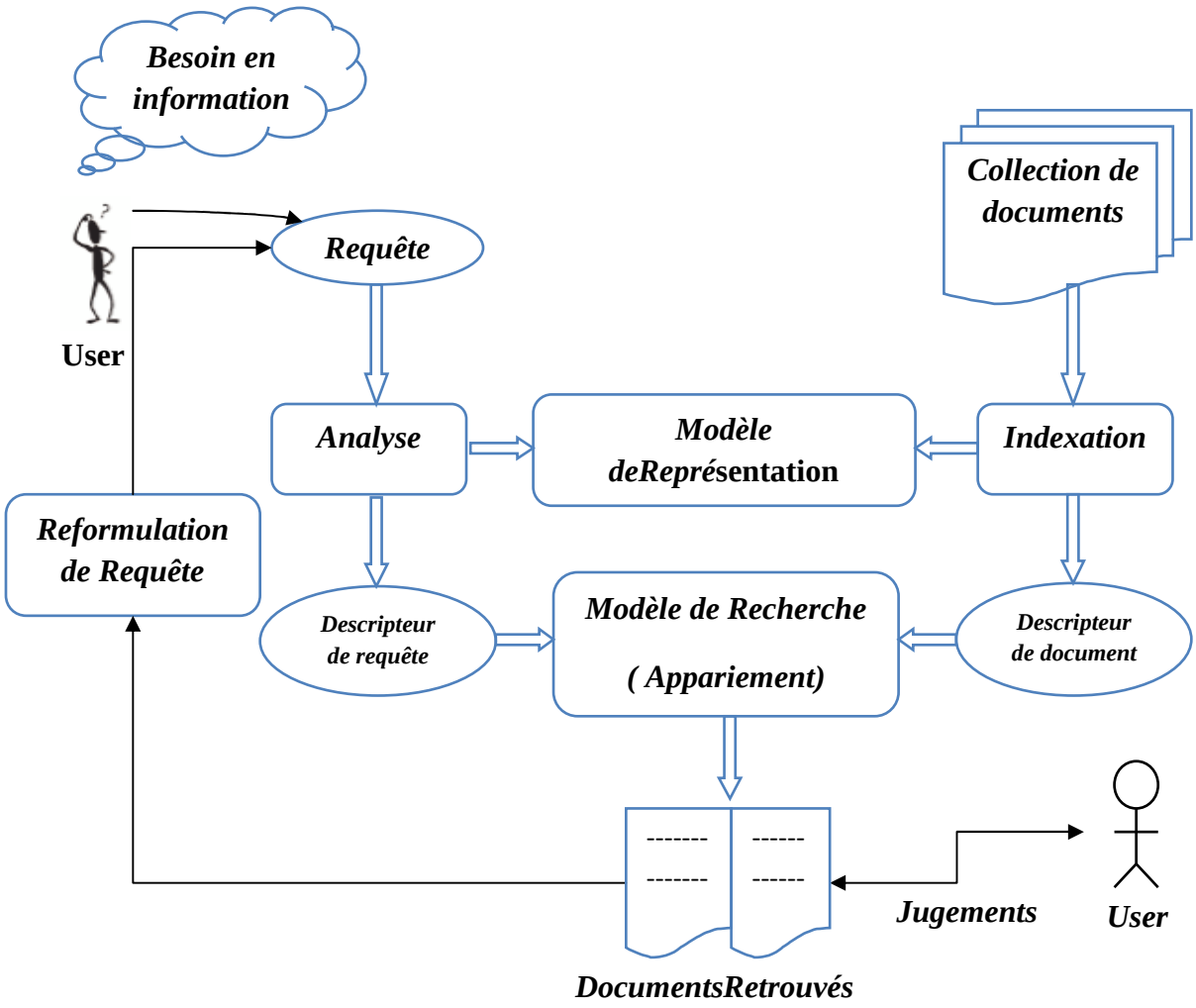


Figure I.1 : Processus en U de la recherche d'information**[17]**

I.5.1. Indexation des documents et des requêtes

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

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

Il existe trois types d'indexation :

I.5.1.1. Indexation manuelle

Cette indexation assure une meilleure pertinence puisqu'elle permet de repérer de façon plus précise les mots clés décrivant un document, mais elle est très coûteuse en terme de temps et subjective du fait qu'elle est fondée sur le jugement d'un être humain. En conséquence, pour un même document, des termes différents peuvent être affectés par des indexeurs différents et il peut même arriver qu'une personne, à des moments différents, indexe différemment le même document**[19].**

I.5.1.2. Indexation semi-automatique

Le processus d'indexation se fait en premier lieu d'une manière automatique, le documentaliste intervient seulement pour ajouter des mots clés qu'il trouve intéressants pour représenter un document**[19].**Dans ce cadre, les indexeurs utilisent un thésaurus ou une base terminologique, qui est une liste organisée de descripteurs (mots clés) obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques.

I.5.1.3. Indexation automatique

Ce type d'indexation est complètement automatisé, il a été mis pour pallier les inconvénients des systèmes manuels. Il comprend un ensemble de traitements automatisés sur les documents qui sont: l'extraction des mots simples, l'élimination des mots vides, la normalisation et enfin la création de l'index**[20]**.Ces étapes sont décrites comme suit:

a) **Extraction des termes :** cette opération consiste à extraire d'un document un ensemble de termes ou de mots simples par une analyse lexicale permettant d'identifier les termes en reconnaissant les espaces de séparation des mots, les caractères spéciaux, les chiffres, la ponctuation, etc**.**

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

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

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

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

c) **Normalisation (lemmatisation ou radicalisation) :** on peut trouver dans un texte, différentes formes d'un mot désignant le même sens. Pour cela, on élimine les différences non significatives et on garde la partie commune (radical, lemme), c'est-à-dire un seul mot suffirait pour représenter le concept et c'est lui seul qu'on va indexer.

Quelques stratégies de lemmatisation selon **Frakes et Baeza-Yates[8] :**

- **Elimination des affixes :** On peut par exemple citer l'algorithme de **porter[21]** qui n'est utilisable que pour la langue anglaise.
- **Troncature :** Cette méthode est utilisée pour la langue française, le moteur de recherche doit reconnaitre la similarité des mots présentés dans un document et dans la requête. Pour y arriver, les moteurs de recherche tronquent les mots.
- **Méthode des n-grammes :** C'est une représentation originale d'un texte en séquence de N caractères consécutifs, elle est très intéressante pour la radicalisation. Cette modélisation correspond au fait à un modèle de **Markov** d'ordre n. Prenons l'exemple suivant :Les tri-grammes au niveau de caractères de l'expression «recherche» sont : 'rec', 'her', 'che'**[22].**

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

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

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

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

Tableau 1.1. Exemple de collection (fichier maître)

- **Le fichier inverse :** Il est créé autour du fichier maître **[23][24].** Ce fichier comme son nom l'indique est le résultat de l'inversion du fichier maître. Plus exactement, au lieu de donner pour chaque document les mots qui le constituent et leurs positions, on donne pour chaque mot les documents qui le contiennent et sa position dans chacun.

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

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

Tableau 1.2. Exemple de fichier inverse

I.5.2. Appariement document-requête

L'objectif de tout SRI est de calculer la pertinence d'un document par rapport à une requête. Cette pertinence est basée sur la fonction d'appariement qui consiste à calculer un score représentatif ou la similarité entre chaque document et la requête de l'utilisateur. Ce score de similarité entre le document et la requête est donné par une fonction nommée Retrieval Status Value ; RSV (Q, D) où Q est une requête et D un document de la collection.

Un système de RI idéal ne renvoie que les documents pertinents à une requête, tous les documents non pertinents ne sont pas retournés ou doivent être retournés après les documents pertinents, c'est-à-dire la fonction d'appariement permet d'ordonner les documents selon leur degré de pertinence.

I.5.3. Reformulation de la requête

Parfois, les documents retournés suite à une requête peuvent être moins intéressants car l'utilisateur n'a pas su utiliser les mots précis pour exprimer son besoin en information.Afin de correspondre au mieuxla pertinence système de la pertinence utilisateur, la reformulation de requête est utilisée**[29].**Cette dernière consiste à modifier la requête initiale, en lui rajoutant des termes significatifs et/ou en réestimant leurs poids associés.Cette technique sera détaillée dans le chapitre suivant.

I.6. Modèles de recherche d'information

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

I.6.1. Modèlesensemblistes :

On distingue les deux modèles suivants :

a. Modèle booléen de base : ce modèle est basé sur la théorie des ensembles et l'algèbre de Boole**[30]**. Les documents et les requêtes sont représentés par des ensembles de mots clés pris de leur contenu. Dans ce modèle, la représentation de la requête doit utiliser l'un des opérateurs logiques$(et\ , ou\ \ et\ non\ )$. Chaque document D est représenté par un ensemble de termes non pondérés, l'index des documents sera constitué par la conjonction des termes t_i.

Exemple *:* $D = t_1 t_2 \dots t_n$.

Où : n : est le nombre de termes dans le document D.

La similarité entre un document ''D'' et une requête ''Q'' est définie par l'une des fonctions suivantes:

$$RSV(Q_i, D) = \begin{cases} 1 si t_i \in D \\ 0 sinon \end{cases}$$

$$RSV(Q_i Q_j, D) = \begin{cases} 1 si RSV(Q_i, D) = 1 et RSV(Q_j, D) = 1 \\ 0 sinon \end{cases}$$

$$RSV(Q_i \ Q_j, D) = \begin{cases} 1 si RSV(Q_i, D) = 1 ou RSV(Q_j, D) = 1 \\ 0 sinon \end{cases}$$

$$RSV(\neg Q_i, D) = \begin{cases} 1 si RSV(Q_i, D) = 0 \\ 0 sinon \end{cases}$$

Ce modèle présente l'avantage suivant :

- ✓ La simplicité de sa mise en œuvre.

Cependant, ilprésente quelques problèmes qui sont :

- ✓ Le système détermine un ensemble de documents non ordonnés comme réponse à une requête car la sélection d'un document est basée sur une décision binaire, c'est-à-dire la correspondance entre un document et une requête est soit 0 ou 1. On ne peut pas donc distinguer quel est le document le plus pertinent.
- ✓ Beaucoup d'utilisateurs trouvent des difficultés dans la formulation de la requête.
- ✓ Le nombre de documents retournés peut être considérable (collection volumineuse).
- ✓ Tous les termes ont la même importance.

Afin de remédier à ces problèmes, une extension du modèle booléen a été proposée: le modèle booléen étendu**[31][48].**

b. Modèle booléen étendu : ce modèle a été proposé par **Salton[31]**, son objectif est de tenir compte de la pondération des termes dans le corpus. Cela permet de résoudre les problèmes cités précédemment en ordonnant les documents retournés par le SRI selon leur similarité à la requête. Dans ce modèle, la requête demeure une expression booléenne classique, tandis que les termes d'un document sont maintenant pondérés.

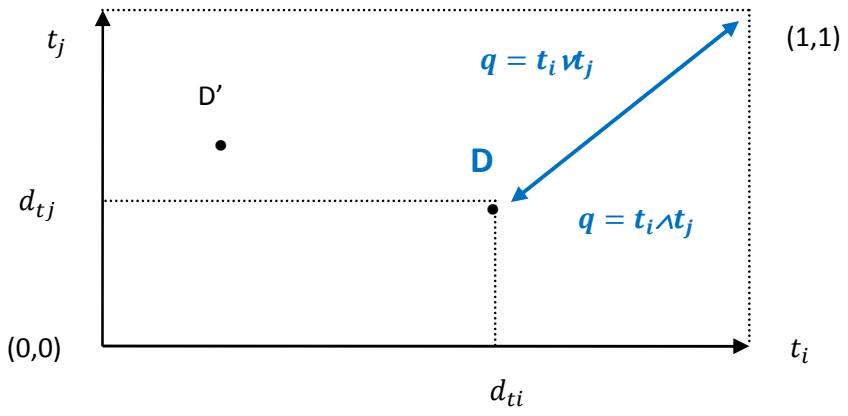


Figure I.2 : Présentation du modèle booléen étendu

Rappelons une des limites du modèle booléen, si $q = t_i \wedge t_j$ est une requête alors si un document contient t_i et ne contient pas t_j, ou si un document ne contient ni t_i ni t_j, ces deux documents sont traités de la même manière, c'est-à-dire ils ne sont pertinents ni l'un ni l'autre.
Avec une telle requête $q = t_i \wedge t_j$, on peut considérer que plus la distance est grande entre $D(d_{ti}, d_{tj})$ et le point de coordonnées (1,1) correspondant à la représentation de $t_i \wedge t_j$, q moins proche de D.
Pour une requête disjonctive binaire $q = t_i \vee t_j$, le point à éviter est le point (0,0), donc plus D est loin de (0,0) plus la requête est satisfaite par D.
Les mesures de similarité requête-document sont données comme suit :
Opérateur ou :

$$sim(D, d_{ti} \vee d_{tj}) = \sqrt{\frac{(d_{ti} - 0)^2 + (d_{tj} - 0)^2}{2}} \qquad (I.1)$$

Opérateur et :

$$sim(D, d_{ti} \wedge d_{tj}) = 1 - \sqrt{\frac{(1-d_{ti})^2 + (1-d_{tj})^2}{2}} \qquad (I.2)$$

Dans le cas de requêtes pondérées, les deux formules précédentes peuvent être étendues de la manière suivante :

$$sim(D, d_{ti} \vee d_{tj}) = \sqrt{\frac{q_{ti}^2 * d_{ti}^2 + q_{tj}^2 * d_{tj}^2}{q_{ti}^2 + q_{tj}^2}} \qquad (I.3)$$

$$sim(D, d_{ti} \wedge d_{tj}) = \sqrt{\frac{q_{ti}^2 * (1-d_{ti})^2 + q_{tj}^2 * (1-d_{tj})^2}{q_{ti}^2 + q_{tj}^2}} \qquad (I.4)$$

I.6.2. Modèle vectoriel

Ce modèle est introduit au début des années 70, par Gérard Salton et son équipe dans le système de recherche d'information SMART**[32]**. En se basant sur une intuition géométrique, le modèle vectoriel représente les documents sous la forme d'un vecteur de termes à t dimensions tel que t est le nombre de termes d'indexation définis par le système. Un document ''D_i'' d'une collection est représenté par :$D_i = (w_{di1}, w_{di2}, \dots, w_{dit})$avec$w_{dij}$: la pondération associée au terme d'indexation t_j dans le documentD_i.La requête est représentée suivant le même formalisme : $Q = (w_{q1}, w_{q2}, \dots, w_{qt})$ avec w_{qi}:poids du terme$t_i$dans la requête*Q*. Ce poids peut être soit une forme de *tf*idf*, soit un poids attribué manuellement par l'utilisateur.

Pour associer un poids à un terme, on peut procéder de différentes manières :

TF (Term Frequency) : elle se calcule par l'une des formules suivantes :

- ✓ $tf = n_{oc}$ $(I.5)$
- ✓ $tf = [0|1]$ $(I.6)$

Les deux formules précédentes expriment l'absence ou la présence d'un terme dans le document .

- ✓ $tf = \frac{n_{oc}}{tf_{max}}$ $(I.7)$
- ✓ $tf = 0.5 + 0.5\frac{n_{oc}}{tf_{max}}$ $(I.8)$

Où :

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

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

IDF (Inverse Document Frequency) : elle se calcule par l'une des formules suivantes :

- ✓ $idf = log\left(\frac{N}{n_i}\right)$ $(I.9)$
- ✓ $idf = log\left(\frac{N-n_i}{n_i}\right)$ $(I.10)$

Où :

N: représente le nombre total de documents dans la collection.

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

Une formule largement utilisée est la suivante :

$$tf * idf = \left(0{,}5 + 0{,}5\frac{n_{oc}}{tf_{max}}\right) * log\left(\frac{N}{n_i}\right) \qquad (I.11)$$

D'autres facteurs sont également considérés comme par exemple la longueur d'un document. En effet, la longueur d'un document doit être prise en compte, sinon les documents longs auront plus de chance d'être sélectionnés que les documents courts car dans les documents longs, les fréquences des termes sont plus élevées, et les similarités avec la requête seront également plus grandes.Pour pallier ce problème, **Robertson[27]** et **Singhal et al[28]** ont introduit la normalisation utilisée pour donner aux documents une chance d'être sélectionnés indépendamment de leurs longueurs.

Elle est donnée par la formule suivante :

$$tf * idf = \frac{tf + log\left(\frac{N-n_i}{n_i+0{,}5}\right)}{k * \left((1-b) + b * \frac{dl}{\Delta d}\right)} \qquad (I.12)$$

b : une constante appartenant à l'intervalle [0,1] et contrôle l'effet de la longueur du document (dans TREC, elle est fixée à 0,75).

k : contrôle l'influence de la fréquence du terme t_i, sa valeur optimale dépend de la longueur et de l'hétérogénéité des documents dans la collection (dans TREC, k=2).

dl : est la longueur du document en nombre de termes et Δd est la longueur moyenne des documents de la collection.

L'appariement s'effectue ainsi avec des fonctions de calcul de similarité entre vecteurs, tel qu'un calcul de distance.

La mesure de similarité peut être calculée par l'une des mesures illustrées dans le tableau ci-dessous :

Les mesures	**Les termes de l'espace vectoriel**	**Calcul de la similarite Sim (D_i , Q)**
Produit scalaire	$\lvert Q \cap D \rvert$	$\sum_{j=1}^{t} w_{dij} * w_{qj}$
Mesure de Jaccard	$\frac{\lvert Q \cap D \rvert}{\lvert Q \rvert + \lvert D \rvert - \lvert Q \cap D \rvert}$	$\frac{\sum_{j=1}^{t} w_{dij} * w_{qj}}{\sum_{j=1}^{t} w_{dij}^2 + \sum_{j=1}^{t} w_{qj}^2 - \sum_{j=1}^{t} w_{dij} * w_{qj}}$
Coefficient de Dice	$\frac{2.\lvert Q \cap D \rvert}{\lvert Q \rvert + \lvert D \rvert}$	$\frac{2 * \sum_{j=1}^{t} w_{dij} * w_{qj}}{\sum_{j=1}^{t} w_{dij}^2 + \sum_{j=1}^{t} w_{qj}^2}$
Mesure cosinus	$\frac{\lvert Q \cap D \rvert}{\sqrt{\lvert Q \rvert} . \sqrt{\lvert D \rvert}}$	$\frac{\sum_{j=1}^{t} w_{dij} * w_{qj}}{\sqrt{\sum_{j=1}^{t} (w_{dij})^2} * \sqrt{\sum_{j=1}^{t} (w_{qj})^2}}$

Tableau I.3. Mesures de similarité utilisées dans le modèle vectoriel**[68].**

La figure suivante montre une représentation d'une requête et de deux documents dans le modèle vectoriel :

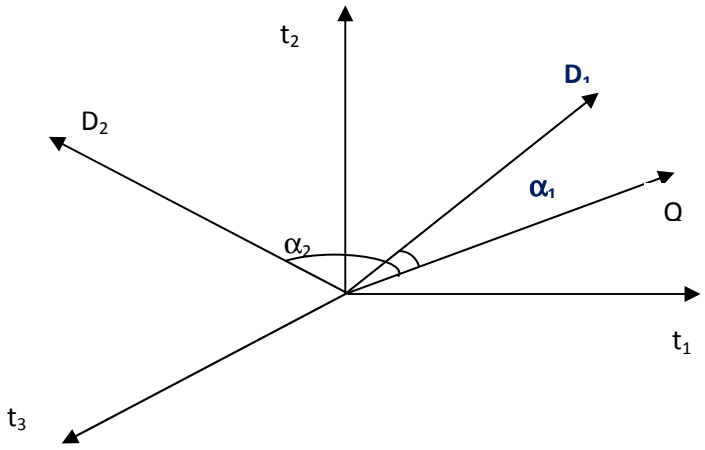


Figure I.3 : Illustration de la représentation d'une requête Q et de deux documents D_1 et D_2 avec le modèle vectoriel dans un espace à trois dimensions.

Remarque : Plus deux vecteurs sont similaires, plus l'angle formé est petit, et plus le cosinus de cet angle est grand. Dans l'exemple de la figure suivante, le document D_1 est plus similaire à la requête que le document D_2.

Le modèle vectoriel présente plusieurs avantages, entre autres :

- ✓ L'amélioration des résultats de recherche grâce à la pondération.
- ✓ Les documents sont ordonnés selon leur degré de pertinence par rapport à la requête utilisateur grâce à la similarité.

Cependant,il présente l'inconvénient suivant :

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

I.6.3. Modèles probabilistes

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

a. Modèle probabiliste de base : ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête**[32][33]**.

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

$$RSV(q,d) = \frac{P(Per|q,d_i)}{P(NPer|q,\ d_i)} \qquad (I.13)$$

L'idée de base de cette fonction est de retrouver les documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents à la requête.

Où : $P(Per|q,d_i)$ et $P(NPer|q,\ d_i)$: la probabilité qu'un document d_i soit pertinent (per)vis-à-vis de la requête q (respectivement non pertinent $(NPer)$).

Plus la valeur de $RSV(q,d)$ est importante, plus le document obtient un meilleur classement. En appliquant le théorème de Bayes pour les deux probabilités, on obtient :

$$P(Per|q,d_i) = \frac{P(Per|q).P(d_i|Per,q)}{P(d_i)} \qquad (I.14)$$

$$P(NPer|q,d_i) = \frac{P(NPer|q).P(d_i|NPer,q)}{P(d_i)} \qquad (I.15)$$

Où :

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

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

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

$P(Per|q)$ et $P(NPer|q)$ indiquent respectivement la probabilité de pertinence et de non pertinence d'un document quelconque (avec $P(Per|q)+P(NPer|q)=1$) qui sont fixes.

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

$$RSV(q,d)=\frac{P(d_i|Per,q)}{P(d_i|NPer,q)} \qquad (I.16)$$

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

$$P(d_i|Per,q)=\prod_{t_j\in d_i} P(t_j|Per,q)\times\prod_{t_j\notin d_i} 1-P(t_j|Per,q) \qquad (I.17)$$

$$P(d_i|NPer,q)=\prod_{t_j\in d_i} P(t_j|NPer,q)\times\prod_{t_j\notin d_i} 1-P(t_j|NPer,q) \qquad (I.18)$$

Où :

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

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

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

$$RSV(d_i,q)=\sum_{t_i\in q}\log\left[\frac{p_i(1-q_i)}{q_i(1-p_i)}\right] \qquad (I.19)$$

Afin de classer les documents avec cette formule, il faut estimer les valeurs des deux probabilités p_i et q_i. En l'absence de collection (documents) d'apprentissage; on peut attribuer la valeur fixe à p_i comme par exemple 0,5**[52]**, comme elles peuvent être estimées à l'aide de l'avis de l'utilisateur sur les résultats d'une première recherche (réinjection de pertinence).

Ce modèleprésente plusieurs avantages, entre autres :

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

Cependant, il présente les inconvénients suivants :

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

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

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

$$score(q,d) = \prod_{i=1}^{n} P(q_i|d) \qquad (I.20)$$

Avec : q_i : sont les termes de la requête.

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

Pour pallier ce problème, des techniques de lissage ont été proposées**[39]**, parmi elles, la technique de lissage de dirichlet (détaillée dans le chapitre 3).

I.7. Evaluation des performances d'un système de recherche d'information

Un système de recherche d'information (SRI) a pour objectif de retrouver tous les documents pertinents et rejeter tous ceux qui ne le sont pas.

Les critères d'évaluation reposant sur le temps de réponse et l'espace utilisé pour le stockage d'information semblent être plus ou moins importants lors de l'évaluation des SRI à cause de la difficulté de leur mise en œuvre, pour cela, d'autres mesures d'évaluation ont été adoptées. Les mesures les plus utilisées sont : la précision, le rappel, le silence et le bruit. Elles sont obtenues en partitionnant l'ensemble des documents restitués par le SRI en deux catégories qui sont : les documents pertinents et les documents non pertinents.

I.7.1. Mesures d'évaluation

La figure I.4 illustre la répartition des documents face à une requête :

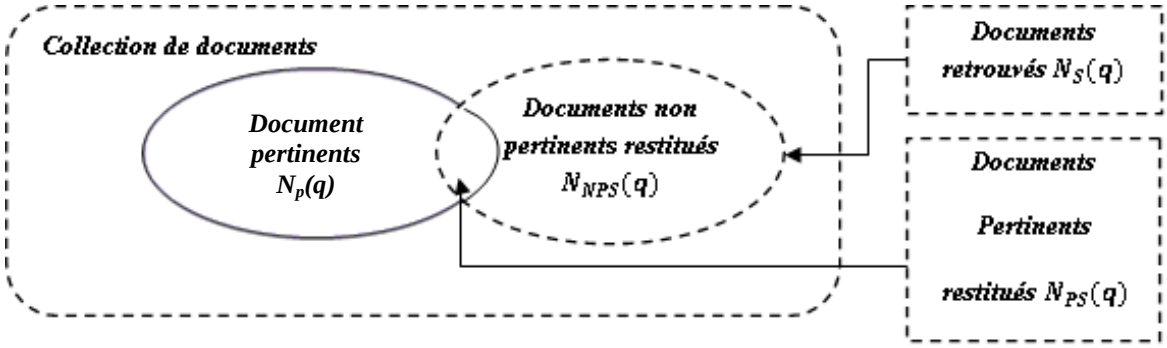


Figure I.4: Représentation des partitions de la collection lors d'une interrogation.

Avec :

$\boldsymbol{N_S(q)}$ **:** Le nombre de documents sélectionnés par le SRI pour une requête $\boldsymbol{q}$.

$\boldsymbol{N_P(q)}$**:** Le nombre de documents pertinents dans la collection pour la requête$\boldsymbol{q}$.

$\boldsymbol{N_{NPS}(q)}$**:** Le nombre de documents non pertinents sélectionnés dans la collection pour la requête $\boldsymbol{q}$.

$\boldsymbol{N_{PS}(q)}$ **:** Le nombre de documents pertinents sélectionnés dans la collection pour la requête $\boldsymbol{q}$.

- **Précision :** mesure la capacité du système à rejeter tous les documents non pertinents à une requête : $Pr(q) = \frac{N_{PS}(q)}{N_S(q)}$ $(I.21)$
- **Rappel :** mesure la capacité du système de retrouver tous les documents pertinents répondant à une requête : $R(q) = \frac{N_{PS}(q)}{N_p(q)}$ $(I.22)$
- **Bruit :** toute réponse non pertinente à une recherche documentaire**[35].** Le taux de bruit est le pourcentage exprimant le rapport entre le nombres de documents non pertinents extraits et le nombre total de documents extraits. Il mesure la difficulté du système à éviter les documents non pertinents : $B(q) = \frac{N_{NPS}(q)}{N_S(q)}$ $(I.23)$
- **Silence :** ensemble de documents pertinents qui ne sont pas repérés lors d'une recherche documentaire**[34].** Dans une recherche documentaire, il y a silence lorsque des documents pertinents répondant à une question et existant dans la mémoire ne sont pas sélectionnés à la suite de l'interrogation**[35].**Il mesure la difficulté du système à retrouver tous les documents pertinents : $S(q) = \frac{N_{NPS}(q)}{N_P(q)}$ $(I.24)$

- **Précision moyenne non-interpolée:** 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 $\left(pr(d_i)\right)$ 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**[38]**.

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

$$AP_q = \frac{1}{N}\sum_{i=1}^{N} per(d_i) \qquad (I.25)$$

  Avec :

$$pr(d_i) = \begin{cases} \frac{r_{n_i}}{n_i} & si\ d_i\ est\ retrouvé \\ 0 & sinon \end{cases} \qquad (I.26)$$

  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 des requêtes utilisées, elle est exprimée ainsi :

$$MAP = \frac{1}{M}\sum_{j=1}^{M} AP_{qj} \qquad (I.27)$$

  Où :

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

I.7.2. Collections de test

La collection ou corpus de test constitue le contexte d'évaluation, c'est-à-dire les éléments qui vont servir à tester le processus de sélection des documents par le SRI. Les corpus diffèrent par le nombre de documents et le nombre de requêtes, mais aussi sur leur domaine de spécialité, la façon de juger la pertinence, etc.

De nombreux projets basés sur les corpus de tests se multiplient depuis les années 1970, on peut citer la collection CACM, la collection CISI, la compagneCLEF(*Cross Language*

Evaluation Forum)ou encore la campagne INEX. La campagne la plus connue est sansconteste TREC (Text REtrieval Conference) organisée annuellement depuis1992 par le NIST (National Institute of Standards and Technology) et la DARPA (Defense Advanced Research Projects Agency).

Dans TREC, les recherches étaient centrées au départ (de TREC1 à TREC6)sur deux tâches principales : la tâche de routage et la tâche ad hoc. La tâchead hoc est constituée d'un ensemble de nouvelles requêtes qui sont lancéessur une collection de documents fixés, et la tâche de routage est composée d'unensemble de requêtes fixes lancées sur une collection de documents en évolution perpétuelle. Autour de ces tâches, bon nombre de pistes ont été explorées, et denouvelles tâches sont apparues. On peut par exemple citer des évaluations de recherche de documents écrits dans une autre langue que l'anglais (exemple : espagnol, français, ou encore chinois, arabe) (à partir de 1994), des évaluations de recherche à travers des langages multiples, des évaluations sur de très grandscorpus (tâche Terabyte en 2004), ou des évaluations portant sur des aspects plus diversifiés (la tâche interactive depuis 1994, la tâche QA (question-réponse) en 1999 etc.) ou encore des évaluations de recherche sur des vidéos (depuis 2001)ou des documents Web (depuis 1999).

Afin de mesurer l'efficacité des SRI de manière standard, il suffit de fournir une collection de test qui est généralement composée d'une collection de documents, aussi appelée corpus de documents, d'une collection de requêtes oucorpus de requêtes, et des jugements de pertinence des documents par rapport à ces requêtes.

I.7.2.1. Corpus de documents

C'est un ensemble de documents sur lesquels les SRI posent des requêtes et récupèrent les documents pertinents. Selon le **TLFI[36]**(Trésor de la Langue Française Informatisé), un corpus de documents est défini comme : "le recueil réunissant ou se proposant de réunir, en vue de leur étude scientifique, la totalité des documents disponiblesd'un genre donné, par exemple épigraphiques, littéraires, etc.".

Il existe de très nombreux ensembles de documents en accès libre, notamment sur le Web : des documents plus ou moins vulgarisés, plus ou moins spécialisés dans un domaine, dans une langue ou une autre, etc. Le choix d'une collection ou autre dépend de la tâche de recherche que l'on veut évaluer, pour garantir une représentativité par rapport à la tâche.De même que la spécification du volume des collections de documents utilisées dans l'évaluation est relativement dépendante de la tâche de recherche impliquée dans le SRI à évaluer, pour garantir une diversité des sujets et du vocabulaire. Les premiers corpus de test développés dans les années 1970 renferment quelques milliers de documents. Les corpus de

test plus récents (par exemple, ceux de TREC) contiennent en général plus de 100000 documents (considérés maintenant comme un corpus de taille moyenne), voir des millions de documents (corpus de grande taille).

I.7.2.2. Corpus de requêtes

Le corpus de requêtes, également appelé "topics", simule l'activité de recherche de l'utilisateur. Pour exploiter au mieux les caractéristiques de la collection de documents et avoir une évaluation assez objective, il est important de créer un ensemble de quelques dizaines de requêtes (TREC utilise de 25 à 50 topics comme une norme du nombre de requêtes de test), et qui soient adéquates par leur longueur, les thèmes abordés, leur forme, etc. Les requêtes sont généralement artificielles formulées par des assesseurs qui participent à la compagne d'évaluation, mais elles peuvent aussi être de vraies requêtes extraites à partir de log de recherche Web comme c'est le cas pour la tâche Web de Trec.

I.7.2.3. Jugements de pertinence

Pour la construction d'un corpus de test, les jugements de pertinence constituent la tâche la plus difficile. Les jugements de pertinence indiquent pour chaque document du corpus s'il est pertinent, et parfois même à quel degré il l'est, pour chaque requête. Pour établir ces listes de documents pour toutes les requêtes, les utilisateurs(ou des testeurs simulant des utilisateurs) doivent examiner chaque document de la base de documents, et juger s'il est pertinent par rapport à une requête donnée. Dans les programmes d'évaluation tels que TREC, les collections de documents contiennent plus d'un million de documents, ce qui rend impossible le jugement exhaustif de pertinence. Ainsi, dans le casde grandes collections, les jugements de pertinence sont construits selon la technique de pooling, effectuée à partir des 1000 premiers documents retrouvés par les systèmes participants.

I.8. Conclusion

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

Nous avons également présenté les métriques d'évaluation d'un SRI à savoir le rappel, la précision, le bruit, le silence et la précision moyenne non interpolée, ainsi que les collections de test.Dans le chapitre suivant, nous allons présenter les différentes techniques d'expansion de requête.

CHAPITRE II

Reformulation de la requête

II.1. Introduction

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

La reformulation de requêtes, est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur. Ellepermet de coordonner le langage de recherche, utilisé par l'utilisateur dans sa requête et le langage d'indexation**[37]**. Par conséquent, elle limite le bruit et le silence, dus à un mauvais choix des termes d'indexation dans l'expression de la requête.

II.2. Reformulation derequêtes

La reformulation de la requête consiste donc à ajouter de nouveaux termes à la requête initiale ou alors repondérer le poids de ses termes dans le but de cibler la recherche vers les documents pertinents**[29]**.

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

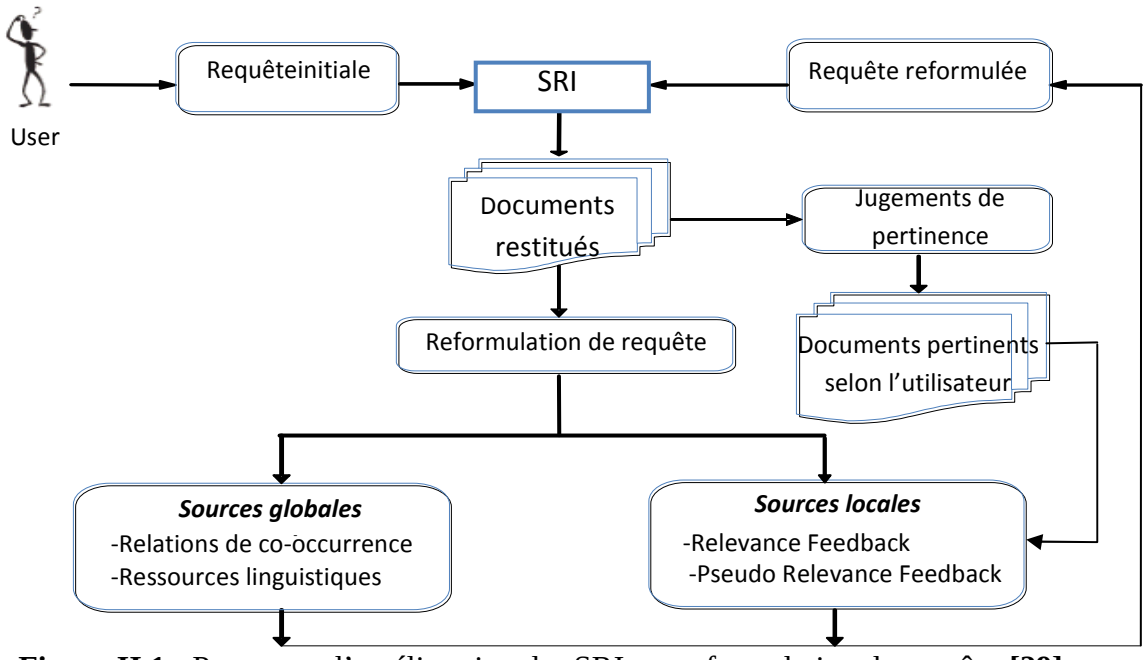


Figure II.1 : Processus d'amélioration des SRI par reformulation de requêtes**[29]**.

II.3. Types de reformulation de la requête

Le but essentiel d'un SRI est de permettre à l'utilisateur d'avoir un résultat satisfaisant par rapport à son besoin exprimé par une requête. La possibilité de reformuler la requête initiale s'avère intéressante dans le processus de la RI. Cela fera en sorte que le résultat retourné soit pertinent. Il existe trois méthodes de reformulation de requêtes :

II.3.1. Reformulation manuelle

C'est à l'utilisateur de sélectionner à partir des documents pertinents ceux dont lesquels va extraire les termes à rajouter à la requête initiale dans le but d'effectuer une nouvelle recherche**.**Cette approche est associée 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 pour compléter sa requête **[34]**.

II.3.2. Reformulation semi-automatique

Cette technique nécessite l'intervention de t'utilisateur qui doit identifier et sélectionner les documents pertinents et ceux qui ne le sont pas. Dans cette approche, ce sont donc le système et l'utilisateur qui sont responsables de la détermination et du choix des termes candidats à la reformulation. Le système joue un grand rôle dans la suggestion des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes. L'utilisateur examine cette liste et décide du choix des termes à ajouter dans la requête **[34]**.

II.3.3. Reformulation automatique (directe)

La reformulation directe de requêtes consiste à ajouter d'autres termes à ceux utilisés par l'utilisateur pour l'interrogation. Les nouveaux termes sont choisis automatiquement, sans l'intervention de l'utilisateur et doivent avoir des sens proches des termes donnés par l'utilisateur. Cette reformulation consiste à ajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Plus précisément, au niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. Ce mécanisme assure la restitution des documents indexés par des variantes des termes composant la requête. Les associations établies manuellement traduisent généralement des relations de synonymie et de hiérarchie. Les thésaurus construits manuellement sont un moyen efficace pour l'expansion de la requête.

En ce qui concerne la seconde catégorie de ressources (ressources construites à partir des collections), elles sont construites en s'appuyant sur une analyse statistique des collections. Il s'agit de chercher des associations de termes afin d'ajouter des termes voisins à la requête. Les associations créées automatiquement sont généralement basées sur la

cooccurrence des termes dans les documents. Les liens inter-termes renforcent la notion de pertinence des documents par rapport aux requêtes.

Le problème avec la reformulation automatique est l'estimation des « bons » termes qui peuvent conduire effectivement à une amélioration du processus de recherche car l'introduction des termes inappropriés peut entraîner un silence ou au contraire augmenter un bruit **[34].**

II.4. Techniques de reformulation de requête

Plusieurs techniques 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 requête a été traitée selon deux classes d'approches. La première, appelée la réinjection de la pertinence (relevance feedback)qui est basée sur les documents sélectionnés par le système (jugés par l'utilisateur). La seconde est basée sur l'utilisation des liens sémantiques ou statistiques établis entre les termes. Ces liens peuvent être construits manuellement par un expert (exemple: thésaurus « WordNet ») ou automatiquement. Dans ce dernier cas, ces liens peuvent être construits à partir des documents retrouvés par le système, on parle alors de reformulation par contexte local, ou à partir de la collection entière de documents, on parle alors de reformulation par contexte global**[41]**.

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

II.4.1. Approche basée sur le contexte global

a. Utilisation de ressources linguistiques : l'expansion directe de la requête consiste à rajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Plus précisément, au niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. Ce mécanisme assure la restitution des documents indexés par des variantes des termes composant la requête.

Il existe deux manières de construire un thésaurus : manuelle et automatique.

✓ **Manuelle :** le principe fondamental des thésaurus construits manuellement est d'ajouter à la requête initiale les termes voisins définis dans le thésaurus. En effet, le système présente le thésaurus à l'utilisateur et lui permet de naviguer parmi les termes. La polysémie y est traduite par la possibilité d'associer à chaque mot, n catégories différentes

représentant ses différents sens. Ce mode de construction est généralement adapté à des collections de petites tailles, à domaine spécifique**[42].**Cependant, les thésaurus manuels sont peu utilisés par les SRI car leur construction et la maintenance des informations sémantiques qu'ils contiennent sont coûteuses en temps et nécessitent le recours aux experts des domaines considérés**[42].**

Les thésaurus permettent d'améliorer le système au niveau de l'indexation et de l'interrogation, en précisant le contexte de la recherche, et en étendant la requête avec des termes considérés comme similaires.

Les types de relations traditionnellement définies dans les thésaurus sont :

- La généralisation ou hyperonyme.
- La spécialisation ou hyponyme.
- La synonymie, réelle ou approchée.
- La composition ou méronymie.

Ces relations seront détaillées dans ce qui suit.

Nous allons définir quelques thésaurus utilisés en recherche d'information :

❖ **Thésaurus basés sur la hiérarchie :** on peut citer WordNet comme exemple de thésaurus hiérarchique. Son avantage réside dans la diversité des informations qu'il contient (grande couverture de la langue anglaise, définition de chacun des sens, ensembles de synonymes, diverses relations sémantiques). En outre, WordNet est librement et gratuitement utilisable pour la recherche. Il couvre la majorité des *noms, verbes, adjectifs* et *adverbes* de la langue anglaise structurés en un réseau de nœuds et de liens. Chaque nœud, appelé **synset** (set of synonyms), est constitué d'un ensemble de termes synonymes. Cela signifie que les synonymes ayant le même sens sont groupés ensemble dans un nœud pour former un synset. Chaque synset représente un sens unique d'un mot particulier. Un terme peut être un mot simple ou une collocation (i.e. deux mots ou plusieurs mots reliés par des soulignés pour constituer un terme complexe correspondant).

Les synsets de WordNet sont reliés par des liens ou relations sémantiques. La relation de base entre les termes d'un même synset est la synonymie. Les synsets sont liés par des relations telles que spécifique-générique ou hyponyme-hyperonyme et la relation de composition meronymie-holonymie**[43].**

1) **La synonymie :** c'est le terme spécifique utilisé pour désigner deux mots qui sont interchangeables dans certains contextes linguistiques. Elle est notée RT (Related Term).

Dans ce cas, l'expansion de la requête initiale se fait à l'aide de l'ensemble des mots contenus dans le synset (sens choisit dans l'étape de désambigüisation), et aussi des mots contenus dans les sens synonymes de ce synset.

2) **L'hyperonymie :** c' est le terme générique utilisé pour désigner une classe englobant des instances de classes plus spécifiques. Elle est parfois notée BT (Broader Term), Y est un hyperonyme de X si X est un type de (kind of) Y. Par exemple, "fruit" est un hyperonyme de "pomme" et de "cerise".

   Dans ce cas, l'expansion de la requête initiale se fait à l'aide de l'ensemble des mots contenus dans le synset (sens choisit dans l'étape de désambiguïsation), et aussi des mots contenus dans les sens pères de ce synset.

3) **L'hyponymie :**c'est le terme spécifique utilisé pour désigner un membre d'une classe (Relation inverse de Hyperonymie). Elle est notée NT (Narrower Term). X est un hyponyme de Y si X est un type de (kind of) Y. Par exemple, "France" est hyponyme de "pays", "cheval" est hyponyme de "animal".

   Dans ce cas, l'expansion de la requête initiale se fait à l'aide de l'ensemble des mots contenus dans le synset (sens choisit dans l'étape de désambigüisation), et aussi des mots contenus dans les sens fils de ce synset.

4) **L'holonymie :**c'est le nom de la classe globale dont les noms meronymes font partie. Y est un holonyme de X si X est une partie de (is a part of) Y. Par exemple, "corps" est un holonyme de "bras", de même que "maison" est un holonyme de "toit".

5) **La méronymie :** représente la composition de concepts, comme les parties d'un objet. Le nom d'une partie constituante (part of), substance de (substance of) ou membre (member of) d'une autre classe (relation inverse de l'holonymie). X est un méronyme de Y si X est une partie de Y. Par exemple, "voiture" a pour meronymes "porte", "moteur".

✓ **Automatique :** la construction automatique de thésaurus est typiquement basée sur la cooccurrence des termes.

❖ **Thésaurus basés sur la similarité :** Les méthodes d'expansion de la requête basées sur un thésaurus de similarité consistent à rajouter à la requête des termes issus d'un thésaurus de similarité. Ceci revient à calculer des valeurs de similarité entre les termes de la requête et ceux d'un thésaurus donné. Les k meilleurs termes de similarité la plus élevée sont rajoutés à la requête initiale. Un thésaurus de similarité est une matrice de similarité terme-terme**[41]**. Pour le construire, au lieu de représenter un document par un vecteur de termes, on représente chaque terme t_i par un vecteur de documents dans un espace de

vecteurs de documents, par exemple, $\vec{t}i = (d_{i1}, \dots, d_{in})$. Avec : d_{ik} : signifie le poids du document d_k par rapport au terme t_i et n est le nombre de documents dans la collection.

La formule de pondération utilisée par Qui Frei**[41]** pour calculer le poids d_{ik} est la suivante :

$$d_{ik} = \frac{\left(0.5 + 0.5\frac{ff(d_k,t_i)}{\max ff(t_i)}\right) * iif(d_k)}{\sqrt{\sum_{j=1}^{n}\left(\left(0.5 + 0.5\frac{ff(d_j,t_i)}{\max ff(t_i)}\right) * iif(d_j)\right)^2}} \qquad (II.1)$$

Avec :

$ff(d_k, t_i)$: la fréquence du terme t_i dans le document d_k .

$iif(d_k) = \log(m/|d_k|)$: la fréquence inverse de d_k ; avec m le nombre de termes dans la collection et $|d_k|$ est le nombre de termes dans le document d_k.

$\max ff(t_i)$: la fréquence maximale du terme t_i dans toute la collection.

Le thésaurus de similarité est construit en calculant la similarité entre toutes les paires de termes (t_i,t_j) de l'indexation. La similarité entre deux termes est exprimée par le produit scalaire suivant :$Sim(t_i, t_j) = \vec{t_i}.\vec{t_j} = \sum_{k=1}^{n} d_{ik} * d_{jk}$ $(II.2)$

Grâce à ce thésaurus, l'expansion d'une requête Q consiste à calculer une similarité, notée $Sim_{qt}(Q,t_j)$, entre chaque terme du thésaurus et la requête Q.

La formule utilisée pour calculer cette similarité est la suivante :

$$Sim_{qt}(Q, t_j) = \sum\nolimits_{t_i \in Q} q_i * Sim(t_i, t_j) \qquad (II.3)$$

Où :

q_i : est le poids du terme t_i dans la requête Q.

Ainsi, tous les termes de la matrice du thésaurus de similarité, associés aux termes de la requête, peuvent être ordonnés par ordre décroissant de leur valeur de similarité Sim_{qt}.

Les poids des termes sélectionnés sont donnés par la formule de pondération suivante :

$$wq_i = \frac{Sim_{qt}(Q, t_j)}{\sum_{t_i \in Q} q_i} \qquad (II.4)$$

Le nombre de termes à rajouter à la requête est un paramètre important en recherche d'information**[41].**L'étude empirique effectuée sur 3 collections documentaires (MED, CACM, NPL), a montré que le nombre de termes à rajouter doit avoisiner les 100. Cependant, d'une collection à l'autre, le nombre optimal de termes à rajouter est très variable, ainsi les valeurs optimales pour MED, CACM et NPL sont respectivement 80, 100, 800**[44]**.

Ainsi, si le nombre de termes rajoutés à la requête est insuffisant, les performances sont médiocres. Inversement, si le nombre de termes rajoutés à la requête est excessif, les performances de la nouvelle requête peuvent être inférieures à la requête initiale.

b. Extension de requêtes par relation de cooccurrence[47] : contrairement aux techniques locales qui ne considèrent que des cooccurrences locales au document considéré, cette approche permet de prendre en compte les cooccurrences de mots dans le corpus de documents tout entier. La création des groupements de mots est basée sur l'hypothèse que deux mots cooccurents dans le même contexte sont sémantiquement similaires. Cette hypothèse s'interprète de la manière suivante : Les mots qui sont souvent utilisés ensemble dans un contexte ont une forte probabilité de synonymie c'est-à-dire, pour décrire un phénomène on utilise souvent dans un contexte local des synonymes relatifs au phénomène.

II.4.2. Approche basée sur le contexte local

La méthode d'analyse du contexte local a été développée par **Croft et Xu [49]**, utilisée dans leur système de recherche Inquery. A la différence avec les autres techniques de reformulation, elle utilise la règle de passage. Elle consiste à modifier la requête initiale de l'utilisateur à partir des proportions des contenus des meilleurs documents retrouvés.

Le principe de base de la méthode est décrit comme suit :

- Sélectionner n passages des n meilleurs documents retrouvés (un passage est une classe de termes de taille fixe, par exemple 300 termes).
- Extraire des concepts (groupes de mots) à partir de ces passages. Ces concepts sont ensuite ordonnés selon l'équation suivante :

$$O(Q,c) = \prod_{t_i \in Q} (\delta + \frac{\log(af_{c,t_i}) * idf_c}{\log(n)})^{idf_i} \qquad (II.5)$$

Où :

$$af_{c,t_i} = \sum_{j=1}^{n} ft_{ij} * fc_j \qquad (II.6)$$

idf_i : Max (1.0, log 10(N/N_i)/5.0)

idf_c : Max (1.0, log 10(N/N_c)/5.0)

ft_{ij} : Nombre d'occurrences du terme t_i dans le passage p_j.

fc_j : Nombre d'occurrences du concept c dans le passage p_j.

N : Nombre de passages dans la collection.

N_i : Nombre de passages contenant le terme t_i.

N_c : Nombre de passages contenant le concept c.

δ: Une constante (égale à 0,1) qui permet d'éviter les valeurs nulles.

- Les 70 meilleurs termes des concepts ordonnés sont utilisés dans l'expansion de la requête initiale.

a) La technique de réinjection de pertinence : la réinjection de pertinence, une des techniques de la reformulation de requêtes, connue aussi sous le nom de Relevance Feedback (RF), permet d'enrichir une requête initiale selon des informations extraites des éléments jugés pertinents par l'utilisateur **[37].**

Le processus de réinjection de pertinence, comporte principalement trois étapes : l'échantillonnage, l'extraction des évidences et la réécriture de la requête.

1. L'échantillonnage : est une étape qui permet de construire un échantillon de documents à partir des éléments jugés par l'utilisateur. Cet échantillon est caractérisé par le nombre d'éléments jugés non pertinents et le nombre d'éléments jugés pertinents.
2. L'extraction des évidences : est l'étape la plus importante, elle consiste en général à extraire les termes pertinents qui serviront à l'enrichissement de la requête initiale. Plusieurs approches ont été développées, la plus connue est celle de Rocchio**[40]** adaptée au modèle vectoriel.
3. La réécriture de la requête : consiste à construire une nouvelle requête en combinant la requête initiale avec les informations extraites dans l'étape précédente.

Le processus général de la réinjection de pertinence peut être renouvelé plusieurs fois pour une même séance de recherche : on parle alors de la réinjection de pertinence à itérations multiples**[37].**

Le schéma suivant illustre le processus de réinjection de pertinence.

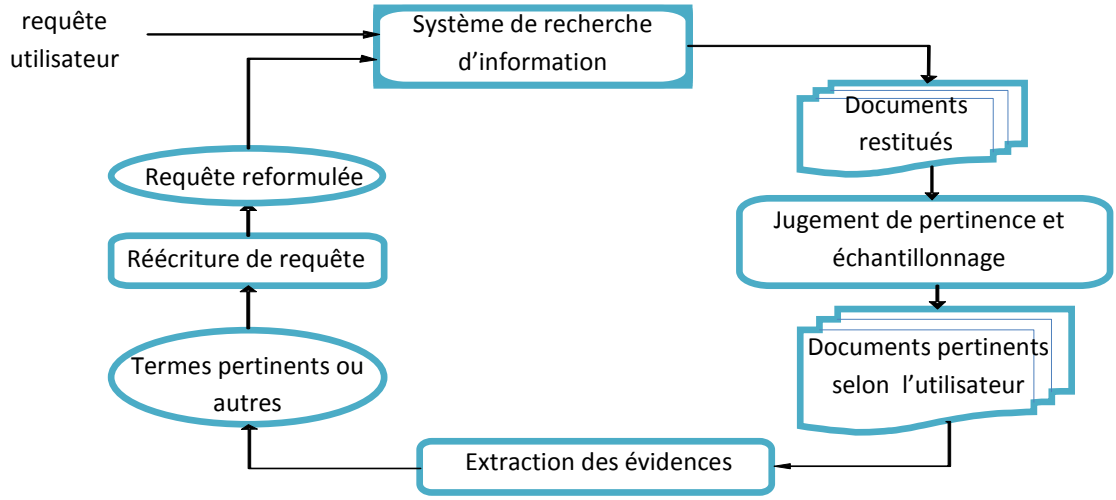


Figure II.2 : Processus général de la réinjection de pertinence**[37].**

D'une manière générale, la phase d'échantillonnage ne présente pas de problématique spécifique. Le seul point abordé à ce niveau concerne le nombre d'éléments à évaluer pour pouvoir effectivement constituer un échantillon représentatif**[37].**

La problématique principale de la réinjection de pertinence réside dans les deux autres phases : l'extraction des termes (ils sont alors pondérés pour sélectionner les éléments les plus pertinents) et la réécriture de la requête avec repondération des termes**[37],[40].**

Dans la plupart des approches de la littérature, les deux phases sont effectuées avec des méthodes de pondération des termes similaires. Cependant, certaines méthodes et particulièrement celles basées sur le modèle probabiliste, utilisent des méthodes de pondération différentes**[40].**

b) Réinjection de pertinence sans jugements de l'utilisateur : 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 ou blind RF 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 requête est générée pour produire une nouvelle liste triée de documents. Le principe de base du pseudo RF est qu'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**[51].**

Selon Croft et Harper**[52]**, cette technique souffre d'un problème majeur qui est appelé « query drift ». Ce problème apparait l'ensemble des documents pertinents (ou ne contient pas dutout). Donc cette technique ne fonctionne bien que dans le cas où la requête initiale permet d'extraire de bons résultats (beaucoup de documents pertinents)**.**

Le schéma suivant explique la procédure de réinjection automatique :

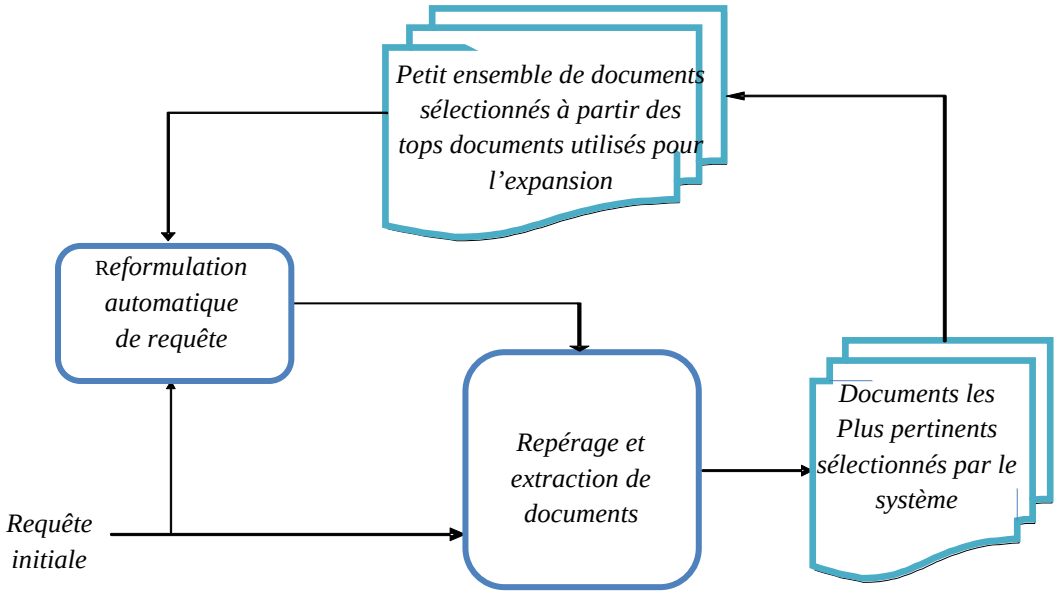


Figure II.3 : Processus de fonctionnement du blind RF

II.5. Reformulation de la requête dans les modèles de RI classique

Il est souvent difficile pour l'utilisateur, de formuler exactement son besoin en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent parfois pas. Retrouver des informations pertinentes en utilisant la seule requête initiale de l'utilisateur est toujours difficile, et ce à cause de l'imprécision de la requête. Afin de faire correspondre au mieux la pertinence de l'utilisateur à la pertinence du système, une étape de reformulation de la requête est souvent utilisée. La requête initiale est traitée comme un essai pour retrouver de l'information désirée ou ciblée. Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'espoir de retrouver des documents plus pertinents. La reformulation de la requête se fait en deux étapes principales : trouver les termes d'extension à la requête initiale, et pondérer les termes dans la nouvelle requête.

II.5.1. Reformulation dans le modèle vectoriel

Les stratégies de reformulation développées dans le modèle vectoriel induisent une repondération de requête avec expansion. La méthode de Rocchio **[41]** est largement utilisée en recherche d'information. Rocchio considère que la restitution des documents pertinents est liée à la notion de « requête optimale ». Cette dernière a pour objectif de maximiser la différence entre le vecteur des documents pertinents et celui des documents non pertinents. Comme l'utilisateur n'est pas en mesure de soumettre une requête optimale, la reformulation doit permettre de rapprocher le vecteur de la requête initiale du vecteur moyen des documents pertinents et de l'éloigner du vecteur moyen des documents non pertinents. Ceci est mis en

œuvre par repondération des termes initiaux et ajout de nouveaux termes pondérés à la requête initiale. La forme standard de l'algorithme de Rocchio est donnée comme suit :

$$Q_{nlle} = \alpha Q_{init} + \frac{\beta}{|D_r|}\sum_{D_j \in D_r} D_j - \frac{\gamma}{|D_n|}\sum_{D_j \in D_n} D_j \qquad (II.7)$$

Où :

| . | : Désigne le cardinal d'un ensemble.

Q_{nlle} : Le vecteur de la nouvelle requête.

Q_{init} : Le vecteur de la requête initiale.

D_r : L'ensemble des documents restitués et jugés pertinents par l'utilisateur.

D_n : L'ensemble des documents restitués et jugés non pertinents par l'utilisateur.

D_j : Le $j^{ème}$ document d'un ensemble.

α, β et γ : sont des constantes.

Le nouveau vecteur de requête est le vecteur de la requête initiale, plus les termes qui différencient au mieux les documents pertinents des documents non pertinents. Une requête reformulée contient les termes originaux et les nouveaux termes (extraits des documents jugés pertinents), associés à de nouveaux poids. Si le poids d'un terme de la requête décroit vers zéro ou au-dessous de zéro, il sera éliminé de l'ensemble des termes de la requête.

II.5.2. Reformulation dans le modèle probabiliste

Dans le modèle probabiliste, Robertson & al **[53]**, Harman **[54]** et Haines & Croft**[55]** ont développé des formules de pondération de requête en utilisant le jugement de l'utilisateur sur la pertinence des documents restitués par le système. Robertson **[53]** calcule la similitude initiale document-requête selon la formule suivante :

$$Sim(Q_k, D_j) = \sum_{i=1}^{r} q_{ki} * d_{ji} * \log\frac{p_i(1-U_i)}{U_i(1-p_i)} + C \qquad (II.8)$$

Où :

$$p_i = p(d_{ji} = 1/Destpertinent)$$

$U_i = p(d_{ji} = 1/Destnonpertinent)$

d_{ji} : Poids du terme t_i dans le document d_j avec :$d_{ji} = 1$Si le terme i occurre dans le document d_j, 0 sinon.

q_{ki} : Poids du terme t_i dans la requête q_k.

$p_{init} = 0$

$U_{init} = \frac{n_i}{N}$

C : Une constante.

n_i : Le nombre de documents de la collection contenant le termet_i.

N : Le nombre total de documents dans la collection.

Les recherches ultérieures exploitent l'occurrence des termes dans les documents jugés pertinents et documents jugés non pertinents. Une liste de termes candidats à l'expansion de requête, sont triés selon une valeur de sélection donnée par la formule:

$$VS(t_i) = \log \frac{p_i * r_i(1 - q_i)}{q_i * R(1 - p_i)} \qquad (II.9)$$

Où :

p_i : Poids du terme t_i dans le document jugé pertinent.

q_i : Poids du terme t_i dans la requête q.

R : Nombre de documents jugés pertinents.

Avec :

$$p_i = \frac{r_i}{R}$$

$$U_i = \frac{n_i - r_i}{NR - R}$$

r_i : Nombre de documents jugés pertinents, contenant le terme candidat t_i.

NR : Nombre de documents non pertinents retrouvés.

La fonction de similitude utilisée lors des itérations feedback devient alors la suivante :

$$Sim(Q_k, D_j) = \sum_{i=1}^{r} q_{ki} * \log\left(\left(\frac{r_i}{R - r_i}\right) + \frac{(n_i - r_i)}{(N - NR - n_i + r_i)}\right) \qquad (II.10)$$

II.6. Reformulation dans le modèle de langue

La reformulation dans le modèle de langue part d'un principe différent des approches traditionnelles de RI; on ne tente pas de modéliser directement la notion de pertinence, mais on considère que la pertinence d'un document face à une requête est en rapport avec la probabilité que la requête puisse être générée par le modèle de langue d'un document. On suppose en effet, qu'à chaque document est associé un modèle de langue, soit M_d, le score de pertinence du document vis-à-vis d'une requête q est déterminé par la probabilité de génération de la requête sachant le modèle de ce document, soit $p(q|M_d)$.

La plupart des modèles de langue développés pour la RI se base sur ce principe de génération de la requête par un modèle de document, néanmoins, il existe d'autres variantes, qui sont discutées dans ce qui suit.

II.6.1. Approches d'exploitation des modèles de langue en RI

En se basant sur la représentation des documents et la fonction de ‘’Ranking’’, les approches de modélisation de langue pour la RI peuvent être classées en trois catégories :

1) Génération de la requête par le modèle de document (Query Likelihood Models) : cette catégorie correspond à ce que nous avons expliqué ci-dessus. Un modèle de langue est associé à chaque document. Les documents sont alors classés selon leurs probabilités de génération de la requête, soit $(q|M_d)$.
2) Génération de document par le modèle de la requête (Document Likelihood Models) : cette approche procède dans le sens inverse. Ainsi, un modèle de langue est associé à la requête, les documents sont alors classés selon leurs probabilités que leur contenu soit généré par le modèle de la requête, soit $p(d|M_q)$.
3) Similarité document-requête : dans cette catégorie, on considère qu'à chaque document et à chaque requête est associé un modèle de langue. Les documents sont alors ordonnés selon la similarité de leurs modèles avec celui de la requête.

II.6.1.1. Génération de la requête par le modèle de document (Query Likelihood Models)

Ponte et Croft **[56]** furent les premiers à proposer un modèle de langue pour la RI. L'intuition derrière leur modèle est que l'utilisateur est supposé avoir une idée des termes qui sont présents dans le document pertinent ou modèle de ce document M_d ; à partir de là, l'utilisateur génère la requête en utilisant ces termes. Ainsi, la requête q est censée fournir des indices (des termes), associés au modèle du document (les documents recherchés).

On cherche alors à estimer la probabilité que la requête provienne du modèle ayant généré ce document, soit :

$$score(q,d) = p(q|M_d) \qquad (II.11)$$

Ce modèle utilisant une distribution de Bernoulli, c'est-à-dire que non seulement les mots présents dans la requête, mais aussi ceux absents de la requête sont pris en compte. Le score du document vis-à-vis de la requête est alors exprimé ainsi :

$$score(q,d) = \prod_{t\in q} p(t|M_d) * \prod_{t\notin q} (1 - p(t|M_d)) \qquad (II.12)$$

Ce score est composé de deux parties : la probabilité d'observer les termes de la requête dans le document et la probabilité de ne pas observer les termes absents de la requête dans le document.

La probabilité $p(t|M_d)$est calculée par une méthode non paramétrique qui utilise la probabilité moyenne d'apparition du terme t dans les documents qui le contiennent $P_{avg}(t)$ et un facteur de risque pour un terme observé dans le document$R(t,d)$. Par contre, la probabilité d'un terme dans la collection est utilisée pour les termes qui n'apparaissent pas dans le document. Le calcul de cette probabilité est exprimé ainsi :

$$p(t|M_d) = \begin{cases} P_{ML}(t|d)^{(1-R(t,d))} * P_{Avg}(t)^{R(t,d)} & si \quad tf(t,d) > 0 \\ \dfrac{tf(t,C)}{|C|} & sinon \end{cases} \qquad (II.13)$$

Hiemstra **[45]** a proposé un modèle où la requête est considérée comme une séquence de termes,$q = (t_1, t_2, \dots, t_n)$. Dans ce modèle, on ne considère que les mots présents dans la requête, une distribution multinomiale est utilisée. Le score d'un document vis-à-vis d'une requête (la probabilité de générer une requête q sachant le modèle de document M_d) est donné par la formule suivante :

$$score(q,d) = p(d) \prod_{i=1}^{n} p(t_i|M_d) \qquad (II.14)$$

Où : $p(d)$ est la probabilité a priori d'un document.

Pour l'estimation de la probabilité $p(t_i|M_d)$, Hiemstra utilise l'approche par interpolation qui combine le modèle de documentM_d avec le modèle de langue de la collection. Le calcul de cette probabilité est exprimé ainsi :

$$P(t_i|M_d) = \lambda P_{ML}(t_i|M_d) + (1-\lambda) P_{ML}(t_i|C) \qquad (II.15)$$

Où :

λ: est le poids d'interpolation qui varie entre 0 et 1. Ce paramètre peut être considéré constant ou il peut être estimé d'une manière sophistiquée en utilisant un processus d'optimisation automatique tel que l'algorithme de maximisation d'espérance (EM).

Les modèles $P_{ML}(t_i|M_d)$ et $P_{ML}(t_i|C)$ sont estimés en se basant sur la vraisemblance maximale, exprimés ainsi :

$$P_{ML}(t_i|M_d) = \frac{tf(t_i,d)}{|d|} \qquad (II.16)$$

$$P_{ML}(t_i|C) = \frac{df(t_i)}{\sum_{t_j \in v} df(t_j)} \qquad (II.17)$$

Tels que :

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

$df(t_i)$: est le nombre de documents contenant le terme t_i.

$|d|$: est la taille du document.

v : est le vocabulaire d'index.

II.6.1.2. Génération de documentà partirdu modèle de la requête (Document Likelihood Models)

Au lieu de modéliser la RI comme processus de génération de la requête, Lavrenko et Croft **[57]** ont proposé de modéliser explicitement le modèle de pertinence. Ils ont en effet, proposé d'estimer ce modèle à partir du modèle de la requête sans utiliser les données d'entrainement, en faisant le parallèle avec la modélisation de la pertinence proposée dans le modèle probabiliste classique. Ils considèrent en effet, que pour chaque requête, il existe un modèle permettant de générer le sujet (thème) abordé par la requête, c'est ce que les auteurs appellent le modèle de pertinence(θ_R).

Le but est alors d'estimer la probabilité $p(t|\theta_R)$, de générer un terme à partir du modèle de pertinence. Comme le modèle de pertinence n'est pas connu, les auteurs ont suggéré d'exploiter les documents retournés les mieux classés (feedback documents) en assumant qu'ils sont générés du modèle de pertinence.

Ce modèle est formalisé comme suit :

$$p(t|\theta_R) = \sum_{d \in R} P(d)P(t|d) \prod_{i=1}^{k} P(q_i|d) \qquad (II.18)$$

Avec :

R : L'ensemble des tops documents pertinents.

$p(d)$: La probabilité de choisir un document d des tops documents pertinents retournés.

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

Les résultats des expérimentations ont montré que cette approche améliore sensiblement les performances de la recherche d'information, de +10% à +29% d'amélioration de précision moyenne par rapport au modèle de langue de base**[38]**.

II.6.1.3. Comparaison des modèles de requête et du document

Cette approche s'est inspirée de l'approche vectorielle de la RI, dans laquelle la requête et le document sont représentés sous forme de vecteurs, la pertinence est alors interprétée par le similarité des vecteurs associés à la requête et au document.

Lafferty et Zhai **[69]** ont proposé un cadre de minimisation de risque basé sur la théorie de décision bayésienne. Dans ce cadre, la requête et les documents sont modélisés par des modèles statistiques. Le modèle de la requête modélise entre autres les préférences et les

besoins des utilisateurs, le modèle du document permet de modéliser le processus de génération de document et de capturer les préférences de l'auteur.

La similarité documents et requête est mesurée par la fonction de divergence Kullback-Leibler (KL) entre le modèle de la requête $(p(t|q))$ et celui du document $(p(t|d))$**[69]**, exprimée comme suit :

$$score(q,d) = -KL(q||d) = -\sum_{t \in V} P(t|q).\log\frac{P(t|q)}{P(t|d)} \qquad (II.19)$$

Où : V est le vocabulaire d'index.

Cette approche peut être vue comme la combinaison de certains aspects des deux approches précédentes. Les expérimentations menées par les auteurs sur des collection TREC ont montré l'utilité de cette méthode.

II.7. Facteurs influant sur les performances de la reformulation

Plusieurs expérimentations ont été effectuées sur les collections de documents, pour l'évaluation de l'impact induit par la reformulation de requête, sur le processus de recherche d'information. Les auteurs révèlent que son apport en performance est en fonction d'un nombre considérable de paramètres **[62]**. Ces paramètres sont très dépendants de :

- La structure modélisant le système.
- La stratégie adoptée pour l'expansion de requête.
- Caractéristiques des collections utilisées pour l'expérimentation : taille des documents, nombre de termes d'indexation, longueur moyenne de requête, etc.

II.7.1. Nombre de termes d'expansion

L'ajout de termes à la requête accroît la performance du SRI. Buckley et al **[63]**ont expérimenté le retour de pertinence dans l'environnement multi-fond documentaire TREC ; ils ont montré que le taux de performance (rappel-précision) est davantage corrélé avec le nombre de termes ajoutés qu'avec le nombre de documents initialement retrouvés.

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

$$RP(N) = A\log(N) + B\log(X) + C \qquad (II.20)$$

Où :

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

N : nombre de documents restitués.

X : nombre de termes ajoutés à la requête.

A, B, C : sont des constantes.

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

- Moins important aux termes ajoutés **[55].**
- Plus important aux termes issus des documents pertinents que ceux issus des documents non pertinents **[50].**

II.7.2. Choix des termes d'expansion

La reformulation de requête telle qu'elle a été initialement utilisée par Wu et Salton **[64]** consistait à ajouter tous les termes des documents pertinents retrouvés en réponse à la requête lors du processus de recherche. Cette méthode de sélection des termes peut avoir à l'origine beaucoup de bruit (restitution de documents non pertinents). En effet, les termes dans les premiers documents pertinents restitués ne sont pas tous significatifs. L'idée d'utiliser seulement une sélection de termes a été proposée par Harman **[37]**. La question est de savoir quels termes utilisés pour étendre la requête initiale de façon à améliorer le rappel et la précision du système.

✓ **Approche de Harman :** Elle consiste à sélectionner les dix premiers documents et à identifier parmi ces derniers ceux pertinents. Harman a utilisé différentes techniques pour ordonner les termes, afin de choisir les vingt meilleurs termes de la liste. Il a démontré la technique utilisée pour le tri des termes pertinents à un large impact sur la performance. Dans plusieurs techniques de tri, il utilise une mesure de bruit n_k calculée comme suit **[37]:**

$$n_k = \sum_{i=1}^{N} \frac{tf_{ik}}{f_k} \log_2 \frac{f_k}{tf_{ik}} \qquad (II.21)$$

Avec :

tf_{ik} : Le nombre d'apparition du terme k dans le document i

f_k : Le nombre d'apparition du terme k dans la collection.

N : Nombre de documents dans la collection.

La technique a été étendue pour tenir compte du nombre de documents dans l'ensemble des documents pertinents contenant le terme k (p_k) et du nombre d'apparition du terme k dans l'ensemble des documents pertinents (rtf_k).

Harman a défini ainsi une autre mesure de bruit par rapport à l'ensemble des documents pertinents. Cette mesure est calculée comme suit **[37]** :

$$rn_k = \sum_{i=1}^{N} p_k \frac{tf_{ik}}{rtf_k} \log_2 \frac{f_k}{tf_{ik}} \qquad (II.22)$$

Harman a défini d'autres techniques de tri des termes. La technique qui conduit à de meilleurs résultats est basée sur une formule de pondération définie par Sparck-Jones et Robertson **[37]** :

$$w_{ij} = \log_2 \frac{p_{ij}(1 - q_{ij})}{q_{ij}(1 - p_{ij})} \qquad (II.23)$$

Avec :

w_{ij} : Poids du terme i dans la requête j.

p_{ij} : La probabilité que le terme i apparaisse dans les documents pertinents pour la requête j.

q_{ij} : La probabilité que le terme i apparaisse dans les documents non pertinents pour la requête j.

La sélection des termes ayant une valeur de poids importante revient à sélectionner les termes caractéristiques des documents pertinents avec une faible probabilité d'apparition dans les documents non pertinents.

Harman a également démontré que la meilleure méthode de sélection des termes issus des documents pertinents devient inefficace au-delà de 20 à 40 termes ajoutés **[37]**.

✓ **Approche de Robertson :** Croft & al et Robertson & al**[37]** ont adopté une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer à chaque terme un nombre traduisant sa valeur de pertinence. Robertson propose la formule suivante pour calculer la valeur de sélection d'un terme :

$$selValue(i) = w_{ij} * (p_i - U_i) \qquad (II.24)$$

Avec :

w_{ij} : Poids du terme i dans la requête j

p_i : La probabilité $(di = 1/D\ estpertinent)$

U_i : La probabilité $(di = 0/Destnonpertinent)$

Les termes sont alors triés en fonction de leur valeur de pertinence, puis sélectionnés en utilisant un seuil prédéfini.

✓ **Approche de Lundquist :** Lundquist & al ont étudié dans une autre technique de tri des termes, l'association d'une valeur pour un terme k. Cette valeur est donnée par: $p_k * nidf \qquad (II.25)$

Où :

p_k : Est le nombre de documents dans l'ensemble des documents pertinents contenant le terme$k$, et $nidf$ est une fréquence absolue inverse normalisée.

En utilisant la collection TIPSTER, Lundquist & al ont démontré que cette formule conduit à de bonnes performances **[37]**. Par ailleurs, ils ont aussi démontré que l'utilisation des dix premiers termes (termes simples ou expressions) conduit à une amélioration de la précision moyenne de 31% par rapport à l'utilisation des cinquante premiers termes et vingt premières expressions.

✓ **Approche de Boughanem :** Boughanem & al ont quant à eux étudié la reformulation de la requête sur un SRI basé sur l'approche connexionniste fondée sur les réseaux de neurones. Les termes ajoutés à la requête sont sélectionnés sur la base d'un seuil de cooccurrence avec les termes de la requête initiale. Ils ont conclu que la valeur idéale du seuil (c'est-à-dire la valeur permettant d'améliorer les résultats) varie de façon inversement proportionnelle à la taille de la base (base de cooccurrence avec les termes de la requête) et à la taille moyenne des documents **[37]**.

II.7.3. Longueur moyenne de la requête

L'accroissement des performances de la reformulation de requête est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite **[63]**. Dans ce sens, des expérimentations intéressantes ont été réalisées sur la base TREC7 et présentées dans **[65].** Les auteurs montrent en effet que la dérivation automatique de courtes requêtes à partir de documents jugés ou supposés pertinents à la suite d'une recherche initiale, permet d'atteindre des résultats très performants pour différentes tâches : recherche, filtrage et routing.

II.7.4. Choix des documents d'expansion

Il existe peu de travaux réalisés pour la sélection des documents comparativement à la sélection des termes, la méthode présentée dans **[66]** considère le document pertinent comme un bon représentatif d'un ensemble de D_{rel} (les documents pertinents dans le corpus) dans la mesure où il peut aider efficacement pour trouver les documents pertinents à partir deD_{rel}par l'intermédiaire d'une recherche effectuée sur le corpus.

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

Deux grandes familles de méthodes sont présentées pour la sélection de documents :

La première estime la représentativité d'un document indépendamment des autres documents dansD_{rel}, en sélectionnant les k-tops documents classés, cette méthode assigne au document d (appartient àD_{rel}) un score, score(d) qui reflète la représentativité de d dansD_{rel}. Les k-tops documents dans D_{rel} sont ensuite utilisés comme entrée à la méthode de sélection des termes d'expansion. Parmi les caractéristiques utilisées pour la sélection des documents:

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

$$score_{(random)}(d)^{def} = \frac{1}{n} \qquad (II.26)$$

Puis choisir les k-documents à partir de D_{rel} au hasard.

Où n : est le nombre de documents dans la collection.

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

$$score_{QuerySim}(d)^{def} = sim(q,d) \qquad (II.27)$$

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

$$score_{length}(d)^{def} = -|d| \qquad (II.28)$$

|d| : est la taille de document d.

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

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

$$p(w\backslash Cent(D_{rel}))^{def} = \frac{1}{n}\sum_{d\in D_{rel}} p(w\backslash d) \qquad (II.29)$$

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

$$score_{centroid}(d)^{def} = -|\mathrm{d}| * (p\ (.\backslash cent\ (D_{rel})\ )||p(.\backslash d)) \qquad (II.30)$$

- Les méthodes basées sur le graphe : Certains travaux sur le reclassement de la première liste de documents trouvés en réponse à une requête, ont montré que les documents qui sont très semblables aux autres documents dans la liste ont une forte probabilité de pertinence. L'idée est que ces documents représentent la liste entière, et

par la vertu de la façon dont la liste a été créée, c'est-à-dire en réponse à la requête, ils pourraient être pertinents au besoin fondamental de l'information. Toutefois, la liste est composée de plusieurs documents à la fois pertinents et non pertinents.

II.8. Conclusion

Dans ce chapitre, nous avons présenté les techniques de reformulation de la requête, les différentes ressources utilisées pour choisir les termes à ajouter soit externes/internes ou bien à partir des jugements de pertinence de l'utilisateur sur les documents restitués par le système. Nous avons abordé la reformulation de la requête dans le cadre des différents modèles de recherche d'information.

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

Dans le chapitre suivant, nous allons présenter notre approche proposée et implémentée, qui consiste à l'extension du modèle de pertinence utilisé pour la reformulationde la requête dans le but de restituer les documents les plus pertinents.

CHAPITRE III

Présentation de l'approche et Expérimentation

III.1. Introduction

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

Nous avons présenté dans le chapitre précédent les différentes approches de reformulation de requête, dans ce chapitre nous allons présenter une nouvelle approche dont l'objectif est de restituer les documents pertinents aux besoins de l'utilisateur en se basant sur le modèle de pertinence.

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

III.2. Architecture générale de notre approche

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

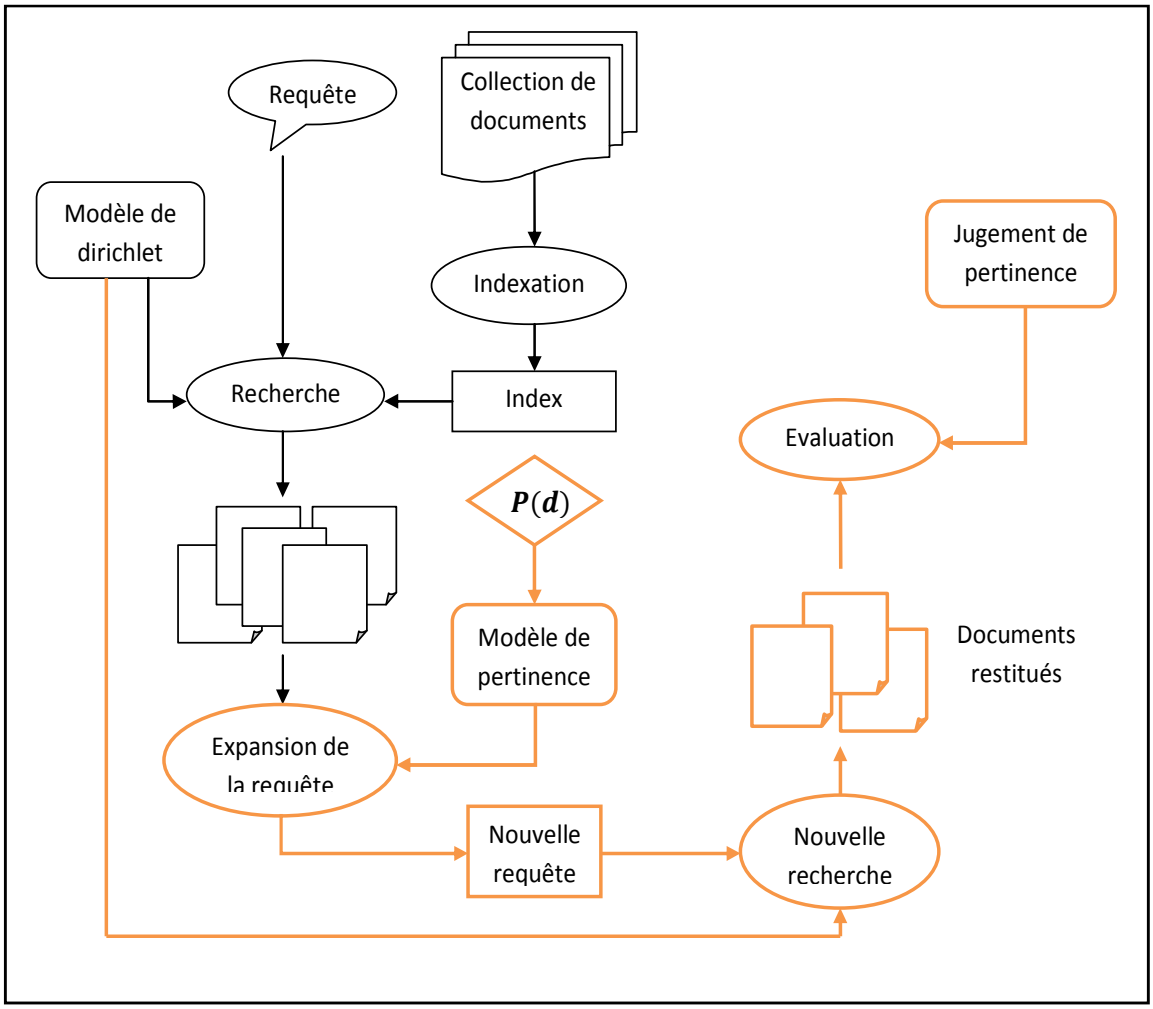


Figure III.1 : Architecture générale de notre approche

III.3. Présentation de notre approche

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

L'objectif de notre travail, est de pouvoir renvoyer à l'utilisateur les documents les plus pertinents répondant à sa requête. Pour cela, nous avons proposé de modifier le processus de recherche de Terrier qui devient comme suit :

a) La recherche simple : en se basant sur la structure de donnée Index produite par le processus d'indexation, le modèle de recherche assigne un score pour chaque terme de la requête dans le document, puis assigne des scores aux documents pour les retourner dans une liste ordonnée (résultat de la première recherche), en se basant sur le modèle de dirichlet.

- **Modèle de dirichlet :** le modèle de dirichlet est l'une des techniques de lissage proposées, afin de remédier au problème des probabilités nulles. Plusieurs études en RI, ont montré que le choix de la méthode de lissage, a un grand impact sur les performances du SRI. Zhai et Lafferty **[69]** ont expérimenté plusieurs techniques de lissage, en utilisant le modèle de langue uni-gramme, et ils ont rapporté que la méthode de lissage de dirichlet donne de meilleurs résultats que les autres méthodes de lissage.

  Le lissage de dirichlet exploite les valeurs de λ (formule(II.15)) en fonction de la taille de l'échantillon. Dans ce cas, la formule s'écrit comme suit :

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

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

$$= \frac{tf(m_i,d) + \mu P_{ML}(m_i|C)}{|d|+\mu} \qquad \text{(III. 1)}$$

  Avec :

$$P_{ML}(m_i|d) = \frac{tf(m_i,d)}{|d|} \qquad (III.2)$$

  $|d|$: est la taille du document (le nombre d'occurrences de mots).

$tf(m_i, d)$: est la fréquence du mot $\boldsymbol{m_i}$dans le document d.

μ : est un paramètre appelé pseudo fréquence.

b) L'expansion de requête :

1. Approche proposée : l'approche proposée consiste à étendre le modèle de pertinence.

1.1. Rappel sur le modèle de pertinence :ce modèle permet de générer les documents pertinents à une requête. Une description détaillée est présentée dans le chapitre précédent (section II.6.1.2).

Il est formalisé ainsi :

$$p(t|\theta_R) = \sum_{d \in R} P(d)P(t|d) \prod_{i=1}^{k} P(q_i|d) \qquad (II.18)$$

1.2. Estimation du facteur $\boldsymbol{p(d)}$: l'objectif de notre travail est précisément d'estimer le facteur$\boldsymbol{p(d)}$,et de voir son impact sur les résultats de la RI. En effet, ce facteur modélise la pertinence a priori des tops documents retournés par la première recherche. Pour son estimation, nous avons proposé plusieurs formules, chacune est basée sur une intuition.

1.2.1. Estimation de$\boldsymbol{p(d)}$ baséesur la clarté du document

1.2.1.1. La clarté de la requête : l'une des premières tentatives ayant eu du succès, semble être un score décrivant la clarté, c'est-à-dire l'absence d'ambiguïté d'une requête qui a été développé à l'université du Massachusetts **[46].**La clarté est basée sur le calcul de l'entropie relative des documents retrouvés pour la requête par rapport au modèle de langage de la collection de documents entière, et est supposée qu'elle reflète la cohérence thématique des documents contenant les termes de la requête.

Elle est formalisée ainsi :

$$score\_clarté = \sum_{t \in V} P(t|q) \log_2 \frac{P(t|q)}{P_{coll}(t)} \qquad (III.3)$$

Avec :

V : est le vocabulaire de l'index.

$P(t|q)$: indique la probabilité d'apparition du terme t dans la requête q.

$P_{coll}(t)$: indique la probabilité d'apparition du terme t dans la collection entière.

Les travaux réalisés sur la clarté ont été proposés dans le contexte de la requête. Pour notre cas, nous proposons de l'estimer sur les documents.

1.2.1.2. La clarté de document : un document constitue l'information élémentaire d'une collection de documents, il peut être un texte, une page web, une image, une vidéo, etc. Il peut aussi être défini par toute unité qui constitue une réponse à un besoin informationnel de

l'utilisateur **[4]**.Dans ce cas, la probabilité de pertinence a priori est proportionnelle à la clarté du document.

→ **Intuition de l'approche :** l'intuition de l'utilisation d'une telle caractéristique est qu'un document clair par rapport aux tops documents pertinents retournés dans la première recherche est un bon document, c'est-à-dire il traite la thématique de la requête. Pour cela, nous avons proposé deux valeurs qui permettent de mesurer le score de clarté pour chaque document retourné.

La première valeur proposée est formalisée ainsi :

$$p(d) = \sum_{i=1}^{|R|} p(t_i|d) * \log \frac{p(t_i|d)}{p(t_i|R)}$$

$$= \sum_{i=1}^{|R|} \left(\frac{tf}{taille(d)}\right) * \log \left(\frac{tf}{taille(d)} \Big/ \frac{TF}{taille(R)}\right) \qquad (F1)$$

Tels que :

R: est l'ensemble des tops documents pertinents retournés dans la première recherche.

$|R|$: est le nombre de documents appartenant à l'ensemble R.

$taille(d)$: est la taille du document d.

$taille(R)$: est la taille totale des documents de l'ensemble R.

tf : est la fréquence locale d'un terme t_i dans un document d.

TF : est la fréquence globale d'un terme t_i dans l'ensemble R.

La deuxième valeur proposée fait intervenir le log dans le but de diminuer l'écart entre les valeurs de clarté.

Elle est formalisée ainsi :

$$p(d) = \sum_{i=1}^{|R|} \log \left(p(t_i|d) * log \frac{p(t_i|d)}{p(t_i|R)} \right)$$

$$= \sum_{i=1}^{|R|} \log \left(\left(\frac{tf}{taille(d)}\right) * \log \left(\frac{tf}{taille(d)} \Big/ \frac{TF}{taille(R)}\right) \right) \qquad (F2)$$

1.2.1.3. Taille de document : dans ce cas, la probabilité de pertinence a priori est proportionnelle à la taille du document.

→ **Intuition :** l'intuition de l'utilisation d'une telle caractéristique est qu'un document plus long tend à contenir plus d'informations et par conséquent, il est plus probable d'être pertinent.

La première valeur proposée est formalisée ainsi :

$$p(d) = \frac{taille(d)}{taille(R)} \qquad (F3)$$

La deuxième valeur fait intervenir le log de la taille d'un document, pour diminuer l'écart entre les valeurs de la taille de document. Elle est formalisée ainsi :

$$p(d) = \frac{\log taille(d)}{\log taille(R)} \qquad (F4)$$

III.4. Exemple illustratif

Nous expliquons dans ce qui suit le fonctionnement de notre approche, plus explicitement, comment les documents de la première recherche seront réordonnés avec notre approche. Le tableau ci-dessous montre l'ordre des 4 documents ($d_0, d_{11}, d_{32}\ et\ d_{94}$), sachant que la fonction de pondération utilisée est : (II.18) étendue avec (F1).

id_doc	**rang_doc**		
	recherche simple	expansion avec le modèle de pertinence	expansion avec l'approche implémentée
AP880731-0085	0	0	0
AP880706-0311	11	1	1
AP880412-0268	32	4	3
AP881119-0051	94	6	4

Tableau III.1. Classement des documents avant et après l'expansion

On peut remarquer que :

Le document AP880731-0085n'a pas changé de rang.

Le document AP880706-0311 qui a un rang égal à 11 dans la recherche simple a passé au rang 1 dans l'expansion avec le modèle de pertinence et est resté au rang 1 dans l'expansion avec notre approche.

Le document AP880706-0311 qui a un rang égal à 11 dans la recherche simple a passé au rang11 dans l'expansion avec le modèle de pertinence et est resté au rang 11 dans l'expansion avec notre approche.

Le document AP880412-0268 qui a un rang égal à 32 dans la recherche simple a passé au rang 4 dans l'expansion avec le modèle de pertinence, puis au rang 3 dans l'expansion avec notre approche.

Le document AP881119-0051 qui a un rang égal à 94 dans la recherche simple a passé au rang 6 dans l'expansion avec le modèle de pertinence, puis au rang 4 dans l'expansion avec notre approche.

III.5. L'environnement technique

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

III.5.1. La plateforme Terrier

TERRIER**, TERabyteRetrIEveR :** est un moteur de recherche robuste et efficace, utilisé avec succès pour la recherche ad-hoc, la recherche sur le web et la recherche multilingue dans des environnements centralisés et distribués.

Terrier offre une plateforme idéale destinée à l'indexation de volumes importants de documents : jusqu'à 25 millions de documents. Il est développé par le département informatique de l'université Glasgow de Scotland.

Comme tous moteurs de recherche, terrier permet :

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

Architecture de Terrier

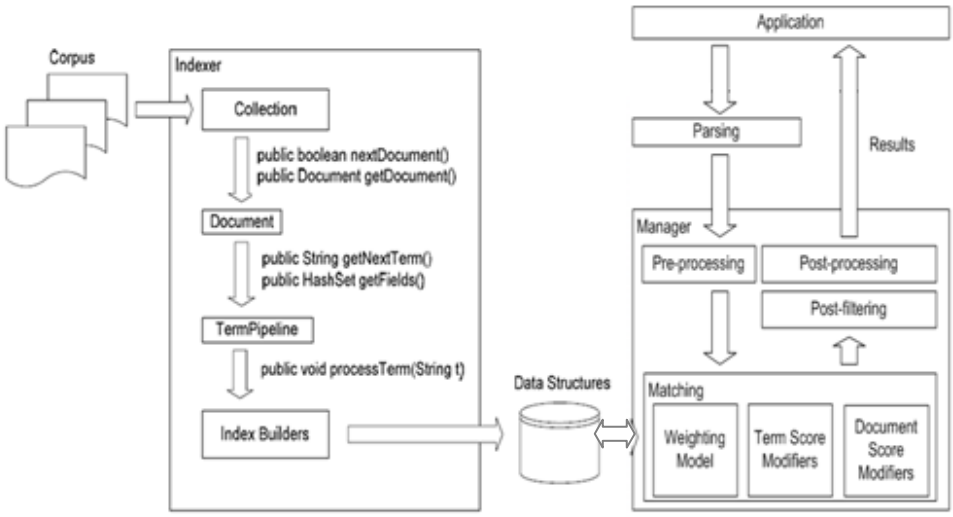


Figure III.2 : Vue d'ensemble d'architecture de Terrier

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

Les quatre procédures sont :

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

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

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

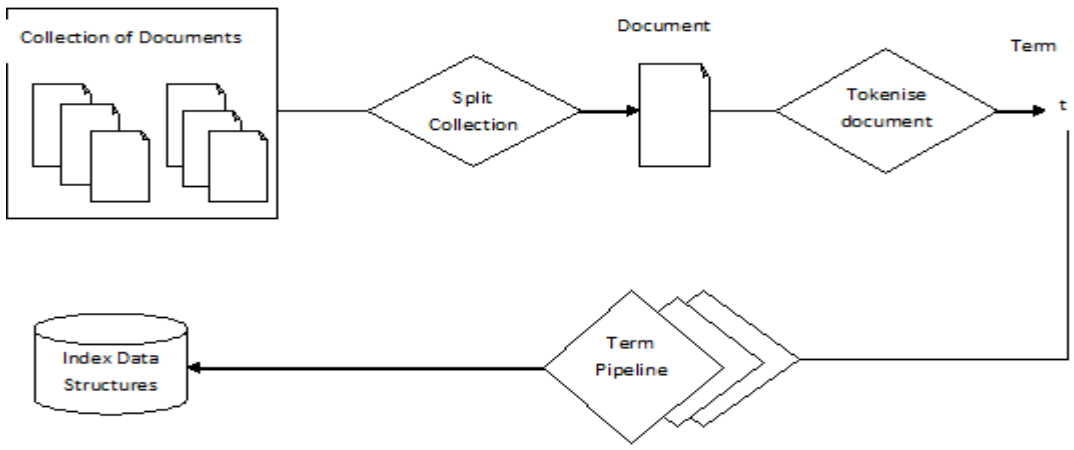


Figure III.3 : Le processus d'indexation dans Terrier

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

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

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

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

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

  Index (Id Terme, Id document, Fréquence terme dans le document, #fields).
- ✓ **Inverted Index** : contrairement à l'index direct, il enregistre pour un terme les documents dans lesquels il apparait, il contient aussi la position de chaque terme et sa fréquence dans ces documents**.**

  Fichier inverse (Id Terme, Id document, Fréquence terme dans le document, #fields).
- ✓ **Document Index :** contient des informations sur les différents documents de la collection (Id Terme, Fréquence terme, #fields).

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

1) Query : classe abstraite qui représente la requête.

   Terrier supporte trois modèles de requête :
   - ✓ *SingleTermQuery :* désigne la requête qui contient un seul terme.
   - ✓ *MultiTermQuery :* désigne la requête qui contient plusieurs termes.
   - ✓ *FieldQuery :* terme qualifié par un champ (Exp : dans le titre du document).
2) Parsing : qui se charge de tokenizer la requête.
3) Pré-processing : qui applique le TermPipeline à la requête. Elimine les mots vides et les lemmatise.
4) Matching : responsable de l'initialisation du Weighting Model et du calcul des scores entre la requête et les documents.
   - ✓ **WeightingModels :** assigne un score pour chaque terme de la requête dans le document (Pondération), plusieurs modèles de pondération sont implémentés : TF_IDF,BM25, etc.
   - ✓ **DocumentScoreModifiers :** il permet de modifier le score d'un document en fonction du langage de la requête.
5) Post-traitement : peut modifier le ResultSet, par exemple, par un procédé QueryExpansion, afin de générer un meilleur classement de documents. Ce procédé

fonctionne en faisant l'extraction des termes informatifs, à partir des tops documents classés (un nombre de documents spécifiés du ResultSet), par attribution du score pour chaque terme en utilisant le modèle de pertinence étendu (notre cas), et ajouter ceux avec les scores les plus élevés à la requête originale. La nouvelle requête est repondérée, et une nouvelle recherche est faite à l'aide du modèle de dirichlet. Un ensemble de documents plus pertinents, qui seront stockés dans le fichier.res (pertinence système) est retourné comme résultat à la nouvelle recherche effectuée.

6) Post-filtering : filtrage des résultats.

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

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

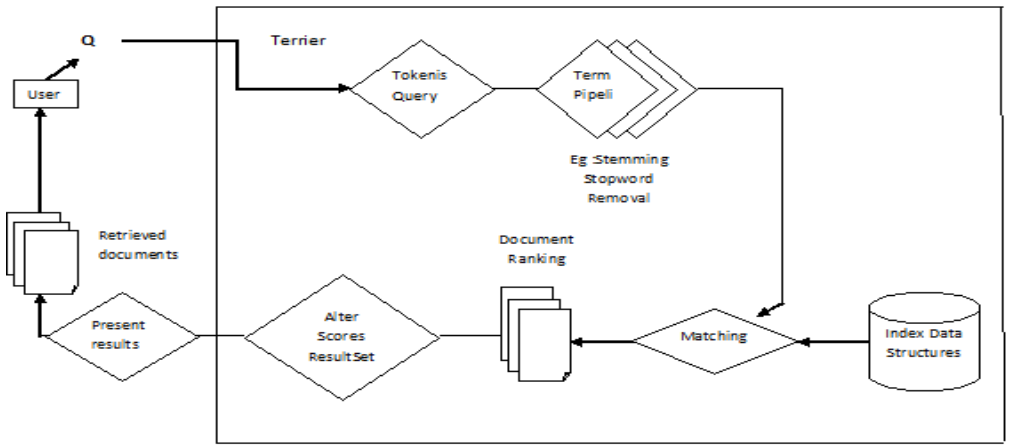


Figure III.4 : Le processus de recherche dans Terrier

III.5.2. Le langage java

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

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

Java se voit attribuer plusieurs qualités dont voici quelques unes :

- **Orienté objet** : la programmation orientée objets présente l'immense intérêt de faciliter la réutilisabilité des composants développés. Java est un langage orienté objet. Ecrire un

programme revient à écrire une classe d'objets avec ses données et les méthodes qui permettent d'y accéder.

- **Simple :** java hérite une grande partie de la syntaxe du langage C++, et dans le but d'écrire des codes facilement et sans erreurs, il en a été dépouillé de tous les mécanismes complexes, redondants ou devenus inutiles, tels que la gestion des pointeurs, la gestion de la mémoire (la libération de la mémoire en particulier n'est plus à la charge du développeur mais est générée de manière automatique par ramasse-miettes intégré), l'héritage multiple,…etc.
- **Robuste :** plusieurs raisons font que le code généré est effectivement plus robuste qu'avec d'autres langages, et risque donc moins de générer des erreurs :
  - Le développeur n'a pas la possibilité d'accéder aux pointeurs, réduisant ainsi le risque d'écraser des données par erreur dans une zone mémoire.
  - Le mécanisme de gestion des exceptions qui permet une meilleure maîtrise des erreurs. Ce mécanisme permet en effet de gérer des évènements non souhaités (ex : division par zéro) en imposant un traitement adapté (ex : arrêt du programme).
  - La déclaration des variables doit obligatoirement être explicite en java. Le code est vérifié (Syntaxe, type) à la compilation et également au moment de l'exécution, ce qui permet de réduire les bugs et les problèmes d'incompatibilité de version.
- **Sécurisé :** au moment de l'exécution d'un programme java, le JRE utilise un processus nommé ClassLoader qui s'occupe du chargement du *byte code* (ou langage binaire intermédiaire) contenu dans les classes java. Le byte code est ensuite analysé afin de contrôler qu'il n'a pas fait de création ou de manipulation de pointeurs en mémoire et également qu'il n'y a pas de violation d'accès.
- **Portable :** cette caractéristique est l'une des celles qui ont contribué à leur grande réputation parmi les communautés d'internet et ceci grâce à son indépendance de toute plateforme d'exécution, car un programme java peut tourner sur n'importe quelle machine possédant une JVM.
- **Multi plates-formes :** les programmes tournent sans modification sur tous les environnements (Windows, Unix,…etc.)

III.5.3. NetBeans

NetBeans est à l'origine un EDI (Environnement de Développement Intégré) Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelques parts en 2002, Sun a décidé de rendre NetBeans open-source. Mais NetBeans n'est pas uniquement un EDI java, c'est également une plateforme. Il vous est possible de créer votre propre application Awt ou Swing, basé sur la plateforme NetBeans.

Pour celles et ceux d'entre vous qui viennent du monde Eclipse, cela correspond à Eclipse RCP. Sa conception est complètement modulaire : tout est module, même la plateforme. Ce qui fait de NetBeans une boite à outils facilement améliorable ou modifiable. La licence de NetBeans permet de l'utiliser gratuitement à des fins commerciales ou non. Les modules que vous pourriez écrire peuvent être open-sources comme ils peuvent être closed-source, ils peuvent être gratuits, comme ils peuvent être payants. Il présente une interface conviviale **GUI** (Graphical User Interface) qui nous permet d'éditer, compiler et exécuter un programme écrit en langage java.

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

III.6. Résultats et expérimentation

III.6.1. Collection de test utilisée

Pour évaluer les différentes formules exposées dans la section précédente, nous avons utilisé la collection de test TREC AP88. Cette dernière contient des documents plats, elle est de taille moyenne. Nous avons également utilisé 50 requêtes (seulement le titre),qui sont dans un fichier nommé topics.51-100.

Le tableau ci-dessous montre quelques statistiques sur la collection et les topics utilisées.

Collection	Taille	Documents	Topics
AP88	237Mo	79919	51-100

Tableau III.2. Statistiques sur la collection de test et les topics utilisées

III.6.2. Evaluation et résultats

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

III.6.2.1. Résultats obtenus avec la recherche simple

L'objectif de la recherche simple est de déterminer la meilleure valeur de paramètre µ du modèle de recherche dirichlet; donc nous avons varié la valeur de ce dernier de 100 à 2400 avec un pas de 100. Le tableau suivant montre les résultats obtenus.

µ	MAP	µ	MAP
100	0.0165	1300	0.0522
200	0.0251	1400	0.0528
300	0.0304	1500	0.0532
400	0.0355	1600	0.0536
500	0.0399	1700	0.0551
600	0.0432	1800	0.0567
700	0.0457	1900	0.0572
800	0.0474	2000	0.0586
900	0.0485	2100	0.0590
1000	0.0497	2200	0.0594
1100	0.0509	**2300**	**0.0597**
1200	0.0517	2400	0.0596

Tableau III.3. Précision moyenne en faisant varier le coefficient µ du modèle de dirichlet

Nous remarquons dans le tableau ci-dessus que le meilleur résultat est obtenu avec **µ=2300** avec une précision moyenne de **0.0597**.Pour cela, nous avons testé l'expansion en considérant le paramètre **µ=2300**.

III.6.2.2. Résultats obtenus avec le modèle de pertinence

Dans l'étape de l'expansion, nous avons remplacé les deux classes ExpansionTerms et QueryExpansion situées dans la librairie de Terrier, avec celles utilisant le modèle de pertinence que nous avons développées. Plus précisément, la classe ExpansionTerms s'est mise à l'emplacement*D:\Prior_QE\terrier\lib\terrier-2.1\uk\ac\gla\terrier\structures*, et la classe QueryExpansion s'est mise à son tour à l'emplacement *D:\Prior_QE\terrier\lib\terrier-2.1\uk\ac\gla\terrier\querying*. Nous avons ensuite testé l'expansion, en faisant varier le nombre de documents d'expansion de **3**, puis de **5** à **25** avec un pas de **5**et celui des termes d'expansion de **3**, puis de **5** à **40** avec un pas de **5**. Les résultats obtenus avec le modèle de pertinence sont montrés dans le tableau ci-dessous.

Nombre de documents	Nombre de termes	MAP	Taux	Nombre de documents	Nombre de termes	MAP	Taux
3	3	0,1178	97,31%	15	3	0,115	92,62 %
	5	0,1176	96,98%		5	0,115	92,62 %
	10	0,1253	109,88%		10	0,1155	93,46 %
	15	0,1283	114,90%		15	0,1209	102,51 %
	20	0,126	111,05%		20	0,1225	105,19 %
	25	0,1271	112,89%		25	0,1254	110,05 %
	30	0,1284	115,07%		30	0,1259	110,88 %
	35	0,1288	115,74%		35	0,1256	110,38 %
	40	0,1286	115,41%		40	0,1254	110,05 %
5	3	0,1115	86,76%	20	3	0,1065	78,39 %
	5	0,1114	86,59%		5	0,1065	78,39 %
	10	0,1183	98,15%		10	0,1149	92,46 %
	15	0,123	106,03%		15	0,1188	98,99 %
	20	0,1256	110,38%		20	0,12	101%
	25	0,1257	110,55%		25	0,1216	103,68 %
	30	0,127	112,73%		30	0,1219	104,18 %
	35	0,1282	114,74%		35	0,1227	105,52 %
	40	0,1277	113,90%		40	0,1221	104,52 %
10	3	0,1028	72,19%	**25**	3	0,1026	71,85 %
	5	0,1028	72,19%		5	0,1026	71,85 %
	10	0,1128	88,94%		10	0,1161	94,47 %
	15	0,1194	100%		15	0,1198	100,67 %
	20	0,1199	100,83%		20	0,1222	104,69 %
	25	0,1197	100,50%		25	0,1269	112,56 %
	30	0,1183	98,15%		**30**	**0,1289**	**115,91 %**
	35	0,1195	100,16%		35	0,1281	114,57 %
	40	0,1191	99,49%		40	0,1278	114,07 %

Tableau III.4. Précision moyenne en faisant varier le nombre de documents et le nombre de termes d'expansion dans le modèle de pertinence

D'après le tableau ci-dessus, nous constatons que le modèle de pertinence utilisé pour l'expansion a apporté des améliorations notables. Le nombre de documents et celui de termes d'expansion jouent un rôle important dans l'amélioration de la précision. Par ailleurs, le meilleur résultat est obtenu avec un nombre de documents d'expansion égal à **25**, et un nombre de termes d'expansion égal à **30**, avec une précision moyenne de **0.1289**et un taux d'amélioration de **115,91%**. Pour cela, nous avons alors testé notre approche en considérant le même paramètre (**25 documents et 30 termes**).

III.6.2.3. Résultats obtenus avec notre approche

Nous procédons de la même manière, nous remplaçons les classes ExpansionTerms et QueryExpansion implémentant le modèle de pertinence, se trouvant dans la librairie de Terrier avec celles implémentant le modèle de pertinence étendu avec F1, F2, F3 et F4 respectivement.

Le tableau suivant montre les résultats obtenus après l'expansion avec le modèle de pertinence étendu avec les différentes formules proposées.

Formules	MAP	Taux d'amélioration
F1	0,1185	-8,06%
F2	0,1135	-11,94%
F3	**0,1294**	**+0,38%**
F4	0,0595	-53,84%

Tableau III.5. Précision moyenne après l'expansion avec le modèle de pertinence étendu avec les différentes formules utilisées.

D'après le tableau ci-dessus, nous constatons que :

Les formules (F1), (F2) et (F4)n'ont pas apporté d'amélioration par rapport au modèle de pertinence (II.18).

La formule (**F3**) a apporté une amélioration avec une précision de **0.1294,** et un taux d'amélioration de **0,38%.**

La figure suivante illustre les différentes précisions retournées, en appliquant l'expansion avec le modèle de pertinence, et le modèle de pertinence étendu avec les formules proposées.

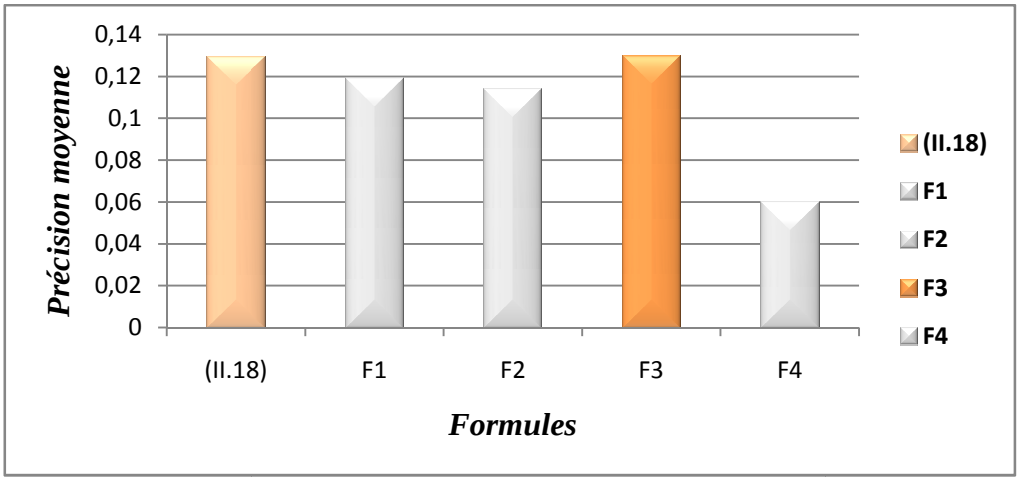


Figure III.5 : Comparaison du modèle de pertinence étendu avec les formules proposées avec le modèle de pertinence (II.18)

Ces résultats montrent que notre approche (modèle de pertinence étendu avec F3) de reformulation de requêtes, offre une précision moyenne plus importante que la reformulation avec le modèle de pertinence.

III.6.2.4. Résultat obtenu avec la formule F1

Dans cette étape, nous avons testé la formule F1, en faisant varier le nombre de documents d'expansion et celui de termes d'expansion.

Les résultats obtenus ont montré que la formule F1 a donné une amélioration meilleure que celle obtenue dans le modèle de pertinence avec une précision de **0,1298** et un taux d'amélioration de **0,69%**, ainsi que celle obtenue par la formule F3, avec un taux d'amélioration de **0,30%**, et cela en utilisant un nombre de documents d'expansion égal à **3** et un nombre de termes d'expansion égal à **35**. Ceci implique que le nombre de documents ainsi que celui des termes d'expansion ont un rôle important dans l'amélioration de la précision.

La figure ci-dessous montre le résultat obtenu.

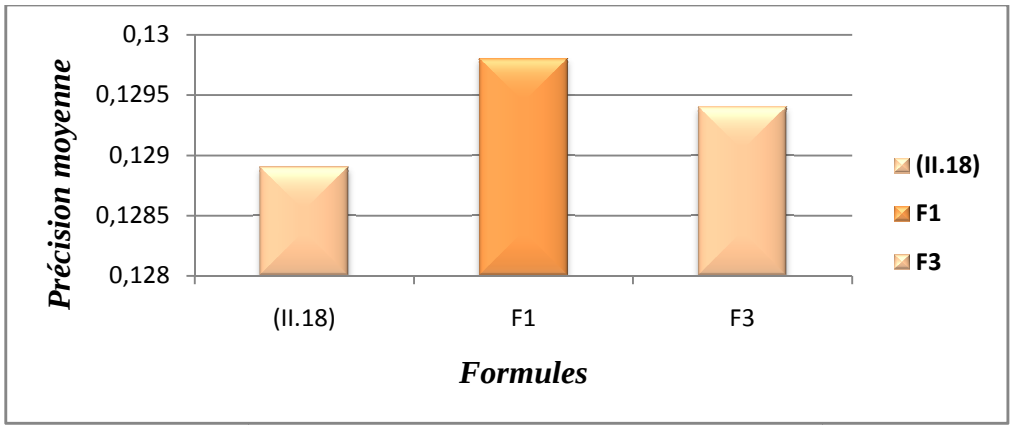


Figure III.6 : Comparaison du modèle de pertinence étendu avec la formule F1 (nombre de documents 3 et nombre de termes 35) avec le modèle de pertinence.

III.6.2.5. Evaluation requête par requête

Les résultats montrés précédemment donnent une appréciation globale; afin d'avoir une vue précise de ces résultats, nous avons analysé les résultats requête par requête. Le tableau ci-dessous montre les résultats obtenus avec la recherche simple (modèle de dirichlet) et le modèle de pertinence.

Requêtes	MAP (modèle de dirichlet)	MAP (modèle de pertinence)	Taux	Requêtes	MAP (modèle de dirichlet)	MAP (modèle de pertinence)	Taux
51	0,2561	0,7855	206,71%	76	0,0012	0,0001	-91,66 %
52	0,1733	0,3255	87,82%	77	0,0327	0	-100%
53	0,2007	0,6312	214,49%	78	0,0696	0	-100%
54	0,1298	0,4207	224,11%	79	0	0	0%
55	0,0126	0,0108	-16,66%	80	0,0088	0,0125	42,04%
56	0,1461	0,2998	105,20%	81	0,058	0,1504	159,31%
57	0,0543	0,0311	-42,72%	82	0,0463	0,0059	-87,25%
58	0,0147	0,0001	-99,31%	83	0,0021	0,0002	-90,47%
59	0,0002	0	-100%	84	0,0106	0,0014	-86,79%
60	0,0006	0	-100%	85	0,0129	0,0074	-42,63%
61	0,1329	0,4474	236,64%	86	0,0037	0	-100%
62	0,0258	0,1674	548,83%	87	0,003	0	-100%
63	0,0134	0,009	-32,83%	88	0,065	0,4296	560,92%
64	0,002	0,0003	-85%	89	0,0044	0,0035	-20,45%
65	0	0	0%	90	0,2743	0,3535	28,87%
66	0,0005	0	-100%	91	0	0	0%
67	0	0	0%	92	0,0001	0	-100%
68	0,0157	0,0082	-47,77%	93	0,0011	0,0001	-90,90%
69	0,0114	0,0001	-99,12%	94	0,0018	0	-100%
70	0,1456	0,8451	480,42%	95	0	0	0%
71	0,0252	0	-100%	96	0,0102	0,0084	-17,64%
72	0,0029	0,0002	-93,10%	97	0,426	0,8199	92,46%
73	0,0002	0	-100%	98	0,4862	0,5112	5,14%
74	0,0002	0,0001	-50%	99	0,0396	0,0277	-30,05%
75	0,0055	0,0026	-52,72%	100	0,0027	0,0001	-96,29%

Tableau III.6. Résultats obtenus avec l'analyse requête par requête avant et après l'expansion avec le modèle de pertinence.

D'après le tableau ci-dessus, les résultats obtenus après l'expansion avec le modèle de pertinence ont révélé:

- ✓ 14 requêtes améliorées.
- ✓ 5 requêtes nulles (non améliorées).
- ✓ 31 requêtes dégradées.

Le tableau ci-dessous montre les résultats obtenus après l'expansion avec le modèle de pertinence étendu avec F1 et les comparer avec ceux obtenus avec le modèle de dirichlet.

Requêtes	MAP (modèle de dirichlet)	MAP F1	Taux	Requêtes	MAP (modèle de dirichlet)	MAP F1	Taux
51	0,2561	0,8234	221,51%	76	0,0012	0,0005	-58,33%
52	0,1733	0,5712	229,60%	77	0,0327	0,0024	-92,66%
53	0,2007	0,3107	54,80%	78	0,0696	0	-100%
54	0,1298	0,1259	-3%	79	0	0	0%
55	0,0126	0,2623	1981,74%	80	0,0088	0,3903	4335,22%
56	0,1461	0,6097	317,31%	81	0,058	0,0087	-85%
57	0,0543	0,0556	2,39%	82	0,0463	0,3715	702,37%
58	0,0147	0	-100%	83	0,0021	0	-100%
59	0,0002	0	-100%	84	0,0106	0	-100%
60	0,0006	0	-100%	85	0,0129	0,0009	-93,02%
61	0,1329	0,3748	182,01%	86	0,0037	0	-100%
62	0,0258	0,168	551,16%	87	0,003	0	-100%
63	0,0134	0,0088	-34,32%	88	0,065	0,012	-81,53%
64	0,002	0,0003	-85%	89	0,0044	0,8018	18122,72%
65	0	0	0%	90	0,2743	0,5397	96,75%
66	0,0005	0,0009	80%	91	0	0,0032	-
67	0	0	0%	92	0,0001	0	-100%
68	0,0157	0,0948	503,82%	93	0,0011	0,0005	-54,54%
69	0,0114	0	-100%	94	0,0018	0,0024	33,33%
70	0,1456	0,557	282,55%	95	0	0	0%
71	0,0252	0	-100%	96	0,0102	0	-100%
72	0,0029	0	-100%	97	0,426	0,3903	-8,38%
73	0,0002	0	-100%	98	0,4862	0,0087	-98,21%
74	0,0002	0,0001	-50%	99	0,0396	0,3715	838,13%
75	0,0055	0,0331	501,81%	100	0,0027	0	-100%

Tableau III.7.Résultats obtenus avec l'analyse requête par requête avant et après l'expansion avec le modèle de pertinence étendu avec F1.

D'après le tableau ci-dessus, les résultats obtenus avec le modèle de pertinence étendu avec F1 ont révélé :

- ✓ 19 requêtes améliorées.
- ✓ 4 requêtes nulles (non améliorées).
- ✓ 27 requêtes dégradées.

Le tableau ci-dessous montre les résultats obtenus avec l'expansion avec le modèle de pertinence étendu et les comparer avec ceux obtenus avec le modèle de pertinence.

Requêtes	MAP (modèle de pertinence)	MAP F1	Taux	Requêtes	MAP (modèle de pertinence)	MAP F1	Taux
51	0,7855	0,8234	4,82%	76	0,0001	0,0005	400%
52	0,3255	0,5712	75,48%	77	0	0,0024	-
53	0,6312	0,3107	-50,77%	78	0	0	0%
54	0,4207	0,1259	-70,07%	79	0	0	0%
55	0,0108	0,2623	2328,70%	80	0,0125	0,3903	3022,4%
56	0,2998	0,6097	103,36%	81	0,1504	0,0087	-94,21%
57	0,0311	0,0556	78,77%	82	0,0059	0,3715	6196,61%
58	0,0001	0	-100%	83	0,0002	0	-100%
59	0	0	0%	84	0,0014	0	-100%
60	0	0	0%	85	0,0074	0,0009	-87,83%
61	0,4474	0,3748	-16,22%	86	0	0	0%
62	0,1674	0,168	0,35%	87	0	0	0%
63	0,009	0,0088	-2,22%	88	0,4296	0,012	-97,20%
64	0,0003	0,0003	0%	89	0,0035	0,8018	-2,20%
65	0	0	0%	90	0,3535	0,5397	52,67%
66	0	0,0009	-	91	0	0,0032	-
67	0	0	0%	92	0	0	0%
68	0,0082	0,0948	1056,09%	93	0,0001	0,0005	400%
69	0,0001	0	-100%	94	0	0,0024	-
70	0,8451	0,557	-34,09%	95	0	0	0%
71	0	0	0%	96	0,0084	0	-100%
72	0,0002	0	-100%	97	0,8199	0,3903	-52,39%
73	0	0	0%	98	0,5112	0,0087	-98,29%
74	0,0001	0,0001	0%	99	0,0277	0,3715	1241,15%
75	0,0026	0,0331	1173,07%	100	0,0001	0	-100%

Tableau III.8. Comparaison de l'analyse requête par requête obtenue dans l'expansion avec le modèle de pertinence étendu avec F1 et celle obtenue avec le modèle de pertinence.

D'après le tableau ci-dessus, les résultats obtenus avec le modèle de pertinence étendu ont apporté d'amélioration par rapport à ceux obtenus avec le modèle de pertinence.

- ✓ 19 requêtes améliorées.
- ✓ 14 requêtes nulles (non améliorées).
- ✓ 17 requêtes dégradées.

La figure suivante illustre la comparaison du modèle de pertinence et celui étendu avec F1 :

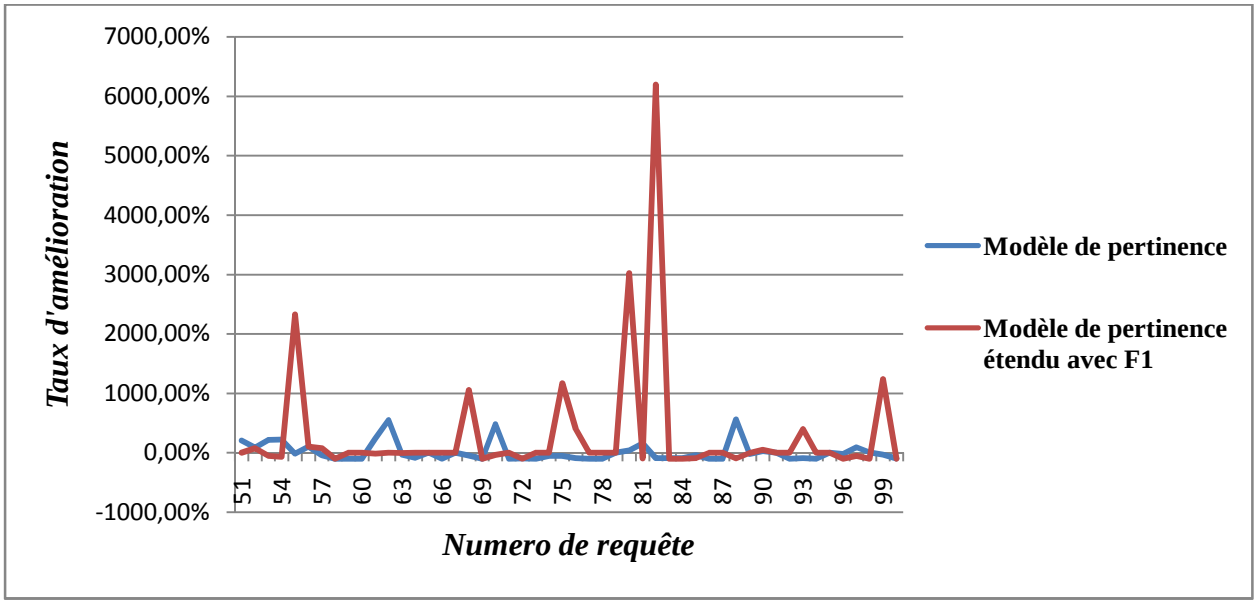


Figure III.7. L'apport de notre reformulation

La figure ci-dessus montre le taux d'amélioration des résultats obtenus après l'expansion avec le modèle de pertinence étendu avec F1 et ceux obtenus après l'expansion avec le modèle de pertinence. Les résultats ont révélé une amélioration considérable en utilisant le modèle de pertinence étendu avec F1 par rapport à ceux obtenus en utilisant le modèle de pertinence.

III.6.2.6. Analyse des résultats obtenus précédemment, en se basant sur le type de la requête

→ **Intuition :**nous proposons dans ce qui suit d'améliorer les résultats selon le type de requête; nous avons classé les requêtes en trois types :

- ambigües si la valeur de MAP <0,05 ;
- moyennes si 0,05 < MAP < 0,2 ;
- claires si MAP > 0,2.

Le tableau ci-dessous montre les résultats obtenus

Requêtes	Recherche simple	Modèle de pertinence	Taux	Modèle étendu avec F1	Taux F1 par rapport à la recherche simple	Taux F1 par rapport au modèle de pertinence
51	claire	+	80%	+,++	60%	60%
53	claire	+		+		
90	claire	+		+,++		
97	claire	+				
98	claire			++		
52	moyenne	+	77,77%	+,++	44,44%	33,33%
54	moyenne	+				

56	moyenne	+		++		
57	moyenne			+,++		
61	moyenne	+		+		
70	moyenne	+		+		
78	moyenne					
81	moyenne	+				
88	moyenne	+				
55	ambigüe			+,++		
58	ambigüe					
59	ambigüe					
60	ambigüe					
62	ambigüe	+		+,++		
63	ambigüe					
64	ambigüe					
65	ambigüe					
66	ambigüe			+,++		
67	ambigüe					
68	ambigüe			+,++		
69	ambigüe					
71	ambigüe					
72	ambigüe					
73	ambigüe					
74	ambigüe					
75	ambigüe			+,++		
76	ambigüe		5,55%	++	**30,55%**	38,88%
77	ambigüe			++		
79	ambigüe					
80	ambigüe	+		+,++		
82	ambigüe			+,++		
83	ambigüe					
84	ambigüe					
85	ambigüe					
86	ambigüe					
87	ambigüe					
89	ambigüe			+,++		
91	ambigüe			+,++		
92	ambigüe					
93	ambigüe			++		
94	ambigüe			+,++		
95	ambigüe					
96	ambigüe					
99	ambigüe			+,++		
100	ambigüe					

Tableau III.9. Les résultats obtenus en utilisant l'expansion avec le modèle de pertinence étendu avec F1 et ceux obtenus en utilisant l'expansion avec le modèle de pertinence en se basant sur le type des requêtes.

Dans la recherche simple, les résultats ont révélé :

- ✓ 5 requêtes claires.
- ✓ 9 requêtes moyennes.
- ✓ 36 requêtes ambigües.

Dans l'expansion avec le modèle de pertinence, les résultats ont révélé :

- ✓ 4 requêtes améliorées sur les 5 requêtes claires de la recherche simple, avec un taux d'amélioration de 80%.
- ✓ 7 requêtes améliorées sur les 9 requêtes moyennes de la recherche simple, avec un taux d'amélioration de 77,77%.
- ✓ 2 requêtes améliorées sur les 36 requêtes ambigües de la recherche simple, avec un taux d'amélioration de 5,55%.

Dans l'expansion avec le modèle de pertinence étendu avec F1, les résultats ont révélé :

- ✓ 3 requêtes améliorées sur les 5 requêtes claires de la recherche simple, avec un taux d'amélioration de 60% et 3 requêtes claires sur les 4 requêtes claires du modèle de pertinence avec un taux d'amélioration de 60%.
- ✓ 4 requêtes améliorées sur les 9 requêtes moyennes de la recherche simple, avec un taux d'amélioration de 44,44% et 3 requêtes améliorées sur les 7 requêtes moyennes du modèle de pertinence avec un taux d'amélioration de 33,33%.
- ✓ 11 requêtes améliorées sur les 36 requêtes ambigües de la recherche simple, avec un taux d'amélioration de 30,55% et 14 requêtes améliorées sur les 2 requêtes ambigües du modèle de pertinence avec un taux d'amélioration de 38,88%.

D'après ces résultats, nous constatons que le modèle de pertinence étendu avec la formule F1 a apporté une amélioration considérable pour les requêtes ambigües avec un taux d'amélioration de **30,55%**, par contre le modèle de pertinence a apporté une amélioration considérable pour les requêtes claires avec un taux d'amélioration de **80%**.

III.7. Conclusion

Dans ce chapitre, nous avons d'une part exposé notre approche qui consiste à étendre le modèle de pertinence en introduisant le facteur p(d); et cela en proposant plusieurs formules. D'autre part, nous avons présenté les résultats obtenus avec les formules proposées; comme nous avons effectué une comparaison avec le modèle de base « modèle de pertinence », obtenu sur une collection de Test TREC AP88.

Conclusion générale

Conclusion générale

Le travail développé dans ce mémoire s'inscrit dans le cadre de la reformulation de requête, nous nous sommes particulièrement intéressés au ré-ordonnancement des documents retournés dans la première recherche.

Afin de sélectionner les documents susceptibles de répondre à une requête, un SRI évalue la pertinence d'un document vis-à-vis d'une requête, mais les documents retournés par le SRI, ne répondent pas toujours aux besoins de l'utilisateur. Pour prendre en compte cette difficulté, des techniques de reformulations (expansion) de la requête sont utilisées, afin d'obtenir des requêtes optimales, le fait que pour sélectionner les bons termes à ajouter à la requête initiale, il faut au préalable choisir les bons documents dans lesquels on recherche les termes.

Pour optimiser les résultats retournés, on a proposé dans notre approche d'améliorer le processus de sélection des documents utilisés pour l'expansion en se basant sur le modèle de pertinence étendu avec la probabilité a priori p(d). Cependant, plusieurs formules ont été évaluées en utilisant une collection de test TREC AP88 sous la plateforme de RI « Terrier ». Les résultats obtenus ont été prometteurs, et parmi les perspectives envisagées pour ce travail est l'expérimentation de notre approche sur d'autres jeux de test (collection plus volumineuse comme par exemple **.**GOV), afin de valider les résultats obtenus.

Bibliographie

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

[2] : Cours « Problématique Générale de la Recherche d'Information », URFIST Bretagne-Pays de Loire, Alexandre Serres, 2002.
http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html

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

[4]: P. Ingwersen. Information retrieval interaction.Taylor Graham, London, 1992.

[5]: S.E. Robertson, S. Walker, M. Beaulieu: OKAPI at TREC 7: Automatic Adhoc Filtering, VLC and Interactive Track, In Proceedings of the 7thText Retrieval Conference TREC7, Juillet 1999.

[6]: C. Bourne, B.Anderson : DIALOG LabWorkbook, second edition, Lockheed Information Systems, PaloAlto, Californie (USA), 1979.

[7]: A. Lelu & C. François : Information Retrieval Based on Neural Unsupervised Extraction of Thematic Fuzzy Clusters, Les Réseaux Neuromimétiques et leurs Applications (Neuronîmes), 1992.

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

[9] : GéryMathias & al., Lyon/Saint-Etienne BM25t : une extension de BM25 pour la Recherche d'Information ciblée Mathias Géry, Christine Largeron, Franck Thollard : Lyon, Saint-Étienne, France (Laboratoire Hubert Curien).

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

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

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

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

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

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

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

[17]:N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? Commun. ACM, 1992.

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

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

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

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

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

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

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

[25]: G. Zipf, « Human Behaviour and the Principal of Least ». Addison-Wesley, 1949.

[26] : Karen SAVAGNAT, Thèse Doctorat, "Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés", Université Paul Sabatier Toulouse, 2005.

[27]:S.E.Robertson and S.Walker, "Some somple effective approximations to the 2-Poisson model for probabilistic weighted retrieval", In proceedings of SIGIR 1994.

[28]:A.Singhal, G.Salton, M.Mitra, and C.Buckley, "Document lengthnormalization", Information Processing and management, 32(5): 619-633, 1996.

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

[30]: G. Salton: « *A comparison between manual and automatic indexing methods* ». Journal of American documentation, 20(1), 1971.

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

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

[33]: Robertson, S. E., & Sparck Jones, K.: "*Relevance weighting of search terms*". Journal of the American Society for Information Science, 27, 129–146.1976.

[34]: Fatiha BOUBEKEUR-AMIROUCHE, Thèse Doctorat, "Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets", 2008.

[35]:S.KARBASI. Thèse Doctorat "Pondération des termes en Recherche d'Information :Modèle de pondération basé sur le rang des termes dans les documents", Université Paul Sabatier de Toulouse, Septembre 2007.

[36] :http://www.atilf.atilf.fr/

[37]: Lobna HLAOUA, Reformulation de Requêtes par Réinjection de Pertinence dans les Documents Semi-Structurés.

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

[39] : Boughanem, M., Kraaij, W., Nie, J.-Y. Modèles de langue pour la recherche d'information. Les systèmes de recherche d'informations, pages 163-182. Hermes-Lavoisier, 2004.

[40] : J. J Rocchio : Relevance Feedback in Information Retrieval, in The Smart System Experiments in Automatic Document Processing, G.Salton, Editor, Prentice-Hall, Inc., Englewood Cliffs, NJ, pp 313-23, 1971.

[41] : Hamid TEBRI, Formalisation et spécification d'un système de filtrage incrémental d'information, l'Université Paul Sabatier de Toulouse, 2004.

[42]: Jian-Yun Nie, Le domaine de recherche d'information, Département d'informatique et recherche opérationnelle, Université de Montréal.

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

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

[45]:D. Hiemstra, F. De Jong, 2002*"Term specific smoothing for the language modeling approach to information retrieval: The importance of a query term"*, In proceedings of the 25th annualinternational ACM SIGIR conference on Research and development in information retrieval,pages 35-41, 2002.

[46]: Steve CRONEN-TOWNSEND, Yun ZHOU et W. BruceCROFT (2002). Predicting query performance. In Proceedings of SIGIR 2002, pages 299–306. ACM Press.

[47] : Moreau et Claveau « Extension de requêtes par relations morphologiques acquises automatiquement » IRISA-CNRS Campus universitaire de Beaulieu Avenue du Général Leclerc 35042 Rennes cedex, France, {Fabienne.Moreau,Vincent.Claveau@irisa.fr}, 2006.

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

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

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

[51]:M'Hamed Mataoui, « thème : reformulation de requête dans les systèmes de recherche d'information dans les documents XML », thèse de magister soutenu le 29-04-2007, université M'hmed BOUGARA de Boumerdes.

[52]: Croft, W.B., Harper D.J. « Using probabilistic models of document retrieval without relevance information ». Journal of Documentation 354 (1979), p. 285-295.

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

[54]: Donna Harman, "Relevance Feedback and Other Query Modification Techniques", Information Retrieval: Data Structures & Algorithms 1992: 241-263.

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

[56]: Ponte, J.M., Croft, W. B. A language modeling approach to information retrieval. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 275-281, 1998.

[57]:LAVRENKO, V. AND CROFT,W. B. 2001. Relevance based language models. In *Proceedings of the 24th AnnualInternational ACM SIGIR Conference on Research and Development in Information Retrieval*. ACMPress, 120–127.

[58]: DEMPSTER, A., LAIRD, N., AND RUBIN, D. 1977. Maximum likelihood from incomplete data via the EMalgorithm. *J. Royal Statist. Soc.* Series B (Methodological) *39*, 1, 1–38.

[59]: Improving the Robustness of Relevance-Based Language ModelsXiaoyan Li, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, Amherst MA 01003.

[60]:LAVRENKO, V. AND ALLAN, J. 2006. Realtime query expansion in relevance models. IR 473, University of Massachusetts.

[61]: METZLER, D. AND CROFT, W. B. 2007. Latent concept expansion using Markov random fields. In *Proceedingsof the 30th Annual International ACM SIGIR Conference on Research and Development inInformation Retrieval*. ACM Press, 311–318.

[62] : Lynda Tamine, « Optimisation de requêtes dans un système de recherche d'information » thèse de Doctorat à l'Université de Paul Sabatier, 2000.

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

[64] : ABDELKRIM BOURAMOUL, Recherche d'information contextuelle et sémantique sur le web, 2001.

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

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

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

[68] : BESANÇON, Romaric **:** Technologies statistiques pour la recherche d'informations : les modèles vectoriels. In : « *Les systèmes de recherche d'informations: modèles conceptuels »*. Sous la direction de M.Ihadjadene. Paris : Lavoisier, 2004. – ISBN : 2-7462-0821-0.

[69]: ZHAI, C. AND LAFFERTY, J. 2001a. Model-based feedback in the language modeling approach to informationretrieval. In *Proceedings of the 10th International Conference on Information and Knowledge Management*.ACM Press, 403–410.

Annexe

1. Introduction

Terrier, TerabyteRetrieval est un projet initialisé par l'université de Glasgow de Royaumes-Unis en 2000, dans le but de fournir une plateforme flexible pour le développement rapide des applications de recherche d'information. Terrier est un produit Open source écrit en java, il a été mis à la disposition du général public depuis novembre 2004 sous la licence de MPL.Il a été conçu pour fonctionner sur la plupart des systèmes d'exploitation actuels tels que Windows ou linux.

Le projet de Terrier a exploré de nouvelles, efficaces et effectives méthodes de recherche pour les collections de documents, nouvelle combinaison et idées de découpage-bord (cuttingedge) de théorie probabiliste, analyse statistique, et techniques de compression de données.Ceci a mené au développement de diverses approches de recherche en utilisant un nouveau et un fort cadre probabiliste pour la RI, dans le but pratique de la combinaison efficace et effective de diverses sources pour augmenter la performance de recherche.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requêtes basés sur le Framework DFR (Divergence From Randomness). Comme tous les moteurs de recherche, Terrier possède les principales facettes suivantes :

Indexation: Permet l'extraction des termes des différents documents du corpus (basic indexed unit).

Recherche: Permet de générer des résultats aux requêtes formulées par les utilisateurs.

2. Installation de Terrier :

Pour pouvoir utiliser terrier il est nécessaire d'installer une JRE. La JRE ou la JDK peut être téléchargée sur le site de java.Puis allersur le site de Terrier pour le télécharger http://WWW.terrier.org

Ensuite dézipper la copie téléchargée dans le répertoire où vous souhaitez installer Terrier.

3. La structure des répertoires de Terrier :

Les répertoires de Terrier sont structurés comme suit :

bin\ : contient les scripts nécessaires pour démarrer Terrier.

doc\ : contient la documentation relative à Terrier.

etc \ : contient les fichiers de configuration de Terrier.

lib\ : contient les classes compilées de Terrier et les différentes librairies externes utilisées par Terrier.

licenses\ : contient les informations sur la licence des différents composants inclus dans Terrier.

share\ : contient la liste des mots vides (stopword list).

src\ : contient le code source de Terrier.

var/index : contient les structures de données : fichier inverse, fichier lexicon, index direct, document index.

var/result : contient les résultats de la recherche.

4. Les applications de Terrier :

Terrier offre trois applications qui sont :

- ✓ ***Batch TREC Terrier* :** permet l'indexation, la recherche et l'évaluation des résultats d'une collection TREC.
- ✓ ***Interactive Terrier* :** permet une recherche interactive en exécutant le script interactive_terrier. C'est un moyen facile pour tester Terrier.
- ✓ ***Desktop Terrier* :** c'est une interface graphique pour la plateforme Terrier.

5. L'API d'indexation de Terrier :

L'indexation dans Terrier est divisée en quatre étapes, et à chaque étape des plug-in (des classes java) peuvent être ajoutés pour la personnalisation du système. Les quatre étapes sont présentées comme suit :

a. Extraction de l'objet ***Document*** de la ***collection*** qui est générée par l'ensemble des corpus reçu en entrée par Terrier.

b. Parcourir chaque document de la collection pour en extraire les ***termes*** ainsi que les ***informations relatives***.

c. Traduction des termes extraits en utilisant ***TermPipelines*** (mots vide, lemmatisation).

d. La construction de l'index via l'utilisation de la classe ***Indexer***.

La figure suivante présente les différentes étapes du processus d'indexation dans Terrier.

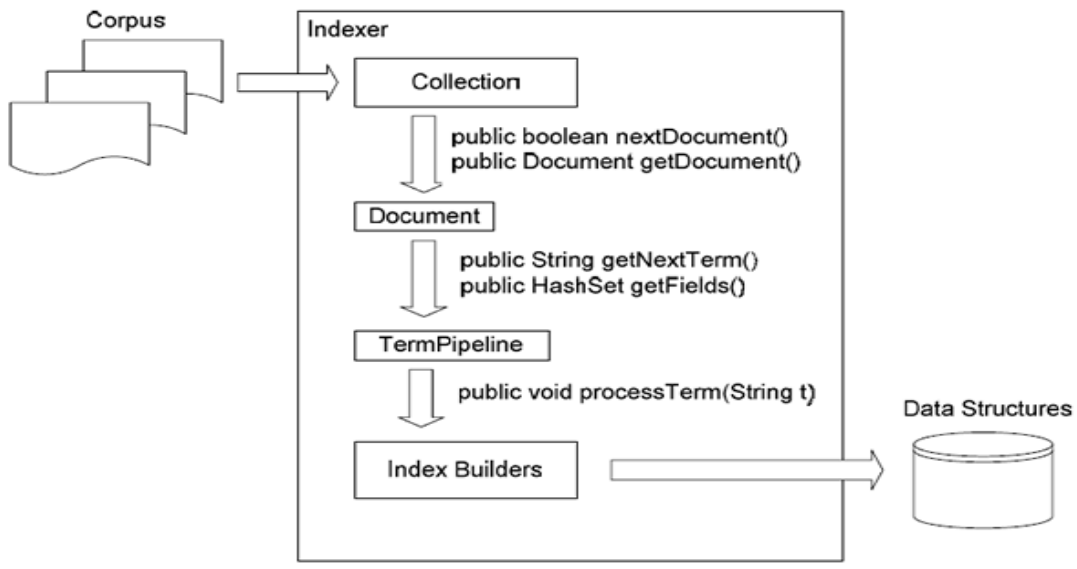


Figure 1 : Présentation du processus d'indexation dans Terrier

Dans ce qui suit, nous présentons les quatre étapes de ce processus :

5.1. Collection :cette composante englobe le concept le plus fondamental de l'indexation avec Terrier. C'est un objet qui représente le corpus, c'est-à-dire un ensemble de documents. ***Collection*** est une interface qui se trouve dans le package ***uk.ac.gla.terrier.indexing***. Elle est utilisée généralement par Terrier pour intégrer de nouvelles sources de données (nouveau corpus) et cela en ajoutant une nouvelle classe java qui implémente cette interface. Dans notre cas, nous avons utilisé la classe ***TRECCollection*** qui permet de traiter les collections de type TREC (à savoir ***AP88***). Elle permet de parcourir un ensemble de documents et de renvoyer un objet document en faisant appel à sa méthode ***nextDocument().*** Terrier offre plusieurs classes qui implémentent cette interface telles que : ***SimpleFileCollection, TRECUTFCollection, SimpleXMLCollection et TRECCollection.***

5.2. Document :c'est une interface qui se trouve dans le package***uk.ac.gla.terrier.indexing***. Cette composante englobe le concept de document. Les classes qui implémentent cette interface s'occupent de parcourir les documents et dont extraire les différents termes. Terrier possède plusieurs parseurs de documents, par exemple : ***HTMLDocument, FileDocument,MSExelDocument, etc.***

5.3. TermPipeline :c'est une interface qui se trouve dans le package ***uk.ac.gla.terrier.terms***. Cette composante reçoit l'ensemble des termes extraits des documents. Elle est considérée comme étant un pipeline qui traite et transforme chaque terme.

5.4. Indexer : cette composante est chargée de la gestion du processus d'indexation. Entre autre la construction de l'index et son écriture dans la structure de données appropriée. Terrier offre deux types d'indexeur ***: BasicIndexer*** et ***BlockIndexer***.

5.5. Les structures d'index : l'index de Terrier est composé de quatre structures de données principales qui sont :

- ✓ ***Vocabulary/Lexicon (data.lex)***
- ✓ ***Inverted Index (data.if)***
- ✓ ***Document Index (data.docid)***
- ✓ ***Direct Index (data.df)***

Structure d'Index	*Contenu*
Lexicon	***Term :*** nom du terme. ***Term id :*** identificateur du terme. ***Document Frequency :*** fréquence en document pour le terme. ***Term Frequency :*** fréquence globale dans la collection. ***Byte offset in inverted file :*** pour avoir la position dans le fichier inverse. ***Bite offset in inverted file :*** pour avoir la position dans le fichier inverse.
Inverted Index	***Document id :***identificateur du document d'appariement. ***Term Frequency :*** la fréquence du terme dans le document. ***Fields(#of fields bits) :*** les champs dans lesquels le terme apparait sont codés en utilisant un bit set. ***Block Frequency :*** fréquence dans une fenêtre. ***[Block id] :*** l'identificateur du bloc dans le document où le terme apparait.
Document Index	***Document id :*** identificateur de document. ***Document Length :*** la longueur de document. ***Document number :*** numéro de document. ***Byte offset in direct file :*** pour avoir la position dans le fichier directe. ***Bit offset in direct file :*** pour avoir la position dans le fichierdirecte.
Direct Index	***Term id :*** identificateur du terme. ***Term Frequency :*** la fréquence du terme. ***Fields (# of fields bits)*** ***Block Frequency :*** fréquence dans une fenêtre. ***[Block id] :*** identificateur du bloc.

Tableau 1 : présentation des structures d'index

L'indexation en Terrier peut être configurée en changeant les propriétés appropriées dans le fichier etc\terrier.properties. Chaque étape citée auparavant possède ses propres propriétés.

6. L'API de recherche de Terrier :

Terrier utilise trois composants principaux pour la recherche : Query, Manager, Matching qui seront décrits ci-dessous :

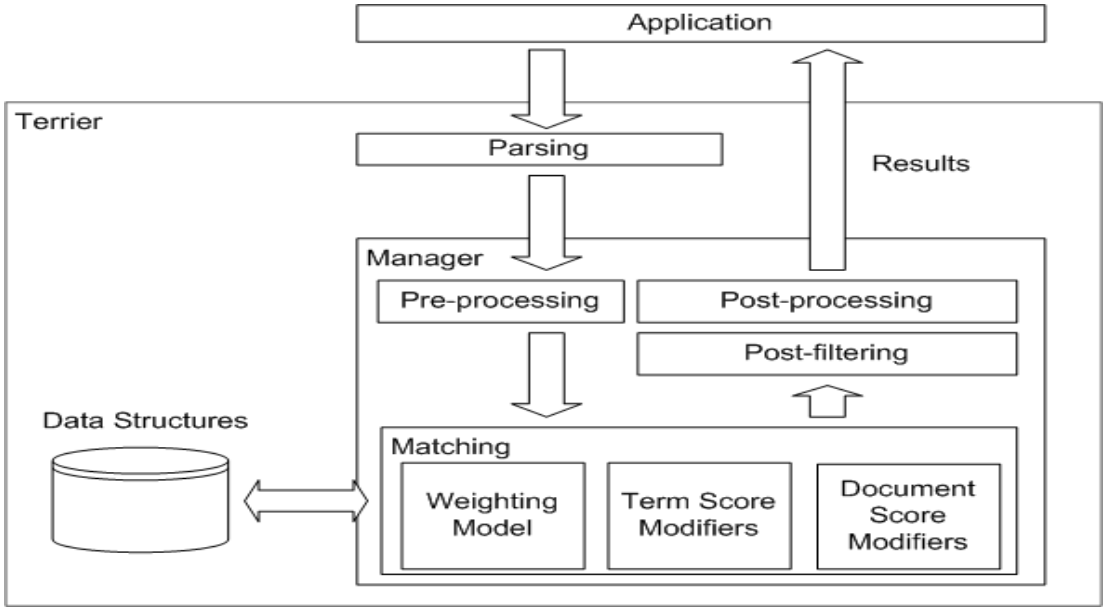


Figure 2 : Présentation du processus de recherche de Terrier

6.1. ***Query :***c'est l'entrée que l'application offre à Terrier. Le module ***Matching*** est responsable de determiner quel document correspond à une requête spécifique et attribut un score au document qui respecte la requête.Query est une classe abstraite, qui se trouve dans le ***package uk.ac.gla.terrier.querying.parser*** et qui modélise les requêtes. Elle consiste en un sub-queries ou bien en un query terms. Un objet query est créé pour chaque requête.

6.2. Manager :c'est le modèle qui est chargé de la gestion de la recherche. Dans un premier niveau, il lit chaque requête en utilisant le parseur approprié. Puis, il crée un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaires et en spécifiant un modèle de pondération.Puis, il crée un fichier résultat (result file) où il associe à chaque requête les documents qui lui sont pertinents. Le résultat de chaque requête est donné par le second niveau de gestion.Le second niveau de gestion est responsable de l'ordonnancement et de la coordination des opérations principales de haut niveau sur une seule requête. Ces opérations sont : ***Pre-processing, Matching, Post-processing et Post-filtering.***

6.3. Matching :ce composant est responsable de déterminer les documents correspondants à la requête spécifiée et donne un score au document qui vérifie la requête. Il utilise l'interface ***weighting models*** pour assigner un score à chaque mot de la requête dans le document. Terrier offre plusieurs classes qui implémentent cette interface, parmi elles : TF-IDF et BM25.

✓ ***Document Score Modifiers :***modifie le score des documents en fonction des propriétés du document.

✓ ***Term Score Modifiers :*** modifie le score des documents en fonction de la position des termes.

- ***Post-processing :***Après l'application de ***TermScoreModifiers*** ou***DocumentScoreModifiers***, l'ensemble de documents recherchés (retournés par l'opération de Matching) sera modifié en appliquant le ***Post-processing*** ou le ***Post-filtering***.***Le Post-processing*** est approprié pour implémenter des fonctionnalités, qui apportent un changement à la requête originale. Un exemple du ***Post-processing*** est l'expansion de requête (EQ), puis exécute une autre fois le ***Matching*** avec cette nouvelle requête. Un autre exemple possible de ***Post-processing*** est l'application de ***clustering***.
- ***Post-filtering :***c'est l'étape finale du processus de recherche de Terrier, où une série de filtres peut enlever les documents déjà recherchés, qui ne satisfont pas une condition donnée. Par exemple, dans le contexte d'un moteur de recherche Web, un Post filtre peut être utilisé pour réduire le nombre de documents recherchés depuis le même site Web, afin d'augmenter la diversité dans les résultats.

7. Modification de Terrier :

L'une des caractéristiques principales de Terrier est son extensibilité, il permet la modification du code source pour intégrer tout changement nécessaire. Pour que toute modification soit prise en compte, il est nécessaire de recompiler tout le code source de Terrier.

8. Utilisation de Batch (TREC) Terrier :

Dans cette section, nous allons montrer comment utiliser Batch (TREC) Terrier pour effectuer les deux opérations suivantes :

8.1. Indexation :

a) Aller dans le répertoire où Terrier est installé en utilisant la commande cd comme suit :

>> cd Prior_QE\terrier

Tel que Prior_QE contient la copie téléchargée de terrier.

b) Initialiser Terrier (supprimer le contenu du dossier index dans var) pour effectuer l'indexation d'une nouvelle collection TREC en utilisant l'option suivante :

>>.\bin\trec_setup<le chemin absolu du répertoire contenant les documents à indexer>

c) Maintenant nous pouvons indexer la collection TREC en utilisant l'option –i (indexer) comme suit :

>>.\bin\trec_terrier –i

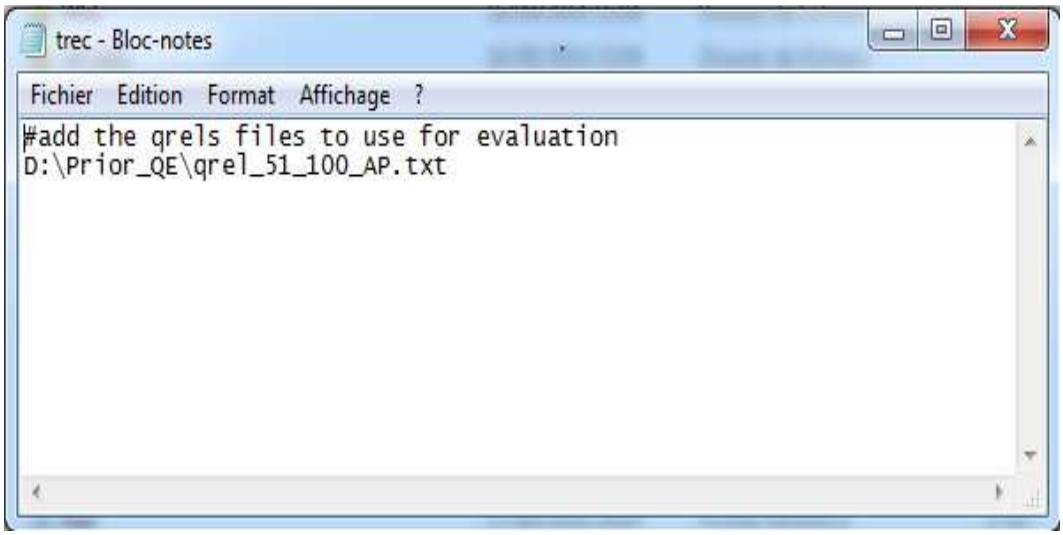
Remarque :

Pour indexer une collection autre qu'une collection TREC, il est nécessaire d'apporter quelques modifications au fichier terrier.properties.

8.2. Recherche :

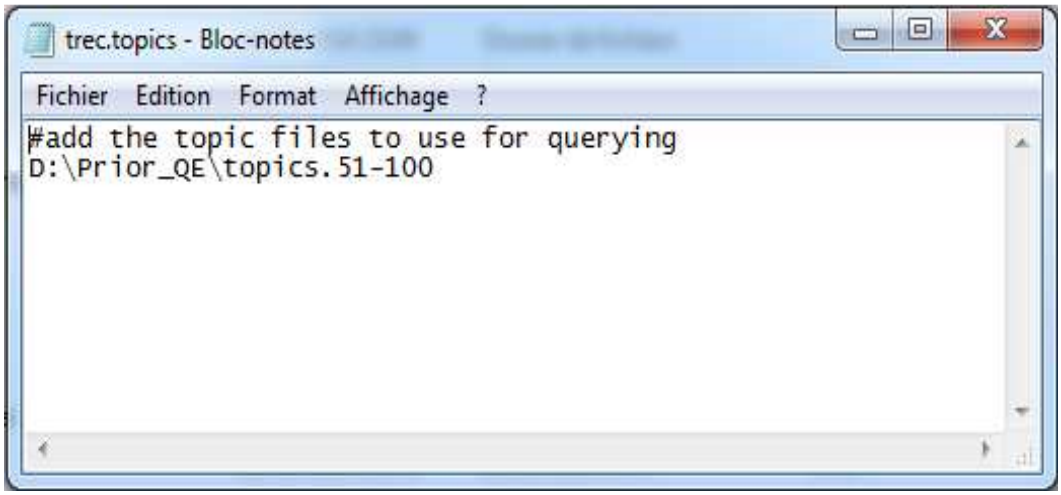
Avant toute recherche et évaluation, il est nécessaire de réaliser les trois étapes suivantes:

a) Spécification du fichier de jugement de pertinence (dans notre cas qrel_51_100_AP.txt) dans le fichier etc\trec.qrls.

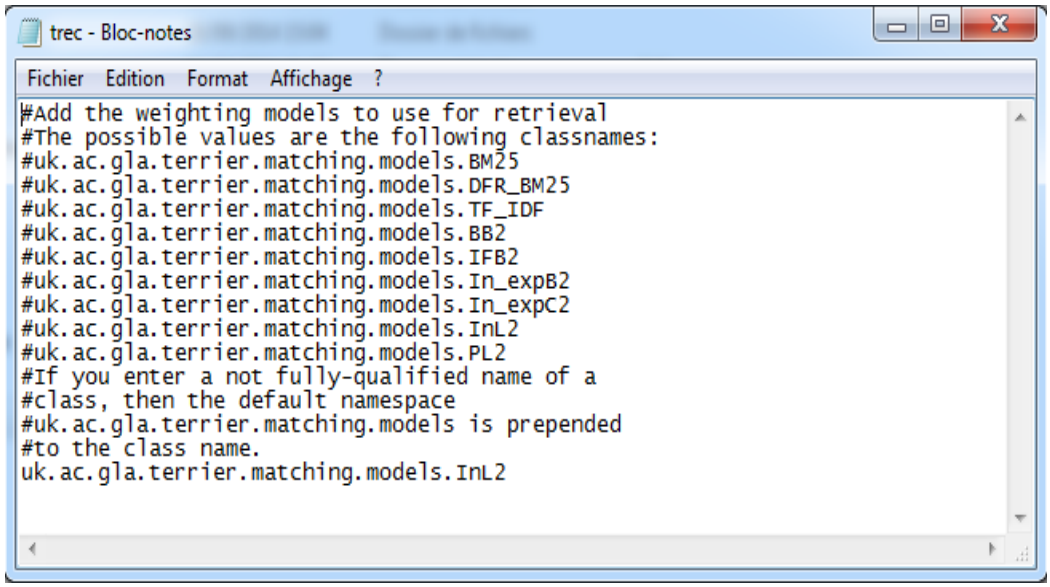


Tel que qrels_.txt est le fichier de jugement de pertinence.

b) Spécification du fichier contenant les requêtes (topics.51-100.txt) dans le fichier etc\trec.topics.list.



c) Spécification du modèle de pondération à utiliser dans le fichier etc\trec.models. Pour cela, il suffit de décocher le modèle choisie comme suit :



Maintenant, on peut lancer la recherche ou l'évaluation.

d) Pour lancer la recherche dans Terrier, on utilise l'option –r (retrieval) comme suit :

.\bin\trec_terrier –r

e) Les résultats de la recherche peuvent être évalués en exécutant la commande :

.\bin\trec_terrier -e