RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Électrique et d'Informatique

Département d'informatique

Mémoire de fin d'étude du Master académique

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Systèmes Informatiques

Thème :

Implémentation et évaluation d'une méthode de sélection de termes d'expansion basée sur les caractéristiques de la requête

Présentsé par :

M^{elle} BELLABIOD Kamilia

M^{elle} LALLOUCHE Dehbia

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

Président : Mr S.Sadou.

Encadreur : Mr A.Hammache.

Examinateur : Mr Y.Yacine.

Examinateur : Mr H.Radja.

Remerciements

D'abord nous remercions le bon Dieu de nous avoir donner santé, courage et foi pour réaliser ce travail.

Nous tenons à exprimer notre gratitude à notre promoteur monsieur Hammache.

Nous remercions infiniment madame Bellabiod Meriem qui nous a réalisé « le k-means ».

Enfin , un grand merci à tous ceux qui ont contribué au bon déroulement de ce modeste travail . En particulier nos chers parents, nos familles et tous nos amies.

Kamilia & Dehbia

Dédicaces

Nous dédions ce modeste travail à nos parents, nos familles, nos camarades, nos proches, nos amies et enfin toutes personne ayant contribué au bon accomplissement de notre projet.

Kamilia & Dehbia

Table des matières

Table des matières

Liste des figures

Listes de tableaux

Introduction générale

INTRODUCTION GÉNÉRALE

La recherche d'information est apparue comme une réponse au besoin de gérer l'explosion de la quantité d'informations. Très tôt, le monde de la recherche s'est intéressé aux systèmes de recherche d'information qui est un ensemble d'outils dans lesquels sont mis en œuvre des techniques et des mécanismes assurant la gestion automatique des informations documentaires, afin de répondre à un besoin d'information croissant.

Un système de recherche d'information a pour fonction de permettre à l'utilisateur d'accéder à des documents qui contribuent à combler son besoin d'information, exprimé sous forme de requête, qui motive sa recherche. Ainsi le système peut être vu par l'utilisateur comme un instrument de prédiction de la pertinence des documents d'un corpus par rapport à sa requête.

Ce travail se situe dans le domaine de la recherche d'information, plus particulièrement dans le cadre de la reformulation de la requête qui a pour objet de modifier la requête de l'utilisateur pour avoir une meilleure description de son besoin de sorte que les informations à retourner seront plus pertinentes.

Pour réaliser cette opération, généralement les premiers documents pertinents sont utilisés comme source de données pour extraire les termes d'expansion. Cependant cette méthode ne convient pas à tous les types de requêtes et précisément les requêtes ambigües, d'où la nécessité de la diversification dans la sélection des documents. La requête "Python" est ambigüe car elle renvoi à deux sens distincts : serpent et langage. Les résultats retournés a ce type de requête doit satisfaire les deux sens de la requête.

Notre travail consiste à mettre en œuvre cette approche qui consiste à choisir les termes d'expansion. Ce choix est basé sur les caractéristiques de la requête (La clarté de la requête).

Pour réaliser notre travail nous l'avons découpé en trois chapitres :

· Le 1er chapitre porte sur des généralités sur le domaine de la recherche d'information, notamment les concepts de base de la recherche d'information, Les modèles de la Recherche d'information et l'évaluation des systèmes de recherche d'information.

· Le 2eme chapitre porte sur l'expansion de requêtes et la diversification : nous définissons d'abord l'expansion de la requête, nous présentons ensuite les méthodes d'expansion de requêtes et le processus d'expansion automatique de la requête et nous décrivons l'expansion de la requête dans le modèle de langue. Enfin nous présentons les caractéristiques de requêtes, à savoir la taille et la clarté de celle-ci, ensuite nous allons définir et étudier la diversification pour résoudre le problème de requêtes ambigües.

· Le 3eme chapitre porte sur la présentation de l'approche proposée et l'expérimentation de l'approche en précisant les outils et langage utilisés pour sa mise en œuvre. Ainsi que les résultats obtenus.

Ce travail se termine par une conclusion générale et quelques perspectives.

Chapitre I:

La recherche d'information

I.1 Introduction

La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information **[1].**

La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations **[2].**

Le but est de pouvoir, parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur.

Ce chapitre a pour but de présenter le domaine de la recherche d'information. Dans la première partie, nous présentons les concepts de base de la recherche d'information (éléments et processus).Dans la seconde partie, nous parlons des modèles de recherche d'information. Dans la dernière partie de ce chapitre nous discutons de l'évaluation des systèmes de recherche d'information.

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

L'opération de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information **[4].**

Le système de recherche d'information (SRI) est un ensemble d'outils dans lesquels sont mis en œuvre des techniques et des mécanismes assurant la gestion automatique des informations documentaires, afin de répondre à un besoin d'information croissant.

La tâche principale d'un système de recherche d'information est de sélectionner dans une collection de documents ceux qui sont susceptibles de répondre aux besoins en information de l'utilisateur. Son but est de retourner à l'utilisateur le maximum de documents pertinents pouvant satisfaire son besoin et le minimum de documents non pertinents **[3].**

Le système de recherche d'information a pour but de mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents (fiche ou enregistrement) au moyen d'une fonction de comparaison (ou de correspondance) **[4].**

L'architecture générale d'un SRI est illustrée par la figure ci-dessous:

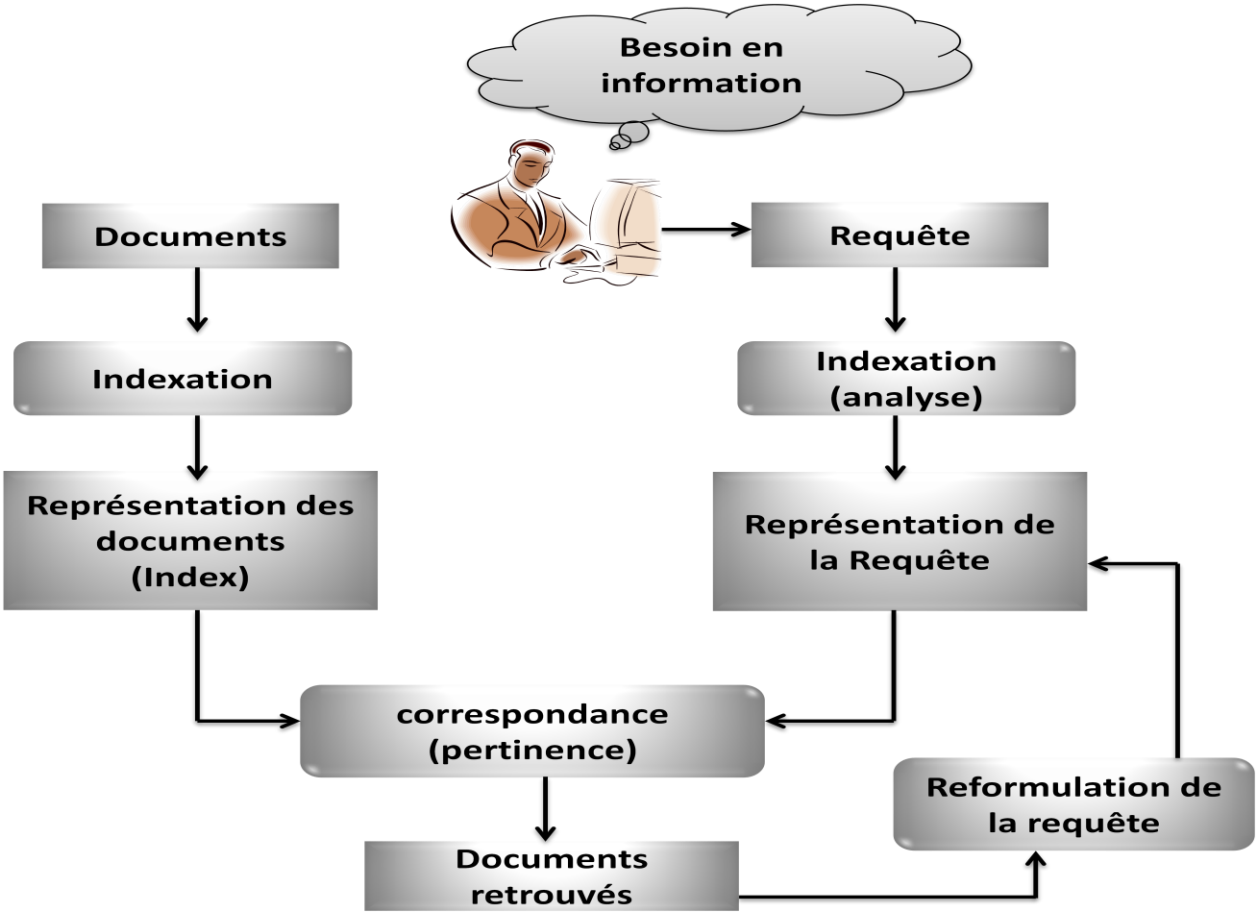

Figure I.3: Architecture générale d'un système de recherche d'information.

La figure I.1 représente le processus de RI: un utilisateur formule son besoin en information sous forme d'une requête, celle-ci est alors indexée par le système. La collection de documents est également indexée. Grâce à l'index (collection et requête indexées), le système est en mesure de construire les représentations puis de mettre en correspondance la représentation de la requête avec les représentations des documents de la collection. Puis il retourne une liste de documents considérés par le SRI comme pertinents par rapport à la requête utilisateur.

Les définitions de la RI et de SRI nous a permis de dégager les éléments suivants: le document, la requête et la pertinence, ainsi que trois principales fonctionnalités : L'indexation, l'appariement document-requête et la reformulation de la requête. Dans ce qui suit, nous détaillons ces éléments et ces processus.

I.2.1 Les éléments de la recherche d'information

Le but d'un système de recherche d'information est de retrouver des documents en réponse à une requête des utilisateurs, de manière à ce que les contenus des documents soient pertinents au besoin initial d'information de l'utilisateur.

Dans cette définition, il y a trois notions clés: documents, requête, pertinence.

I.2.1.1 Requête

Une requête exprime le besoin d'information de l'utilisateur écrite sous plusieurs formes. On distingue trois types de requêtes **[6]** :

- les requêtes basiques, la requête est un ensemble de mots-clés,
- les requêtes logiques (booléennes), la base des requêtes est un ensemble d'opérateurs logiques (et, ou et non),
- les requêtes structurées, ce type de requêtes porte des informations non seulement sur le contenu mais aussi des informations sur la structure des documents telles que en-têtes, titres. . .

I.2.1.2 Document

Un document est toute unité qui peut constituer une réponse à une requête d'utilisateur. Il peut être un texte, une page web, une image, une vidéo, etc.

I.2.1.3 Pertinence

La pertinence est la notion centrale dans la recherche d'information, car toutes les évaluations s'articulent autour de cette notion.

Le but de la RI est de trouver seulement les documents pertinents : un document pertinent est un document qui doit contenir l'information que l'utilisateur recherche.

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

La pertinence système : est déterministe, objective et elle est définie à travers les modèles de la recherche d'information. Elle est souvent traduite par un score évaluant l'adéquation du contenu des documents vis-à vis de celui de la requête **[5]**.

La pertinence utilisateur : 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. 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 à évolué.

I.3 Les processus de la recherche d'information

Pour répondre aux besoins en information de l'utilisateur, un système de recherche d'information met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans un fonds documentaire d'une part, et les besoins en information des utilisateurs d'autre part. Un système de recherche d'information est défini par ses modèles de représentation des documents, des besoins de l'utilisateur et sa fonction d'appariement **[6]**.

Ce processus est composé de trois fonctions principales : l'Indexation des documents et des requête, l'appariement requête-document et la reformulation de la requête.

I.3.1 Le processus d'indexation

L'étape d'indexation consiste à analyser les documents et les requêtes afin de créer une représentation de leur contenu textuel qui soit exploitable par le système de recherche d'information. Chaque document (et requête) est alors associé à un descripteur représenté par l'ensemble des termes d'indexation extraits **[7]**.

On distingue trois types d'indexation :

I.3.1.1 Indexation manuelle

Dans une indexation manuelle chaque document de la collection est analysé par un spécialiste du domaine ou un documentaliste. L'indexation manuelle assure une meilleure précision dans les documents restitués par le SRI en réponse aux requêtes des utilisateurs **[8]**.

Néanmoins, cette indexation est très couteuse en terme de temps et subjective du fait qu'elle est fondée sur le jugement de l'être humain. En conséquent, différents spécialistes peuvent indexer un document avec des termes différents. Il se peut même arriver qu'un spécialiste indexe différemment un document, à différents moments.

I.3.1.2 Indexation semi-automatique

L'indexation semi-automatique est un processus partiellement automatisé, le documentaliste intervient pour apporter les rectifications qu'il voit nécessaires **[9]**.

Le processus d'indexation se fait en premier lieu d'une manière automatique, le documentaliste intervient seulement pour ajouter ou supprimer des mots-clés qu'il trouve intéressants pour représenter un document **[6]**.

I.3.1.3 Indexation automatique

Dans l'indexation automatique, le processus d'indexation est complètement informatisé. Pour créer l'index d'une façon automatique, ce type d'indexation passe par un ensemble de traitements automatisés: l'analyse lexicale, l'élimination des mots vides, la normalisation (lemmatisation ou radicalisation), la sélection des descripteurs, le calcul de statistiques sur les descripteurs et les documents (fréquence d'apparition d'un descripteur dans un document et dans la collection, la taille de chaque document, etc.) et enfin la création de l'index et éventuellement sa compression. Nous détaillons ces différentes étapes ci-dessous.

a- L'analyse lexicale

L'analyse lexicale constitue la première étape du processus d'indexation. Sa fonctionnalité principale est de connaître une unité lexicale ou un radical **[10]**.

La mission de l'analyse lexicale est alors de transformer une suite de caractères en une suite de mots reconnaissables. Autrement dit, l'analyse lexicale consiste à découper le document en unités lexicales tels que chaque unité lexicale est une séquence de caractères entourée par des séparateurs d'unités.

b- L'élimination des mots vides

Les mots vides sont généralement des mots dits « grammaticaux » (comme les prépositions : à, de, les articles le, la, un, des, les pronoms : ce, lui ou encore les auxiliaires : être, avoir...etc.), ou des mots très fréquents au sein d'une collection de textes donnée **[7]**.

On distingue deux techniques pour éliminer les mots vides :

– L'utilisation d'une liste préétablie de mots vides (aussi appelée anti-dictionnaire ou stop-list).

– L'élimination des mots ayant une fréquence qui dépasse un certain seuil dans la collection.

b- La lemmatisation

La lemmatisation consiste à prendre la forme canonique du mot. Dans le document les mots peuvent apparaitre sous différentes formes.

La lemmatisation permet de substituer chaque mot par sa racine (lemme). La racine d'un mot est soit la forme infinitive si le mot est un verbe, soit la forme singulier masculin si le mot est un nom. L'utilisation de la lemmatisation contribue à l'amélioration des performances des systèmes de recherche d'information **[11]**.

c- Le choix des descripteurs

Elle consiste à déterminer le type d'unités élémentaires pour représenter les documents. On parle aussi de descripteur. L'objectif est d'avoir une représentation des documents permettant une moindre perte d'information sémantique possible **[12]**. On distingue plusieurs types de descripteurs, nous citons **[13] :**

Les mots simples : les mots simples du texte de document en éliminant les mots vides, les lemmes ou les racines des mots extraits.

Les N-grammes: qui sont une représentation originale d'un texte en séquence de N caractères consécutifs. On trouve des utilisations de bi-grammes et trigrammes dans la recherche d'information.

Les mots composés : groupes de mots ou expression (phrase en anglais) sont souvent plus riches sémantiquement que les mots qui les composent pris séparément.

Les concepts : qui sont des expressions pris généralement d'une structure conceptuelle, tels que les thésaurus ou les ontologies.

d- La création de l'index

Les index sont utilisés pour représenter le contenu des documents (substituts capables de les représenter).

Des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation afin d'accélérer la réponse à une requête. Les moyens de stockage les plus utilisés sont :

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

Le fichier inverse : Il est créé autour du fichier maître. 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 et les fréquences qui le constituent, on donne pour chaque mot les documents qui le contiennent et sa fréquence dans chacun des documents.

Le tableau I.2 illustre un exemple de fichier inverse construit à partir de la collection illustrée par le tableau I.1 : **[6]**

Document	Contenu
d_1	La(1) recherche(2) d'(3) information(4) gère(5) des(6) textes(7).
d_2	Un(1) système(2) de(3) recherche(4) d'(5) information(6) doit(7) restituer(8) l'(9) information(10) pertinente(11) à(12) l'(13) utilisateur(14).
d_3	Une(1) information(2) est(3) pertinente(4) si(5) elle(6) satisfait(7) l'(8) utilisateur(9).

Tableau I.4: Exemple d'un fichier maître.

Terme	d_1	d_2	d_3
recherche	2	4	
information	4	6,10	2
gère	5		
textes	7		
système		2	
restituer		8	
pertinente		11	4
utilisateur		14	9
satisfait			7

Tableau1.25: Exemple d'un fichier inverse.

I.3.2 L'appariement document-requête

La fonction d'appariement document-requête permet de mesurer la valeur de pertinence d'un document vis-à-vis d'une requête. Afin de réaliser cela, le système de recherche d'information représente le document et la requête avec un même formalisme, puis le système de recherche d'information compare les deux représentations. Le résultat de cette comparaison se traduit par un score qui détermine la probabilité de pertinence (degré de similarité ou degré de ressemblance) du document vis-à-vis de la requête. **[13]**

Ce score de pertinence est calculé à partir d'une fonction ou d'une mesure de similitude, notée : **RSV** (d, q) (**R**etrieval **S**tatut **V**alue)

d : un document de la collection.

q : la requête.

Il existe deux méthodes d'appariement :

Appariement exact (« exact match retrieval »)

Le résultat est une liste de documents respectant exactement la requête spécifiée avec des critères précis. Les documents retournés ne sont pas triés. **[14]**.

Appariement approché (« best match retrieval »)

Le résultat est une liste de documents sensés être pertinents pour la requête. Les documents retournés sont triés selon leur score de pertinence vis-à-vis de la requête. **[15]**

I.3.3 La reformulation de la requête

Dans les systèmes de recherche d'information, la requête initiale seule est souvent insuffisante pour permettre la sélection de documents répondant au besoin de l'utilisateur, il est difficile pour la plupart des utilisateurs de formuler la requête « idéale » qui va permettre de retrouver l'information exacte recherchée. Pour cette raison, plusieurs techniques ont été proposées pour améliorer les performances des SRI.

II.4 Les modèles de la Recherche d'information

L'indexation choisit les termes pour représenter le contenu d'un document ou d'une requête, le modèle permet de donner une interprétation des termes choisis pour représenter le contenu d'un document **[16].**

Un modèle de la Recherche d'Information modélise la fonction d'appariement qui joue un rôle central dans la RI.

Un modèle de la Recherche d'Information représente le noyau d'un Système de Recherche d'Information, c'est le modèle qui détermine le comportement clé d'un Système de Recherche d'Information. Il remplit les deux (2) principaux rôles suivants :

– Créer une représentation formelle, pour un document et pour une requête, basée sur leurs termes ;

–Définir une méthode de comparaison entre une représentation de document et une représentation de requête, pour déterminer leur degré de correspondance (similarité ou appariement).

Selon Baeza **[17]**, un modèle de la Recherche d'Information est défini par un quadruplet (D, Q, F, R (q, d)) : où

– D est l'ensemble de documents ;

– Q est l'ensemble de requêtes ;

– F est le schéma du modèle théorique de représentation des documents et des requêtes ;

– R (q, d) est la fonction de pertinence du document d à la requête q.

Nous pouvons distinguer trois (3) grandes classes de modèles regroupés selon les fondements mathématiques sur lesquels ils se basent :

– Ensemblistes, par exemple le modèle booléen, basé sur la théorie des ensembles ;

– Algébriques, comme le modèle vectoriel, basé sur la théorie d'algèbre ;

– Probabilistes, basés sur la théorie des probabilités et estiment des probabilités de pertinence d'un document vis-à-vis de la requête.

Nous présentons dans ce qui suit les modèles les plus connus :

II.4.1 Les modèles booléens

Le modèle booléen **[18]**, est le premier modèle de la Recherche d'Information à avoir été mis en œuvre, même aujourd'hui beaucoup de systèmes commerciaux (moteurs de recherche) utilisent le modèle booléen. Cela est dû à la simplicité et à la rapidité de sa mise en œuvre.

Ce modèle est basé sur la théorie des ensembles et de l'algèbre de Boole.

Le modèle booléen de base

Dans le modèle booléen de base, les requêtes sont exprimées par des expressions booléennes, dont les termes sont reliés par des opérateurs logiques : OR(∨), AND(∧), NOT (¬) [**13],** permettant d'effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associés à chaque terme.

Dans ce modèle chaque document est représenté par une conjonction logique des termes non pondérés qui constitue l'index du document.

La pertinence (noté par RSV (q, d) [pour Retrieval Status Value]) entre une requête (q) et un document (d) est déterminée de la manière suivante :

–Si la requête (q) contient un seul (1) terme, soit q= t (avec t : un terme)

$$\text{RSV}= \begin{cases} 1 & si\ t \in à\ d \\ 0 & sinon \end{cases} \quad (I.1)$$

–Si la requête (q) contient deux (2) termes reliés par l'opérateur AND, soit q= $t_1 \wedge t_2$:

$$\text{RSV (d, q)} = \begin{cases} 1 & si\ \text{RSV}(d, t_1) = 1 \text{ et } \text{RSV}(d, t_2) = 1 \\ 0 & sinon \end{cases} \quad (I.2)$$

–Si la requête (q) contient deux (2) termes reliés par l'opérateur OR, soit q=$t_1 \vee t_2$:

$$\text{RSV (d, q)} = \begin{cases} 1 & si\ \text{RSV}(d, t_1) = 1 \text{ ou } \text{RSV}(d, t_2) = 1 \\ 0 & sinon \end{cases} \quad (I.3)$$

–Si la requête (q) est composée de négation d'un seul (1) terme, soit q= ٦t :

$$\text{RSV (d, ٦q)} = \begin{cases} 1 & si\ \text{RSV}(d, q) = 0 \\ 0 & sinon \end{cases} \quad (I.4)$$

Exemples

La représentation d'une requête (q) est comme suit : q = $(t_1 \wedge t_2) \vee (t_3 \wedge t_4)$.

La représentation d'un document (d) est comme suit : d = $t_1 \wedge t_2 \wedge t_3 \ldots \wedge t_n$.

La représentation de la pertinence entre un document (d) et une requête (q) est comme suite RSV (d, q).

Ce modèle présente un certain nombre d'avantages :

– Il est efficace à condition que l'utilisateur maîtrise bien le langage de requêtes.

– Ce modèle est facile à implémenter.

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

– Les documents retournés à l'utilisateur ne sont pas ordonnés selon leur pertinence.

– Il est difficile pour les utilisateurs de formuler de bonnes requêtes. Par conséquent, l'ensemble des documents trouvés est souvent trop grand, pour les requêtes courtes, ou complètement vide dans le cas de requêtes longues.

– Les documents qui ne contiennent pas tous les termes de la requête sont considérés automatiquement comme non pertinents.

– Tous les termes dans un document ou dans une requête ont le même poids.

Tous l'art est de trouver le moyen permettant une utilisation plus flexible des opérateurs booléens. Le modèle booléen étendu **[19]** est l'un des premiers modèles qui ont été proposés à cette fin, nous trouvons aussi le modèle booléen basé sur la théorie des ensembles flous **[20]**, **[21].**

II.4.2 Les modèles vectoriels

Le modèle vectoriel de base (nommé aussi VSM pour Vector Space Model), a été introduit par Gerard Salton **[14]**, concrétisé dans le cadre du système SMART.

C'est un modèle algébrique qui se base sur une formalisation géométrique. En effet, les documents et les requêtes sont représentés dans un même espace, défini par un ensemble de dimensions, chaque dimension représente un terme d'indexation.

Les requêtes et les documents sont alors représentés par des vecteurs, dont les composantes représentent le poids du terme d'indexation considéré dans le document (requête).

Supposons nous avons un espace T= $\{t_1, t_2, \dots, t_j, \dots, t_n\}$ termes d'indexation de dimension *n*, un document (d_i) et est une requête (q) sont définis de la manière suivante :

–Un vecteur document $d_j = (w_{i1}, w_{i2}, \dots, w_{ij}, \dots, w_{in})$.

–Un vecteur requête $q = (w_{q1}, w_{q2}, \dots, w_{qj}, \dots, w_{qn})$.

Où

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

La pondération est une fonction très importante en recherche d'information (RI), elle permet de mesurer l'importance ou le poids d'un terme dans un document.

Dans la littérature, plusieurs schémas de pondération ont été proposés. La majorité de ses schémas prennent en compte la pondération locale et la pondération globale **[22].**

Pondération locale

La pondération locale permet de :

–Mesurer l'importance du terme dans le document.

–La pondération locale prend en compte les informations locales du terme qui ne dépendent que du document.

–La pondération locale correspond en général à une fonction de la fréquence d'occurrence du terme dans le document (noté tf pour *t*erme *f*requency), exprimée de la manière suivante :

– $tf_{ij} = 1 + \log\ (f(t_i, d_j))$ (I. 5)

Où $f(t_i, d_j)$ représente la fréquence du terme t_i dans le document d_j.

Nous pouvons citer d'autres fonctions de calcul de tf (pour *t*erme *f*requency) :

– $tf_{ij} = \mathrm{n_{oc}}$ (I. 6)

– $tf_{ij} = [0|1]$ Présence, absence (I. 7)

– $tf_{ij} = \frac{\mathrm{n_{oc}}}{tf_{\max}}$ (I. 8)

– – $tf_{ij} = 0.5 + 0.5\frac{\mathrm{n_{oc}}}{tf_{\max}}$ … (I. 9)

Avec

– $\mathrm{n_{oc}}$: représente le nombre d'occurrences du terme dans le document ;

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

Pondération globale

La pondération globale prend en compte :

–Les informations concernant le terme dans la collection.

Un facteur de pondération globale est alors introduit. Ce facteur nommé idf (pour *i*nverted *d*ocument *f*requency), c'est la fréquence du terme dans l'ensemble des documents, exprimé comme suit :

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

– $idf = \log\left(\frac{N-n_i}{n_i}\right)$ (I. 11)

– $idf = \log\left(\frac{N}{n_i}\right) + 1$ (I. 12)

– …

Où

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

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

Les fonctions de pondération

Les fonctions de pondération combinant la pondération locale et globale sont référencées sous le nom de la mesure $(tf \times idf)$.

Cette mesure donne une bonne approximation de l'importance du terme dans les collections de documents de taille homogène.

Cependant, elle ne tient pas compte d'un aspect du document : sa longueur.

En effet, la mesure $(tf \times idf)$ favorise les documents longs, car ils ont tendance à répéter le même terme, ce qui accroit leur fréquence, par conséquent augmentent la similarité de ces documents vis-à-vis de la requête.

Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération, comme facteur de normalisation **[23]**, **[24].**

La comparaison de la requête au document dans le modèle vectoriel est effectuée en évaluant la similarité entre leurs vecteurs respectifs. Les documents considérés comme les plus

pertinents sont ceux dont le vecteur est le plus proche de celui de la requête, en tenant compte d'une mesure de similarité définit au préalable.

Plusieurs mesures de similarité ont été définies **[25]**, dont les plus utilisées sont décrites de la manière suivante :

–Le produit scalaire: $\mathbf{RSV(q, d_i)} = \sum_{j=1}^{|T|} w_{qj}.w_{ij}$ (I. 13)

– La mesure de cosinus : $\mathbf{RSV(q, d_i)} = \frac{q.d_i}{||q||.||d_i||} = \frac{\sum_{j=1}^{|T|} w_{qj}.w_{ij}}{\sqrt{\sum_{j=1}^{|T|} w_{qj}{}^2.\sum_{j=1}^{|T|} w_{ij}{}^2}}$ (I. 14)

– La mesure de Dice: $\mathbf{RSV}(q, d_i) = \frac{2 \times \sum_{j=1}^{|T|} w_{qj}.w_{ij}}{\sum_{j=1}^{|T|} w_{qj}{}^2 + \sum_{j=1}^{|T|} w_{ij}{}^2}$ (I. 15)

– La mesure de Jaccard: $\mathbf{RSV}(q, d_i) = \frac{\sum_{j=1}^{|T|} w_{qj}.w_{ij}}{\sum_{j=1}^{|T|} w_{qj}{}^2 + \sum_{j=1}^{|T|} w_{ij}{}^2 - \sum_{j=1}^{|T|} w_{qj}.w_{ij}}$ (I. 16)

Le modèle vectoriel est caractérisé par :

– Une représentation uniforme des documents et des requêtes.

– Le modèle vectoriel permet de pallier à l'un des inconvénients majeurs du modèle booléen, en permettant de trier les documents répondant à une requête.

– Le modèle prend en considération le poids des termes dans les documents.

– Permet de retrouver des documents qui répondent partiellement à une requête.

– Les documents sont restitués dans un ordre décroissant de leur degré de similarité avec la requête.

Malgré la large utilisation du modèle vectoriel, il présente un inconvénient majeur :

– Il repose sur l'hypothèse de l'indépendance des termes d'indexation, or que ces termes dans les documents sont sémantiquement liés.

Plusieurs variantes du modèle vectoriel ont été proposées, pour remédier à cette limitation. Parmi elles, nous trouvons le modèle vectoriel généralisé **[26]** (**GVSM** pour **G**enaralized **V**ector **S**pace **M**odel) proposé par Wong & al en 1985 qui lève l'hypothèse d'indépendance des termes, le modèle **LSI** (pour **L**atent **S**emantic **I**ndexing) **[27]**, **[28]**, **[29]**, **[30]**, **[31]** et le modèle connexionniste **[32]**, **[33]**.

II.4.3 Les modèles probabilistes

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête **[34]**, **[35]**, **[36].**

Nous présentons ci-dessous deux(2) types de modèles probabilistes :

–Le modèle probabiliste de base.

–Le modèle de langue.

II.4.3.1 Le modèle probabiliste de base

Le modèle probabiliste de base trie les documents selon leur probabilité de pertinence vis-à-vis d'une requête.

La fonction de classement (tri) de ce modèle est exprimée ainsi :

$$\text{–RSV (q, d)} = \frac{P(Per \mid q, d_i)}{P(NPer \mid q, d_i)} \qquad (I.17)$$

Le principe de base de cette fonction consiste à retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents.

Où

–P (Per | q, d_i) : représente la probabilité qu'un document d_i soit pertinent (Per) vis-à-vis de la requête (q) ;

–P (NPer | q, d_i) : représente la probabilité qu'un document d_i soit non pertinent (NPer) vis-à-vis de la requête(q).

Selon la formule de bayes, les deux (2) probabilités sont obtenues de la façon suivante :

$$\text{–} \qquad P(\text{Per} \mid \text{q}, \text{d}_\text{i}) = \frac{P(\text{Per} \mid \text{q}).P(\text{d}_\text{i} \mid \text{Per,q})}{P(d_i)} \qquad (I.18)$$

$(I.11)$

ET

$$\text{–}P(\text{NPer} \mid \text{q}, \text{d}_\text{i}) = \frac{P(\text{NPer} \mid \text{q}).P(\text{d}_\text{i} \mid \text{NPer,q})}{P(d_i)} \qquad (I.19)$$

Où

–P (d_i) représente la probabilité de choisir le document (d_i);

–P (d_i | Per, q) représente la probabilité que le document (d_i) fait partie des documents pertinents pour la requête (q);

–P (d_i | NPer, q) représente la probabilité que le document (d_i) fait partie des documents non pertinents pour la requête (q);

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

Après remplacement dans la fonction de tri, nous aurons la formule suivante :

$$\text{– RSV (q, d)} = \frac{P(d_i \mid Per,q)}{P(d_i \mid NPer,q)} \qquad (I.20)$$

Le calcul de probabilité dans ce modèle est trop complexe, il a pour avantage de restituer les documents sélectionnés comme pertinents dans l'ordre de leurs pertinences.

II.4.3.2 Le modèle de langue

L'utilisation des modèles de langue en RI remonte à 1998, Pont et Croft **[37]** furent les premiers à proposer ce modèle.

L'objectif de base de leur modèle est que l'utilisateur a une idée (un modèle) des termes qui sont présents dans le document pertinent ou modèle de ce document (M_d).

A partir de document (M_d), 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).

La probabilité de génération de la requête à partir de modèle de document est considérée comme la pertinence du document, elle est présentée de la manière suivante :

$$-\text{RSV}(q, d) = P(q|M_d) \qquad (I.21)$$

Le modèle de langue utilise une distribution de Bernoulli, c'est-à-dire que les mots présents et ceux qui sont absents dans la requête sont pris en compte. La pertinence du document vis-à-vis de la requête est alors exprimée de cette manière :

$$-\text{RSV}(q, d) = \prod_{t \epsilon q} P(t|M_d) \times \prod_{t \notin q} (1 - P(t|M_d)) \qquad (I.22)$$

Cette pertinence est composée de deux(2) parties :

– La probabilité d'observation les termes de la requête dans le document ;

– La probabilité de ne pas observer les termes absents de la requête dans le document.

La probabilité P ($q|M_d$) est calculée par une méthode non paramétrique qui utilise :

– La probabilité moyenne d'apparition du terme (t) dans les documents qui le contient $P_{avg}(t)$;

– 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 présenté sous cette forme :

$$pr(t|M_d) = \begin{cases} P_{ML}(t|d)^{(1-R(t,d))} \times P_{\text{Avg}}(t)^{R(t,d)} & si\ tf\ (t,d) > 0 \\ \frac{tf(t,C)}{|C|} & sinon \end{cases} \qquad (I.23)$$

Où

– tf (t, d) : représente la fréquence de terme (t) dans le document (d) ;

– tf (t, c) : représente la fréquence du terme (t) dans la collection (C).

Dans le modèle proposé par Hiemstra **[38]**, la requête (q) est considérée comme une séquence de terme. Dans ce modèle nous ne considérons que les mots présents dans la requête (q), une distribution multinomial est utilisée.

La pertinence d'un document vis-à-vis d'une requête (q) (la probabilité de générer une requête (q)) sachant le modèle de document (M_d) est donnée sous cette forme :

– $RSV(q,d) = P(d)\prod_{i=1}^{n} P(t_i|M_d)$ (I. 24)

Où

– P(d) représente 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 document (M_d) avec le modèle de langue de la collection (C). Le calcul de cette probabilité est exprimé comme suit :

– $P(t_i|M_d) = \lambda P_{ML}(t_i|M_d) + (1-\lambda)P_{ML}(t_i|C)$ (I. 25)

Où

– λ : représente 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).

Le calcul des deux (2) probabilités $P_{ML}(t_i|M_d)$ et $P_{ML}(t_i|C)$ est réalisé selon une estimation vraisemblance maximale, exprimée ainsi :

– $P_{ML}(t_i|M_d) = \frac{tf(t_i,d)}{|d|}$ (I. 26)

ET

– $\mathrm{P_{ML}(t_i|C)} = \frac{\mathrm{d}f(t_i)}{\sum_{j\epsilon V}^{t} \mathrm{d}f(t_j)}$ (I. 27)

Où

– $tf(t_i,d)$: représente la fréquence du terme (t_i) dans le document (d) ;

– $\mathrm{d}f(t_i)$: représente le nombre de documents contenant le terme (t_i) ;

– V : représente le vocabulaire d'index.

Miller et al **[39]** ont proposé un modèle similaire à celui de Hiemstra.

La différence entre les deux états, illustrés par la figure suivante :

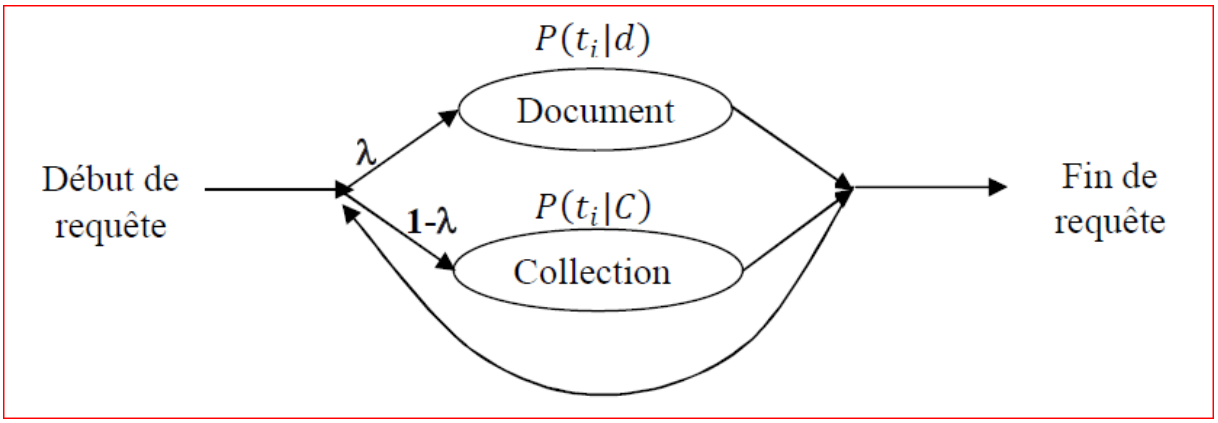

Figure I.2 : Modèle de Markov à deux(2) états

Ce modèle correspond à la formule suivante :

$$- P(\mathrm{q}|\mathrm{M_d}) = \prod_{t \epsilon q}[\lambda P_{ML}(\mathrm{t_i}|\mathrm{M_d}) + (1-\lambda)P_{ML}(\mathrm{t_i}|\mathrm{C})] \quad (\mathrm{I.28})$$

L'estimation de la probabilité P $(t_i|M_d)$ est réalisée de la même manière que le modèle précédent(Hiemestra). Par contre, la probabilité $P(t_i|C)$ est estimée différemment :

$$- \mathrm{P_{ML}}(\mathrm{t_i}|\mathrm{C}) = \frac{\mathrm{t}f(t_i)}{\sum_{t_j \epsilon V} tf(t_j)} \quad (\mathrm{I.29})$$

Où

– $tf\ (t_i)$: représente la fréquence du terme (t_i) dans le corpus ;

– $\sum_{t_j \in V} tf(t_j)$: représente le nombre de termes dans le corpus ;

– V : représente le vocabulaire d'index.

I.5 Evaluation des SRI

L'évaluation des méthodes et des modèles de Recherche d'Information (RI) a été un centre d'intérêt important et cela dès l'apparition des premiers Système de Recherche d'Information (SRI), les premières évaluations datent de 1953 **[40]**.

Plusieurs quantités mesurables ont été proposées pour l'évaluation d'un Système de Recherche d'Information (SRI):

– Le temps de réponse ;

– La pertinence ;

– La qualité ;

– La présentation des résultats ;

– Les performances de stockage (espaces mémoire nécessaire pour le stockage d'index par exemple)

L'évaluation des Système de Recherche d'Information (SRI) est abordée selon deux (2) angles différents :

– Paradigme système : vise à évaluer les performances du système essentiellement en termes de qualité des documents retournés par le système.

– Paradigme usager : vise à satisfaire l'utilisateur, et non les capacités du système.

L'approche « système » est la plus utilisée dans la Recherche d'Information (RI), elle se base sur deux (2) éléments essentiels :

– Mesures d'évaluation des SRI (Système de Recherche d'Information) ;

– Les collections de test.

I.5.1 Mesures d'évaluation des SRI

Le principal objectif d'un système de recherche d'Information est de restituer à l'utilisateur tous les documents pertinents et de rejeter tous les documents non pertinents.

Cet objectif est évalué à l'aide de différentes mesures d'évaluation **[41]**. Nous présentons ci-dessous les plus utilisées:

– La précision (P) ;

– Le rappel (R) ;

– Le Bruit (B) ;

– Le silence (S) ;

– Les mesures alternatives.

La figure suivante illustre la répartition des documents face à une requête :

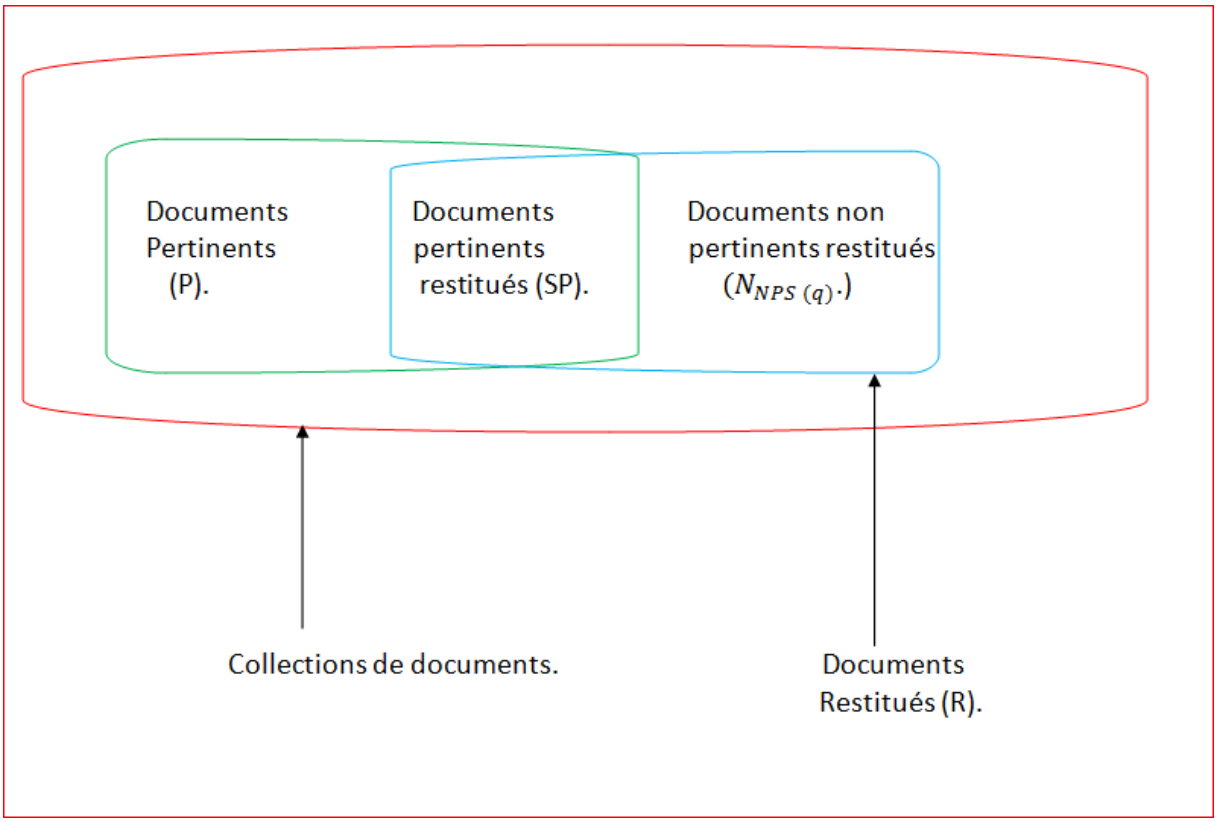

Figure I.3 : Représentation des partitions de la collection lors d'une interrogation.

Avec :

– R : représente le nombre de documents sélectionnés par le Système de Recherche d'Information (SRI) pour une requête (q) ;

– P : représente le nombre de documents pertinents dans la collection pour la requête (q) ;

– $N_{NPS(q)}$: représente le nombre de documents non pertinents sélectionnés dans la collection pour la requête (q) ;

– SP : représente le nombre de documents pertinents sélectionnés dans la collection pour la requête (q).

I.5.1.1 La précision (P)

La précision est le rapport du nombre de documents pertinents restitués par le système (SP) sur le nombre total de documents restitués (R), exprimée ainsi :

$$-précision = \frac{SP}{R} \qquad (I.30)$$

La précision mesure la capacité du système à trouver seulement les documents pertinents. Elle vaut 1 si tous les documents retrouvés sont pertinents, et elle vaut 0 si aucun document retrouvé n'est pertinent.

I.5.1.2 Le rappel(P)

Le rappel est calculé par le rapport du nombre de documents pertinents restitués(SP) sur le nombre total de documents pertinents(P), exprimé ainsi :

$$-rappel = \frac{SP}{P} \quad (I.31)$$

Quand le rappel vaut 1 c'est que le nombre total de documents pertinents ont été retrouvés par le système. Quand il vaut 0 alors aucun document pertinent n'a été retrouvé.

Des mesures complémentaires au rappel et précision ont été définies, il s'agit de bruit et de silence.

I.5.1.3 Le bruit(B)

La mesure d'évaluation « bruit » est une notion complémentaire à la précision, elle est définie par cette première méthode:

$$- B = 1 - P \quad (I.32)$$

Où

– P : représente la précision du Système de Recherche d'Information (SRI).

Le taux de bruit est le pourcentage exprimant le rapport entre le nombre de documents non pertinents restitués et le nombre total de documents restitués, il est défini par cette deuxième méthode :

$$- B(q) = \frac{N_{NPS(q)}}{N_{S(q)}} \quad (I.33)$$

I.5.1.4 Le silence(S)

La mesure d'évaluation «silence» est une notion complémentaire au rappel, elle est définie par cette première méthode:

$$- S = 1 - R \quad (I.34)$$

Où :

– R est le rappel du Système de Recherche d' Information (SRI).

Il mesure la difficulté du système à retrouver tous les documents pertinents, elle est définie par cette deuxième méthode :

$$- S\ (q) = \frac{N_{NPS(q)}}{SP} \quad (I.35)$$

I.5.1.5 Les mesures alternatives

Plusieurs mesures alternatives ont été proposées, parmi elles, nous trouvons :

–La précision exacte(ou R-Précision)

C'est une précision qui est calculée sur les « R » premiers documents retournés par le système, sachant que la requête admet « R » documents pertinents.

–La précision moyenne non interpolée(MAP)

La précision moyenne non interpolée (**A**verage **M**ean **P**recision) est une mesure de performance globale, elle est calculée en deux étapes :

Première étape

Nous calculons la précision moyenne pour une requête donnée (AP_q), et nous calculons pour chaque document pertinent restitué sa précision (pr(d_i)) qui est égale au nombre de documents pertinents restitués sur le rang de ce document ; pour les documents retrouvés non pertinents leur précision est égale à zéro (0).

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

$$- AP_q = \frac{1}{N}\sum_{i=1}^{N} pr(d_i) \qquad (I.36)$$

Avec :

$$- pr(d_i) = \begin{cases} \frac{r_{n_i}}{n_i} & si\ d_{ij} est\ retrouvé \\ 0 & sinon \end{cases} \qquad (I.37)$$

Où

– n_i: dénote le rang du document d_i qui a été retrouvé et qui est pertinent pour la requête ;

– r_{n_i}: représente le nombre de documents pertinents retrouvé au rang (ni) ;

– N : représente le nombre total de documents pertinents pour la requête (q).

Deuxième étape

Nous calculons la précision moyenne pour un ensemble de requête, en effectuant la moyenne de la précision moyenne de chaque requête, elle est exprimée ainsi :

$$- \text{MAP} = \frac{1}{M}\sum_{j=1}^{M} AP_{q_j} \qquad (I.38)$$

Où

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

I.5.2 Les collections de test

Une collection de test représente le moyen d'évaluation des Système de Recherche d'Information (SRI). Elle comprend généralement :

– Un corpus de documents : c'est un ensemble de documents à indexer, sur lesquels le système sera évalué.

– Une liste de requêtes prédéfinies.

– Les jugements de pertinence : ils sont établis manuellement et ils contiennent la liste des documents pertinents pour chaque requête.

L'évaluation d'un Système de Recherche d'Information (SRI) consiste à comparer les résultats retournés par ce dernier par rapport aux jugements de pertinence.

Depuis les années 70, les collections de test se sont multipliées. Elles sont mises en place dans le cadre de compagne d'évaluation des Systèmes de Recherche d'Information (SRI), dont la collection CACM , la collection CISI, la compagne CLEF et la compagne TREC qui constitue à ce jour la compagne de référence pour ce qui concerne l'évaluation des Systèmes de Recherche d'Information (SRI) et cela depuis l'année de son lancement 1992 **[42]**, **[43].**

L'objectif de la compagne TREC est de proposer une plate-forme qui réunit :

–Les collections de test ;

–Les tâches spécifiques ;

–Les protocoles d'évaluation pour chaque tâche afin de mesurer les différentes stratégies de recherche **[44].**

Les tâches proposées se sont diversifiées d'une compagne à une autre ; parmi les tâches proposées dans TREC 2012 nous pouvons citer :

– Recherche d'information sur le web ;

– Recherche d'information médicale ;

– Recherche d'information dans les micros blogs ;

– Recherche d'information contextuelle.

I.6 Conclusion

Dans ce chapitre, nous avons discuté, en général, des principaux concepts de la RI. Nous avons, particulièrement, introduit des éléments de base : requête, document et pertinence. Nous avons aussi décrit les processus de base de la RI : l'indexation, l'appariement requête-document et la reformulation de la requête. Enfin, nous avons traité l'évaluation des systèmes de recherche d'information ainsi ses mesures alternatives.

Ces différents concepts traités dans ce chapitre, nous aiderons à mieux comprendre les notions fondamentales pour mener à bien notre travail.

Le chapitre suivant est consacré à la présentation des deux concepts clés de notre travail à savoir l'expansion de la requête et la diversification des résultats de recherche.

Chapitre II:

Expansion de requêtes & Diversification

II.1 Introduction

Les performances d'un Système de Recherche d'Information, mesurées en général par la double mesure rappel-précision, dépendent d'une part de l'efficacité du modèle de recherche mise 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 des 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 (SRI) proposent des techniques, appelées expansion de requête, basée sur les caractéristiques de celles-ci pour améliorer automatiquement la requête initiale de l'utilisateur.

Ce chapitre est consacré à cette technique qui consiste à ajouter des termes à la requête initiale, dans lequel nous allons définir deux (2) partie, la première est consacrée pour l'expansion de requêtes, et la seconde est consacrée pour les caractéristiques de celles-ci.

II.2. Expansion de requêtes

Dans les Systèmes de Recherche d'Information, souvent la requête initiale est insuffisante pour permettre de sélectionner des documents pertinents qui répondent aux besoins des utilisateurs.

Pour cela nous allons présenter la technique d'expansion de requêtes qui permet de résoudre ce problème.

Dans cette première partie, nous allons présenter les méthodes d'expansion de requêtes, et les principales approches d'expansion de celles-ci, ensuite nous allons détailler le processus d'expansion automatique de la requête, et enfin nous allons présenter l'expansion de requêtes dans le modèle de langue.

II.2.1 Définition de l'expansion de requêtes

L'expansion de requêtes est une méthode conçue pour améliorer le processus de Recherche d'Information. Elle est utilisée dans le domaine de conception des Système de Recherche d'Information adaptatifs aux besoins des utilisateurs. C'est un processus qui permet de construire une nouvelle requête plus adéquate à la Recherche d'Information dans

l'environnement du Système de Recherche d'Information, comparée à celle formulée initialement par l'utilisateur.

L'expansion de la requête consiste donc à ajouter de nouveaux termes à la requête initiale ou alors revoir le poids des ses termes dans le but de cibler la recherche vers les documents pertinents **[45]**.

Plusieurs autres définitions ont été attribuées à l'expansion de requêtes :

– Abberley **[46]** : définit l'expansion de requêtes comme un moyen qui permet de reformuler les requêtes et d'améliorer le processus de Recherche d'Information.

– Elle est considérée selon **[47]** : comme un processus qui a pour but de préciser et d'éclaircir les résultats en permettant à l'utilisateur de modifier sa requête afin d'améliorer la pertinence de ses résultats.

– Efthimiadis **[48]**, en plus de proposer des classifications des méthodes d'expansion de requête, il a aussi donné les définitions suivantes : " l'expansion de requête est un processus qui vise à compléter la requête initiale en proposant des termes supplémentaires, elle est considérée comme une amélioration de la Recherche d'Information ".

Le schéma suivant permet de représenter les principales approches des Système de Recherche d'Information par l'expansion de requête initiale :

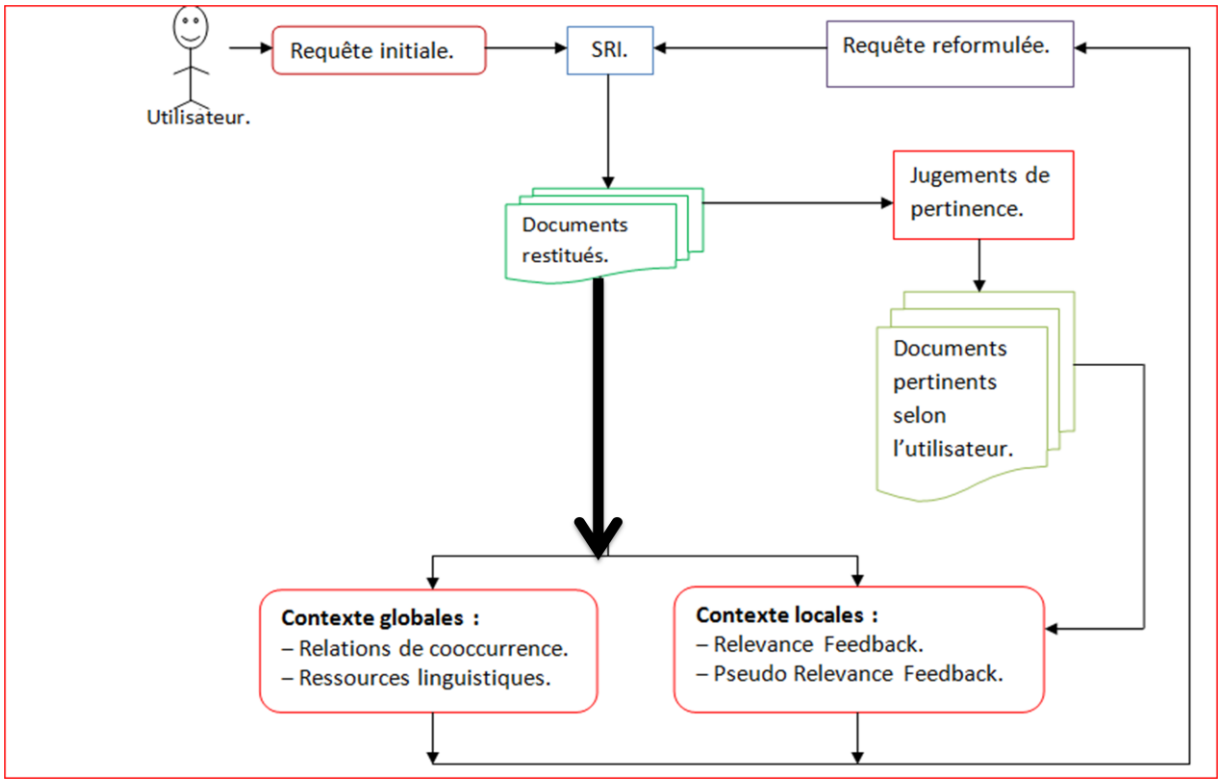

Figure II.1: Processus d'amélioration des SRI par expansion de requêtes.

II.2.2 Les méthodes d'expansion de requêtes

Nous décrivons ci-dessous les facteurs de classifications des méthodes d'expansion de requêtes ainsi les principales approches de l'expansion de celle-ci.

II.2.2.1 Les facteurs de classifications des méthodes d'expansion de la requête

Les techniques d'expansion de requêtes peuvent être classées selon trois (3) principaux facteurs **[49]**:

– Classification selon l'intervention de l'utilisateur ;

– Classification selon la source de données utilisée ;

– Classification selon la méthode de sélection des termes.

II.2.2.1.1 Classification selon l'intervention de l'utilisateur

L'expansion de la requête peut être réalisée par l'utilisateur (dite manuelle), ou par le système (dit automatique) comme elle peut être réalisée conjointement par l'utilisateur et le système, dans ce cas elle est dite semi-automatique ou interactive **[50]**.

Cependant, le rôle de l'utilisateur dans ce processus peut être actif/passif ; il est actif dans l'expansion manuelle et l'expansion interactive de la requête, et passif dans l'expansion automatique de celle-ci.

a) L'expansion 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. Nous pouvons procéder à l'expansion de la requête en utilisant un vocabulaire contrôlé (thésaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes pour compléter sa requête **[51]**.

b) L'expansion semi-automatique

Cette technique nécessite l'intervention de l'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 l'expansion. Le système joue un rôle important 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 **[51].**

c) L'expansion automatique (directe)

L'expansion 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 lui. Cette expansion 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 (équivalence, association, hiérarchie), 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. La construction du thesaurus se fait généralement pendant le processus d'indexation.

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 (pondération des termes). Il s'agit de chercher des associations de termes afin d'ajouter des termes voisins à la requête. Parmi les thesaurus construits automatiquement, nous pouvons citer un thesaurus basées sur les similarités **[52]**, un thesaurus statistique **[53]**, ou bien des mini-thesaurus construits seulement d'après la requête et à partir de techniques de clustering **[54]**.

Dans ce cadre d'expansion, nous pouvons citer également la réinjection de pertinence automatique (réinjection de pertinence aveugle), dans laquelle nous utilisons le même principe de la réinjection de pertinence mais en considérant les **n** premiers documents renvoyés par le système comme pertinents **[55]**, **[56].**

Le problème avec l'expansion 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 entrainer un silence ou au contraire augmenter un bruit **[51].**

II.2.2.1.2 Classification selon la source de données utilisée

Différentes sources de données sont utilisées pour reformuler la requête initiale. Elles peuvent être des ressources externes **[57]**, telles que les ontologies, les thésaurus et la relation de cooccurrence ente termes dans la collection. Les méthodes basées sur ces ressources sont dite méthodes globales, ou des documents résultats de la première recherche ; les méthodes basées sur ces ressources sont dites méthodes locales. Ces méthodes sont également connues sous le nom de réinjection de pertinence.

La réinjection de pertinence a montré son efficacité avec différents modèles de la Recherche d'Information et affiche de meilleurs résultats que les méthodes globales **[58]**.

II.2.2.1.3 Classification selon la méthode de sélection des termes

Une méthode permet de sélectionner les termes à ajouter à la requête initiale, plusieurs méthodes existent, nous trouvons :

– La relation de cooccurrence ;

– Les mesures d'information ;

– Les techniques de classification.

II.2.3 Principales approches de l'expansion de requêtes

Plusieurs approches ont été proposées pour l'expansion de requêtes afin d'améliorer les performances des Système de Recherche d'Information (SRI).

L'expansion de requête a été traitée selon deux (2) classes :

– Approche basée sur le contexte global ;

– Approche basée sur le contexte local ;

II.2.3.1 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 (2) techniques de construire un thésaurus :

– Manuelle ;

– 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 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 spécifiques **[59]**.

Cependant, les thésaurus manuels sont peu utilisés par les Système de Recherche d'Information car leur construction et la maintenance des informations sémantiques qu'ils contiennent sont couteuses en temps et nécessitent le recours aux experts des domaines considérés **[59]**.

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 métonymie.

Ces relations seront détaillées dans ce qui suit.

Nous allons définir quelques thésaurus utilisés en Recherche d'Information (RI) :

– Thésaurus basés sur la hiérarchie :

Nous pouvons 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 constitue 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 collection.

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 **[60]**.

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**, est aussi des mots contenus dans les sens synonymes de ce **synset**.

1) 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, <<légume>> est un hyperonyme de <<carotte>> et de << courgette>>.

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

2) 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>>, <<chat>> 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**, et aussi des mots contenus dans les sens fils de ce **synset**.

3) 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>>.

4) 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 **[61]**. Pour le construire, au lieu de représenter un document par un vecteur de termes, nous représentons 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 **[61]** pour calculer le poids d_{ik} est la suivante

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

Où

– $ff(d_k, t_i)$: 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.

– $maxff(t_i)$: 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 :

$$-Sim(t_i, t_j) = \vec{t_i}.\vec{t_j} = \sum_{k=1}^{n} t_{ik} * t_{jk} \quad \text{(II.2)}$$

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 suivante :

$$Sim_{qt}(Q, t_j) = \sum_{t_i \epsilon Q} q_i * Sim(t_i, t_j) \quad (II.3)$$

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 :

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

Le nombre de termes à rajouter à la requête est un paramètre important en Recherche d'Information (RI) **[61]**. L'étude empirique effectuée sur trois (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 **[62]**.

a) 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 nous utilisons souvent dans un contexte local des synonymes relatifs au phénomène.

II.2.3.2 Approche basée sur le contexte local

La méthode d'analyse du contexte local a été développé par Croft et Xu **[63],** 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 :

$$- O(Q,c) = \prod_{t_i \epsilon Q} \left(\delta + \frac{\log(af_{c,t_i}) * idf_c}{\log(n)}\right)^{idf_i} \quad \text{(II.5)}$$

Où

– $(af_{c,t_i}) = \sum_{j=1}^{n} ft_{ij} * fc_j$;

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

– ft_{ij} : représente le nombre d'occurrences du terme t_i dans le passage P_i ;

– fc_j : représente le nombre d'occurrences du concept c dans le passage P_j ;

– N : représente le nombre de passage dans la collection ;

– N_i : représente de passages contenant le terme t_i ;

– N_c : représente de passages contenant le concept c.

– δ : représente 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 technique de réinjection de pertinence est l'une des méthodes de L'expansion 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 **[64]**.

Le processus de réinjection de pertinence, comporte principalement trois étapes :

– L'échantillonnage ;

– L'extraction des évidences ;

– La réécriture de la requête.

1. *L'échantillonnage* : est une étape qui permet de construire un échantillon de documents à partir des éléments jugés non pertinents et le nombre 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 **[65]** 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 : nous parlons alors de la réinjection de pertinence à itérations multiples **[64]**.

Le schéma suivant illustre le processus de réinjection de pertinence :

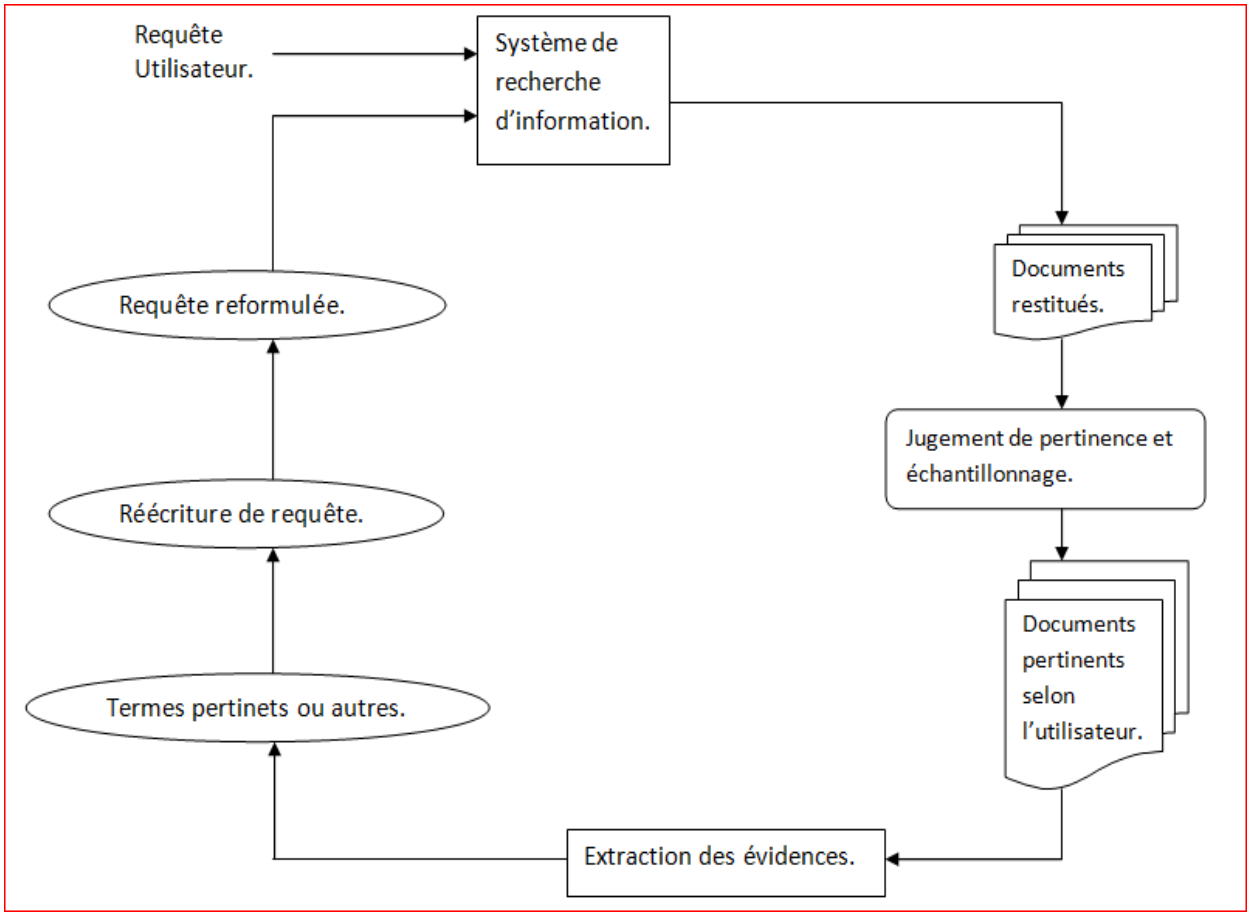

Figure II.2: Processus générale de la réinjection de pertinence [64].

D'une manière générale, la phase d'échantillonnage ne présente pas de problème spécifique. Le seul point abordé à ce niveau concerne le nombre d'élément à évaluer pour pouvoir effectivement constituer un échantillon représentatif **[64]**.

La problématique principale de la réinjection de pertinence réside dans les deux (2) 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 **[64]**, **[65]**.

Dans la plupart des approches de la littérature, les deux (2) 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 **[65]**.

La technique de réinjection de pertinence a été mise en place à l'origine dans le modèle vectoriel. Rocchio **[65]** a proposé le modèle de reformulation de requête suivant :

$$Q_N = \alpha . Q_O + \beta . \frac{1}{|R|} \sum_{r \epsilon R} r - \frac{1}{|R'|} \sum_{r' \epsilon R'} r' \quad \text{(II.6)}$$

Où :

− Q_N : représente le vecteur de la nouvelle requête (reformulée) ;

− Q_O : représente le vecteur de la requête originale ;

− R : représente l'ensemble des vecteurs r des documents jugés pertinents par l'utilisateur ;

−R' : représente l'ensemble des vecteurs r' des documents jugés non pertinents par l'utilisateur ;

− α, β , δ: représentent les paramètres de la reformulation.

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

Dans le modèle probabiliste, la réinjection de pertinence est mise en place directement dans le modèle de mesure de pertinence. Elle consiste à revoir les poids des termes de la requête **[66]**, comme suit :

$$- w_{q_j} = log \frac{r' + 0.5/(r - r' + 0.5)}{(df_j - r' + 0.5)/(n - df_j - r + r' + 0.5)} \quad \text{(II.7)}$$

Où :

– r : représente le nombre de documents pertinents ;

– r': représente le nombre de documents pertinents contenant le terme q_j ;

– df_j : représente le nombre de documents contenant le terme q_j ;

– n : représente le nombre total de documents dans la collection.

a) La réinjection par pseudo feedback (réinjection aveugle)

Ces méthodes de reformulation nommées aussi, pseudo-réinjection de pertinence (ou blind) sont effectuées de manière automatique. Elles se basent sur l'hypothèse que les

documents les mieux classés (les premiers) sont considérés comme pertinents. Le système utilise alors les premiers documents pour reformuler la requête.

Selon Croft et Harper **[67]**, cette technique souffre d'un problème majeur qui est appelé << query drift>>. Ce problème apparait l'ensemble des documents pertinents (ou ne contient pas du tout). Donc cette technique ne fonctionne bien que dans le cas où la requête initiale permet d'extraire de bons résultats (beaucoup de documents pertinents).

Le schéma suivant explique la procédure de réinjection automatique :

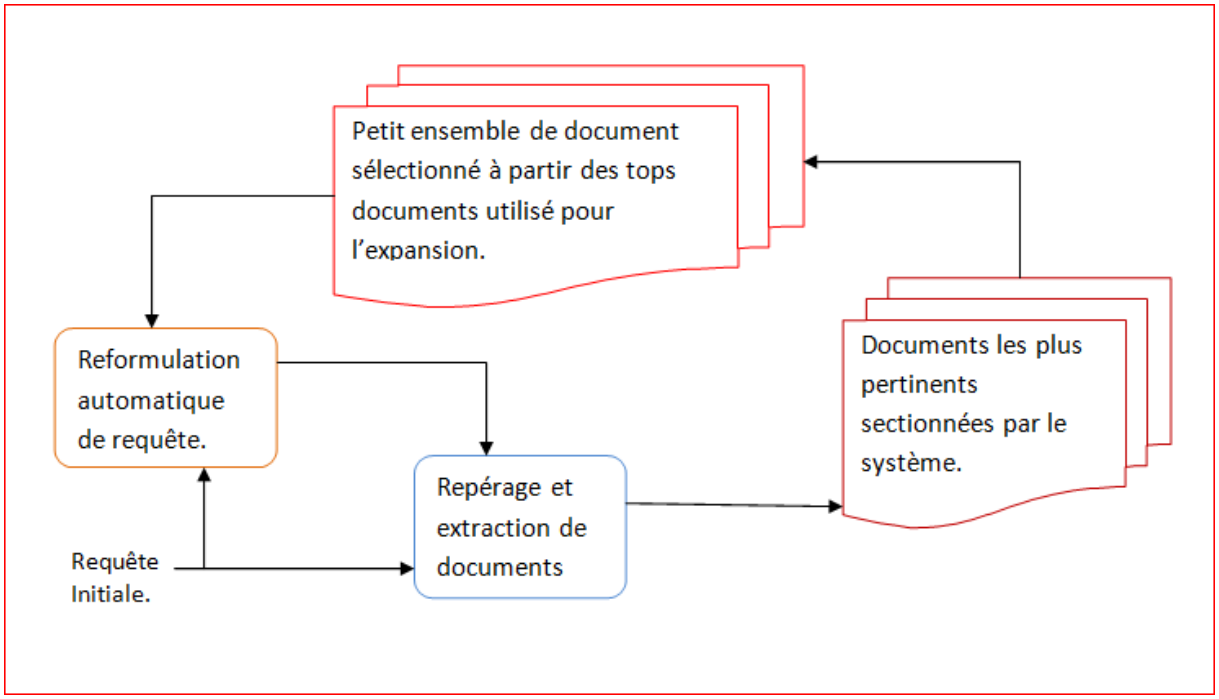

Figure II.3 : Processus de fonctionnement du blind RF.

La variante de la formule de Rocchio (formule ((II.6)) pour la réinjection automatique de la requête est exprimée par la formule suivante :

$$Q_N = \alpha . Q_O + \beta . \frac{1}{|R|} \sum_{r \epsilon R} r \qquad \text{(II.8)}$$

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

Plusieurs travaux **[68]**, **[69]** ont tenté d'évaluer l'impact de la pseudo-réinjection, en variant le nombre de termes à rajouter à la requête. Ils montrent que la performance du système est obtenue lorsque la requête est construite entre 20 et 40 termes.

II.2.4 Le processus d'expansion automatique de la requête

Le processus d'expansion de requêtes contient quatre (4) principales étapes que nous allons définir par la suite, présentées dans la figure (Figure II.4) :

– Traitement de données ;

– Génération et classement des termes ;

– Sélection des termes ;

– Reformulation de la requête.

La requête initiale de l'utilisateur et la source de données, sont considérées comme l'entrée du processus et la requête reformulée comme sa sortie.

Nous schématisons ce processus avec la figure suivante :

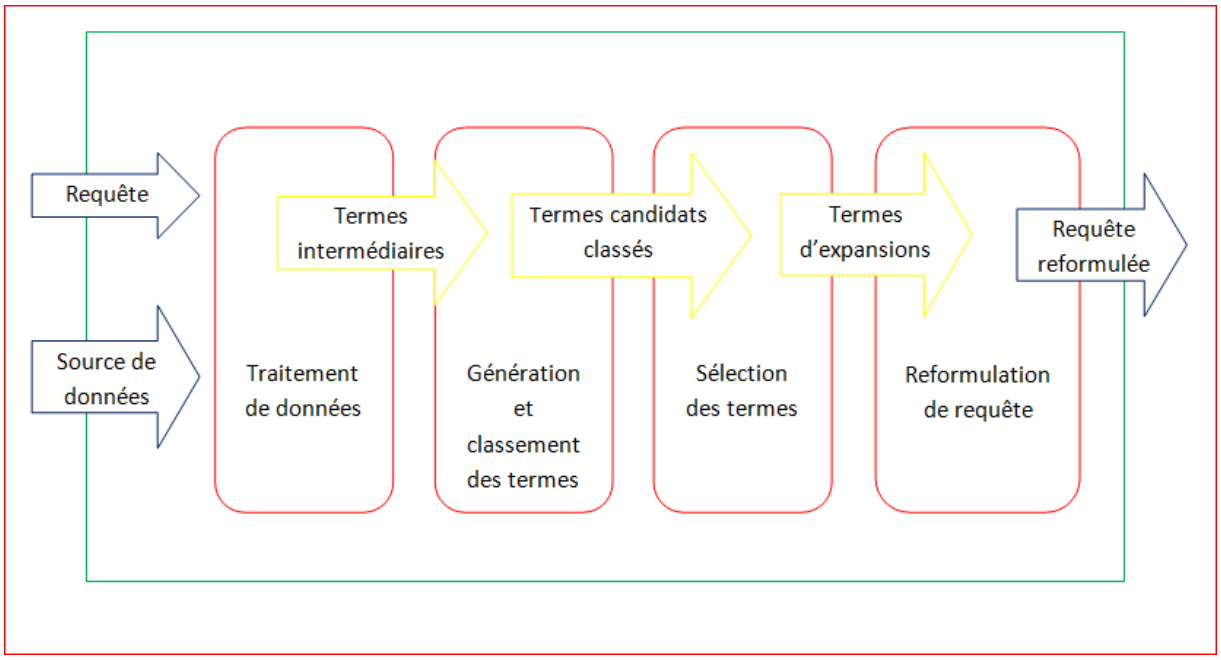

Figure II.4: le processus d'expansion automatique de la requête.

II.2.4.1 Traitement de données

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

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

De nombreuses techniques d'expansion de requêtes sont basées sur les informations contenues dans les tops documents, extraits en réponse à la requête de l'utilisateur initiale à partir d'une collection de documents.

Dans cette étape de traitement de données, il est indispensable d'indexer la collection et d'exécuter la requête.

En conséquence, chaque document est représenté comme un ensemble de termes pondérés, avec un fichier complémentaire inversé de l'indice qui associe les termes du document aux termes de la requête. Le système d'indexation peut également stocker les positions des termes afin de fournir la recherche basée sur la proximité. Lorsque la collection utilisée pour l'expansion de la requête est la même que celle en cours de recherche, le système de classement à lequel la requête élargie sera soumise est généralement utilisé pour effectuer aussi un premier passage de classement. Si un corpus externe est utilisé
(par exemple, des données Web pour les recherches sur l'Internet, ou données de bureau personnel pour des recherches sur le Web), comme dans **[70]**, **[71]**, **[72]**, **[73]**, en générale un Système de Recherche d'Information (SRI) différent, est nécessaire ; plusieurs options sont disponibles, telles que l'installation et le fonctionnement d'un moteur de recherche de bureau(commercial ou disponible gratuitement), en utilisant les récupération API Web, ou encore le développement de son propre système pour l'indexation des documents et le classement.

D'autres techniques d'expansion de requêtes, basées sur l'analyse du corpus, nécessitent l'extraction des termes particuliers de la collection manuellement, qui sont généralement différents de ceux utilisés à des fins d'indexation par un Système de Recherche d'Information (SRI) classique. Une approche bien connue de Qiu et Frei **[74]**, où chaque terme est représenté comme un vecteur de documents pondéré en utilisant les statistiques d'une collection non-standard.

Un autre exemple de Crouch et Yang **[75]**, qui construit un thésaurus de statistique en regroupant d'abord l'ensemble de la collection de documents via l'algorithme complet de clustering.

Il existe peu de travaux réalisés pour la sélection des documents comparativement à la sélection des termes, la méthode présentée dans **[76]** 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 de D_{rel} par l'intermédiaire d'une recherche effectuée sur le corpus.

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

Deux (2) grandes familles de méthodes sont présentées pour la sélection de documents :

La première :

Elle estime la représentativité d'un document indépendamment des autres documents dans D_{rel} , en sélectionnant les k-tops documents classés, cette méthodes assigne au documents d (appartient à D_{rel}) une pertinence, RSV(d) qui reflète la représentativité de d dans D_{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 méthodes Random :

Elle affecte une pertinence à chaque document selon la fonction suivante :

$$- RSV_{(random)}(d)^{def} = \frac{1}{n} \quad \text{(II.9)}$$

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

Où

– n : représente le nombre de documents dans la collection.

– La méthode QuerySim :

Elle considère le document (d) qui a une similarité élevée avec la requête comme un bon représentant et le calcul du la pertinence (RSV) pour chaque document se fait selon la fonction suivante :

$$- RSV_{QuerySim}(d)^{def} = sim(q, d) \quad \text{(II.10)}$$

– La méthode basée sur la longueur d'un document :

Elle suppose que les documents pertinents courts sont les meilleurs représentants que les documents longs pertinents :

$$- RSV_{length}(d)^{def} = -|d| \quad \text{(II.11)}$$

Où

– $|d|$: représente la taille de document d

La deuxième :

Elle est basée sur l'hypothèse suivante : les documents représentatifs sont semblables les une des autres en exploitant les relations (similitude) entre les documents dans D_{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.

$$- p(w\backslash Cent(D_{rel}))^{def} = \frac{1}{n}\sum_{d\epsilon\, D_{rel}} p(w\backslash d) \quad \text{(II.12)}$$

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

$$score_{\text{centroid}}\,(d)^{def} = -|d| * \left(p.\backslash cent(D_{rel})\right)||p(.\backslash d)\right) \quad \text{(II.13)}$$

– Les méthodes basées sur le graphe:

Certains travaux sur le reclassement de la première liste de 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, 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.2.4.2 Génération et classement des termes candidats d'expansion

Dans la deuxième étape de l'expansion automatique de la requête, le système génère et classe les termes candidats d'expansion.

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

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

Nous classons les techniques utilisées pour effectuer la génération et le classement des termes candidats selon :

– Le type de relation entre les termes d'expansion générés ;

– Les termes de la requête initiale.

Il existe deux (2) types d'association:

– Association un-à-un.

– Association un-à-plusieurs.

II.2.4.2.1 Association un à un

La forme la plus simple de génération et de classement des termes candidats est basée sur les associations un-à-un entre les termes d'expansion et les termes de la requête initiale, c'est-à-dire, chaque terme d'expansion est associé à un terme unique de la requête initiale.

Dans la pratique, un ou plusieurs termes d'expansion sont générés et marqués pour chaque terme de la requête avec l'aide d'une variété de techniques.

L'une des ces techniques consiste à s'appuyer sur les associations linguistiques, comme l'utilisation d'un algorithme de lemmatisation qui consiste à regrouper les mots d'une même famille et les réduire en une entité appelée lemme (forme canonique).

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

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

L'idée générale est que les deux termes sont sémantiquement liés lorsqu'ils apparaissent dans les même documents, tout comme deux documents sont considérés comme similaires s'ils contiennent les même termes. Deux (2) simples mesures de similarité sont :

– Le coefficient Dice (D) ;

– L'indice de Jaccard (J).

Compte tenu des termes u et$v$, le coefficient Dice (D) est définit comme suit :

$$- \mathrm{D} = \frac{2.\mathrm{d}f_{u \wedge v}}{\mathrm{d}f_u + \mathrm{d}f_v} \qquad (\mathrm{II}.14)$$

Où

– $\mathrm{d}f_{u \wedge v}$: représente le nombre de documents qui contiennent à la fois les termes u et v ;

– $\mathrm{d}f_u, \mathrm{d}f_v$: représentent les nombres de documents contenants les termes u et v respectivement.

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

– $\mathrm{J} = \frac{\mathrm{d}f_{u \wedge v}}{\mathrm{d}f_{u \vee v}}$ (II. 15)

Où

– $\mathrm{d}f_{u \vee v}$: représente le nombre de documents contenants le terme u ou le terme v.

Une approche plus générale est définit dans ce qui. Si on considère une matrice **A** terme-document où chaque cellule $\mathbf{A_{t,d}}$ représente le poids $\mathbf{W_{t,d}}$ pour le terme (t) dans le document (d), et si on considère une matrice **C** de similarité terme-à-terme alors est calculée ainsi :

– $\boldsymbol{C} = \boldsymbol{A}A^t$ (II. 16)

Où

– $\mathrm{C}_{u,v}$: représente le degré de similarité entre le terme u et le terme v. $\mathrm{C}_{u,v}$

–$\mathrm{C}_{u,v}$ est donnée comme suit :

–$\mathrm{C}_{u,v} = \sum \mathrm{d}_j \, \mathrm{W}_{u,j} . \mathrm{C}_{v,j}$ (II. 17)

Ainsi, la corrélation entre chaque terme de la requête et chaque terme dans la collection peut être calculée à l'aide de la formule définit ci-dessus et pour introduire la notion de fréquence des termes, il est préférable de générer des facteurs de similarité normalisés, comme la mesure de cosinus donnée par la formule suivante :

– $\frac{\mathrm{C}_{u,v}}{\sqrt{\sum \mathrm{d_j} \mathrm{W}_{u,u}^2 \sum \mathrm{d_j} \mathrm{W}_{v,v}^2}}$ (II. 18)

La formule (II. 17) peut produire différentes méthodes de similarité terme-à-terme, en se basant sur la façon dont les documents et la fonction de pondération sont choisis.

Une technique bien connue proposée par Attar et Fraenkel dans **[77]** repose sur l'ensemble des documents retournés en réponse à la requête initiale et utilise la fréquence des termes pondérés.

La cooccurrence des termes dans l'ensemble du document est simple, mais la position des termes pose un inconvénient car elle n'est pas prise en compte, alors que deux termes qui apparaissent loin l'un de l'autre dans un document. Cet aspect est généralement abordé en considérant la proximité des termes c'est-à-dire en utilisant des documents textuels restreints tels que les documents de longueur fixe pour mesurer la similarité des termes.

Cependant, la cooccurrence simple, dans un contexte grand ou petit ne signifie pas nécessairement que les termes sont similaires. L'information mutuelle est une mesure plus complète pour l'association de mot, qui intègre la dépendance entre termes **[78]**, **[79]**.

Elle est définit ainsi :

$$- \mathrm{I}_{u,v} = \log_2 \left[\frac{\mathrm{P}(u,v)}{\mathrm{P}(u),P(v)} + 1\right] \quad \text{(II. 19)}$$

Où

– $\mathrm{P}(u,v)$: représente la probabilité conjointe que le terme u et le terme v apparaissent dans un certains contexte (généralement un document) ;

– $\mathrm{P}(u), P(v)$: représentent les probabilités d'occurrence des termes u et v respectivement.

L'un des inconvénients de l'information mutuelle est sa tendance à favoriser les termes rares plus que les termes communs. Ce qui peut devenir un problème plus aigu pour les données clairsemées. Sinon, nous pourrions envisager la définition classique de la probabilité conditionnelle. Elle consiste à mesurer le degré de l'association du terme v au terme u donnée comme suit :

$$- \mathrm{P}(v,u) = \frac{\mathrm{P}(u,v)}{\mathrm{P}(u)} \quad \text{(II. 20)}$$

De ce fait, les règles d'association ont été utilisées afin de trouver les termes d'expansion en corrélation avec les termes de la requête **[80]**, **[81]**.

II.2.4.2.2 Association un à plusieurs

Association un-à-un à tendance à ajouter un terme quand il est fortement lié à l'un des termes de la requête initiale. Dans certains cas, cela ne peut pas refléter exactement les relations des termes d'expansion à la requête dans son ensemble.

Ce problème a été analysé par Bai & al **[82]**, par exemple, si le <<programme>> est fortement lié à une requête contenant le mot <<ordinateur>>, alors l'expansion automatique pourrait fonctionner seulement pour certaines requêtes comme :<<programme java>>, <<programme d'application>>, mais pas pour d'autres requêtes comme : <<programme TV>>, <<programme spécial>>.Ici encore, nous rencontrons la question de l'ambiguïté de la langue.

Le principe de l'approche association un-à-plusieurs est d'étendre l'association un-à-un pour les autres termes dans la requête. L'idée est que si un terme d'expansion est corrélé à plusieurs termes de la requête, donc in est corrélé à la requête dans son ensemble.

La formule définit dans ce qui suit, calcule les facteurs de corrélation d'un terme d'expansion candidat v pour chaque terme de la requête, en utilisant des corrélations terme-à-terme, puis elle combine les pertinences trouvés pour trouver la corrélation de la requête (q) globale :

$$- C_{q,v} = \frac{1}{|q|}\sum_{u\epsilon q} C_{u,v} \quad (II.21)$$

Une approche similaire a été proposé dans **[83]** et **[84],** et plusieurs autres travaux de recherche ont suivi **[85]**, **[86]**, **[87]** et **[88]**.

La formule (II. 17), dans **[89]** est utilisée pour déterminer la similarité terme-à-terme dans toute la collection. Elle est vue comme un espace concept-terme, où les documents sont utilisés pour extraire les termes d'indexation.

Dans un document, le poids du terme est donné comme étant le produit de la fréquence du terme dans le document par la fréquence inverse du terme associe à ce document, et la fréquence inverse du terme d'un document (d_j) est exprimée comme suit :

$$- \log \frac{T}{DT_j} \quad (II.22)$$

Où

–T : représente le nombre de terme dans la collection ;

–DT_j : représente le nombre de termes distincts dans le document (d_j).

L'inverse de la fréquence du terme est similaire à celle du document utilisée pour le classement du document.

La formule (II. 17) est utilisée pour calculer la corrélation d'un terme-concept (plutôt qu'une corrélation terme-à terme)

Où

– $W_{u,j}$: représente la fréquence du terme de la requête dans le j-ième passage ;

– $W_{v,j}$: représente la fréquence du concept **c** dans le j-ième passage.

Un concept représente un groupe de mots adjacents (proches) dans les documents les plus recherchés, et le passage est une portion de texte de taille fixe, utilisée pour analyser les concepts candidats.

Le calcul de la valeur de corrélation terme-concept exacte est basé sur la fréquence inverse du terme et du concept dans les passages contenus dans l'ensemble de la collection.

L'approche un-à-plusieurs est aussi basée sur une combinaison de relations multiples entre les paires de termes dans un cadre de la chaine de Markov **[90]**. Un réseau d'expansion est construit pour chaque requête, il contient des paires de mots qui sont reliés par des types

de relations, comme des occurrences, des synonymes, des radicaux et les probabilités de transition. Ces relations peuvent être générées à partir de plusieurs sources.

Ensuite les mots avec une forte probabilité de pertinence sont sélectionnés en tant que termes d'expansions. En tenant compte des données rares, l'approche un-à-plusieurs est plus robuste. Cette approche prend en charge des données complexes impliquant des chaines de termes. Elle présente des limitations des relations entre les termes simples, et afin de les surmonter, nous pouvons voir la requête comme une expression, et ensuite chercher des phrases qui lui sont liées. Les phrases sont caractérisées par un contexte riche et une moindre ambiguïté que leurs mots constructifs même si une évaluation de similarité au niveau de la phrase ne peut pas être simple.

II.2.4.3 Sélection des termes

Après avoir classé les termes candidats. Dans la deuxième étape, les principaux éléments (termes) sont sélectionnés pour l'expansion de la requête.

Les termes avec une probabilité supérieurs à un certain niveau peuvent être sélectionnés seulement quand les pertinences des termes sont considérées comme des probabilités.

Plusieurs techniques pour la sélection des termes ont été proposées, elles utilisent des informations et pas seulement que les poids attribués aux termes candidats.

Une des ces techniques utilise plusieurs fonctions de classement de termes, et sélectionne pour chaque requête les termes les plus courants.

Une autre stratégie consiste à choisir une quantité variable de termes d'expansion en fonction de la difficulté de la requête.

Dans **[91]** les auteurs utilisent un classificateur afin de distinguer entre la pertinence et le non pertinence du classement des termes d'expansion. Pour apprendre les paramètres du classificateur, un ensemble d'information est créés dans lequel les termes simples sont étiquetés comme bons ou mauvais selon leurs influences sur les résultats de la recherche.

L'ajout de termes à la requête accroit la performance du Système de Recherche d'Information (SRI). Buchley & al **[92]** 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 de documents initialement retrouvés.

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

$$– RP(N) + A \log(N) + B \log(X) + c \qquad (II.23)$$

Avec

– RP (N): représente la performance du système pour N documents restitués ;

– N : représente le nombre de documents restitués ;

– X : représente le nombre de termes ajoutés à la requête ;

– A, B, C : représentent 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 accroit la performance du système. On attribue alors un poids :

– Moins important aux termes ajoutés **[93]**.

– Plus important au terme issu des documents pertinents que ceux issus des documents non pertinents **[94]**.

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

- **Salton et Buckley [95]** : ont expérimenté séparément, l'ajout de tous les nouveaux termes, tous les termes issus des documents pertinents et les termes les plus fréquents dans les documents restitués à la requête initiale. L'expansion de la requête avec tous les nouveaux termes offre de meilleurs résultats que les autres méthodes, toutefois l'écart de performance n'est pas très considérable relativement aux exigences de temps et d'espace mémoire.

Robertson **[96]** et Haines **[97]** adoptent une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer pour chaque terme un nombre traduisant sa valeur de pertinence. Les termes sont alors triés puis sélectionnés sur la base d'un seuil.

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

$$-SV(i) = w(P_i - U_i) \quad \text{(II.24)}$$

Avec

$$- w = log \frac{P_i(1- U_i)}{U_i(P_i- P_i)} \quad \text{(II.25)}$$

Avec

P_i : représente la probabilité (d_i=1/D est Pertinent) ;

U_i : représente la probabilité (d_i=1/D est Non Pertinent).

- **Harman [98]** propose les fonctions suivantes

$$SV\ (i) = \frac{RT_j * df_i}{N} \quad \text{(II.26)}$$

Avec :

– RT_j : représente le nombre total de documents retrouvés par la requête ;

– df_i : représente la fréquence d'occurrence du terme t_i dans la collection ;

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

$$SV\ (i) = \frac{r_i}{R}\ \frac{df_i}{N} \quad \text{(II.27)}$$

Avec

– r_i : représente le nombre de documents pertinents contenant t_i ;

– R : représente le nombre de documents pertinents.

$$SV\ (i) = log_2 \frac{p_i(1-q_i)}{(1-p_i)} \quad \text{(II.28)}$$

Avec

– p_i : représente la probabilité que t_i appartienne aux documents pertinents ;

– q_i : représente la probabilité que t_i appartienne aux documents non pertinents.

Les expérimentations réalisées sur différentes collections standards, ont révélé que la troisième fonction est la meilleure.

II.2.4.4 La reformulation de la requête

La reformulation de la requête est la dernière étape du processus d'expansion automatique de la requête. Elle décrit la requête élargie qui sera soumise au Système de Recherche d'Information (SRI) ce qui revient généralement à affecter un poids à chaque terme qui décrit la requête étendu appelé <<pondération de la requête>>.

Il existe plusieurs techniques de pondérations de requêtes, la plus populaire est reproduite à partir de la formule Rocchio pour la réinjection de pertinence **[99]**.

La formule générale est donnée comme suit :

$$-W'_{t,q'} = (1-\lambda).W_{t,q} + \lambda.RSV_t \quad \text{(II. 29)}$$

Où

– q′ : représente la requête étendue ;

– q : représente la requête originale ;

– λ : représente le paramètre de pondération ;

– RSV_t : représente le poids attribué au terme d'expansion t.

Les poids basés sur les documents pour la requête qui n'est pas étendue el les pertinences en fonction de différence de distribution utilisés pour les termes d'expansion

possèdent de différentes échelles, alors leurs valeurs doivent être normalisées avent de les additionner dans la formule (II. 29).

De ce fait, plusieurs techniques de normalisation simples, discutées dans **[100]**, ont été proposées, elles produisent des résultats comparables en général.

Afin d'optimiser les performances, la valeur λ peut être ajustée si les données sont disponibles. Donner plus d'importance par exemple aux termes de la requête initiale deux (2) fois plus que les termes d'expansion. Ou autres possibilités comme la technique suggérée dans **[101]** et qui consiste à utiliser une formule de pondération de requêtes sans paramètres.
La formule ci-dessus peut être utilisée dans le cas où les termes sont extraits d'un thésaurus ou de Word Net.

Les pondérations peuvent être fondées sur des critères multiples comme le nombre de termes, le nombre de cooccurrence, la longueur du document et le type de la relation.
L'étape de pondération de termes est naturellement prise en charge si l'approche de modélisation du langage effectue le classement des documents
Dans la formule suivante, le modèle de requête est estimé généralement, en ne considérant que des mots de la requête initiale :

$$- \mathrm{Sim}(q, d) \alpha \sum_{t \epsilon v} P(t|\, \theta_q) \log \frac{P(t|\theta_q)}{P(t|\theta_d)} \qquad (II.30)$$

Tandis que le modèle de document est estimé en considérant également les mots invisibles :

$$- P\left(t\middle|\theta'_q\right) = (1-\lambda).P(t|\theta_d) + \lambda.P(t|\theta_c) \qquad (II.31)$$

Où

– θ_d : représente le modèle de document ;

– θ_c : représente le modèle de la collection.

Ainsi la question qui se pose est de savoir s'il est possible de créer un meilleur modèle de requête en trouvant les mots connexe avec leurs probabilités associées et ensuite d'utiliser le modèle de requête correspondant pour unir le modèle de la requête initiale de la même manière que le modèle de document est lissé avec le modèle de la collection.

De ce fait, plusieurs méthodes ont été explorées. Elles sont basées non seulement sur les documents feedback **[102]**, **[103]**, mais également sur les relations entre les termes **[104]**.
Indépendamment de méthode de génération spécifique, le modèle final étendu est donnée par la formule suivante :

$$-\mathrm{P}(\mathrm{t}|\theta'_{\mathrm{q}}) = (1-\lambda).\mathrm{P}(\mathrm{t}|\theta_{\mathrm{q}}) + \lambda.\mathrm{P}(\mathrm{t}|\theta_{QEM}) \quad \text{(II. 32)}$$

Où

– θ_{q} : représente le modèle de la requête initiale.

Cette expression peut être considérée comme une généralisation de la formule (10).la pondération de la requête est la même dans le processus d'expansion automatique de requête, mais elle n'est pas toujours assurée.

Une approche alternative simple consiste en l'augmentation du nombre de termes qui décrivent la requête sans effectuer la pondération de requête. Une autre approche vise à augmenter le nombre de termes de la requête, et ensuite appliquer une version modifiée de la fonction de pondération utilisée par le système de classement pour explicitement traiter avec les fonctionnalités d'expansions.

II.2.5 L'expansion de la requête dans le modèle de langue

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

II.2.5.1 Modèle de Lavrenko et Croft

Au lieu de modéliser la Recherche d'Information (RI) comme processus de génération de la requête, Lavrenko et Croft **[105]** 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 document) en assumant qu'ils sont générés du modèle de pertinence. Ce modèle est formalisé comme suit :

$$-\ p(t|\theta_R) = \sum_{d\epsilon R} P(d)p(t|d)\prod_{i=1}^{K} P(q_i|d) \quad \text{(II. 33)}$$

Où

– R : représente l'ensemble des tops documents pertinents ;

– $P(d)$: représente 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 (RI), de +10% à +29% d'amélioration de précision moyenne par rapport au modèle de langue de base **[106]**, **[107]**.

II.2.5.2 Modèle de Zhai et Lafferty

Dans la même option que **[105]**, Zhai et Lafferty **[108]** ont proposé un modèle nommé model-based feedback, où le nouveau modèle de la requête est obtenu par l'interpolation du modèle original de la requête avec un modèle de matière θ_T (Topic model), obtenu en utilisant les documents les mieux classés (retour de pertinence). La construction du modèle θ_T consiste en l'extension d'une partie des documents retournés pertinents qui est distincte de l'ensemble des documents de la collection. Comme les documents les mieux classés sont susceptibles de contenir à la fois des informations pertinents génériques (ou même non pertinents), ils peuvent être représentés par un modèle génératif mixte qui combine le modèle θ_T (à estimer) et le modèle de langue de la collection. Le logarithme de la probabilité des documents les mieux classées est donnée comme suit :

$$logP(r|\theta_T) = \sum_{d\epsilon R} \sum_t c(t,d) \log\left((1-\lambda)p(t|\theta_T) + \lambda p(t|C)\right) \quad \text{(II.34)}$$

Où

– R : représente l'ensemble des documents les mieux classés ;

– $c(t,d)$: représente le nombre d'occurrence du terme t dans le document d ;

– λ : représente le poids d'interpolation.

L'algorithme EM (Expectation-Maximisation) est ensuite utilisé pour extraire le modèle θ_T.

II.3 La diversification des résultats de recherche

Un nombre considérables d'expérimentations ont été effectuées sur les collections de documents pour l'évaluation de l'impact induit par l'expansion de requête sur le processus de Recherche d'Information (RI), des améliorations ont été constatées. Cependant, pour certaines requêtes "difficile" ou "ambigüe", l'expansion de requêtes n'améliore pas les résultats. Par exemple pour la requête "Java" elle est ambigüe car elle fait référence à trois sens distincts : langage, café et ile. Si un seul sens est pris en considération dans l'expansion de cette

requête, alors les utilisateurs ayant un besoins différents ne seront pas satisfaits. Pour satisfaire tous les besoins, il faut diversifier les résultats de recherche. Cette diversification est contrôlée grâce aux caractéristiques de la requête.

Nous présentons dans cette section la notion de diversification des résultats et les caractéristiques de requêtes pour identifier les requêtes nécessitant de la diversification.

II.3.1 Définition et objectifs de la diversification [49]

La diversification des résultats de recherche vise à sélectionner divers documents à partir des résultats de recherche afin de couvrir autant d'intentions que possible. Dans les approches existantes, on suppose que les résultats initiaux sont suffisamment diversifiés et couvrent bien les aspects de la requête. Or, on observe souvent que les résultats initiaux n'arrivent pas à couvrir certains aspects.

La diversification des résultats de recherche essaie de sélectionner des résultats pertinents mais diversifiés parmi les meilleurs résultats.
Il est peut être considéré comme un problème d'optimisation dont le but est de déterminer un ordre (ou un classement) des documents, de manière à couvrir autant que possible les différents aspects de la requête.
La diversification des résultats de recherche vise à identifier les informations pertinentes dans l'incertitude posée par une requête ambigüe. Son efficacité dépend à la fois de la pertinence des documents retournés, et de leurs capacités à remplir de multiples intentions de l'utilisateur, par rapport à la requête de l'utilisateur.

La diversification des résultats de recherche peut être considérée comme une généralisation du problème standard de classements des informations où les défis à relever sont les suivants:

- Satisfaire les besoins multiples en informations au-delà de la requête de l'utilisateur : Ce premier défi est dû au problème de l'ambiguïté de la requête, ce qui signifie qu'une requête peut avoir plusieurs aspects (ou interprétations). Cependant, l'aspect de l'utilisateur avec lequel est concerné n'est pas clair.
- Eviter la redondance dans le classement des informations : ce deuxième défi est dû au fait que, une fois un document **d** satisfaisant les besoins en informations de l'utilisateur **a** été obtenu, un autre document **d'** qui satisfait les mêmes besoins en

informations de l'utilisateur **a** que **d**, est considérée comme plus utile (ou redondant) pour l'utilisateur. Il a été montré que la pertinence d'un document dans un classement doit être estimée dépendamment de la pertinence des documents classés au-dessus de ce dernier. Autrement dit, un bon moteur de recherche doit tenir compte de la pertinence d'un document à la lumière des autres documents récupérés. Il est donc important d'enlever ce document redondant **d'** du la liste de classement.

II.3 .2 Classification des approches de diversification [49]

Nous pouvons classer les méthodes en deux catégories, selon la façon dont les aspects de la requête sont représentés : les approches de diversification implicites et les approches de diversification explicites.

II.3.2.1 L'approche de diversification implicite

L'approche de diversification implicite favorise des documents différents à travers les relations entre les documents pour produire des listes de classement qui véhiculent à la fois l'information pertinente et diversifiée sur une requête explicite pour couvrir autant que possible les différents aspects de la requête (qui sont soit définis manuellement ou extraites automatiquement) sur la base de la relation entre les documents et la requête.

Parmi les approches les plus importantes de la diversification implicite, nous citons :

La méthode basée sur «Novelty » : La majorité des diversifications implicites adoptent cette stratégie de nouveauté ou de non redondance. Son importance été démontrée dans plusieurs études ainsi certains montrent qu'il est important que les moteurs de recherche actuels prennent en compte le critère de «Novelty ».

Carbonell et Goldstein [116] proposent une méthode appelée Pertinence marginale maximale, qui est la première méthode de diversification implicite basée sur « Novelty ».

La pertinence marginale maximale : est une méthode représentative précoce des diversifications implicites et elle est l'une des approches les plus populaires dans le document de diversification. Elle vise à équilibrer la pertinence et la diversité d'une liste de classement, en sélectionnant les documents qui optimisent la pertinence et en réduisant la redondance par rapport aux documents de rang supérieur.

II.3.2.2 L'approche de diversification explicite

La majorité des approches explicites adoptent une stratégie basée sur la couverture « coverage » puisque ces approches exigent que les aspects de la requête soient connus (qui sont souvent déterminées manuellement).

La méthode basée sur « coverage » : Le travail de **Radlinski et Dumais [117]** a été le premier qui représente la direction vers des approches de diversification explicites. Les auteurs proposent de trouver pour chaque requête Q d'autres requêtes qui sont liés à Q. Plus les reformulations de la requête sont exécutées, plus la diversité est.

Dans **Radlinski et Dumais [117]**, les requêtes reformulés sont définis comme les requêtes qui sont trouvées dans les journaux de recherche ayant au moins un mot commun avec la requête Q original, et que l'utilisateur soumis au moins deux requêtes liées à Q dans une minute de chaque autre. Les auteurs développent des méthodes pour générer l'ensemble des requêtes connexes à partir de laquelle (Résultat Maximum Variety) est le plus puissant.

Dou et al [118] affirment que les résultats de la recherche doivent être diversifiés d'une manière multidimensionnelle, car les requêtes sont généralement ambiguës à différents niveaux et dimensions.

II.3.3 Caractéristiques de la requête pour contrôler la diversification

Nous allons détailler les deux (2) caractéristiques suivants:

– La taille de la requête ;

– Clarté de la requête.

II.3.3.1 Taille de la requête

Des expérimentations intéressantes ont été réalisées sur la base TERC7 et présentées dans **[110]**. Les auteurs montrent en effet que la dérivation automatique de courtes requêtes à partir de documents jugés ou supposés pertinents à la suite d'une recherche initiale, permettent d'atteindre des résultats très performants pour différentes tâches : recherche, filtrage et routing.

II.3.3.2 Clarté de requête

La clarté de la requête a été utilisée pour prédire la difficulté de requête **[111]**. Lorsqu'une requête est difficile, cela pourrait signifier qu'il a différents interprétations, donc afin de compléter l'image de tous les aspects de la requête, nous avons besoin de fournir un ensemble diversifié de documents pour vous assurer que tous les aspects de la requête sont couverts.

Selon la définition **[111]**, la clarté d'une requête est la divergence de Kullback-Leibler de modèle de la requête et du modèle de collection. Pour notre cas, le modèle de requête est estimé à partir des documents pertinents en F_{rel} qui est définie comme suit :

$$- QClar_1 = \sum_{w \epsilon F_{rel}} p(w|\, \theta_{F_{rel}})\, log_2 \frac{P(w|\theta F_{rel})}{p(w|C)} \quad \text{(II.35)}$$

Où

– $P(w|\theta F_{rel})$ est estimée comme suit :

$$- \frac{c(w,F_{rel})}{\sum_{w' \epsilon F_{rel}} c(w',F_{rel})}, \quad \text{(II.36)}$$

Avec $c(w,\ F_{rel})$: représente le nombre de mots w in F_{rel} (à savoir, les documents pertinents).

II.4 Conclusion

Dans ce chapitre, nous avons présenté deux concepts, dans la première section nous avons détaillé l'expansion de requêtes, dans laquelle, nous avons cité les trois (3) facteurs de classifications des méthodes d'expansion de requêtes, puis nous avons énuméré et expliqué les principales approches de l'expansion de celle-ci, à savoir l'approche basée sur le contexte global et l'approche basée sur le contexte locale, après nous avons cité les étapes de la reformulation de la requête, à titre d'exemple, le prétraitement des données, la génération et le classement des termes candidats d'expansion, la sélection des termes d'expansion et la réécriture de la requête, et en fin, nous avons abordé l'expansion de requête dans le cadre du modèle de langue.

Dans la deuxième partie, nous avons présenté le concept de la diversification des résultats de recherche et ses objectifs, nous avons parlé de la classification des approches de diversification (implicites et explicites). En fin nous avons expliqué les caractéristiques des requêtes qui prédisent les requêtes nécessitant la diversification, à savoir la taille et la clarté.

Dans le chapitre suivant, nous allons présenter notre méthode proposée, qui consiste à utiliser les caractéristiques de requêtes pour choisir les termes d'expansions afin d'assurer la diversification des résultats de la recherche.

Chapitre III:

Présentation & Implementation de l’approche

III.1 Introduction

Dans la première section, nous décrivons l'approche proposée. Dans la seconde section, nous abordons l'expérimentation de l'approche en précisant les outils et langage utilisés pour sa mise en œuvre.

Enfin nous présentons quelques résultats expérimentaux obtenus sur la collection de test TREC AP88.

III.2 Présentation de l'approche proposée

III.2.1 Principe de l'approche

Le besoin en information de l'utilisateur est exprimé par une requête qui est exécutée par le système de recherche d'information. Dans certains cas cette requête n'est pas complète et claire, alors elle retourne des résultats qui ne sont pas satisfaisants.

Pour remédier à ce problème, nous utilisons l'expansion de requête pour améliorer la requête initiale en ajoutant des termes générés, classés et choisis pour reformuler enfin cette requête.

Cependant, il existe des requêtes non améliorées avec cette méthode classique et cela est dû au fait que ces requêtes traitent de plusieurs thématiques (les requêtes ambigües).

Pour remédier à ce problème nous proposons une approche qui se base sur le principe contenant les éléments suivants :

1. Une requête multithématique est une requête ambigüe, cette requête nécessite alors de la diversification dans le choix des termes d'expansion ce qui implique l'obligation d'effecteur un travail de regroupement (clustering) des documents pseudo pertinents afin de capter toutes les thématiques de cette requête une fois que ce travail est effectué on sélectionne les documents d'expansion à partir des clusters obtenus.

   Nous avons utilisé l'algorithme de k-means pour le clustering .Il consiste à diviser les documents retournées en plusieurs sous ensembles, ces sous ensembles sont formes par des documents qui se ressemblent au sens d'un critère de similarités.

2. Si au contraire la requête n'est pas multithématique alors elle est claire, donc les premiers documents pseudo pertinents traitent de la thématique de la requête.
3. Sur la base des deux éléments énoncés précédemment, on établit un seuil pour l'ambigüité de la requête pour spécifier quelle source (documents d'expansion) à utiliser. Ce seuil est obtenu par expérimentation.

Enfin le calcul du score de clarté (l'ambigüité) de la requête est obtenu par la formule suivante :

$$Score\ de\ clarté = \sum_{i=1}^{n} P(t_i|M) * \ log_2 \frac{P(t_i|M)}{P(t_i|C)} \qquad \text{(III.1)}$$

Où **n** représente tous les termes des top documents « **M** » et $\mathbf{P(t_1|M)}$ est la fréquence globale relative du terme « t_i » dans ces top document. $\mathbf{P(t_1|C)}$ est la fréquence globale relative du terme « t_i » dans la collection C.

III.2.2 Formalisation de l'approche

Nous présentons dans cette section, l'emplacement de notre approche dans le SRI et les algorithmes mis en œuvre pour l'implémenter.

III.2.2.1 Emplacement de notre approche dans un système de recherche d'information

La figure suivante illustre l'emplacement de notre approche dans un SRI :

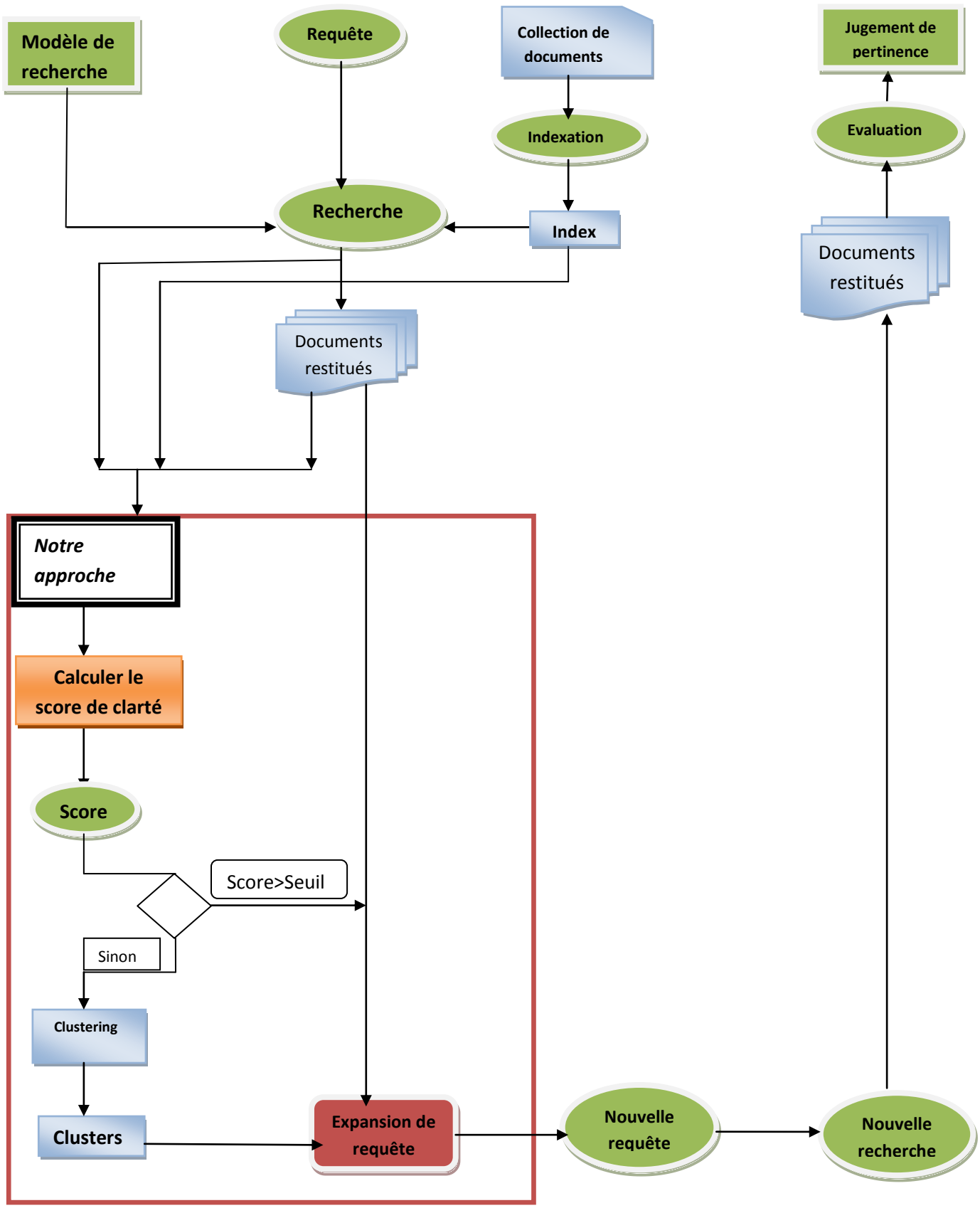

Figure III.6: Emplacement de notre approche dans un SRI.

III.2.2.2 Processus d'expansion avec notre approche

Nous schématisons le processus d expansion de requête avec notre approche dans la figure suivante :

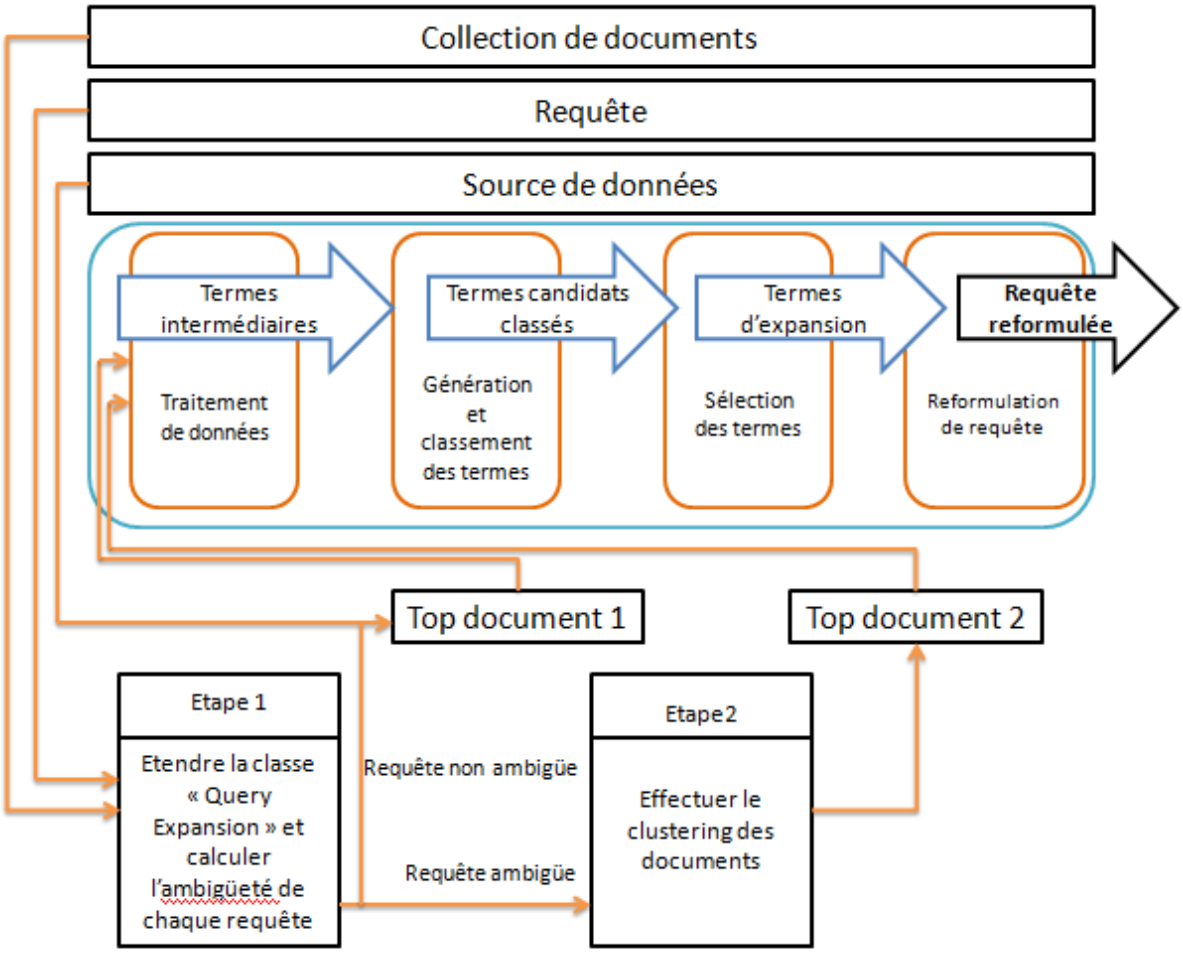

Figure III.2 Processus d'expansion avec notre approche.

Etape1 :

Cette étape consiste à calculer l'ambigüité de chaque requête dans le but de choisir ensuite la méthode de classement des documents à utiliser. Pour faire ce calcul nous avons modifié la classe Query Expansion de plate forme Terrier. Voici l'algorithme de calcul de l'ambigüité qui présente la manière de choix de méthode de classement des documents :

Entrées :Top_Document[]

Sorties :score_clarté

Début

//Remplir la table

Table < clé,valeur>

Pour (i=0 jusqu'à taille de Top_Document)

Récupérer toutes les informations sur les termes du document i dans la matrice Id_terms[][]

Pour (j=0 jusqu'à taille Id_terms [0])

Mettre Id_terms [0][j] ,Id_terms [1][j] dans table //Id_terms [0][1]comme premier argument

// et Id terms [j][1]comme deuxième argument

Fin pour

Fin pour

// calculer le score de clarté

Pour (i=0 jusqu'à taille de table)

Récupérer la fréquence globale du terme i dans top_documents notée **TFdp** // de table

Récupérer la fréquence globale du terme i dans la collection de document notée **TFg**

// calculer la fréquence relative du terme dans la collection notée **P (ti|C)**

$P(t_i|C)$= TFg/ Taille de la collection // C représente la collection

//calculer la fréquence relative du terme dans top documente **$P(t_i|M)$**

P(ti, M)= TFdp/taille-totale-de-top-document-document // M représente les premiers documents de top document

// calculer le score de clarte

Score_clarte += P (ti| M)* $\log_2 P(t_i/M) / P(t_i/C)$

Fin pour

Fin

Etape2 :

Cette étape est effectué si le score de clarté calculé dans l'étape1 est inferieur à un seuil définit, c'est-à-dire elle est effectuée si la requête est ambigüe. Nous avons utilisé pour ça l'algorithme de « k-means » qui a pour but de faire le clustering dans le cas d'une requête ambigüe qui a pour but de classer les tops documents dans des clusters ensuite de sélectionner les documents d'expansion à partir de ces clusters.

Entrées :

Int [] docIDs *//Vecteur des documents*

Nbr_documents *// nombre de documents à utiliser dans le clustering*

Int K *// nombre de clusters considérés*

Sorties : « liste de documents obtenus après le clustering »

Si score > seuil

ExpansionTerms (KL)

Sinon

Clustering avec k-mean:

-Récupérer les documents et les parcourir un par un

-calculer la similarité entre chaque deux documents

-Comparer doc[i] et doc[i+1] grâce au similarité

-Choisir k documents formant k cluster

-Affecter chaque document au cluster le plus pertinent *

(Le plus pertinent est celui qui a la plus grande similarité)

-Recalculer les similarités

-Aller a l'étape * en cas de nouvelle affectation

Retourner le nouveau classement

finsi

III.3 L'environnement de développement

Dans ce qui suit nous allons présenter notre environnement technique et préciser les différents outils utilisés : la plate forme Terrier, le langage de programmation JAVA et Netbeans.

III.3.1 La plateforme Terrier

Terrier est l'abréviation de TERabyteRetrIEver qui est un moteur de recherche flexible, robuste, et efficace. Il est facilement déployable à grande échelle des collections de documents. Il a été crée par le département d'informatique de l'université Glasgow de Royaume-Uni en 2000. Dans le but de fournir une plateforme flexible pour le développement rapide des applications de recherche d'information. La recherche peut être effectuée facilement sur les collections de test standard TREC et CLEF.

Terrier est un produit open source écrit en Java, qu'il a été mis à disposition, du général public depuis novembre 2004 .Il a été conçu pour fonctionner sur la plupart des systèmes d'exploitation actuels tels que Windows ou Linux.

Avec sa conception modulaire, il facilite son extension en cas de besoin, et il propose aussi une grande palette de configuration et cela dans le but de satisfaire ses utilisateurs et leurs fournir une meilleure maniabilité. Ce système est sauvant utilisé dans l'innovation et la mise en œuvre de nouvelles méthodes pour la recherche d'information.

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.

III.3.1.1 Architecture de Terrier

La figure ci-dessous montre l'architecture générale de Terrier :

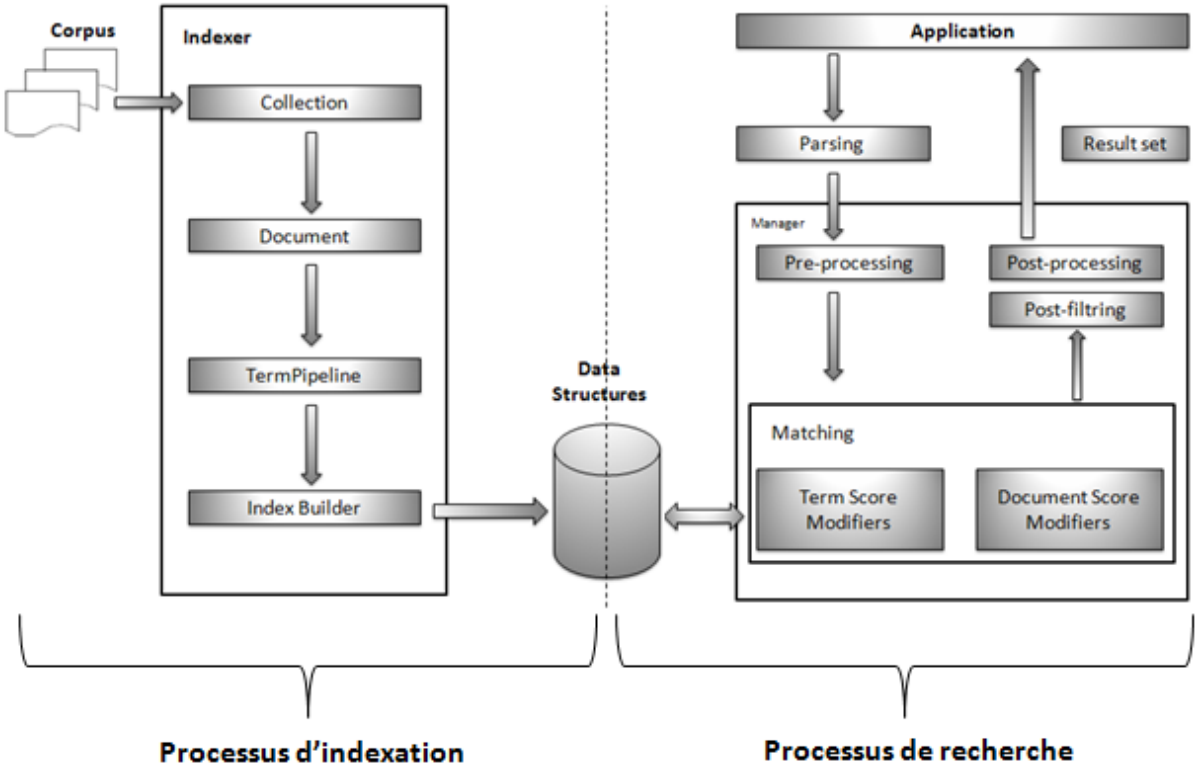

Figure III.3 Vue d'ensemble de l'architecture Terrier.

III.3.1.1.1 Le processus 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) : pour chaque document de la collection, un processus visant à extraire les termes du document nommé « tokenisation ».
   Ce processus retourne comme résultat l'ensemble de termes de document qu'on peut adopter comme termes d'indexation.
3. Traitement des termes extraits avec Term-Pipeline : consiste à l'élimination des mots vides et la lemmatisation des termes.
4. La construction de l'index.

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

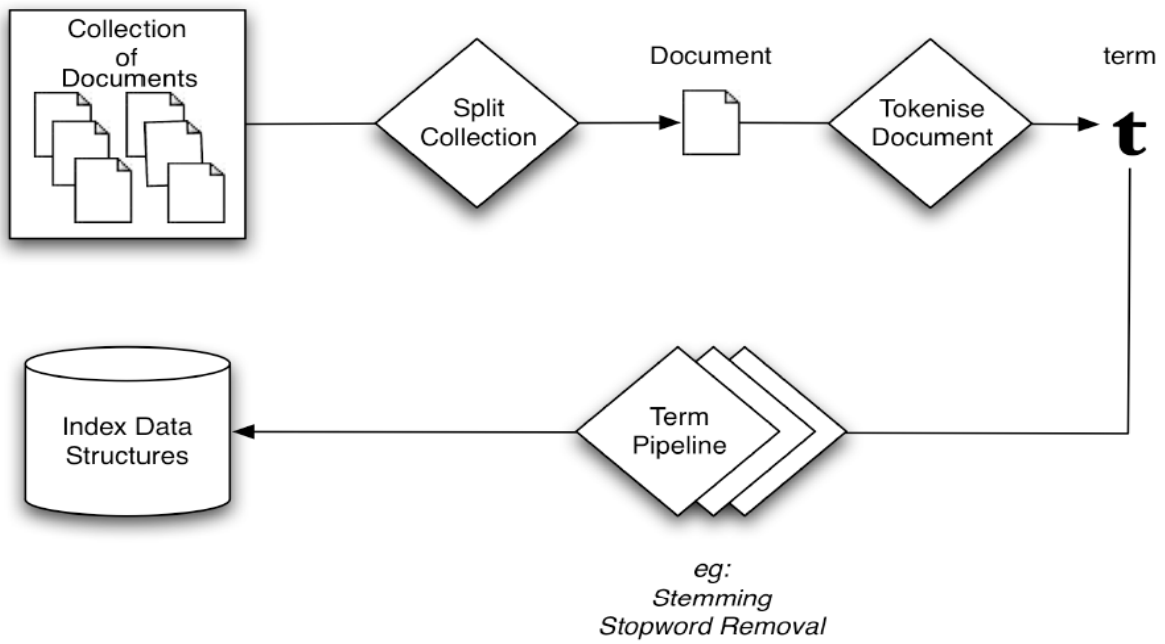

Figure III.4 Le processus d'indexation dans Terrier.

Le processus d'indexation avec Terrier consiste à analyser tous les documents de la collection spécifiée afin de produire un ensemble de termes d'indexation « index » et les ranger dans la structure « index data structure ».

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

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 de 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.
- **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.
- **Document index :** contient des informations sur les différents documents de la collection.

III.3.1.1.2 Le processus de recherche

Un des principaux objectifs de Terrier est de faciliter la recherche d'information. Terrier implémente pour cette raison certain nombre de fonctionnalités de recherche qui offre un large choix pour le développement de nouvelles applications et pour les testes en recherche d'information.

Durant le processus de recherche, chaque requête doit passer par les étapes suivantes :

1. **Query :** est une classe abstraite qui représente la requête.
   Terrier supporte trois modèles de requête :
   - *Single Term Query :* désigne la requête qui contient un seul terme.
   - *Multi Term Query :* désigne la requête qui contient plusieurs termes.
   - *Field Query :* terme qualifié par un champ.
2. **Parsing :** est l'opération qui se charge de tokenizer la requête.
3. **Pré-processing :** est l'oppération qui applique le TermPipeline à la requête. Elimine les mots vides et les lemmatises.
4. **Matching :** est l'opération responsable de l'initialisation du Weighting Model et du calcul des scores entre la requête et les documents.
   - **Weighting Models :** est une interface qui assigne un score pour chaque terme de la requête dans le document.
     Terrier offre plusieurs classes qui implémentent cette interface, parmi eux (TF-IDF, BM25).

- **Document Score Modifiers :** permet de modifier le score des documents en fonction du langage de la requête.
- **Term Score Modifiers :** permet de modifier le score des documents en fonction de la position des termes.

5. **Post-processing :** est l'opération appropriée pour implémenter des fonctionnalités qui apportent un changement à la requête originale. Un exemple de Post-processing est l'expansion de requête, puis exécute une autre fois le Matching avec cette nouvelle requête. Un autre exemple de Post-processing est l'application de clustering.
6. **Post-filtering :** 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 recherchée depuis le même site Web, afin d'augmenter la diversité des résultats.

A la suite des étapes précédentes, Terrier s'occupe de retourner un ensemble de documents triés selon l'ordre décroissant de leurs scores.

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

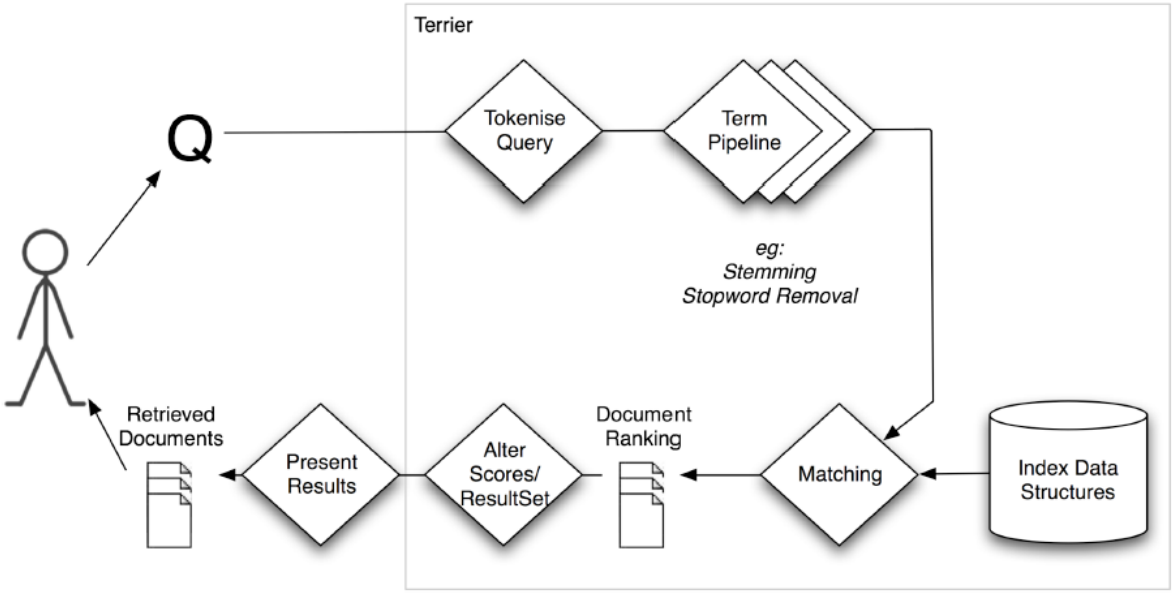

Figure III.5 Le processus de recherche dans Terrier.

III.3.2 Le langage Java

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

Cyrille Herby Apprenez à programmer en Java

Parmi les caractéristiques de base du langage Java on cite : Mohamed N. Lokban

- **Java est simple :** Java est un langage simple à prendre en main, basé sur le langage C/C++ mais laisse de coté les sources de problèmes (pointeurs, structures, gestion de la mémoire, héritage multiple, macros etc.).
- **Java est orienté objet :** Java est un langage purement orienté objet.
  - Tout est classe.
  - Héritage simple.
  - Une librairie plus de classes est fournie.

- **Java est distribué :** Java propose une API réseau standard. Cette dernière permet de manipuler, par exemple, les protocoles HTTP & FTP avec aisance.

Des API pour la communication entre des objets distribués (Remote Method Invocation).

Java est interprété : Un code source doit être traduit dans le langage machine avant d'être exécuté.

Compilateur: traduction du code source dans le langage binaire de la machine sur laquelle il sera exécuté.

Interpréteur: idem qu'un compilateur, sauf qu'il procède par étapes successives de compilation et exécution. Chaque instruction est compilée puis exécutée, puis le tour à l'instruction qui suit etc.

Compilateur Java traduit le code source Java en bytecode (code portable). Par la suite un interpréteur Java spécifique à une machine donnée (Java Virtual Machine : JVM Machine Virtuelle), traduit et exécute le bytecode.

- **Java est indépendant de l'architecture :** Le bytecode généré n'est pas lié à un système d'exploitation en particulier. De ce fait, il peut être interprété très facilement sur n'importe quel environnement disposant d'une JVM.

- **Java est portable :** Java est portable d'un système à un autre : int 32 bits alors qu'en C/C++ 16 ou 32 bits.
- **Java est robuste :**
  - Pas de pointeurs.
  - Gestion de mémoire indépendante.
  - Mécanisme d'exceptions pour la gestion des erreurs.
  - Compilateur très contraignant.
  - Pas d'héritage multiple ni surcharge des opérateurs.
- **Java est sécurisée :** quatre niveaux de sécurité :
  - Langage et son compilateur contraignant.
  - Vérifier : vérifier le bytecode.
  - Class Loader : le chargeur de classe.
  - Security Manager : protection des fichiers et accès au réseau.

III.3.3 NetBeans

L'EDI NetBeans est un environnement de développement intégré, placé en open source par Sun en juin 2000, se concentrant principalement sur simplifier le développement

d'applications Java. Il fournit du support pour tous les types d'applications Java, depuis le client riche jusqu'aux applications d'entreprises multi-couches, en passant par les applications pour les mobiles supportant Java.

L'EDI NetBeans a une architecture modulaire qui permet les 'plug-ins'.
Cependant, l'étendue des fonctionnalités dans l'installation de base est tellement riche que vous pouvez probablement utiliser l'EDI pour votre travail sans du tout vous souciez des modules externes.

L'EDI est lui-même écrit en Java, ce qui vous permet de le faire tourner sur n'importe quel système d'exploitation.

NetBeans est disponible sous Windows, Linux, Solaris , Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

La figure suivante illustre l'environnement de développement Netbeans :

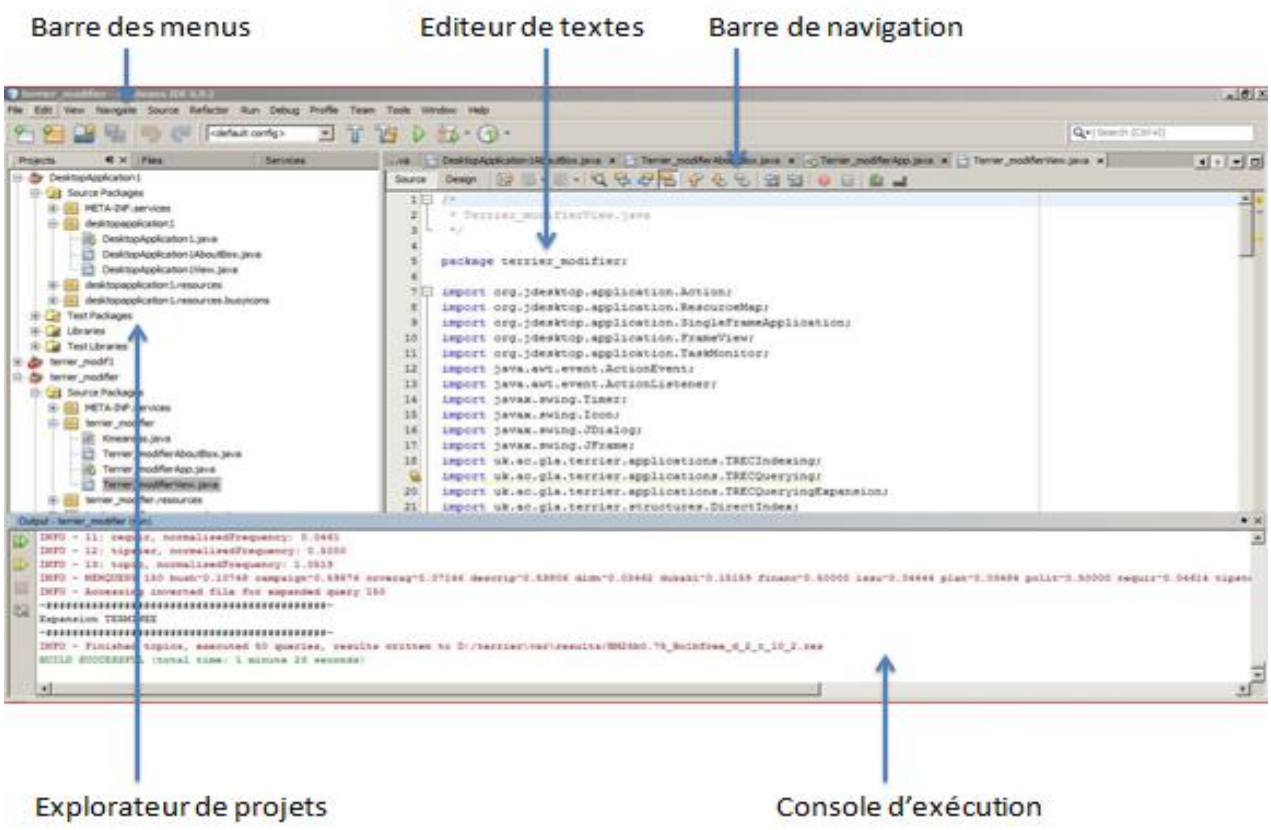

Figure III.6 Environnement de développement de Netbeans.

III.4 Expérimentation

III.4.1 Collection de test utilisée

Différentes collections de tests sont utilisées en recherche d'information. La collection que nous avons utilisée pour nos expérimentations est : TREC AP88 (Associated Press newswire, 1988).

Pour la recherche nous avons utilisé 50 requêtes issues des topics numérotées « 101-150 » de la collection TREC.

III.4.2 Evaluation & Résultats

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base : recherche simple et recherche avec le modèle de pertinence.

Ces résultats seront comparés par la suite à ceux obtenus après la reformulation avec notre approche en appliquant les paramètres de recherche :

III.4 .2.1 Résultats de la recherche simple

Avant de procéder à notre approche, nous allons commencer par illustrer les résultats obtenus avec la recherche simple.

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

Recherche simple	
Modèle de recherche	**MAP**
TF-IDF	**0.1105**

Tableau III .1 Résultat obtenu avec la recherche simple (TF-IDF).

III.4 .2.2 Résultats obtenus avec l'expansion de requêtes

Nombre de documents	Nombre de termes	MAP
1	1	0.1625
	5	0.1625
	10	0.1625
	15	0.1625
	20	0.1625
	25	0.1625
5	1	0.2001
	5	0.2001
	10	0.2010
	15	0.2048
	20	0.2057
	25	0.206
10	1	0.2043
	5	0.2043
	10	0.2065
	15	0.208
	20	0.2074
	25	0.2088
15	1	0.2107
	5	0.2107
	10	0.217
	15	0.215
	20	0.2149
	25	0.2143
20	1	0.2074
	5	0.2074
	10	0.2081
	15	0.2093
	20	0.2093
	25	0.2117
25	1	0.218
	5	0.218
	10	0.2185
	15	0.224
	25	0.2231
30	1	0.2175
	5	0.2175
	10	2.2197
	15	2.2251
	20	**0.2283**
	25	0.2261

Tableau III .2 Résultat obtenu avec l'expansion de requête.

Le tableau ci-dessus résume les obtenus avec le modèle de pertinence en utilisant le modèle de pertinence en utilisant le modèle de recherche TF-IDF, et le modèle d'expansion dans lequel nous constatons la meilleure précision est de **0.2283**, elle est obtenue avec un nombre de document égal à **30** et un nombre de termes est égal à **20**.

III.4 .2.2 Résultats obtenu avec notre approche

Dans le tableau ci-dessous, nous présentons les résultats obtenus avec notre approche en utilisant le modèle **TF-IDF**.

En faisons varier le nombre de clusters de **3** jusqu'à **12** avec un pas de **3** et des valeurs allant du minimum au maximum des score de clarté des requêtes «seuil », nous obtiendrons le tableau suivant :

Seuil	Nombre de clusters	MAP	Seuil	Nombre de clusters	MAP
0.9310890758078855	3	0.2258	1.1046360992715014	3	**0 .2262**
	6	0.2257		6	0.2265
	9	0.2257		9	0.2231
	12	0.2257		12	0.2232
0.9859114783702482	3	0.2283	1.2430354462190756	3	0.2012
	6	0.2269		6	0.2013
	9	0 .2274		9	0.1862
	12	0 .2280		12	0.1962
1.0849039913444734	3	0.2292	1.870419906965994	3	0.2087
	6	0.2287		6	0.1802
	9	**0.2305**		9	0.1694
	12	0.2278		12	0.1870

Tableau III .3 Résultat obtenu avec notre approche.

Le tableau ci-dessus nous montre que le modèle de pertinence étendu avec un seuil de **« 1.0849039913444734»** et un nombre de cluster qui est égale à **9**, nous apporte une importante amélioration avec une MAP de **0.2305** par rapport au modèle de pertinence de base qui a obtenu une MAP de **0.2283**.

III.5 Conclusion

Dans ce chapitre, nous avons présenté le principe de notre approche et son fonctionnement, nous avons aussi présenté l'environnement d'implémentation et les testes de notre approche. Enfin, nous avons présenté et discuté les résultats de notre approche vis-à-vis le modèle de base TF-IDF et le modèle d'expansion et les résultats obtenu avec notre approche.

CONCLUSION GENERALE

Conclusion Générale

Notre projet se situe dans le domaine de la recherche d'information, plus particulièrement dans le cadre de la reformulation de la requête. Il porte sur l'Implémentation et l'évaluation d'une méthode de sélection de termes d'expansion basée sur les caractéristiques de la requête « clarté de la requête ».

Pour mener à terme notre travail, nous avons donné un aperçu général sur la recherche d'information ainsi les systèmes de recherche d'information et nous avons traité l'expansion de requêtes et la diversification des documents, ce qui nous a permis d'enrichir nos connaissances pour le bon déroulement de notre travail. De même nous avons défini et suivi, la plate-forme de recherche d'information Terrier, le langage de programmation 'Java' et l'environnement 'Netbeans' afin d'implémenter notre approche.

Ce travail nous a permis d'aborder le domaine de la recherche d'information, d'enrichir nos connaissances et plus précisément :

- Approfondir nos connaissances sur la recherche d'information.
- Mettre l'accent sur la manière dont les systèmes de recherche d'information fonctionnent dans la plate-forme Terrier.

D'après nos résultats, qui ont apporté une amélioration dans les résultats de recherche en basant sur le score de clarté de la requête, nous pensons que l'utilisation de ces modèles d'expansion et les caractéristiques de la requête peut encore évoluer vers plus de performance.

BIBLIOGRAPHIE

Bibliographie

[1] Boubekeur F. "Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets", thèse de doctorat en informatique, Université Paul Sabatier. 2008

[2] Hernandez N. "Ontologie de domaine pour la modélisation du contexte en recherche d'information", thèse de doctorat en informatique, Université Paul Sabatier. (2006)

[3] Boughanem M , Jaques S : « Recherche d'information état des lieux et perspectives ». Lavoisier, 2008.

[4] Abdelkrim Bouramoul : « Recherche d'information conceptuelle et sémantique sur l web ».Thèse de doctorat en informatique, Université MENTOURI de Constantine, 2011.

[5] C. CLEVERDON. Progress in documentation. evaluation of information retrieval systems. Journal of Documentation, 1970.

[6] Mohamed Ben Aouicha **:** « Une approche algébrique pour la recherche d'information structurée ». Thèse de doctorat en informatique, université Paul Sabatier.

[7] Fabienne Moreau : « Revisiter le couplage traitement automatique des langues et recherche d'information ». Thèse de doctorat en informatique, université de Rennes 1,2006.

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

[9] C. Jacquemin, B. Daille, J. Royanté, and X. Polanco. In vitro evaluation of a program for machine-aided indexing.Inf. Process. Manage, 2002.

[10] C. Fox. Lexical analysis and stoplists.Frakes WB, Baeza-Yates (eds) Prentice Hall, New Jersey, pages 102–132, 1992.

[11] Boubekeur-Amirouche F. Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets, thèse en informatique, Université Toulouse III-Paul Sabatier,2008 .

[12] Baziz M., « Indexation Conceptuelle Guidée Par Ontologie Pour La Recherche d'Information ». Thèse de Doctorat en Informatique de l'Université Paul Sabatier de Toulouse, 2005.

[13] Hammache Arezki: "le recherché d'information : modèle de langue combinant mots simple et mots composés". Thèse de doctorat en informatique, Université Mouloud Mammeri de Tizi Ouzou, 2013.

[14] G. Salton. The SMART Retrieval System-Experiments in Automatic Document Processing . Prentice-Hall Inc,NJ,1971.

[15] S. Robertson and K. Sparck Jones. Relevance weighting for search terms. Journal of The American Society For Information Science,1976.

[16] Y.Champclaux. "Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information". Thèse de doctorat en informatique, Université de Toulouse 3- Paul Sabatier, page 44,4 décembre 2009.

[17] R. Baeza-Yates and R. A. Ribeiro-Neto. "Modern Information Retrieval". New York : ACM Press; Harlow England : Addison-Wesley, cop., 1999.

[18] G. Salton. "A comparison between manual and automatic indexing methods. Journal of American Documentation", 20(1) : 61–71, 1971.

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

[20] Kraft, D.H, and Buell, D.A.Fuzzy sets and generalized Boolean retrieval systems.International Journal on Man-Machine Studies, 19:pp. 49-56, 1983.

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

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

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

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

[25] VAN Rijibergen, C.J Information retrieval. London: Butterworth, 1979.

[26] Wong, S., Ziarko, W., Wong, P. Genralized vector space modele in information retrievl.*Proceeding of the 8th ACM SIGIR Conference on Research and Development in information retrieval*,New-York, USA, pp.18-25, 1985.

[27] Berry,M.W, Dumains, S.T.,O'Brien, G. W. 1995. Using linear algebra for intelligent information retrieval. SLAM Rev.37 (4), pp. 573-595, 1995.

[28] Dumais, S, Latent Semantic Indexing (LSI). *Processing of* TERC -3, 1994.

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

[30] Furnas, G, W, Landauer, T.K, G omez, L.M, Dumais S.T. The Vocabulary Problem in Human-System Communication, *Communications of* the ACM 30, PP. 964-971, 1987.

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

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

[33] Kwork, K.L, A neural network for probabilistic information retrieval. International ACM SIGIR Conference on Research and Developpement in Information Retrieval, pp 21-30, 1989.

[34] S. Robertson and K. Sparck Jones. "Relevance weighting for search terms". Journal of the American Society for Information Science, 27(3):129–146, 1976.

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

[36] M. E. Maron and J. L. Kuhns. "On relevance, probabilistic indexing and information retrieval". J. ACM, 7(3):216–244, 1960.

[37] A language modeling approach to information retrieval. Proceeding of the 21th annual international ACM SIGIR conference on Reserch and development in information retrieval, pp.257-281, 1998.

[38] A linguistically motivated probabilistic model of information retrieval. In Nicolaou, C., and Stephanides, C., editors, Research and Advanced Technology for Digital Libraries-Second European Conference, ECDL '98, Proceedings, number 1513 in Lecture Notes in Computer Science. Springer Verlag pp.569-584, 1998.

[39] Miller, D.R.H, Leek, T., Schwartz, R.M.A hidden markov model information retrieval system.*Hearst et al.*, pp. 214-221, 1992.

[40] Lancaster, F. Information Retrieval Systems: characteristics, testing and evaluation, John Wiley, New York, 1979.

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

[42] Voorhees, E.M & Harman, D.K. TEREC: Experiment and Evaluation in Information Retrieval Digital Libraries and Electronic Publishing, MIT Press, 2005.

[43] Voorhees, E.M. TERC: Continuing information retrieval's tradition of experimentation. *Communication of the ACM*, 50, pp. 51-54, 2007

[44] Boughanem, M Outils de validation en recherché d'information. La campagne d'évaluationTREC, 2003.

[45] Mustapha BAZIZ : << *indexation conceptuelle guidée par ontologie pour la recherche d'information*>>. Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, Décembre 2005.

[46] Abberly, D, Kirby, S. Renals, and T. Robinsob. THISL broadcast news retrieval system, In Proc. ESCA Workshop on Accessing Information in Spoken Audio, pages 19 {24, Cambridge, 1999}.

[47] Improving Retrieval Performance by Relevance Feedback. Journal of the America Society for Information Science 41, 4, 288-297.

[48] Efthimis N. Efthimiadis.Query Expansion. Annual Review of Information Science and Technology, ARIST.31:121, {187, 1996}.

[49] Arbi Bouchoucha , « Diversified Query Expansion » , Université de Montréal, Juin, 2015.

[50] Spécification de logiciels et traitement de l'information, << Reformulation de requête dans les systèmes de recherche d'information dans des documents XML>>

[51] Fatiha BOUBKEUR-AMIROUCHE, Thèse Doctorat, « Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets », 2008.

[52] Fatiha Boubekeur-Amirouche, Thèse Doctorat, « Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets », 2008.

[53] Experiments in automatic statistic thesaurus construction. In proceedings of the ACM-SIGHIR Conference on Research and Development in Information Retrieval, Copenhage, Denmark, pages 77-88.

[54] Local feedback in full-text retrieval systems. ACM 24, 3, 397- 417.

[55] Using probabilistic models of information wi-thout relevance information. Journal of docu

mentation, 35(4) :285,295.

[56] Improving automatic query expansion. In Proceedings of the twenty-First Annual International ACM SIGIR Conference on Research and Developments in Information Retrieval, 206-214. Melbourne.

[57] Introduction to Information Retrieval combridge University Press.

[58] Improving the effectiveness of information retrieval with local context analysis.ACM Trans.Info.Syst 18,1, 79-112 la reinjection de pertinence.

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

[60] 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.

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

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

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

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

[65] 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 313-323, 1971.

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

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

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

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

[70] Xu. J., Croft W.B., <<Improving the effectiveness of information retrieval with local context analysis>>, ACM Trans. *Inf.Syst*., vol. 18, n° 1, 2000, pp. 79-112, ACM Press.

[71] Voorhees, E, 2004. Overview of the trec 2004 robust track. *In Processing of the* 13^{th} *Text Retrieval Conference* (TREC-7), NIST Special Publication 500-261. National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA.

[72] Diaz, F.and Metzler, D. Improving the estimation of relevance models using large external corpora. In proceedings of the 29^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, Seattle, Washington, USA, pp. 154-161. 2006.

[73] Chirita, P.-A., Firan, C. S., AND Nejdl, W. Personalized query expansion for the web. In proccedings of the 30^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, Amsterdam, The Netherlands, pp. 7-14. 2007.

[74] Qiu Y., Frei H. –P., << Concept-based query expansion>>, Proceedings of SIGIR-93, 16^{th} ACM *International Conference on Research and Development in Information Retrieval*, Pittsburg, US, 1993, pp. 160-169.

[75] C. and Yang, B. Experiments in automatic statistical thesaurus construction. In Proceedings of the 15^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, Copenhagen, Denmark, pp. 77-88, 1992.

[76] Fianna Raiber, Oren Kurland, « On Identifying Representative Relevant Documents ». Faculty of Industrial Engineering and Management, 2000.

[77] Local feedback in full-text retrieval systems.J. ACM 24, 3, 397- 417.

[78] Word association norms, mutual information and lexicography.Computat.Linguist. 16, 1, 22-29.

[79] Information Retrieval .Butterworths.

[80] Query expansion using fuzzy association rules between terms. In Proceedings of the 4th International Conference Journ'ees de l'Information Messine(JIM'03).

[81] Integration of association rules and ontologies for semantic query expansion. Data Knowl.Engin. 63, 1, 63-75.

[82] Extending query translation the cross-language query expansion with markov chain models.In Proceedings of the 16th Conference on Information and Knowledgs Management (CIKM'07).ACM Press.

[83] Concept-based query expansion.In Proccdings of the 16the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.ACM Press, 160-169.

[84] Query expansion using local and global document analysis. In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, 4-11.

[85] Query expansion using term relationships in language models for information retrieval. In Proceedings of the 14th ACM International Conference on Information and Knowledgs Management.ACM Press, 688-695

[86] Query expansion by mining user logs.IEEE Trans.Knowl.Data Engin. 15,4,829-839.

[87] **I**mproving retrieval performance by global analysis.In Proceedings of the 18th International Conference on Pattern Recognition.IEEE Computer Society, 703-706.

[88] Mining dependency relations for query in passage retrieval.In Proceedings of the 29th Annual International ACM SIGIR Conference on Reseatch and Develoment in Information Retrieval.ACM Press, 382-389.

[89] Concept-based query expansion.In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.ACM Press, 160-169.

[90] Query expansion using random walk models. In Proceedings of the 14th Conference on Information and Knowledge Management (CIKM'05).ACM Press, 704-711.

[91] Selecting good expansion terms for pseudorelevance feedback. In Proceedings of the 31st Annual International ACM SIGIR Conference on Researchaud Devlopment Information Retrieval. ACM Press, 243-250.

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

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

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

[95] G.Salton & C.Buckley: Improving Retrieval Performance by Relevance Feedback, Journal of The Amercican Society for Information Science, Vol.41, N=4, pp 288-297, 1990.

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

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

[98] D.Harman: Relevance Feedback Revisited: Conference on Research and Development in Information Retrieval (SIGIR), pp 1-10, 1992.

[99] Relavance feedback in information retrieval. In The SMART Retrieval System, G.Salton Ed, Prentice-Hall, Englewood Cliffs, NJ, 313-323.

[100] Re-examining the effects of adding relevance information in a relevance feedback environment.Info.Process.Manage.44, 3, 1086-1116.

[101] Probabilistic models for information retrieval based on divergence from randomness.Ph.D.thesis, Department of Comuting Science, University of Glasgow, UK.

[102] Relevance based language models. In Proceedings of the 24th Annual International ACM SIGIR Conference on Reseaech and Development in information Retrieval.ACMPress, 120-127.

[103] Zhai, C, Lafferty, J.Model-based feedback in the language modeling approach to information rretrieval.Proceedings of the 10th International Conference en Information and Knowledge ManagementACM Press, pp 403-410, 2001.

[104] Query expansion using term relationship in language models for information retrieval in Language models for information retrieval. In Proceedings of the 14th ACM International Conference on Information and Knowledgs Management.ACM Press, 688-695.

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

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

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

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

[109] 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.

[110] 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.

[111] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. *In proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 299-306, 2002.

[112] B. He and I. Ounis Inferring query performance using pre-retrieval predictors. *In Proceedings of Symposium on String Processing and Information Retrieval*, pages 43-54, 2004.

[113] Abdelkrim Bouramoul, Recherche d'Information contextuelle et sémantique sur le web, 2001.

[114] V. Faber. Clustering and the continuous K-means algorithm. *Los Alamos Science*, pages 138-144, 1994.

[115] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. *In Proceedings of ACM International Conference on Information and Knowledge Management*, pages 255-264, 2009.

[116] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In Proceedings of SIGIR, pages 335–336, Melbourne, Australia, 1998.

[117] Davood Rafiei, Krishna Bharat, and Anand Shukla. Diversifying web search results. In Proceedings of WWW, pages 781–790, Raleigh, North Carolina, USA, 2010.

[118] Zhicheng Dou, Sha Hu, Kun Chen, Ruihua Song, and Ji-Rong Wen. Multi-dimensional search result diversification. In Proceedings of WSDM, pages 475–484, Hong Kong, China, 2011.