REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET
DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE

MEMOIRE

De Fin d'étude

En vue de l'obtention du diplôme
Master académique en Informatique
Option : système d'information

THEME

Implémentation et évaluation d'une approche qui réduire les problèmes d'ambiguïté et de disparité des termes en recherche d'information

Proposé et dirigé par :
Mr HAMMACHE
Arezki

Réalisée par
Melle LAMRAOUI
Farida

2013/2014

Remerciement

Je tiens tout d'abord à remercie l'encadreure : Dr. HAMMACHE Arezki ma voire fait confiance, qui ma guider avec compréhension et gentillesse, je le remercie pour toute les soutiens qu'il a eu à m'apporter durant l'étude, pour l'aide scientifique et morale qu'il a sus m'apporter tout au long de ce mémoire.

Je tiens également à remercier l'ensemble des enseignants qui m'ont suivi durant mon cursus.

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

Un grand merci à Zina , Safia ,tofike ,Yassine et mes collègues de travail.

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

Je dédie ce travail à…

A ma très chère mère : *Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.*

A la mémoire de mon Père : *Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous.*

A mon très cher fiancé : *qui ma avoir donné le courage, le soutien et l'amour pour avoir terminé se travaille.*

A mes très chers frères : *Ils vont trouver ici l'expression de mes sentiments de respect et de reconnaissance pour le soutien qu'ils n'ont cessé de me porter.*

A mes très chères sœurs : *En témoignage de l'attachement, de l'amour et de l'affection que je porte pour vous. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et réussite.*

A mes chers ami (e)s : *Ils vont trouver ici le témoignage d'une fidélité et d'une amitié infinie.*

Liste des figures

Liste des tableaux

Sommaire

Sommaire

Sommaire

INTRODUCTION GENERALE

Introduction générale

Vue du nombre sans cesse croissant de documents électroniques disponibles sur Internet et dans les bases de données, retrouver des informations correspondant à un besoin est bien souvent considéré comme un processus cognitif très complexe, qui fait appel à de nombreux savoirs et se compose de diverses tâches, allant de la prise en compte du manque d'information jusqu'au traitement des données identifiées. Cette discipline est appelée Recherche d'Information (RI). Elle s'intéresse au développement des techniques et des outils qui permettent de retrouver une information intéressante afin de satisfaire un besoin en information, dite information pertinente. Ces outils sont appelés des Systèmes de Recherche d'Information (SRI). Ainsi, un SRI permet de sélectionner parmi un volume d'information, les informations pertinentes vis-à-vis d'un besoin en information. Dans ce système, ce dernier est exprimé sous forme de requête.

Pour retrouver les documents pertinents vis-à-vis d'une requête, le SRI compare la représentation de cette requête à la représentation de chaque document. Cette comparaison est réalisée au moyen d'une fonction de correspondance (Retrieval Status Value: RSV) et un score de pertinence est affecté à chaque document. Ces scores permettent de présenter à l'utilisateur les documents pertinents ordonnés.

Les modèles de recherche et représentation d'information sont basés sur le processus de mise en correspondance entre la requête utilisateur et documents de la collection .Le mécanisme de recherche détermine alors sur la base de degré de pertinence supposé des documents, ceux qui répondent au besoin de l'utilisateur. De nombreux modèles et stratégies sont développés.les modèles classiques, le modèle booléen, le modèle vectoriel et le modèle probabiliste sont fondés sur l'utilisation de théories mathématiques tant pour représentation des unités textuelles que pour le calcul de la pertinence des documents.

Dans le processus d'indexation, l'ensemble des descripteurs ou mots clés représentent des documents, ce qu'on appelle une présentation par le sac des mots. Plusieurs méthodes en développements, parmi où on trouve celle prenant en compte la proximité entre les termes (des termes adjacents) et utilisation d'unité de présentation plus complexe (N-grammes).

La difficulté pour l'utilisateur de formulée sa requête de recherche de manière efficace. Il a été montré que la faible pertinence des systèmes de recherche est du

principalement a des requêtes mal formés.les termes d'une requête formuler peuvent avoir plusieurs sens, termes nulle ou une requête très courte.

Notre travail s'inscrit dans le cadre de recherche d'information .Nous basons sur les deux points: (1) l'expansion de la requête automatique pour remédie le problème la disparité des termes (2) utilisation des mots composée comme unité d'indexe pour réduire problème d'ambiguïté.

L'expansion de la requête : c'est l'approche la plus utilisée pour pallier le problème de disparité des termes en recherche d'information. Le rôle major de cette technique est la sélection des termes à ajouter à la requête (extension de la requête utilisateur. Notre travail s'intéresse à la technique réinjection de pertinence. Le choix de terme se base sur la relation de cooccurrence entre les termes de la requête initiale et les termes des premiers documents retournes par la première recherche.

Traitement le problème d'ambiguïté : Pour résoudre ce problème, nous proposons une approche permettant une représentation plus précise des documents et des requête elle se base sur les termes composés. Il est généralement supposé que les termes composés sont moins ambigus que les mots simples, et ils représentent un sens plus précis.

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

Chapitre 1, nous dressons un état de l'art la de recherche d'information classique. La description des concepts de base de la recherche d'information, suivi des méthodes de sélection des termes d'indexation. Ensuite nous décrirons les différents modèles de recherche d'information. Nous décrivons aussi l'indexation avec mots composée et finalement nous présenté l'évaluation des SRI.

Chapitre 2, nous présentons l'expansion de requête automatique. Dans le premier point, nous définissons l'expansion de requêtes suivie du fonctionnement d'expansion de requête. Le second point, traite les différentes approches d'expansion de requêtes.

Chapitre 3, nous présentons notre approche et la présentation des résultats.

Chapitre I
La Recherche d'Information

I-1 Introduction

Avec l'augmentation rapide du volume documentaire stocké sous format numérique, il est devenu très difficile de trouver une information ou un document qui répond à un besoin utilisateur. La recherche d'Information (RI) est une discipline relativement ancienne qui s'intéresse à répondre à la problématique : comment retrouver un document parmi une collection de documents qui satisfait un besoin utilisateur **[01].**

Le but de la recherche d'information est de trouver les documents qui satisfont un besoin utilisateur. Si l'utilisateur juge qu'un document répond à son besoin, le document est dit pertinent. Dans un Système de Recherche d'Information (SRI), L'utilisateur exprime son besoin d'information sous forme d'une requête. Le SRI tente de trouver tous les documents pertinents et de rejeter les documents qui ne sont pas pertinents. Dans la pratique, l'ensemble des documents renvoyés par un SRI pour une requête est composé d'un sous-ensemble de documents pertinents et un sous-ensemble de documents non pertinents. Ces sous-ensembles déterminent la performance d'un SRI.

I-1Définition de la RI

La recherche d'Information (RI) est un domaine apparu en même temps que les ordinateurs. Au début, la RI se concentrait sur les applications dans les bibliothèques. A la fin des années 1960 et au début des années 1970, G. Salton a développé le système SMART **[02],** qui a grandement influencé le domaine de la RI.

La recherche d'information (RI) est un ensemble de méthodes et procédures ayant pour objet d'extraire d'un ensemble de documents, les informations voulues. Dans un sens plus large, la RI est toute opération (ou ensemble d'opérations) ayant pour objet la recherche, la collecte et l'exploitation d'informations en réponse à une question sur un sujet précis **[03].**

I-2 Système de recherche d'information (SRI)

I-2-1 Définition de SRI

Un SRI intègre un ensemble de modèles pour la représentation des unités d'information (documents et requêtes) ainsi qu'un processus de recherche/décision qui permet de sélectionner l'information pertinente en réponse au besoin exprimé par l'utilisateur à l'aide d'une requête **[04].**

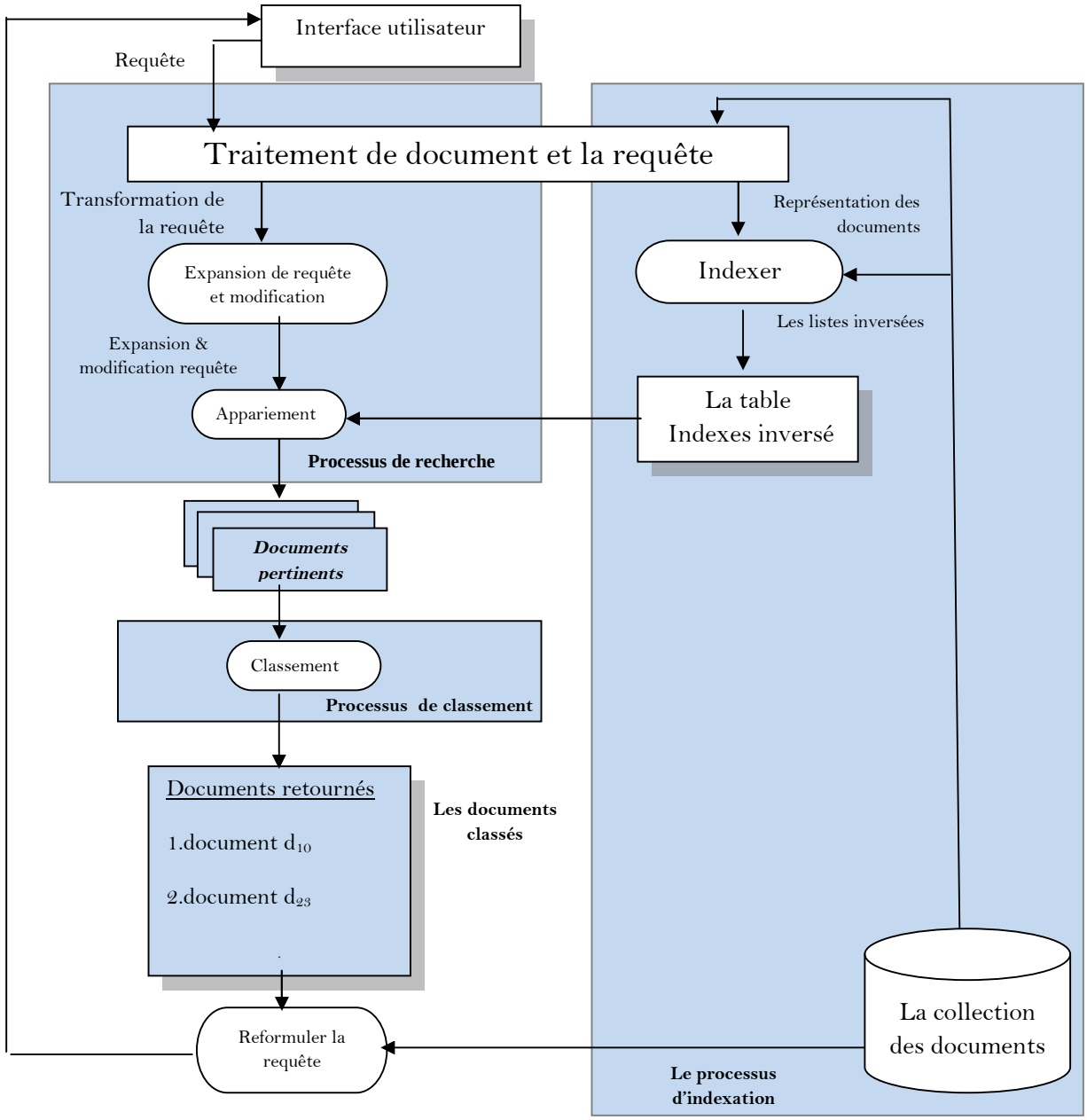


Figure I-1 : L'architecture d'un SRI [**05]**

I-2-2 les concepts de base d'un SRI

- **Collection de documents** : la collection de documents (ou fond documentaire, corpus) représente l'ensemble des informations exploitables, compréhensibles et accessibles par l'utilisateur. Une collection comporte un ensemble de granules documentaires **[06].**
- **Document**: Un document peut être un texte, une page WEB, une image, une bande vidéo, etc. Dans notre contexte, nous appelons document toute unité qui peut constituer une réponse à une requête d'utilisateur.

➨ **Requête**: Une requête exprime le besoin d'information d'un utilisateur. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.

➨ **Besoin d'information :** La notion de besoin en information en recherche d'informations est souvent assimilée au besoin de l'utilisateur. Trois types de besoin utilisateur ont été définis par **[07]** :

✓ **Besoin vérificatif** : l'utilisateur cherche à vérifier le texte avec les données connues qu'il possède déjà. Il recherche donc une donnée particulière, et sait même souvent comment y accéder. La recherche d'un article sur Internet à partir d'une adresse connue serait un exemple d'un tel besoin. Un besoin de type vérificatif est dit stable, c'est-à-dire qu'il ne change pas au cours de la recherche.

✓ **Besoin thématique connu** : l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et domaine connus. Un besoin de ce type peut être stable ou variable ; il est très possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche. Le besoin peut aussi s'exprimer de façon incomplète. C'est ce qu'on appelle dans la littérature le label.

✓ **Besoin thématique inconnu** : cette fois, l'utilisateur cherche de nouveaux concepts ou de nouvelles relations hors des sujets ou domaines qui lui sont familiers. Le besoin est intrinsèquement variable et est toujours exprimé de façon incomplète.

➨ **Pertinence :** La pertinence est une notion fondamentale et cruciale dans le domaine de la RI. Les travaux de recherche récents **[08] [9]** s'accordent sur la difficulté de la définition de la pertinence et mettent en exergue deux types de pertinence. La pertinence système **[10]** et la pertinence utilisateur **[11] [08] [12] .**

✓ ***La pertinence système*** : est déterministe, objective et de finir `a travers les modèles de RI. Elle est souvent traduite par un score cherchant à évaluer la pertinence des documents vis-à-vis d'une requête. Cette pertinence est mesurée par une similarité de représentation document-requête (modèle vectoriel), une probabilité de pertinence des documents étant donnée une requête (modèle probabiliste).

✓ ***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, et évolué dans le temps d'une

recherche.**[11] [08] [12].**Une information non pertinente pour une requête peut être jugée pertinente plus tard par les connaissances de l'utilisateur sur le sujet à évolué.

I-3 Les processus de Système Recherche D'information

L'architecture générale d'un SRI est donnée par la Figure I-1. Il met en œuvre les trois processus : le processus d'indexation, le processus d'appariement et le processus de reformulation. Ces processus seront présentés d'une manière détaillée dans les sections qui suivent.

Le processus d'indexation consiste à décrire chaque document de la collection et chaque requête par un ensemble de descripteurs. Ces descripteurs permettent de refléter au mieux le contenu du document ou de la requête. Ainsi, les documents et les requêtes sont représentés dans un même espace dit espace d'indexation. Dans cet espace, chaque document et chaque requête possède une représentation interne unique.

Le processus d'appariement ou de comparaison consiste à mettre en correspondance la représentation de la requête avec les représentations des documents. Cette mise en correspondance permet de calculer un degré de ressemblance ou de similarité entre chaque requête et chaque document de la collection. En se basant sur ce degré, les documents qui sont jugés similaires (par le SRI) à la requête sont par la suite retournés à l'utilisateur. Certains systèmes permettent de présenter ces documents dans une liste triée selon l'ordre décroissant de leur ressemblance avec la requête utilisateur.

I-3-1. Indexation

L'indexation est le processus permettant de créer une représentation des documents et des requêtes facilement manipulable par un système de recherche d'information. Elle consiste à analyser les documents afin d'extraire un ensemble de mots clés servant comme descripteurs des documents. Ils peuvent être des mots simples ou bien des mots composés Il existe trois types d'indexation **[13]**:

A. L'indexation manuelle

L'indexation manuelle est réalisée par des documentalistes. Ces experts ont pour tâche de caractériser au mieux les idées contenues dans les unités documentaires. Cette indexation requiert un important effort intellectuel et cognitif pour identifier et décrire l'essence des unités documentaires.

Ce type d'indexation permet d'obtenir une caractérisation performante mais subjective du contenu des unités documentaires car cette approche dépend fortement des connaissances du domaine des documentalistes. L'indexation manuelle trouve ses limites pour de grandes bases de documents qui nécessitent énormément de temps pour être traitées.

B. L'indexation automatique

L'indexation automatique est la plus communément utilisée. Ce type d'indexation regroupe un ensemble de traitements automatisés sur un document : l'extraction automatique des termes du document, l'élimination des mots vides, la lemmatisation ou la radicalisation des mots et enfin la création de l'index.

C. L'indexation semi automatique

Ce type d'indexation combine les méthodes d'indexation manuelle et automatique en privilégiant toutefois l'intervention humaine. Ainsi, les experts caractérisent les idées contenues dans une unité documentaire sous la forme de méta-informations. Une indexation automatique est ensuite réalisée pour l'unité documentaire en tenant compte de ces méta-informations.

I-3-1-1 Les méthodes de l'indexation automatique

De nombreuses méthodes (linguistiques, statistiques, assignation, etc.) ont été développées pour concevoir, ou améliorer dans certains cas, les systèmes ou les logiciels d'indexation automatique**.**

A) Les méthodes linguistiques

L'un des problèmes qui se posent en utilisant des mots comme index est qu'ils sont ambigus, c'est-à-dire qu'un mot peut désigner plusieurs sens, la conséquence est qu'un document contenant un mot ne désigne pas nécessairement le même sens que ce même mot dans la requête.

Ainsi, c'est une source de bruit (documents non pertinents retournés). C'est pour cela qu'on a recours aux méthodes linguistiques.

Ces méthodes utilisent différentes analyses linguistiques pour le traitement du langage naturel appelées TAL. On y distingue les niveaux d'analyse suivants :

- *Niveau morphologique* : à ce niveau on isole chaque terme par le biais d'un dictionnaire qui permet le contrôle des chaînes de caractères et le repérage des mots.

- *Niveau lexical* : il s'agit ici de polymorphisme de mot appartenant à un même concept, le traitement se traduit par la suppression des variantes combinatoires (flexion, dérivation, conjugaison) pour obtenir une forme canonique par lemmatisation (extraction de la racine du mot). Les outils nécessaires à ce procédé de lemmatisation sont des dictionnaires de correspondances entre formes fléchies ou dérivées et formes canoniques (par exemple produira, produisent, etc., auront la même forme canonique *produire*).

- *Niveau syntaxique* : les techniques syntaxiques partent des phrases et consistent à déterminer les regroupements structurels des mots au sein des phrases et les relations entre les mots, afin de lever les ambiguïtés portant sur les termes retenus au sens de cooccurrence.
- *Niveau pragmatique* : l'être humain fait appel à ses connaissances du monde et du contexte pour résoudre les cas de polysémie, dans le cas du traitement automatique, on fait appel aux réseaux sémantiques.

B) La méthode statistique

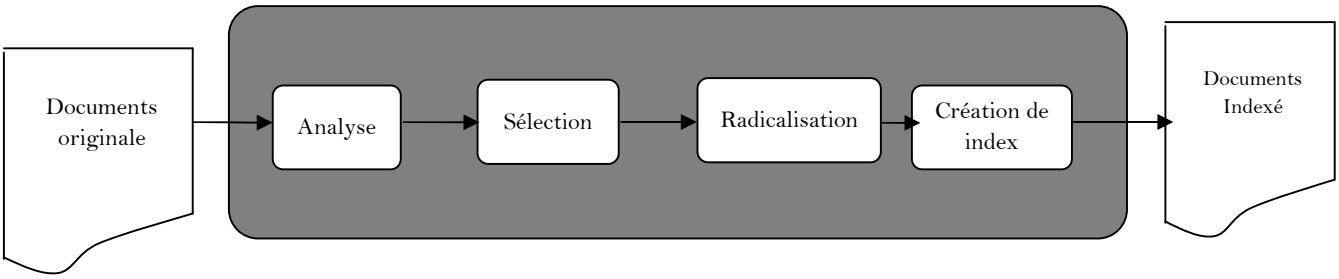


Figure I-2 Suite du traitement effectué lors de l'indexation

Les méthodes statistiques, ou encore méthodes par extraction, sont basés sur deux types de traitement :

✓ Le premier est basé sur le calcul de fréquence statistique (avec prise en compte des synonymes (en utilisant un dictionnaire) ou sur racines) ;
✓ Le deuxième type de traitement est construit sur une recherche de voisinage (que l'on appelle également méthodes par cooccurrence (présence simultanée de deux ou plusieurs mots dans une phrase)).

La méthode statistique est, en fait basée, sur le mot plein (le lexique). En effet, une fois que tous les mots vides, ceux qui ne portent pas de sens en soi (mots grammaticaux, articles, etc), sont éliminés, il ne reste que les mots pleins. On tient compte du fait que plus un mot plein est

présent dans un texte et plus il est significatif, et servira ainsi de descripteur et pourra apparaître lors d'une interrogation. Les différentes phases d'indexation en utilisant la méthode statistique sont :

1) L'analyse lexicale :

L'analyse lexicale est un processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est une unité lexicale ou un radical **[14].** L'analyse lexicale permet de reconnaître les espaces de séparation des mots, les chiffres, les ponctuations, etc.

2) L'élimination des mots vides :

Un des problèmes majeurs de l'indexation consiste à extraire les termes significatifs et à éviter les mots vides (pronoms personnels, prépositions, etc).Les mots vides peuvent aussi être des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc.). On distingue deux techniques pour éliminer les mots vides:

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

3) La lemmatisation :

Un mot donné peut avoir différentes formes dans un texte, mais leur sens reste le même ou très similaire. On peut par exemple citer économie, économiquement, économétrie, économétrique, etc. il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème, une substitution des termes par leur racine, ou lemme, est utilisée.

On distingue plusieurs types stratégies de lemmatisation, on cite : les dictionnaires, l'élimination des affixes, la troncature, les N-gramme, etc.

- *L'élimination des affixes* : cette stratégie consiste à éliminer par itération les suffixes en utilisant un dictionnaire contenant tous les suffixes possibles, exemple : algorithme de Porter **[15].**
- *La troncature :* la troncature des mots consiste à tronquer un mot à partir d'un rang précis, afin d'obtenir son radical. Cette technique permet ainsi de réduire les variables morphologiques des mots issus de la même racine. Par exemple, pour la langue française, une troncature à sept caractères est communément utilisée (exemple: référence, référencement, référen).

4) Le choix des descripteurs

Le processus d'indexation consiste à extraire à partir du texte des documents, les *termes clés* appelés aussi *descripteurs*. Les *descripteurs* représentent l'information atomique d'un index. Ils sont censés indiquer de quoi parle le document **[16].**

D'après BAZIZ **[17]** Les descripteurs peuvent être :

- ***Les mots simples*** du texte du document en soustrayant les mots outils (ou *mots vides*),
- ***Les lemmes*** ou les *racines* des mots extraits,
- **Les *concepts*** qui sont des expressions contenant un ou plusieurs mots. Ces concepts peuvent être écrits de manière libre par l'utilisateur ou, ce qui est souvent le cas, choisis dans une liste de concepts (on parle alors de vocabulaire contrôlé).
- ***Les N-grammes*** : qui sont une représentation originale d'un texte en séquence de N caractères consécutifs. On trouve des utilisations de bigrammes et trigrammes dans la recherche d'information.
- ***Les contextes*** : ici les descripteurs peuvent êtres des termes n'apparaissant pas explicitement dans le texte du document mais ayant un lien sémantique et/ou de cooccurrence avec les mots du document.
- ***Les groupes de mots*** : un groupe de mots ("phrase" en anglais), est souvent plus riche sémantiquement que les mots qui le composent pris séparément. En effet, à titre d'exemple, le terme composé "moteur diesel" est plus précis que "moteur " et "diesel" pris isolément. Cet argument a conduit à considérer les groupes de mots comme unité de base dans le langage d'indexation.

5) La création de l'index :

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants : les fichiers inverses (”inverted files”), les tableaux de suffixes (”suffix arrays”) et les fichiers de signatures (”signature files”).

Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications. Les fichiers inverses sont composés de deux éléments principaux :

- Le vocabulaire, qui est l'ensemble de tous les mots différents du texte ;
- Les occurrences (posting) : pour chaque mot, il s'agit de la liste de toutes les positions dans le texte pour lesquelles le mot apparaît.

Généralement, les structures de données sont compressées avant d'être enregistrées sur le disque, ce qui permet de réduire la taille de l'index. Parmi les méthodes de compression utilisées on peut citer la méthode Elias Gamma **[18]** qui opère au niveau bit requérant ainsi beaucoup d'opérations pour la compression et la décompression. D'autres méthodes plus efficaces, opérant au niveau octet ont été proposées dans **[19].**

c) La méthode mixte

De nombreuses méthodes sont développées dans le but d'avoir une meilleure description du contenu des documents. Les méthodes combinent les avantages de plusieurs méthodes en même temps, il s'agit pour la plupart d'une combinaison des méthodes linguistique et statistique.

I-3-2 Appariement document-requête

Le processus d'appariement consiste à comparer la représentation de requête avec les représentations des documents. Il calcule pour chaque couple requête document, une mesure appelée pertinence système qui reflète le degré de similarité entre la requête et le document considéré. Le processus d'appariement se base sur une fonction de similarité (ou de correspondance) noté RSV (Retrieval Status Value). Cette fonction est différente d'un modèle de recherche d'information à un autre. D'ailleurs un modèle de RI est caractérisé par sa fonction de similarité et son modèle d'indexation.

Selon **[20]** plusieurs modèles de recherche d'information ont été proposés. Nous citons dans la suite les principaux modèles.

I-3-3 Reformulation de requête

Dans un processus de recherche d'information, l'utilisateur a souvent tendance à ajuster sa requête initiale. La requête initiale subit alors plusieurs ajustements selon les documents qui sont retournés par le SRI. En effet, l'utilisateur trouve des difficultés pour formuler sa requête d'une manière exacte et par la suite les documents retournés par le SRI ne lui conviennent pas. Cette opération est dite la reformulation de la requête. Elle consiste à construire une nouvelle requête à partir de la requête utilisateur initiale. Cette nouvelle requête exprime mieux le besoin utilisateur. On distingue différentes méthodes de reformulation de requête, elles sont définies dans le chapitre suivant.

I-3-4 Les modèles de recherche d'information

I-3-4-1 Les modèles booléen

Le modèle booléen **[21]** est le premier modèle utilisé en RI. Il est basé sur la théorie des ensembles et sur l'algèbre de Boole.

Les requêtes étant aussi représentées par un ensemble de mots-clés exprimant le besoin d'information, le processus de recherche consiste à trouver les documents qui décrivent exactement par la requête, c'est-à-dire les documents contenant tous les termes de la requête (intersection des ensembles). Cette approche est donc très stricte et ne classe les documents que dans deux catégories : les pertinents et les non pertinents ($RSV\ (q,d) \in \{1,0\}$).

A) Le modèle booléenne et booléenne étendu

Afin de remédier aux insuffisances du modèle booléen de base, un modèle booléen étendu a été proposé par Salton **[22]** Dans celui-ci les termes dans les documents et les requêtes sont pondérés afin de permettre un appariement document requête approché.

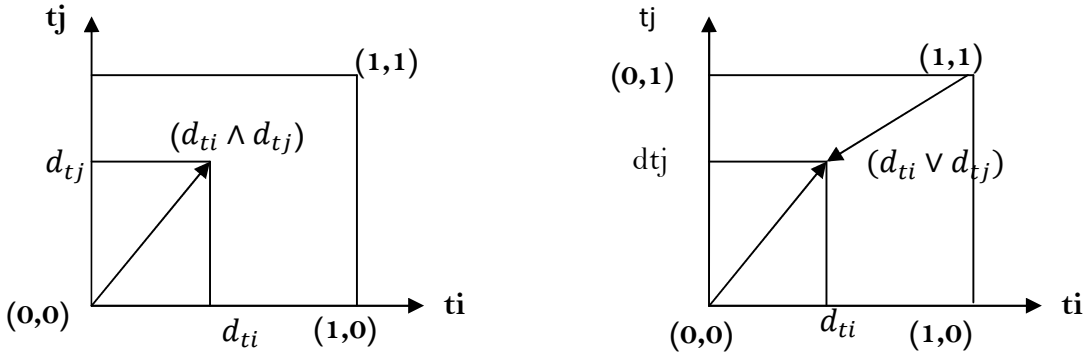


Figure I-3 Principe du modèle booléen étendu

Considérons les requêtes portant sur deux termes $q_1 = t_i \vee t_j$ et $q_2 = t_i \wedge t_j$ et un document d=(d_{ti}, d_{tj}) où d_{ti}, d_{tj} sont respectivement les points des termes t_i, t_j dans d.

La similarité sim(d,q_1) entre requête $q_1 = t_i \vee t_j$ et le document d sera la distance normalisé entre le point (0,0) et le point d=(dti, dtj) est :

$$sim((D, d_{ti} \vee d_{tj}) = \sqrt{\frac{(d_{ti}-0)^2 + (d_{tj}-0)^2}{2}} \qquad \textbf{(I-6)}$$

De manière analogue la similarité sim(d,q_2) entre la requête $q_2 = t_i \wedge t_j$ et le document d sera distance inverse entre le point(1,1) et le point d=(dti, dtj)

$$sim((D, d_{ti} \wedge d_{tj}) = \sqrt{\frac{(1-d_{ti})^2 + (1-d_{1-tj})^2}{2}} \quad \textbf{(I-7)}$$

Dans le cas de requête pondérés les deux formules précédentes peuvent être étendues de la manière suivante :

$$sim((D, d_{ti} \vee d_{tj}) = \sqrt{\frac{q_{ti}^2 * d_{ti}^2 + q_{tj}^2 * d_{tj}^2}{q_{ti}^2 + q_{tj}^2}} \quad \textbf{(I-8)}$$

$$sim((D, d_{ti} \wedge d_{tj}) = \sqrt{\frac{q_{ti}^2 * (1-d_{ti})^2 + q_{tj}^2 * (1-d_{1-tj})^2}{q_{ti}^2 + q_{tj}^2}} \quad \textbf{(I-9)}$$

B) Le modèle basé sur les ensembles flous

Contrairement à la logique binaire, la logique flou **[23]** est une logique fondée sur des variables pouvant prendre d'autre valeurs outre les valeurs "vrai" ou "faux"(0 ou 1), des valeurs intermédiaires comprise dans l'intervalle [0, 1].

Le modèle booléen flou est basé sur la théorie des ensembles flous. Il a été introduit par Salton en 1989 **[24].** Dans ce modèle et contrairement au modèle booléen de base, les résultats des opérations logiques, i.e. le coefficient de similarité entre la requête est un document, peuvent prendre des valeurs comprises dans l'intervalle [0, 1].

I-3-4-2 Les modèles vectoriels

Le modèle algébrique est un modèle vectoriel, créé au début des années 70 par Gerard Salton pour le système SMART (System for the Mechanical Analysis and Retrieval of Text) où l'on représente les documents et les requêtes par des vecteurs dans un espace multidimensionnel dont les dimensions sont les termes issus de l'indexation **[19].** Les documents et la requête sont représentés comme des vecteurs dans le repère des termes. On ramène une proximité sémantique à une mesure de distance géométrique.

A) Le modèle vectoriel standard

Dans ce modèle, un document est représenté sous forme d'un vecteur dans l'espace vectoriel composé de tous les termes d'indexation. Les coordonnées d'un vecteur document représentent les poids des termes correspondants. Formellement, un document *di* est représenté par un vecteur de dimension *n*,

$$d_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{in}) \qquad \text{pour i = 1, 2, …, m.}$$

Où

w_{ij} est le poids du terme *tj* dans le document *di*,

m est le nombre de documents dans la collection,

n est le nombre de termes d'indexation.

Une requête *Q* est aussi représentée par un vecteur de mots-clés défini dans le même espace vectoriel que le document.

$$Q = (wQ_1, wQ_2, wQ_3, \dots, wQ_n), \qquad \textbf{(I-10)}$$

Où

wQ_j est le poids de terme *tj* dans la requête *Q*. Ce poids peut être soit une forme de *tf*idf*, soit un poids attribué manuellement par l'utilisateur.

Sous l'angle de ce modèle, le degré de pertinence d'un document relativement à une requête est perçu comme le degré de corrélation entre les vecteurs associés. Ceci nécessite alors la spécification d'une *fonction de calcul de similarité* entre vecteurs mais également du principe de construction qui se traduit par la *fonction de pondération*.

1) Les fonctions de pondération

La pondération des termes permet d'exprimer le pouvoir discriminant d'un terme *tj* dans le document *di*. Le pouvoir discriminant d'un terme est sa capacité à distinguer les documents les uns des autres. La pondération consiste à affecter à chaque terme ti d'un document dj un poids wij. Ce poids exprime le degré de représentativité du terme dans le document **[01].**

La plupart des formules pondération proposées dans la littérature de RI se basent sur deux facteurs: un facteur de pondération locale et un facteur de pondération globale.

La pondération locale tfij : La pondération locale exploite des informations locales spécifiques au document dans lequel le terme d'indexation apparait. En général, la pondération locale d'un terme ti dans un document dj, est une fonction de fréquence de ce

terme dans le document dj. Cette pondération est notée souvent tfij. Dans la pondération locale exploite des informations locales spécifiques au document dans lequel le terme d'indexation apparait. En général, la pondération locale d'un terme ti dans un document dj, est une fonction de fréquence de ce terme dans le document dj. Cette pondération est notée souvent tfij. Dans la littérature on trouve de nombreuses formules de la pondération locale. Nous citons en particulier **[25][26]:**

$$tf_{ij} = f(t_i, d_j) \quad \textbf{(I-11)}$$

$$tf_{ij} = 1 + \log(f(t_i, d_j)) \quad \textbf{(I-12)}$$

$$tf_{ij} = 0.5 + 0.5 * \frac{f(t_i,d_j)}{max_{t_i \epsilon\, d_j}(f(t_i,d_j))} \quad \textbf{(I-13)}$$

Où (ti, dj) est la fréquence du terme ti dans le document dj

La pondération globale IDFi : La pondération globale d'un terme est exprimée en fonction du nombre total de documents dans le corpus et en fonction du nombre de documents contenant ce terme. Cette pondération est notée IDFi (Inverse Document Frequency). Dans la littérature plusieurs formules de calcul de cette mesure ont été proposées, dont nous citons **[21] [27] :**

$$IDF_i = \log\left(\frac{n_i}{N}\right) \quad \textbf{(I-14)}$$

$$IDF_i = \log\left(\frac{N-n_i}{N}\right) \quad \textbf{(I-15)}$$

Où

ni est le nombre de documents où le terme ti apparait et N est le nombre total de documents dans le corpus.

Les fonctions de similarité

La fonction de similarité permet de mesurer la ressemblance des documents et de la requête. Les types de mesures les plus répandus sont représentées dans le tableau ci-dessus :

Les mesures	**Les termes de l'espace vectoriel**	**Calcule de la similarité Sim (Q , D_j)**
Produit scalaire	$\lvert Q \cap D \rvert$	$\sum_{i=1}^{M} qi \,.\, di\, j$
Mesure de Jaccard	$\frac{\lvert Q \cap D \rvert}{\lvert Q \rvert + \lvert D \rvert - \lvert Q \cap D \rvert}$	$\frac{\sum_{i=1}^{M} qi \,.\, dij}{\sum_{i=1}^{M} qi^2 + \sum_{i=1}^{M} dij^2 - \sum_{i=1}^{M} qi.\, dij}$

Coefficient de DICE	$\frac{2.\lvert Q \cap D\rvert}{\lvert Q\rvert + \lvert D\rvert}$	$\frac{2\ .\ \sum_{i=1}^{M} qi\ .\ dij}{\sum_{i=1}^{M} qi^2 + \sum_{i=1}^{M} di^2 - \sum_{i=1}^{M} qi.\ di}$
Mesure cosines	$\frac{\lvert Q \cap D\rvert}{\sqrt{\lvert \mathbf{Q}\rvert}\ .\ \sqrt{\lvert \mathbf{D}\rvert}}$	$\frac{\sum_{i=1}^{M} qi\ .\ dij}{\sqrt{\sum qi^2}.\sqrt{\sum dij^2}}$
Mesure recouvrement	$\frac{\lvert Q \cap D\rvert}{\min(\lvert Q\rvert, \lvert D\rvert)}$	$\frac{\sum_{i=1}^{M} qi\ .\ dij}{\min(\sum di\ , \sum qi)}$

Tableau I.1 les mesures de la similarité utilisé dans le modèle vectoriel

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. Les documents dans le modèle vectoriel, sont en effet restitués dans un ordre décroissant de leur degré de similarité avec la requête. Plus le degré de similarité d'un document est élevé, plus le document ressemble à la requête, et donc pertinent pour l'utilisateur.

I-3-4-3 Les modèles probabilistes

Le modèle probabiliste consiste à calculer la pertinence d'un document en fonction de pertinences connues pour d'autres documents.

A) Modèle probabiliste classique (de base)

Selon ROBERTSON et al **[25],** dans le modèle probabiliste, les documents et la requête sont représentés par des vecteurs dans l'espace d'indexation comme dans le modèle vectoriel. Dans ces vecteurs les pondérations des index sont binaires. Pour une requête Q l'ensemble des documents disponibles est divisé en deux sous ensembles : l'ensemble R des documents pertinents et l'ensemble NR des documents non pertinents. A chaque document D on associe deux probabilités :

P(R/D) : la probabilité que le document D soit pertinent pour la requête Q

P(NR/D) : la probabilité que le document D soit non pertinent pour la requête Q

La similarité entre le document D et la requête Q est alors calculée en fonction de ces deux probabilités de la manière suivante :

$$\mathrm{sim}(D, Q) = \frac{P(R \backslash D)}{P(NR \backslash D)} \qquad \textbf{(I-16)}$$

I-3-4-4Modèle de langue

A) Idée de base :

Par « modèle de langue », on désigne une fonction de probabilité *P* qui assigne une probabilité *P*(S\M) à un mot ou à une séquence de mots *S* en une langue. Une fois cette fonction définie, il est possible d'estimer la probabilité d'une séquence de mots quelconque dans la langue, ou d'un point de vue générative, d'estimer la probabilité de générer cette séquence de mots à partir du modèle de la langue M.

Considérons la séquence S composée des mots suivants : m_1, m_2,…, m_k. La probabilité *P*(*s*) peut être calculée comme suit :

$$P(S\backslash M) = \prod_{i=1}^{k} P(m_i \backslash m_{i-n+1} \dots m_{i-1}) \qquad \textbf{(I-17)}$$

Si on utilise la règle de chaîne en théorie de probabilité pour calculer cette probabilité, il y a souvent trop de paramètres (c'est-à-dire $P(m_i|m_1 \dots m_{i-1})$) à estimer, et ceci est souvent impossible

de réaliser. Ainsi, dans les modèles de langue utilisés en pratique, des simplifications sont souvent faites. En général, on suppose qu'un mot m_i ne dépend que de ses *n*-1 prédécesseurs immédiats, c'est-à-dire :

$$\mathrm{P}(\mathrm{m_i}\backslash \mathrm{m_1} \dots \mathrm{m_{i-1}}) = \mathrm{P}(\mathrm{m_i}\backslash \mathrm{m_{i-n+1}} \dots \mathrm{m_{i-1}}) \qquad \textbf{(I-18)}$$

On utilise, dans ce cas, un modèle de langue n-gramme. En particulier, les modèles souvent utilisés sont les modèles uni-gramme, bi-gramme et tri-gramme comme suit :

Uni-gramme : $$\boldsymbol{P(S\backslash M) = \prod_{i=1}^{k} P(m_i)} \qquad \textbf{(I-19)}$$

Bi-gramme : $$\boldsymbol{P(S\backslash M) = \prod_{i=1}^{k} P(m_i \backslash m_{i-1}) = \prod_{i=1}^{k} \frac{P(m_{i-1} m_i)}{P(m_{i-1})}} \qquad \textbf{(I-20)}$$

Trie-gramme:
$$\boldsymbol{P(S\backslash M) = \prod_{i=1}^{k} P(m_i \backslash m_{i-2}\, m_{i-1})}$$
$$\boldsymbol{= \prod_{i=1}^{k} \frac{P(m_{i-2} m_{i-1} m_i)}{P(m_{i-2} m_{i-1})}} \qquad \textbf{(I-21)}$$

Ce que l'on doit estimer sont les probabilités $P(m_i)$ (un-gramme n=1), $P(m_{i-1}m_i)$ (bi-gramme n=2) et $P(m_{i-2}m_{i-1}\, m_i)$ (tri-gramme n=3) pour la langue. Cependant, il est difficile d'estimer ces probabilités pour une langue dans l'absolue. L'estimation ne peut se faire que par rapport à un

corpus de textes ***C***. Si le corpus est suffisamment grand, on peut faire l'hypothèse qu'il reflète la langue en général. Ainsi, le modèle de langue peut être approximativement le modèle de langue pour ce corpus $P(\bullet|\boldsymbol{C})$. Selon les fréquences d'occurrence d'un n-gramme α, sa probabilité $P(\alpha|C)$ peut être directement estimée comme suit :

$$\mathrm{P}(\alpha) = \frac{|\alpha|}{\sum_{\alpha_j \in C}|\alpha_j|} = \frac{|\alpha|}{|C|} \qquad \textbf{(I-22)}$$

Où

$|\alpha|$: est la fréquence d'occurrence du n-gramme α dans ce corpus, α_i est un n-gramme de la même longueur que α, et $|C|$ est la taille du corpus (c'est-à-dire le nombre total d'occurrences de mots). Ces estimations sont appelées les estimations de vraisemblance maximale (***M**aximum **L**ikelihood*, ou ML). On désignera aussi ces estimations par P_{ML}.

Un problème bien connu des modèles de recherche d'information est la prise en compte des probabilités nulles par le lissage. En effet si un mot m n'apparaît pas dans le document D la probabilité P(m|D) est égale à 0. Les probabilités étant multipliées entre elles, l'absence d'un seul mot donnerait une probabilité P(S\M) = 0. Nous réglons ce problème en lissant le modèle de langue avec les techniques de lissage. Ces techniques sont détaillées dans la section suivant.

B) Les techniques de Lissage

Plusieurs techniques de lissage ont été proposées, nous présentons ci-dessous quelques unes d'entre elles :

Lissage de Laplace

Le lissage de Laplace consiste à ajouter la fréquence 1 à tous les n-grammes. Cette méthode est aussi appelée la méthode « ajouter-un ». Pour un n-gramme α, sa probabilité est estimée comme suit (où *V* est l'ensemble du vocabulaire d'indexes) :

$$\mathrm{P_{LL}}(\alpha) = \frac{|\alpha| + 1}{\sum_{\alpha_j \in C}|\alpha_j| + 1} = \frac{|\alpha| + 1}{|C| + N} \qquad \textbf{(I-23)}$$

Où

N est le nombre de n-grammes (distincts) et |C| est la taille du corpus.

Lissage Good-Turing

L'idée de ce lissage est de modifier la fréquence d'occurrence observée de la façon suivante : Soit un n-gramme α qui apparaît *r* fois dans le corpus. On modifie cette fréquence à *r** suivante :

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \qquad \textbf{(I-24)}$$

Où :

$\boldsymbol{n_r}$ est le nombre de n-grammes apparus ***r*** fois dans le corpus. Ainsi, l'estimation de la probabilité devient la suivante :

$$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_j \in C} r^*_j} \qquad \textbf{(I-25)}$$

Dans cette méthode la fréquence d'ordre $\frac{r^*}{r}$ pour un n-gramme vu sera redistribuée sur les n-grammes non vus dans le corpus. La méthode de Good-Turing est recommandée pour les n-grammes de faibles fréquences, car elle n'effectue pas de grandes modifications comme c'est le cas pour les n-grammes de grandes fréquences.

Lissage Backoff

Le lissage « Backoff » consiste à utiliser un modèle du même ordre (par exemple, bi-gramme) si le n-gramme est observé dans le corpus, mais utiliser un modèle d'ordre inférieur (par exemple, uni-gramme) si ce n'est pas le cas. Par exemple, dans le lissage Katz, on peut combiner le modèle bi-gramme avec un modèle uni-gramme comme suit :

$$P_{Katz}(m_i|m_{i-1}) = \begin{cases} P_{GT}(m_i|m_{i-1}) & \text{Si } |m_i|m_{i-1}| > 0 \\ \alpha(m_i)P_{Katz}(m_{i-1}) & \text{sinon} \end{cases} \qquad \textbf{(I-25)}$$

Dans cette méthode, la diminution de fréquence utilisée dans P_{GT} est redistribuée au modèle d'ordre inférieur (uni-gramme). $\alpha(m_i)$ est un paramètre qui détermine la part de cette redistribution à m_i, déterminée comme suit :

$$\alpha(m_{i-1}) = \frac{1-\sum_{m_i:|m_{i-1}m_i|>0} P_{GT}(m_i|m_{i-1})}{1-\sum_{m_i:|m_{i-1}m_i|>0} P_{ML}(m_i)} \qquad \textbf{(I-26)}$$

Le lissage de Katz est proposé pour palier au problème posé par les n-grammes de hautes fréquences.

Lissage par interpolation (Jelinek Mercer)

Le lissage par interpolation, par exemple de Jelinek-Mercer, consiste à combiner un modèle avec un ou des Modèles d'ordre inférieur systématiquement, plutôt que d'utiliser ce dernier seulement pour dans le cas de fréquence 0 Pour une combinaison de modèle bi-gramme avec le modèle uni-gramme, on a

$$P_{JM}(m_i|m_{i-1}) = \lambda_{m_{i-1}} P_{ML}(m_i|m_{i-1}) + (1 - \lambda_{m_{i-1}}) P_{JM}(m_i) \qquad \textbf{(I-27)}$$

Où

$\boldsymbol{\lambda_{m_{i-1}}}$ est un paramètre déterminé de telle manière à maximiser l'espérance des données. Ce paramètre peut dépendre du mot $\boldsymbol{m_{i-1}}$ ou il peut être attribué d'une valeur identique pour tous les mots. Sa valeur est souvent déterminée avec le processus de maximisation de l'espérance (EM).

Comme on pourra voir plus loin, dans le contexte d'utilisation des modèles de langue en RI, on se limite souvent aux modèles uni-gramme. Ce type de lissage consiste à combiner le modèle de langue considéré avec un ou plusieurs modèles de références estimés sur d'autres corpus d'apprentissages. Typiquement, dans le cas de collection de documents, on pourrait par exemple estimer le modèle de document en le combinant avec le modèle de la collection. Dans ce cas, le modèle de document est exprimé ainsi

$$P_{JM}(m_i|d) = \lambda P_{ML}(m_i|C) + (1 - \lambda) P_{ML}(m_i|d) \qquad \textbf{(I-28)}$$

Lissage Dirichlet

Le lissage précédent ne tient pas compte de la taille des échantillons. Pour remédier à cela, le lissage de Dirichlet exploite les valeurs de $\boldsymbol{\lambda}$ en fonction de la taille de l'échantillon. Dans ce cas cette formule s'écrit comme suit:

$$P_{Dir}(m_i|\, d) = \frac{|d|}{|d| + \mu} P_{ML}(m_i|d) + \frac{\mu}{|d| + \mu} P_{ML}(m_i|C) = \frac{|d| P_{ML}(m_i|d) + \mu P_{ML}(m_i|C)}{|d| + \mu} \qquad \textbf{(I-29)}$$

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

Avec $P_{ML}(m_i|d) = \frac{tf(m_i d)}{|d|}$ **(I-30)**

Où

$|\boldsymbol{d}|$ est la taille du document (le nombre d'occurrences de mots), $\boldsymbol{tf(m_i d)}$est la fréquence du mot $\boldsymbol{m_i}$ dans d et $\boldsymbol{\mu}$ est un paramètre appelé pseudo fréquence.

I-4Au-delà des mots simples

La majorité des approches (modèles) développées en RI se basent sur l'utilisation des mots simples comme unités de représentation des documents et des requêtes, souvent appelé représentation en sac de mots. Ces approches posent deux problèmes, l'ambiguïté des mots et leur disparité.

I-4-1 L'ambiguïté des mots

Dite ambiguïté lexicale, se rapporte à des mots lexicalement identiques et portant des sens différents. Elle est généralement divisée en deux types Krovetz **[28]**: l'ambiguïté syntaxique et l'ambiguïté sémantique.

L'ambiguïté syntaxique se rapporte à des différences dans la catégorie syntaxique. Par exemple, « *play* » peut apparaître en tant que nom ou verbe.

L'ambiguïté sémantique se rapporte à des différences dans la signification, et est décomposée en homonymie et polysémie selon que les sens sont liés ou non.

Ce problème conduit à avoir des documents non pertinents en réponse à une requête contenant des mots ambigus. Par exemple, des documents sur la « programmation java » peuvent être renvoyés en réponse à la requête « aéroport de java », car le terme java contient plus d'un sens (ile, programmation, etc.).

I-4-2 La disparité des mots

Se réfère à des mots lexicalement différents mais portant un même sens. Ceci implique que des documents, pourtant pertinents, ne partagent pas de mots avec la requête, ne sont pas retrouvés.

En plus de l'expansion de la requête (Chapitre II), diverses approches ont été proposées pour remédier à ces problèmes. Ces approches permettent d'incorporer ou d'utiliser des informations conceptuelles ou sémantiques dans les méthodologies de recherche (l'indexation sémantique, l'indexation conceptuelle et l'indexation par des mots composés).

Nous présentons ci-dessous une parmi ses approches les plus utilisées, l'indexation par des mots composés.

I-4-3 L'indexation par des mots composés

L'indexation par des mots composés est une technique qui permet l'utilisation des mots composés comme unités d'indexation. Ceci a pour objectif une représentation plus précise du contenu sémantique des documents et des requêtes **[29].**

L'idée d'utiliser les mots composés comme unités d'indexation est que ces derniers sont moins ambigus et plus précis que les mots simples. Par exemples : le terme «java» est ambigu, par contre les mots composés « *ile de java* » et « *langage java* » sont non ambigus ; le terme « *voiture électrique* » est plus spécifique que l'un des deux termes « *voiture* » et « *électrique* ».

L'intuition est claire, les mots composés aident à construire des unités d'indexation non ambiguës et plus précises et peuvent par conséquent améliorer la précision de la RI. Cinq paramètres sont généralement à considérer dans l'exploitation des mots composés comme unités d'indexation **[29].**

A) *La directionnalité* :

C'est-à-dire l'ordre des termes. Dans certains cas la préservation de l'ordre est importante pour préserver le sens de l'unité d'indexation. Par exemple, « Recherche d'information », dans d'autre cas l'ordre n'est pas important, « Recherche et développement ». Peu de travaux existent en RI où sont utilisés les mots composés directionnels, la plupart des travaux exploitant les mots composés sont basés sur la non directionnalité de ces derniers**. [29]**

B) *La distance* :

La distance entre les termes formant le mot composé (l'adjacence ou la non-adjacence des termes) ; l'intensité de liens entre termes opérationnalisée à travers la distance reflète la proximité sémantique entre termes. La capture de cette proximité est importante pour la recherche d'information **[29].**

Les études effectuées en RI sur l'extraction des mots composés supposent que la cooccurrence des mots dans les éléments fortement structurés (c.-à-d., une phrase) est plus significative que dans les éléments moins structurés (c.-à-d., des paragraphes ou des sections).

Ainsi, la recherche sur l'extraction des mots composés a été dominée par l'analyse de phrase L'analyse empirique justifie de limiter l'extraction des mots composés aux combinaisons des termes apparaissant dans la même phrase **[29].** Martin *et al* **[30]** ont constaté que 98% de combinaisons syntaxiques associent les termes qui sont dans la même phrase et sont séparés par cinq mots au plus.

Fagan [**31]** a constaté que la restriction de l'extraction des mots composés à une fenêtre de distance de cinq termes est presque aussi efficace que des mots composés extraits dans une phrase sans une telle restriction, soutenant ainsi les résultats de Martin *et al* [**30].**

C) *La taille des mots composés*

En principe la taille d'un mot composé peut être de n'importe quelle longueur (supérieure ou égale à 2). Dans la pratique les mots composés longs conduisent à des index très spécifiques qui sont généralement moins utiles pour la RI **[29].**

D) *La pondération des mots composés* :

Les différents schémas de pondération proposés pour l'attribution d'un poids à un mot simple dans un document, prennent généralement en considération trois facteurs : le facteur de pondération local (*tf*), qui mesure.

L'importance du terme dans le document ; un facteur de pondération globale, mesurant la représentativité globale du terme dans la collection (***idf***) et un facteur de normalisation qui prend en compte la longueur du document **[29]**.

Cependant, pour les mots composés, il n'y pas de schéma de pondération bien accepté. En général, trois approches sont proposées pour la pondération des mots composés :

L'utilisation de la fréquence (***tf)*** du mot composé dans le document **[32]**; en se basant sur le fait que la fréquence d'un terme est corrélée avec son importance **[33]**.

L'adaptation de schéma de pondération (***tf*** X ***idf***) appliqué pour les mots simples. Comme c'est le cas dans **[34]**.

L'utilisation des mesures d'association, telle que l'information mutuelle **[35]**.

I-4-4 Repérage des mots composés

Trois approches principales existent dans la littérature pour le repérage et l'extraction des mots composés.

***Approches linguistiques* :** Ces approches se basent sur une analyse syntaxique partielle ou l'utilisation de patrons (templates) syntaxiques pour détecter les mots composés . Le plus souvent, un ensemble de patrons syntaxiques comme (NOM NOM) ou (NOM PREP NOM) est utilisé pour l'identification. Malgré les nombreuses études consacrées à ce problème, il n'existe pas encore, à notre connaissance, une méthode effective qui permette de distinguer les termes des non termes d'un point de vue syntaxique **[29]**. Des exemples d'outils issus de ces approches sont TreeTagger et AZNOUN PHRASER de l'université de l'Arizona.

Cependant, ces approches souffrent d'un inconvénient majeur puisque elles sont basées sur des règles, et ces règles sont dépendantes de la langue.

Approches statistiques : Les approches se basent sur la cooccurrence des termes dans le corpus pour extraire les mots composés et cela en partant de l'hypothèse que des termes (souvent réduits à deux ou trois mots) qui apparaissent ensemble dans le texte sont susceptibles de représenter un concept. **[22]**

Les mots composés sont extraits ici soit en se basant sur leurs fréquences observées dans le corpus soit par l'utilisation des mesures d'association qui déterminent le degré d'association entre les mots composants. **[29]**

Les mesures d'association : Les mesures d'association permettent de calculer « un score d'association » pour chaque paire de termes candidat dans le corpus ; ce score indique le potentiel de ce candidat d'être reconnu comme un mot composé. **[29]**

Plusieurs mesures d'association ont été proposées dans la littérature, telles que l'information mutuelle et le coefficient de Dice **[36]**. Toutes ces métriques adoptent le postulat suivant : « les mots composés sont ceux dont les composants apparaissent ensembles plus souvent que par hasard », cela est obtenu en comparant la fréquence observée dans le corpus et la fréquence attendue (qui se base sur l'hypothèse d'indépendance des termes). **[29]**

Les approches statistiques ont un avantage considérable puisqu'elles ne nécessitent aucune autre information ou ressource pour l'extraction des mots composés. Elles exploitent seulement les informations apparaissant dans le corpus, d'où leurs flexibilité et portabilité (i.e. : elles ne dépendent ni de la langue du corpus ni du domaine traité par le corpus). **[29]**

Approches mixtes : Ces approches se basent sur les régularités statistiques et les patrons syntaxiques pour l'extraction des mots composés. Fagan **[37]** a comparé l'apport pour la RI

des mots composés extraits statistiquement et des mots composés extraits linguistiquement, en utilisant l'analyse syntaxique, la troncature et la normalisation.

L'évaluation a montré que les mots composés extraits linguistiquement ont donné des résultats semblables ou plus faibles que les résultats obtenus avec les mots composés extraits statistiquement. Les gains de performance constatés en utilisant les mots composés extraits statistiquement dans son expérience étaient de l'ordre de 17% à 39%. L'exploitation des mots composés dans le contexte du modèle de langue est présentée en détails dans le chapitre trois.

I-5 Evaluation des SRI

L'évaluation des SRI est abordée selon deux angles différents. L'un est dit « paradigme système », qui vise à évaluer les performances du système essentiellement en termes de qualité des documents retournés par le système, c'est-à-dire leur pertinence vis-à-vis des besoins en information des utilisateurs. L'autre est dit « paradigme usager », qui est centré sur la satisfaction de l'utilisateur, et non sur les performances intrinsèques du système, en modélisant le comportement des utilisateurs en situation de recherche

I-5-1 Mesures d'évaluation de SRI

Dès l'apparition des premiers SRI, la pratique d'évaluation desdits systèmes est apparue; les premières évaluations datent de 1953**[38]**.

L'évaluation des SRI est abordée selon deux angles différents. L'un est dit « paradigme système», qui vise à évaluer les performances du système essentiellement en termes de qualité des documents retournés par le système, c'est-à-dire leur pertinence vis-à-vis des besoins en information des utilisateurs. L'autre est dit « paradigme usager », qui est centré sur la satisfaction de l'utilisateur, et non sur les performances intrinsèques du système, en modélisant le comportement des utilisateurs en situation de recherche.

1-La précision :

Mesure la proportion de documents pertinents (Dp) relativement à l'ensemble des documents restitués par le système(Dr). Elle est exprimée par :

$$\text{precision} = \frac{|\text{Dr}|}{|\text{Dp}|} \qquad \textbf{(I-31)}$$

2- Le rappel :

Mesure la proportion de documents pertinents restitués par le système relativement (Dr) à l'ensemble des documents pertinents P contenus dans la base documentaire. Il est exprimé par:

$$\text{rappel} = \frac{|Dr|}{|P|} \quad \textbf{(I-32)}$$

D'autres mesures d'évaluation d'un SRI existent. Ainsi, des mesures complémentaires au rappel et à la précision :

Le bruit **:** la mesure d'évaluation bruit est une notion complémentaire à la précision, elle est définie par :

$$B = 1 - \frac{|Dr|}{|Dp|} \quad \textbf{(I-33)}$$

Le silence **:** la mesure d'évaluation silence est une notion complémentaire au rappel, elle est définie par :

$$S = 1 - \frac{|Dr|}{|P|} \quad \textbf{(I-34)}$$

L'erreur de système **:** Il définit le pourcentage de documents non pertinents qui ont été retrouvés elle est définie par :

$$a = 1 - \frac{|Dr|}{|P|} \quad \textbf{(I-35)}$$

Elimination **:** Elle définit le pourcentage de documents non pertinents non retrouvés. Elle est définie par :

$$E = 1 - \frac{|Dr|}{|P|} \quad \textbf{(I-36)}$$

3- La précision moyenne non interpolée (MAP)

Une des mesures très largement employée est la moyenne des précisions aux rangs où se trouvent les documents pertinents par rapport à une requête (abrégée en AP pour average precision). Plus formellement, soit D = $\{d_1,\ldots,d_n\}$ une liste de documents renvoyés pour une requête Q donnée, la précision moyenne de cette liste se définit comme :

$$AP = \frac{1}{|R|}\sum_{k=1}^{n} rel(d_k) \quad \textbf{(I-37)}$$

Où

|R| représente le nombre total de documents jugés pertinents, n est le nombre de documents renvoyés par le système.

$\boldsymbol{rel(d_k)}$ est un indicateur de la pertinence du document au rang k ; il est égal à 1 si $\boldsymbol{d_k}$ est pertinent, à 0 sinon.

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

$$\mathrm{MAP} = \frac{1}{\mathrm{M}} \sum_{\mathrm{k}=1}^{\mathrm{M}} \mathrm{AP}_{\mathrm{qj}} \qquad \textbf{(I-38)}$$

Où

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

I-6 Conclusion

Dans ce chapitre nous avons présenté les principaux concepts de la RI, à travers l'architecture commune à tous les SRI, permettant l'appariement entre les requêtes formulées par des utilisateurs et les documents de la collection. Nous avons également présenté les différents modèles et stratégies utilisés lors de la mise en œuvre de ces concepts.

Nous présentons dans le prochain chapitre les techniques d'expansion de requêtes automatique utilisée pour améliorer les résultats de la recherche.

Chapitre II
Expansion de Requêtes

II-1 Introduction

L'un des problèmes majeurs de la recherche d'information est la formulation des requêtes. Blair et Maron [BM85] ont montré que la faible performance des systèmes de recherche d'information est due généralement à l'incapacité des utilisateurs de formuler les requêtes adéquates.

L'utilisateur ne sait pas choisir les bons termes qui expriment le mieux ses besoins d'information **[39], [40],** En introduisant la reformulation de requête, la RI est alors envisagée comme une suite de formulations et de reformulations de requêtes jusqu'à la satisfaction du besoin d'information de l'utilisateur, la requête initiale permettant rarement d'aboutir à un résultat qui satisfait ce dernier. Il s'agit en particulier d'ajouter des termes à la requête initiale de l'utilisateur et on parle alors d'expansion de la requête de l'utilisateur **[41] [42].**

Dans ce chapitre en présentant la reformulation de la requête automatique plus claire expansion de requête automatique (les fonctionnalités et les différentes approches).

II-2 Définition d'expansion de requête

Plusieurs définitions d'expansion de requêtes « Query Expansion (QE) » ont été données dans la littérature, nous en présentons dans cette section quelques unes.

Salton et McGill **[43]** définissent l'expansion de requêtes dans les systèmes de recherche d'information (SRI) comme un processus qui vise à rendre les résultats plus clairs et précis en permettant à l'utilisateur de modifier sa requête pour améliorer la pertinence de ses résultats.

Selon Abberley et al **[44],** l'expansion de requêtes dans leur système permet de reformuler les requêtes et améliorer le processus de recherche d'information.

Efthimiadis **[45]** qui a également proposé une classification des méthodes d'expansion de requêtes, donne la définition suivante : " l'enrichissement (expansion) de requêtes ou l'enrichissement de termes est un processus qui vise à compléter la requête en proposant des termes supplémentaires, et est considéré comme une amélioration de la recherche d'information ". Il donne également les définitions suivantes :

- L'expansion de requêtes est une approche qui peut être appliquée quelle que soit la recherche ou les méthodes utilisées.
- La requête initiale telle qu'elle est saisie par l'utilisateur peut être une représentation inadéquate ou incomplète des besoins de l'utilisateur, soit de lui-même ou de la représentation des idées dans les documents, base de données, etc.
- L'expansion de requêtes peut avoir lieu à la formulation de requête initiale, _a la phase de reformulation de requêtes, ou bien les deux.
- Le concept plus général de modification de requête peut impliquer la suppression des termes de la requête. Dans notre travail, nous nous intéressons à l'ajout de nouveaux termes.

A l'ère du Web sémantique, Vechtomova et Wang **[46]** présentent l'expansion de termes des requêtes comme une amélioration de la formulation de la requête initiale dans la recherche de documents. Ces termes sont généralement choisis parmi les documents entiers, des paragraphes ou des parties du document où apparaissent les termes de la requête. Ils ajoutent que la relation sémantique entre les termes diminue en fonction de la distance qui les sépare dans le texte.

II-3 Les méthodes d'expansion de requêtes

Quand on parle de requêtes, selon Efthimis **[45]** la simplicité de la recherche peut être réduite à deux étapes :

1-Formulation de la requête initiale. L'utilisateur construit la première stratégie de recherche et la soumet au système.

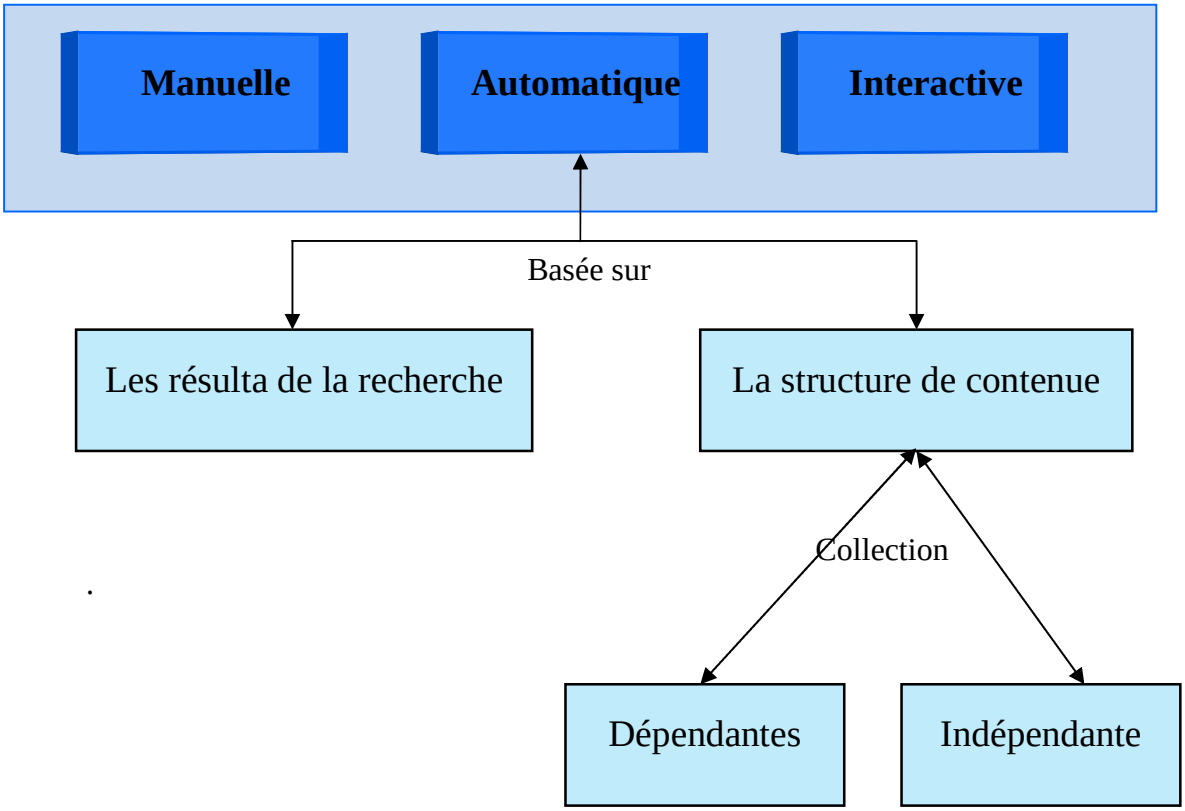
2-Reformulation de la requête. Après avoir eu quelques résultats de sa recherche, l'utilisateur améliore les résultats en modifiant sa recherche (i) manuellement, (ii) semi-automatiquement, ou (iii) automatiquement.

L'expansion de la requête est un cas particulier de la reformulation de requêtes, et est basé sur des méthodes proposées dans la classification donnée par **[45].** La figure **[II-1]** illustre cette classification et est réalisée en fonction de trois différents critères :

1- L'interaction de l'utilisateur dans le processus pour améliorer la sélection des termes supplémentaires.

2- Les types de ressources utilisées pour trouver les termes d'une requête supplémentaire.

3- Les méthodes et algorithmes utilisées pour sélectionner les termes à ajouter



.

Figure II-1Classification selon Efthimiadis [45]

Nous pouvons classer les différentes méthodes d'enrichissement comme suit : manuelle, automatique (médiation de l'utilisateur) ou interactive (assistée par l'utilisateur).

- *MQE, Manual Query Expansion*. L'expansion manuel de requêtes est basé sur les modèles de recherche booléenne et les modes d'interaction entre l'utilisateur et le système de recherche.

- *AQE, Automatic query Expansion*. L'expansion automatique de requêtes consiste à la proposition automatique de nouveaux termes par le système. Cette étape est transparente pour l'utilisateur dans le processus de recherche d'information.

- *IQE, Interactive Query Expansion*. L'expansion de requêtes interactives consiste à l'interaction entre le système et l'utilisateur pour le choix des termes de la requête enrichie. D'une part, le système propose les termes et les présente à l'utilisateur et d'autre part, les utilisateurs sélectionnent des termes en fonction de leurs préférences.

Ces différentes méthodes sont basées sur :

1- Les résultats de la recherche (Based on Search Results, BSR) : en utilisant les documents issus de la requête de l'utilisateur, en prenant les plus pertinents afin d'extraire de nouveaux termes qui peuvent servir à enrichir la requête initiale.

2- La structure du contenu (Based on Knowledge Structures, BKS) : qui est indépendante du processus de recherche. Elle peut avoir deux méthodes différentes :

- collection dépendante (collection dependent, CD), être basée sur un corpus
- collection indépendante (collection independent, CI), être indépendante du corpus.

II-4 Fonctionnement de l'expansion de requête

Comme le montre la figure II-2, le processus de expansion est divisé en quatre grandes étapes: prétraitement de la collection, génération et classement les technique d'expansion applicable, la sélection les fonctionnalités et reformulation de requêtes. **[47]**

II-4-1 Le prétraitement de la collection

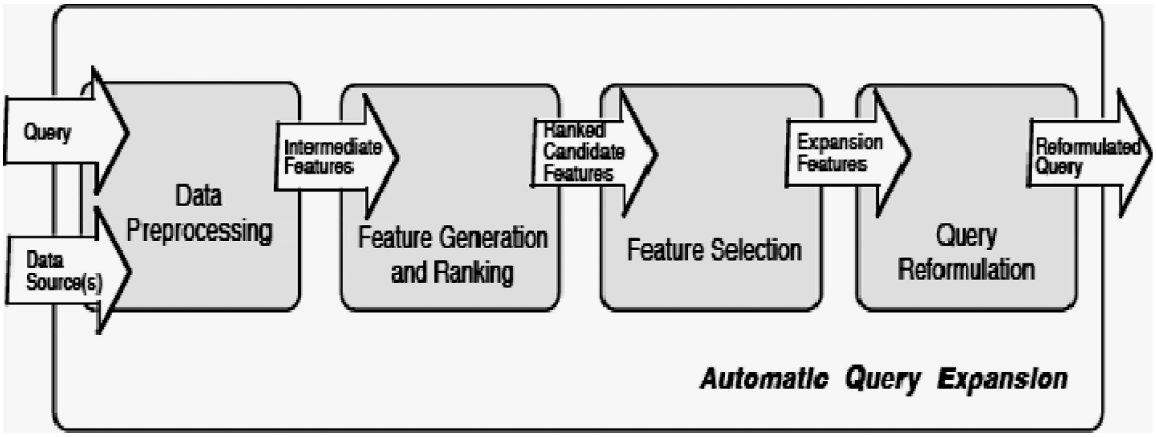


Figure II-2 le processus d'expansion de requête

La première étape dans expansion de la requête est de prendre la source de données que la requête de l'utilisateur est dirigé contre et le reformater pour un traitement ultérieur plus efficace. Les étapes suivantes sont généralement effectuées à la source de données **[47]**:

1. l'extraction de texte à partir de documents comme HTML, PDF, MS Word, etc (si l'collection est faite de ces documents),
2. tokenization (c'est à dire, l'extraction de mots individuels, en ignorant la ponctuation et convertir les termes en minuscules).

3. mot d'arrêt retrait (par exemple en enlevant des mots communs tels que les articles et les prépositions).

4. lemmatisation les termes extraire (un radical de mot).

5. Attribuez une valeur d'importance pondérée de chaque terme dans l'ensemble de mot résultant.

En conséquence, chaque document est représenté comme un ensemble de termes pondérés, avec un fichier d'index inversé complémentaire qui fait correspondre le plan de documents au moment de la requête.

II-4-2 Génération et classement des techniques d'expansion

Au cours de cette étape, expansion de requête automatique prend requête initiale de l'utilisateur et la collection transformé et génère un ensemble de conditions possibles à ajouter à la requête d'origine sur la base des relations entre la requête et la collection. La requête d'origine peut être reformatée tout comme la source de données était à l'étape précédente pour faciliter la génération de la fonction. Les fonctionnalités de candidats générés dans cette étape seront classées selon la fonction de classement terme du système. **[47].**

II-4-3 Sélection de fonctionnalité d'expansion

Une fois les fonctionnalités potentielles ont été générées et classés, les fonctionnalités avec les plus hauts rangs sont sélectionnées. Dans cette étape, les caractéristiques de candidats ne sont pas évaluées plus loin et sont simplement sélectionnés en fonction de rang. Seul un nombre limité de fonctionnalités d'extension sont choisis pour permettre un traitement plus rapide des requêtes et de la recherche a montré que l'utilisation de toutes les fonctionnalités de candidats n'est pas nécessairement mieux que d'utiliser seulement quelques-uns. La recherche suggère aussi qu'il est typique pour sélectionner entre dix et trente caractéristiques d'expansion. On pourrait aussi mettre en œuvre cette stratégie afin que seuls les termes dans une gamme de rang spécifique soient sélectionnés. **[47].**

II-4-4 reformulation de requêtes

La dernière étape consiste à modifier la requête initiale en ajoutant les fonctionnalités de candidats sélectionnés à la requête d'origine et effectuer la recherche à la requête reformulée.

II-5 Classification des approches de l'expansion de requêtes

II-5-1Analyse locale linguistique

L'expansion de requêtes, qui consiste à compléter les termes originaux de la requête en leur associant des termes sémantiquement proches, est une des solutions envisagées pour permettre de repérer plus de documents pertinents. Cette procédure a été automatisée de plusieurs manières. Ils sont généralement basés sur les *dictionnaires*, *thésaurus*, ou *d'autres sources* de représentation des connaissances similaires, telles que *WordNet*. Que les caractéristiques de dilatation sont généralement générées indépendamment de la requête complète et du contenu de la base de données consultée, ils sont généralement plus sensibles à mot ambiguïté de sens.

Certains types de ressources sont populaires, nous reprenons leur définition issue du grand dictionnaire dans le domaine des technologies de l'information :

Vocabulaire contrôlé : « Dans un domaine préalablement défini (d'ordre scientifique, technique, professionnel ou autre, et en général pour une langue donnée), le choix de termes sélectionnés, classés et indexés en vue de faciliter l'indexation, le stockage et la recherche des publications traitant des concepts apparentés à ces termes».

Taxonomie : « Construction d'un plan de classification de cpncepts utilisant des classes disjointes de concepts agrégés» .

Ontologies : « Ensemble d'informations dans lequel sont définis les concepts utilisés dans un langage donné et qui décrit les relations logiques qu'ils entretiennent entre eux ».

Thésaurus : « Vocabulaire contrôlé et dynamique de termes ayant entre eux des relations sémantiques et génériques, et qui s'applique à un domaine particulier de la connaissance ». Ainsi, il permet d'améliorer le système au niveau de l'indexation et de l'interrogation, en précisant le contexte de la recherche, ce qui peut lever un certain degré d'ambigüité 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, désigne les termes ayant un sens plus large. Elle est parfois notée BT pour « broader term ».

- ❖ la spécialisation ou hyponyme, désigne les termes ayant un sens plus spécifique, et est donc la relation symétrique de la précédente. C'est la relation is a. Elle est notée NT pour « narrower term ».
- ❖ la synonymie, réelle ou approchée, désigne les termes ayant un sens équivalent ou proche et est notée RT pour « related term ».
- ❖ la composition, ou méronymie, représente la composition de concepts, comme les parties d'un objet.

Ces relations seront détaillées dans ce qui suit :

On peut citer WordNet comme exemple de thésaurus hiérarchique. C'est une base de données lexicale pour l'anglais, basée sur des principes linguistiques. Dans cette hiérarchie, les éléments sont en fait des cliques de termes synonymes appelés synset. WordNet est un réseau sémantique organisé autour de la notion de synset. Un synset regroupe des termes (simples ou composés) ayant un même sens dans un contexte donné. Les synsets sont liés par des relations telles que spécifique-générique ou hyponyme-hyperonyme et la relation de composition meronymie-holonymie**.**

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

2. **L'hyperonymie**

   C'est le terme générique utilisé pour désigner une classe englobant des instances de classes plus spécifiques. Elle est parfois notée BT (Broader Term), Y est un hyperonyme de X si X est un type de (kind of) Y.

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

3. **L'hyponymie**

C'est le terme spécifique utilisé pour désigner un membre d'une classe (Relation inverse de Hyperonymie). Elle est notée NT (Narrower Term). X est un hyponyme de Y si X est un type de (kind of) Y.

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

4. **L'holonymie**

C'est le nom de la classe globale dont les noms meronymes font partie. Y est un holonyme de X si X est une partie de (is a part of) Y.

5. **La méronymie**

Représente la composition de concepts, comme les parties d'un objet. Le nom d'une partie constituante (part of), substance de (substance of) ou membre (member of) d'une autre classe (relation inverse de l'holonymie). X est un méronyme de Y si X est une partie de Y.

II-5-1-1 Thesaurus basé sur linguistique :

L'idée de base est de distinguer les polysémies par définition de contextes d'utilisation des termes dans la collection. A chaque terme est ainsi associé plusieurs vecteurs contexte dépendants de leur usage dans les documents.

Dans **[48],** les auteurs définissent le contexte d'un terme t_l à une position voisine i , VCi=(Wi1 , …, Wi,200) comme formé des 200 termes à plus grande valeur de cooccurrence avec le terme t à la position i,

Où :

$$W_{lk} = \log\left(\frac{N * df_{lk}}{tf_l * tf_k} + 1\right) \qquad \text{(II-1)}$$

Avec :

df_{1k} : Fréquence de cooccurrence de contexte du terme t_1 avec le terme f_k

tf_1 : Nombre total d'occurrences du terme t_1 dans la collection

tf_k: Nombre total d'occurrences du terme $f_k$dans la collection

II-5-2Analyse locale

Les techniques basées sur l'analyse locale permettent d'identifier les relations entre termes afin d'enrichir les requêtes par l'analyse des documents retrouvés les mieux classés **[49].** Dans **[50],** les auteurs proposent une technique qui suppose que les premiers documents retrouvés sont pertinents, ensuite la requête est enrichie suivant la méthode standard de réinjection de la pertinence **[51].** Une méthode similaire est utilisée où les premiers documents retrouvés sont utilisés pour ré-estimer les probabilités des termes.

Nous décrivons dans les sous sections suivantes trois méthodes d'expansion de requête par analyse locale : la méthode d'expansion par classification locale proposée par Attar et Frankel **[52],** la méthode d'expansion par analyse du contexte local et enfin la méthode d'expansion par réinjection locale (local feedback) **[49].**

II-5-2-1Classification locale

La classification découpe l'espace des documents en sous-espaces homogènes appelés classes Celles-ci sont constituées à partir de critères discriminatoires restreignant l'espace de recherche à un échantillon plus pertinent; les documents d'une même classe sont caractérisés par la même valeur du critère **[49].**

La méthode d'expansion de requête par classification locale proposée par Attar et Frankel **[52]** est la première méthode à utiliser les résultats de la recherche pour enrichir la requête de manière automatique Elle consiste à étendre la requête initiale à partir d'une classe de termes extraite des résultats de la recherche de la requête initiale. Le processus d'expansion par classification locale peut se résumer comme suit :

1- trouver un ensemble de documents en effectuant une recherche avec la requête initiale,

2- classer les termes des documents retrouvés (i.e. plusieurs formules pour calculer les distances entre termes ont été proposées par les auteurs **[52])**,

3- étendre la requête initiale en utilisant les classes de documents, pour cela chaque terme ti de la requête initiale, les m termes les plus similaires et de la même classe que ti sont considérés.

Les résultats expérimentaux obtenus par Attar et Fraenkel ont montré une amélioration des résultats. Cependant les collections de documents utilisés sont très petites et ne permettent pas de tirer des conclusions définitives.

II-5-2-2 Analyse du contexte local

Dans le cas de cette stratégie de recherche plus connue sous l'expression anglaise « ad hoc feedback », les informations utilisées pour la reformulation de requête dépendent en grande partie de la recherche en cours : documents retrouvés, termes et poids associés.

A l'origine, les travaux relatifs à l'utilisation de cette stratégie consistent essentiellement en l'application de techniques de classification de termes issus des *n* tops documents retrouvés **[52].** Actuellement, de nouvelles techniques sont mises en œuvre en vue d'analyser le contexte local de la recherche et de l'exploiter pour l'expansion de requête.

L'approche proposée par Xu & Croft **[53]** combine les atouts de l'analyse globale et analyse locale en procédant comme suit :

1- sélectionner les n premiers documents retrouvés par la requête initiale,

2- sélectionner un passage par document (un passage est une fenêtre de mots, sa longueur optimale fixée empiriquement est de 300 mots),

3- extraire à partir de ces passages les concepts d'expansion. Un concept peut être un nom ou un groupe nominal.

Un concept c est représenté par un ensemble de tuples {< t1, a1 >,< t1, a1 >, ..}, où ti est un terme en cooccurrence avec le concept c, et ai est le nombre de cooccurrences entre ti et c. L'extraction des concepts est faite via l'utilisation de l'étiqueteur morpho-syntaxique Jtag **[54].**

4- calculer la similarité f(c,Q) entre la requête initiale Q et chacun des concepts extraits c,

5- ordonner les concepts selon leur similarité f(c,Q) avec la requête initiale q,

6- sélectionner les k concepts les plus similaires à la requête pour l'expansion de la requête initiale (la valeur optimale de k fixée empiriquement est 30)

$$f(c,Q) = \prod_{w_i \in Q} \left(\delta + co\_degree(c,w_i)\right)^{idf(w_i)} \qquad \textbf{(II-2)}$$

Où:

δ : est une constante qui permet d'éviter les valeurs nulles,

$$idf(w_i)\text{i} = \min\left(1.0, \frac{\log_{10}\left(\frac{\text{N}}{\text{N}w_i}\right)}{0.5}\right) \quad \textbf{(II-3)}$$

Où :

N représente le nombre de passages, i.e. nombre de documents retrouvés,

$\text{N}w_i$ représente le nombre de passages contenant le terme $\mathbf{w_i}$,

codegree(c, $\mathbf{w_i}$) :est la fonction de calcul de similarité, i.e. degré de cooccurrence, entre le concept c est le terme $\mathbf{w_i}$,

Avec :

$$co_{degree(c,w_i)} = \log_{10}(co(c,w_i)+1)\frac{idf(c)}{\log_{10}(n)} \quad \textbf{(II-4)}$$

Où :

idf(c) = min(1.0, log10(N/Nc)/0.5) et Nc représente le nombre de passages contenant le concept c

$$co(c,w_i) = \sum_{p\in Q} tf(c,p) * tf(w_i,p) \quad \textbf{(II-5)}$$

Où :

$tf(c,p)\ et\ tf(w_i,p)$ Représentent respectivement la fréquence du concept c et du terme w_i dans le passage p.

Notons que une fois les concepts choisis, la requête est reformulé de manière brute, c'est à dire que les concepts sont rajoutés à la requête initiale tel que saisie par l'utilisateur.

III-5-2-3 Réinjection de la pertinence locale

L'expansion de la requête par la réinjection locale (local feedback) est semblable aux méthodes d'expansion par réinjection de la pertinence utilisateur, la différence majeure est que les k premiers documents retrouvés sont supposés pertinents. Les résultats obtenus sont tr`es encourageants, cependant le problème majeur de cette méthode est que les résultats peuvent être sensiblement dégradés si parmi les k premiers documents retrouvés peu d'entre eux sont pertinents **[49].**

II-5-2-4 Réinjection de la pertinence dans le modèle Vectoriel

Dans le modèle vectoriel, la réinjection de la pertinence utilisateur se fait généralement par l'ajout au vecteur requête initial des poids des termes des documents jugés pertinents et la soustraction au vecteur requête initial des poids des termes des documents non pertinents. Rocchio **[51]** décrit une stratégie permettant de dériver itérativement le vecteur requête optimal à partir d'opérations sur les vecteurs documents pertinents et vecteurs documents non pertinents. La formule posée est la suivante :

$$Q_{i+1} = \propto Q_i + \frac{\beta}{P}\sum_{dp \in Dp} dp - \frac{\delta}{Np}\sum_{dnp \in Dnp} dnp \qquad \textbf{(II-6)}$$

Où :

Qi+1 : Requête construite à la i+1 ème itération de feedback

Qi : Requête construite à la i ème itération de feedback

Dp : Ensemble des documents jugés pertinents

Dnp : Ensemble des documents jugés non pertinents

P : Nombre de documents jugés pertinents

Np : Nombre de documents jugés non pertinents

a, b, d : Constantes

Salton et Buckley **[55]** ont comparé l'effet de formule de Rocchio, Ide-Regular et Ide-Dechi, sur différentes collections :

Ide-Regular $$Q_{i+1} = \propto Q_i + \beta \sum_{dp \in Dp} dp - \delta \sum_{dnp \in Dnp} dnp \qquad \textbf{(II-7)}$$

de-Dec-Hi $$Q_{i+1} = \propto Q_i + \beta \sum_{dp \in Dp} dp - \delta\, dnp \qquad \textbf{(II-8)}$$

Où :

dnp : premier document jugé non pertinent.

Les auteurs ont mené une série d'expérimentations pour évaluer la reformulation par injection de pertinence, en comparant l'impact de l'utilisation des trois formules sur les résultats de recherche d'information, effectuées dans les collections CRANFIELD, CISI et MED. Les résultats présentés montrent que la formule Ide-Dec-Hi donne les meilleurs résultats avec les paramètres a= 1, b=0.75, d=0.25.

Buckley & al **[50]** se sont intéressés à l'application de la technique de relevance feedback dans la base TREC. La nouvelle requête est obtenue selon la formule Ide_Regular avec les paramètres a=8, b=16, d=4.

Les résultats obtenus dans la base TREC2, pour la tâche de routing, montrent un accroissement de performance de 24% lors de l'expansion et repondération de requêtes.

II-5-2-5 Réinjection de la pertinence dans le modèle probabiliste

Sur la base du modèle probabiliste, Harman**[56],** Haines **[57]** et Robertson **[58]** ont développé des formules de pondération de requête en utilisant le jugement de l'utilisateur sur la pertinence des documents restitués par le système.

Robertson calcule la similitude initiale Document-requête selon la formule :

$$\mathrm{Sim}(Q_k, D_j) = \sum_{i=1}^{T} q_{ki} * d_{ji} * \log \frac{P_i(1-U_i)}{U_i(1-P_i)} + C \qquad \textbf{(II-9)}$$

Où :

Pi : Probabilité (dji = 1/ Dj est Pertinent).

Ui : Probabilité (dji = 1/ Dj est Non Pertinent).

C : Constante.

Avec :

Dji = 1 si ti occurre dans Dj, 0 sinon

Pinit = 0.5

Uinit = ni / N

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

$$VS(t_i) = \log \frac{P_i * r_i (1 - q_i)}{q_i * R(1 - P_i)} \qquad \textbf{(II-10)}$$

Où :

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

R : Nombre de documents jugés pertinents.

Les liens de dépendance conditionnelles sont repondérés et calculés comme suit :

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

Où :

NR : Nombre de documents non pertinents retrouvés.

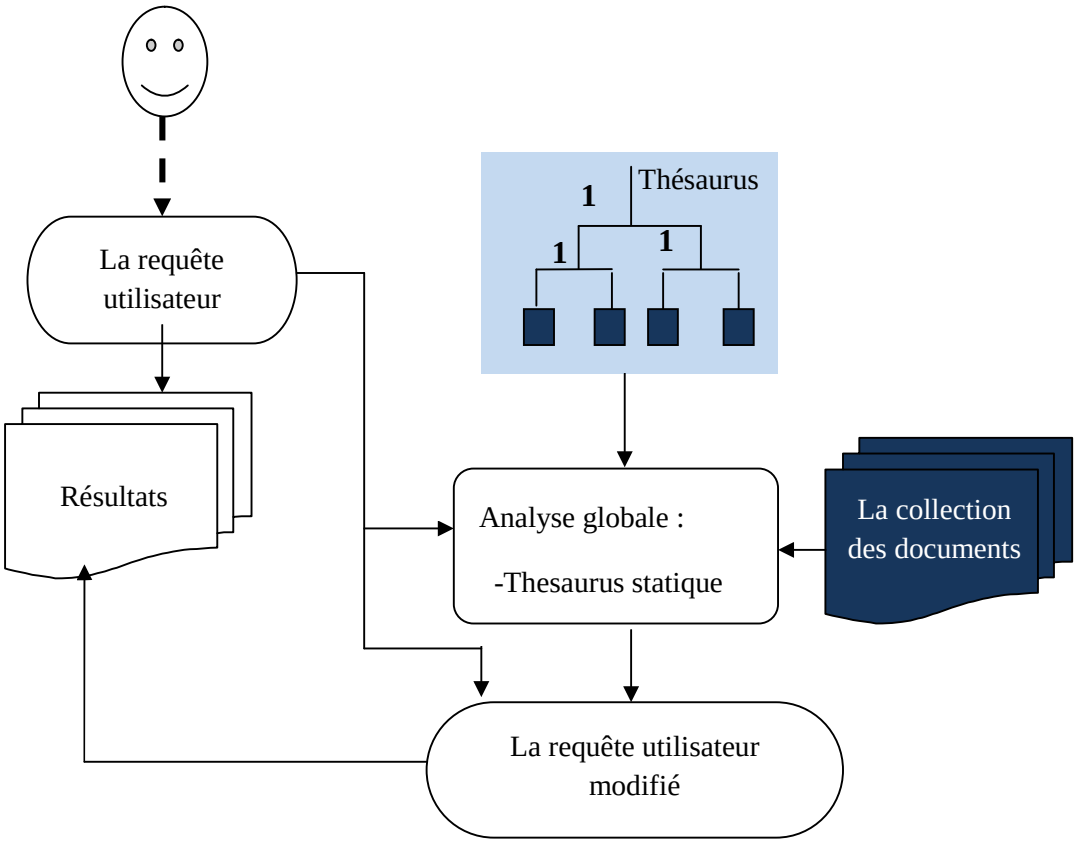
La fonction de similitude utilisée lors des itérations feedback devient alors la suivante :

$$Sim(Q_k, D_j) = \sum_{i=1}^{T} q_{ki} * \log\left(\left(\frac{r_i}{R - r_i}\right) + \frac{(n_i - r_i)}{(N - NR - n_i + r_i)}\right) \qquad \textbf{(II-11)}$$

Il en résulte ainsi une repondération de la requête avec expansion.

II-5-3 Analyse global :

Dans cette section, nous discutons d'un modèle d'expansion de requête basée sur un thesaurus de similarité globale construite automatiquement. La figureII-3 illustre les l'analyse globale d'une requête.



. **Figure II-3** analyse globale d'expansion de la requête

II-5-3-1 Thésaurus de la similarité

Le thésaurus de similarité est basé sur terme de relations à long terme plutôt que sur une matrice de cooccurrence. La distinction est clairement dans la discussion ci-dessous. En outre, une attention particulière est portée à la sélection des termes d'expansion et à la repondération de ces termes. Contrairement aux approches d'analyse locale précédente, les conditions de développement sont choisies en fonction de leur similarité à la requête entière plutôt que de leurs similitudes avec les termes de requête individuels **[59].**

Un thésaurus de similarité est construit en utilisant terme de relations à long terme, qui sont calculés en considérant que les termes sont des concepts dans un espace de concept. Dans ce concept, l'espace, chaque terme est indexé par les documents dans lesquels il apparaît. Ainsi, les termes assument le rôle original de documents alors que les documents sont interprétés comme des éléments d'indexation. Les définitions qui suivent établissent le cadre approprié **[21].**

Soit **t** est le nombre de termes dans la collection, **N** est le nombre de documents de la collection, et $f_{i,j}$ est la fréquence d'apparition du terme $\mathbf{k_i}$ dans le document $\mathbf{d_j}$. En outre, $\mathbf{t_j}$ est nombre de termes d'indexation distincts dans le document $\mathbf{d_j}$ et ITFj est la fréquence de terme inverse pour le document $\mathbf{d_j}$ défini par: **[60]**

$$ITF_j = log\left(\frac{t}{t_j}\right) \qquad \textbf{(II-19)}$$

Qui est analogue à la définition de la pondération de terme fréquence inverse tf.

Dans ce cadre, chaque terme $\mathbf{k_i}$ est associé un vecteur $\overrightarrow{\mathbf{k_i}}$ donnée par:

$$\overrightarrow{\mathbf{k_i}} = (w_{i,1}, w_{i,2}, w_{i,3}, \ldots, w_{i,N})$$

Lorsque, $\mathbf{w_{i,j}}$ est un poids associé à l'index de documents paire ($\mathbf{k_i}$, $\mathbf{d_j}$). Ici, cependant, les poids de document sont utilisés pour des vecteurs de terme de l'index, au lieu d'avoir des documents de l'indice des coefficients des vecteurs termes comme dans le modèle classique de vecteur indiqué dans la section. En outre, ces poids sont terminés sous une forme assez distincte de la manière suivante .

$$w_{i,j} = \frac{\left(0.5+0.5\frac{f_{i,j}}{\max_j(f_{i,j})}\right)ITF_j}{\sqrt{\sum_{l=1}^{N}\left(\left(0.5+0.5\frac{f_{i,l}}{\max_k(f_{i,k})}\right)*ITF_l\right)^2}} \qquad \textbf{(II-20)}$$

Où:

$\boldsymbol{max_k(f_{i,j})}$Représente la fréquence du terme ti dans le document d_k,

La relation entre deux termes $\mathbf{K_u}$ et $\mathbf{K_v}$ est calculée comme un facteur de corrélation $\mathbf{C_{u,v}}$ donnée par: **[61]**

$$C_{u,v} = \overrightarrow{k_u} . \overrightarrow{k_v} = \sum_{\forall d_j} w_{u,j} . w_{v,j} \qquad \textbf{(II-21)}$$

Compte tenu de thésaurus global de similitude, l'expansion de la requête se fait en trois étapes comme suit **[61]**.

1- La premier, représentent la requête dans le même espace de vecteur utilisé pour représenter les termes d'index.

2- Deuxième, sur la base de la similarité de thesaurus global, calculer une similitude **sim (q, k_j)** entre chaque terme de $\mathbf{k_v}$ en corrélation avec les termes de requête et l'ensemble de la requête **q.**

3- Troisième, expansé (élargir) la requête de par les top (haut) **r** classé termes selon **sim (q, k_v).**

Dans la premier le vecteur est représenté par :

$$\vec{q} = \sum_{k_i \in q} w_{i,q} . \vec{k_i} \qquad \textbf{(II-22)}$$

Où

$w_{i,q}$: est un poids lié aux paires **(K_i,q)** d'index de la requête. Ce poids est indiqué par l'équation 5 avec le document d remplacé par la requête q.

Pour la deuxième étape la similarité s **sim (q, k_j)** entre chaque terme $\mathbf{k_v}$ corrélé avec les termes de requête et la requête d'utilisateur q dans calculé par :

$$sim(q, k_v) = \vec{q} . \vec{k_v} = \sum_{k_i \in q} w_{i,q} . c_{i,v} \qquad \textbf{(II-23)}$$

Et pour la troisième étape, les r top termes sélectionnés par $\mathbf{sim(q, k_v)}$ sont ajoutés à la requête originale **q** à d la requête expansé $\mathbf{q_m}$. À chaque terme k d'expansion dans la requête q est assigné un poids w_{v,q_m} indiqué par **[62]** :

$$w_{i,q_m} = \frac{sim(q,k_v)}{\sum_{k_i \in q} w_{i,q}.} \qquad \textbf{(II-24)}$$

L'expansion de la requête q_m est utilisée pour chercher des nouveaux documents.

Les expérimentations réalisées dans des collections standards montrent l'intérêt de cette stratégie d'expansion. Cependant, cette dernière donne des résultats très dépendants des paramètres de l'algorithme de classification : nombre de classes, taille min d'une classe… Ces paramètres sont en outre très variables en fonction des collections interrogées **[05].**

II-5-3-2 Thésaurus statique

L'objectif recherché de toute méthode d'expansion de requêtes est de sélectionner des termes discriminants pour l'expansion de requêtes. Ainsi dans **[62]** les termes sélectionnés pour la

construction des classes de termes sont ceux dont la fréquence en document est faible, car ce type de termes sont discriminants **[63].** Cependant il est difficile de classer ce type de termes car ils apparaissent dans peu de documents.

Par conséquent, au lieu de classer les termes, les auteurs proposent de classer les documents et d'associer aux classes ainsi construites les termes dont la fréquence en documents est faible.

La classification des documents s'effectue de la manière suivante **[62]**:

1- chaque document de la collection est associé à une classe distincte, i.e. le nombre de classes correspond au nombre de documents de la collection,

2- calculer la similarité entre chaque paire de classes,

3- déterminer la paire de classes (**Cu, Cv)** dont la valeur de similarité est la plus grande,

4- fusionner les classes **Cu et Cv**,

5- si le nombre de classes est supérieur à 1 alors allez à l'étape 2,

6- retourner la structure hiérarchique des classes de documents.

La similarité entre deux classes de documents correspond au minimum des similarités entre toutes les paires de documents, i.e. deux documents n'appartenant pas à la même classe. Les documents sont représentés par des vecteurs de termes et la similarité entre deux documents est calculée par le cosinus entre les vecteurs représentant les documents.

La classification ainsi effectuée produit une structure hiérarchique des documents de la collection. La détermination des classes de termes s'effectue, selon trois paramètres :

- **TC** : seuil de similarité entre deux classes de documents, il permet de déterminer les classes de documents qui seront utilisés pour la génération des classes de termes, i.e. la paire de classes s **(Cu, Cv)** sera fusionnée si la similarité **sim(Cu, Cv)** entre **Cu** et **Cv** est supérieure ou égale à TC. *Si* ***sim(Cu, Cv)*** < ***TC*** *alors* Cu et Cv seront considérés comme des classes distinctes.
- **NDC** : nombre maximum de documents d'une classe. Il représente une condition supplémentaire pour la génération des classes de documents. Ainsi si le nombre de documents de la paire de classes **(Cu, Cv)** est supérieur à **NDC**
  alors Cu et Cv ne seront pas fusionnées même si la similarité **sim(Cu, Cv)** entre **Cu** et **Cv** est supérieure ou égale à TC.

- **MIDF** : seuil de la fréquence inverse en documents.

Une fois les classes de documents construites (via les paramètres **TC** et **NDC**), les documents de chaque classe sont utilisés comme source de termes. Ainsi à chaque classe de documents on associe une classe de termes, i.e. les termes contenus dans les documents de la classe de document.

Afin de sélectionner des termes discriminants seuls les termes dont la fréquence en document est faible seront considérés, i.e. les termes dont la fréquence inverse en documents est inférieure à un seuil donnée **MIDF**.

Une fois les classes de termes construites, i.e. les classes du thésaurus, celle-ci peuvent être utilisées pour l'expansion de requêtes. L'expansion se fait par l'ajout de tous les termes de la classe la plus similaire à la requête. Le poids des termes d'une classe **C** sont égaux et dont la valeur est donnée par la formule :

$$wt_c = \frac{\sum_{i=1}^{|C|} w_{i,C}}{|C|} \qquad \textbf{(II-25)}$$

Les expériences effectuées sur 4 collections documentaires ont montré une amélioration significative des performances.

II-6 Paramètres de performance

II-6-1 Nombre de termes ajoutés à la requête

L'ajout de termes à la requête accroît la performance du SRI dans le cas des deux stratégies décrites précédemment. Buckley & al **[63]** ont expérimenté la relevance feed-back dans l'environnement multi-fond documentaire TREC; ils ont montré que le taux de performance est d'avantage corrélé avec le nombre de termes ajoutés qu'avec le nombre de documents initialement retrouvés. Ils ont abouti à la mise au point de l'équation de variation :

$$RP(N) = A \log(Ns) + B \log(X) + C \qquad \textbf{(II-27)}$$

Où :

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

Ns: Nombre de documents restitués

X : Nombre de termes ajoutés à la requête

A, B, C : Constantes

II-6-2 Méthode de sélection des termes

L'expansion des requêtes consiste à étendre automatiquement la requête initiale afin d'améliorer la qualité des documents retrouvés. Nous citerons les principales méthodes expérimentées :

Salton et Buckley **[64]** 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.

Robertson **[65]** et Haines **[66]** 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

$$SV(i) = w(P_i - U_i)$$

Où :

$$w = \log \frac{P(1-U_i)}{U(1-P_i)}$$

Avec :

P_i : Probabilité (d_i = 1/ D est Pertinent)

U_i : Probabilité (d_i = 1/ D est Non Pertinent)

Harman **[68]** propose une liste des fonctions suivantes

$$SV(i) = \frac{RT_j * df_i}{N}$$

Où :

RT_j : Nombre total de documents retrouvés par la requête

df_i: Fréquence d'occurrence du terme t_i dans la collection

N : Nombre total de documents dans la collection

$$SV(i) = \frac{r_i}{R} - \frac{df_i}{N}$$

Où :

r_i : Nombre de documents pertinents contenant t_i

R : Nombre de documents pertinents

$$SV(i) = \log \frac{P_i(1-q_i)}{(1-P_i)}$$

Avec :

P_i : Probabilité que ti appartienne aux documents pertinents

q_is: Probabilité que ti appartienne aux documents non pertinents

Parmes les trois fonctions la troisième fonction est la meilleur.

Lundquist **[68]** ont expérimenté la fonction **pi*nidf** en utilisant la fonction de pondération des documents normalisée par la longueur **[69].** Les résultats montrent un accroissement de 31% des performances à l'ajout des 10 top termes et ce, dans des collections moyennes.

II-6-3 Longueur moyenne de requête

L'accroissement des performances est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite **[63].**

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.

II-7 Conclusion

Dans ce chapitre nous avons traité le fonctionnement d'expansion de requêtes, les principales approches d'expansion de requête ainsi que les paramètres de performances

Dans notre cas nous utilisons l'expansion de requêtes basée sur la technique réinjection de pertinence dans le cadre du modèle de langue ; ce point est traité en détail dans le chapitre suivan

Chapitre III : Description de notre approche et évaluions

Chapitre III : Évaluions et Expérimentation

III-1 Introduction

Un système de recherche d'information doit faire deux opération l'indexation de documents (et de la requête) et la recherche. Un des problèmes dans la RI est l'ambiguïté des termes l'objectif principale de notre approche est de dépasser ce problème. Nous proposons d'utiliser une nouvelle unité d'indexation « les mots composés. Un autre problème d'utilisation des termes simple est la disparité des termes pour résoudre ce problème divers approches sont proposées. En ce que nous concerne, nous intéressons à l'expansion de requêtes.

Le chapitre est organisé comme suit : (1) présentation de notre approche (le processus d'indexations et de recherche), (2) présentation de la recherche classique et de l'expansion de requêtes sous la plateforme Terrier, le dernier point concerne la présentation des résultats d'expérimentation obtenus.

III.2. Architecture générale de notre approche

La figure suivante illustre l'architecture de notre approche, plus précisément.

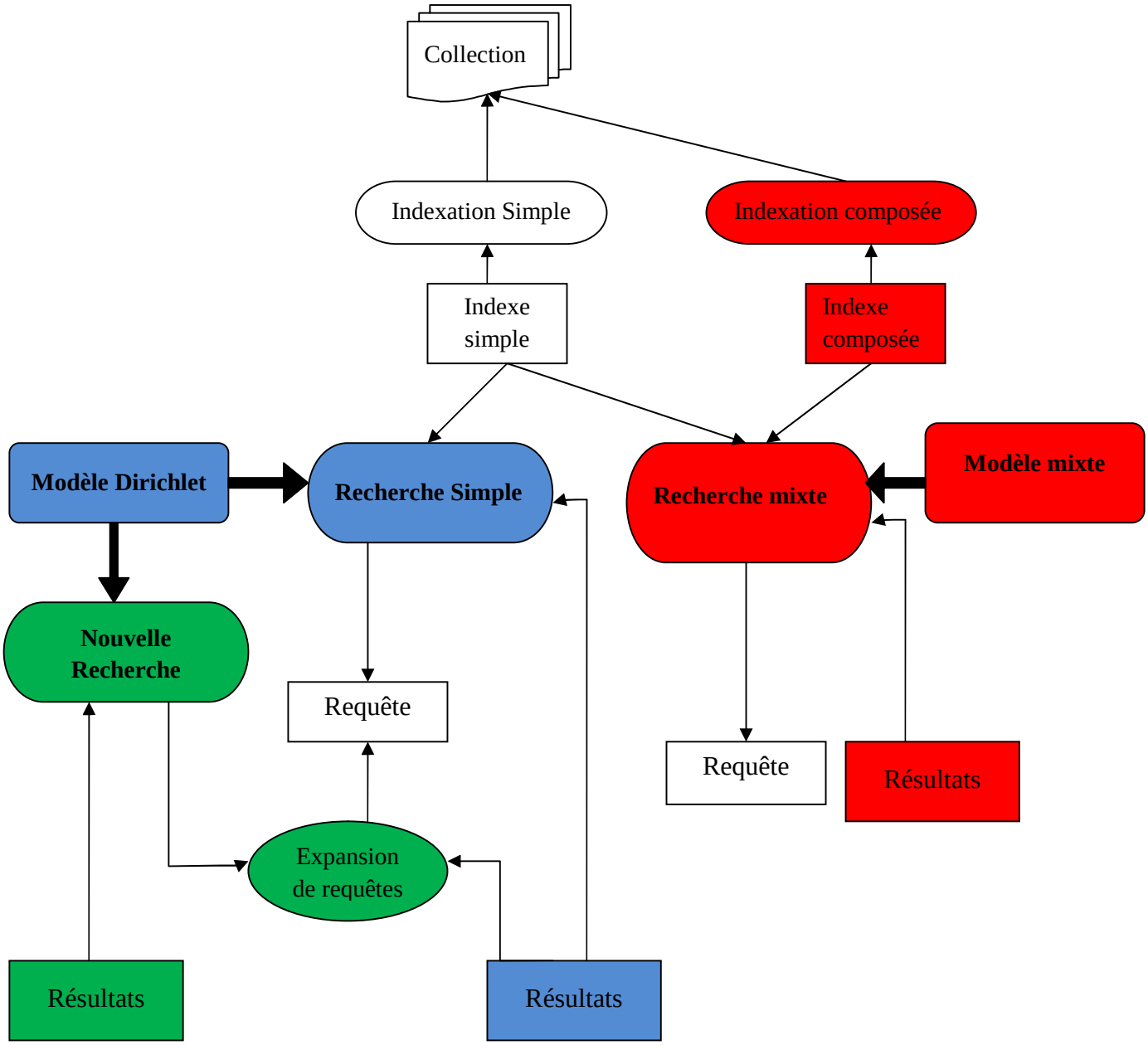


Figure III.1 : Architecture générale de notre approche

III-2 Présentation notre approche

Notre travail se base sur une approche qui tente réduire l'ambiguïté des termes. Dans le processus d'indexation l'unité d'indexe est un terme (mot non vide), cette approche d'indexation rencontre le problème d'ambiguïté (les phrases perdent leur sens).Dans le cadre de notre travail s'intéresse à ce problème c'est-à-dire comment éliminer ou réduire cet ambiguïté. Nous proposons l'approche d'indexation avec les mots composées, cette approche se base sur l'unité d'indexation, un mot composée défini comme tout terme adjacent non vide.

En plus de mot composé l'expansion de requête est une approche qu'y remédie le problème d'ambiguïté et la disparation des termes. Nous avons implémenté cette approche.

Dans le premier pont, nous allons présenter notre travail qui consiste en l'implémentation d'une nouvelle approche d'indexation et de recherche des documents sous la plateforme Terrier.

Dans le deuxième point, l'implémentation la recherche et l'expansion de requête simple sous la plateforme de terrie.

III-2-1 le processus d'indexation

A) Les prétraitements de la collection

En premier lieu, nous procédons au prétraitement de la collection. Nous parsons les documents, nous éliminons les mots vides et nous appliquons l'algorithme de Porter [14] à l'aide la plateforme de terrie. Les documents traités obtenus sont ensuite utilisés comme entrées pour l'outil Text-NSP pour l'extraction des mots composés.

B) L'extraction des mots composés

L'extraction des mots composés : pour l'extraction des mots composés nous avons utilisé l'outil Text-NSP. Le package Text-NSP est un outil permettant l'identification et la sélection de n-grammes ou séquence de mots dans une collection de texte. Dans le processus d'extraction des mots composés, nous tenons compte des paramètres suivants:

La directionalité entre mots simples : est très important pour garder le sens pour unité d'indexation. L'ordre des mots est important pour garder le sens de indexe, dans d'autres cas l'ordre n'est pas important, par exemple le terme «*énergie renouvelable*». dans notre cas la contrainte d'ordre est respectée lors de l'identification des mots composés.

La distance : la distance entre uni_gramme qui forme les bi_gramme (ou l'adjacence ou la non- adjacence des termes) : l'intensité de liens entre termes – opérationnalisée à travers la distance- reflète la proximité sémantique entre termes. La capture de cette proximité est importante pour la recherche d'information. Les études réalisées en RI sur l'extraction des mots composés suppose que la cooccurrence des mots dans les éléments fortement structurés (c.-à-d., une phrase) est plus significative que dans les éléments moins structurés (c.-à-d., des paragraphes ou des sections). Ainsi, la recherche sur l'extraction des mots composés a été dominée par l'analyse de phrases. Dans notre cas, nous avons adopté l'adjacence entre termes, un mot composé est reconnu si et seulement s'il est composé de mots simples adjacents.

La taille des mots composés : Touts N-gramme (N supérieure ou égale à 2). Forment les mots composés .Dans notre cas, nous s'sintérisons à la taille égale à deux ,un mot composé de deux mot simple. Pour former la liste des termes nous utilisons l'outil Text-NSP, ce

dernière est composé de deux processus principale pour former la liste des mots composées : (1) « *count.pl* » est une module qui permet de sélectionner

es bi-grammes avec une fréquence minimale précisé, une fréquence de terme supérieur un seuil donnée noté(seuil_freq)).une fois la liste des mots composées formré en sortie de « count.pl »,cette liste passe en entrée de deuxième module. (2) « *statistic.pl* », donne en sortie une liste des bi-gramme avec leur les différentes fréquences calculer avec des mesures statique de. Dans notre cas, nous avons utilisé la mesure de Point wise Mutual information (PMI). L'étude menée par Petrovic et al [69] a montré que la mesure PMI permet l'identification de mots composés pertinents pour la RI. Nous ne gardons dans la liste finale que les bi-grammes ayant un score supérieur à un seuil noté « *seuil_PMI* ». Cette liste est ensuite utilisée dans les étapes d'indexation et de recherche

III-2-2 Recherche simple

Dans notre approche la base des indexes produite par le processus d'indexation simple sous terrier, la recherche se base sur le modèle de Dirichlet. Ce dernier est une technique de lissage de modèle de langue, son principe de ne pas avoir une probabilité à nulle .La formule s'écrit comme suit :

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

$$= \frac{|d| P_{ML}(m_i|d) + \mu P_{ML}(m_i|C)}{|d|+\mu} \qquad \textbf{(III. 6)}$$

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

Avec :

$$P_{ML}(m_i|d) = \frac{tf(m_i,d)}{|d|} \qquad \textbf{(III. 7)}$$

Où :

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

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

μ : est un paramètre appelé pseudo fréquence.

III-2-3 Recherche mixte

En se basant sur la nouvelle structure de donnée Index produite par le processus d'indexation précédant, le modèle de recherche assigne un score pour chaque terme de la requête (indexé avec les composées) dans le document, avec le modèle mixte.

A)Le modèle mixte

En suivant la logique du modèle de langue et en considérant que le contenu d'un document comporte à la fois des mots simples et mots composés. Chacun produisant un type de terme.

Nous supposons donc que le modèle de document peut être estimé à l'aide de deux modèles : un modèle des mots simples (M_{D_t}) et un modèle des mots composés (M_{D_T}). Ainsi, étant donné une requête Q, exprimée par des mots simples et des mots composés le modèle d'appariement document-requête que nous proposons combine les deux modèles de la manière suivante :

$$P(Q|D) = \prod_{t_i \in Q} P(t_i|D) \times \prod_{T_j \in Q} P(T_j|D) \quad \textbf{(III-1)}$$

Avec :

$$P(t_i|D) = \lambda P\left(t_i \middle| M_{D_t}\right) + (1-\lambda) P\left(t_i \middle| M_{D_T}\right) \quad \textbf{(III-2)}$$

$$P\left(T_j \middle| D\right) = \alpha P\left(T_j \middle| M_{D_T}\right) + (1-\alpha) \prod_{t_k \in T_j} P\left(t_k \middle| M_{D_t}\right) \quad \textbf{(III-3)}$$

Où

$\lambda \; et \; \alpha \in [0,1]$ sont des paramètres de lissage,

$P\left(T_j \middle| M_{D_T}\right)$ et $P\left(t_i \middle| M_{D_t}\right)$ peuvent être évaluées en utilisant n'importe quel modèle de langue uni-gramme. Nous avons pour notre part opté pour le lissage de Dirichlet,

$$P_{Dir}\left(t_i \middle| M_{D_t}\right) = \frac{F(t_i, D_t) + \mu P(t_i|C_t)}{|D_t| + \mu} \quad \textbf{(III-4)}$$

Où

$F(t_i, D_t)$ est la fréquence du mot simple t_i dans le document D,

$P(t_i|C_t)$: est le modèle de langue de la collection (la fréquence globale du terme est utilisée),

$|D_t|$: est la longueur du document exprimée avec des mots simples,

μ est le paramètre de lissage.

De la même manière on a :

$$P_{Dir}(T_j|M_{D_T}) = \frac{F(T_j,D_T)+\mu P(T_j|C_T)}{|D_T|+\mu} \quad \textbf{(III-5)}$$

Où

$P(T_j|C_T)$: est la fréquence du mot composé T_j dans la collection des mots composéesC_T.

$F(T_j, D_T)$: est la fréquence du mot composé T_j dans le document D.

La probabilité$P(t_i|M_{D_T})$ donne Le passage d'un mot simple vers un document D est réalisé à travers tous les mots composés contenant le mot simple. Cette probabilité est exprimée comme suit :

$$P(t_i|M_{D_T}) = \sum_T \ P(t_i|T) \times P_{Dir}(T_j|M_{D_T}) \quad \textbf{(III-1)}$$

Avec :

$P(t_i|T)$: La probabilité de dominance de ti dans le mot composé T.

$$P(t\,|T) = \frac{imp(t)}{\sum_{t_i \in T}(imp(t_i))}$$

Où :

imp.(t) : la dominance de terme t

$$imp(t) = \frac{N}{df}$$

Avec :

N est le nombre de document dans la collection et df est le nombre de documents où le t apparaît.

III-2-3 Expansion de requête dans la recherche simple (EQ_TS)

Dans notre approche nous utilisons la plateforme terrier pour modèle d'expansion de requêtes. Terrier mise en œuvre un module d'expansion de requêtes automatiquement. Le modèle d'expansion de requêtes utilisée est modèle par défaut Bo1.

III-3-Evaluation

III-3-1 Collection de test utilisée

Pour arriver à une telle évaluation, on doit connaître d'abord les réponses idéales de l'utilisateur. Ainsi, l'évaluation d'un système se fait à l'aide d'un corpus de test.
Dans un corpus de test, il y a:

1- un ensemble de documents (La collection).

2- un ensemble de requêtes (les topic) .
3- la liste de documents pertinents pour chaque requête.

Nous avons évalué notre modèle en utilisant les collections TREC la *WT10g*. La collection WT10g (*TREC Web 2000-2001*) a servi de support à la tâche de recherche Web de TREC pour les années 2000 et 2001. Il s'agit d'un ensemble de pages Web récupérées en 2000 ne contenant que des documents en langue anglaise. Nous utilisons ici les requêtes des années 2000 et 2001 (50 pour chaque année). Ce sont des requêtes soumises par des utilisateurs réels, extraites à partir des historiques de requêtes du moteur de recherche excite.

Un document TREC est généralement présenté sous le format SGML. Il est identifié par un numéro et décrit par un auteur, une date de production et un contenu textuel. Une requête TREC est également identifiée par un numéro. Elle est décrite par un sujet générique, une description brève et une description étendue sur les caractéristiques des documents pertinents associés à la requête. La table suivante montre quelques statistiques sur les collections et requêtes utilisées

Collection	**#documents**	**Topics**
WT10g	*1, 692,096*	*451-550*

Les
figures suivantes illustrent un exemple d'un document et une requête respectivement de TREC

<DOC>
<DOCNO>*WTX001-B01-1*</DOCNO>
<DOCOLDNO>IA001-000000-B001-3</DOCOLDNO>
<DOCHDR>
http://www.ram.org:80/ramblings/movies/jimmy_hollywood.html 208.194.41.61 19970101152051 text/html 2080
HTTP/1.0 200 Document follows
MIME-Version: 1.0
Server: CERN/3.0
Date: Wednesday, 01-Jan-97 15:20:23 GMT
Content-Type: text/html
Content-Length: 1873
Last-Modified: Thursday, 23-Nov-95 03:11:57 GMT
</DOCHDR>
<title> Jimmy Hollywood movie review </title>
<h1> Jimmy Hollywood </h1>
<hr>
<p>
As one critic put it, "this is a satire whose aim is so unsure that its principal casuality is itself." That may well be true, but it does bring out one good point: throughout the entire movie, Jimmy Hollywood is pursued relentlessly by the cops even though he is doing good for
<p>
<hr>
Ram Samudrala || <var>me@ram.org</var>
<hr>
</DOC>

FigureIII-2 Exemple d'un document TREC.

<top>

<num> Number: 452

<title> do beavers live in salt water

<desc> Description:

Describe the normal habitat for beavers; note exceptions, if any.

<narr> Narrative:

Relevant documents describe the habitat range

as well as references to specific areas and bodies of water.

</top>

Figure III-3Exemple d'une requête TREC

III-3-2 Les outils utilisés

III-3-2 -1 La plateforme Terrier

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

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

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

Architecture de Terrier

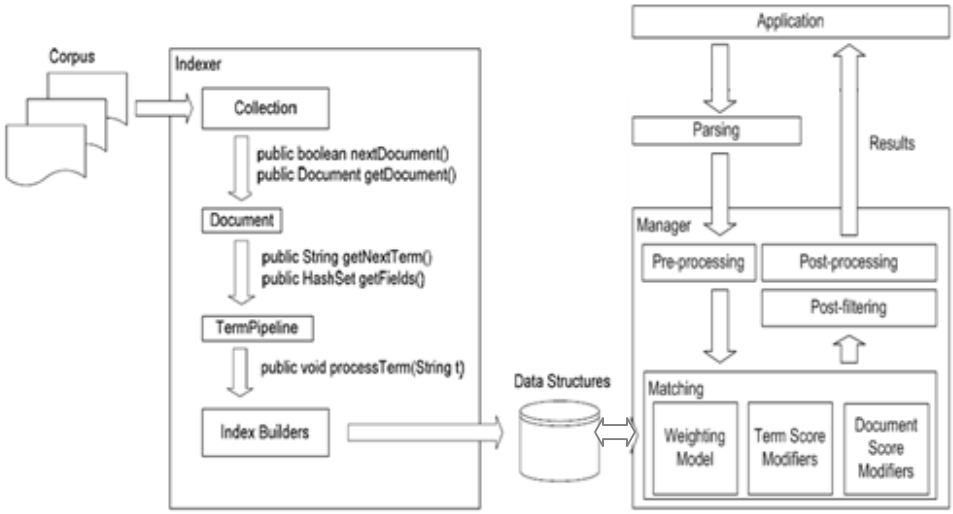


Figure III-4 : Vue d'ensemble d'architecture de Terrier

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

Les quatre procédures sont :

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

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

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

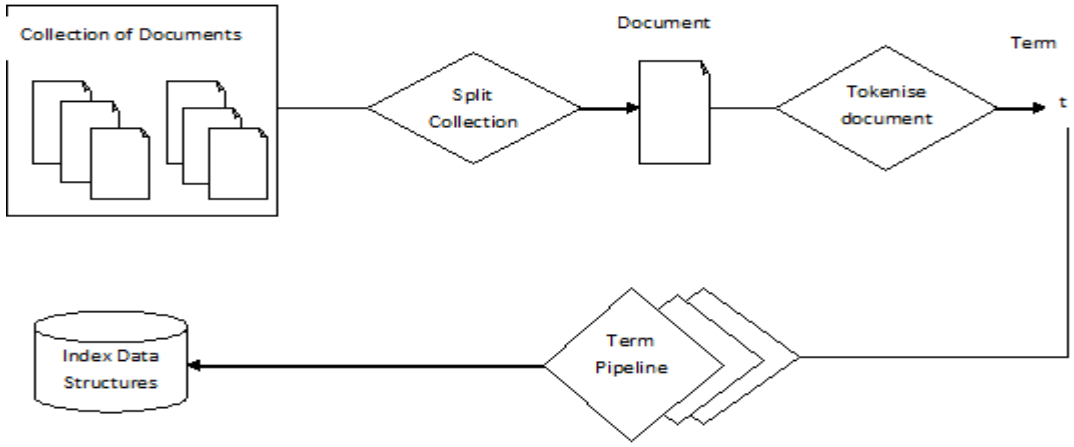


Figure III.-5 : Le processus d'indexation dans Terrier

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

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

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

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

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

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

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

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

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

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

1) Query : classe abstraite qui représente la requête.

   Terrier supporte trois modèles de requête :

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

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

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

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

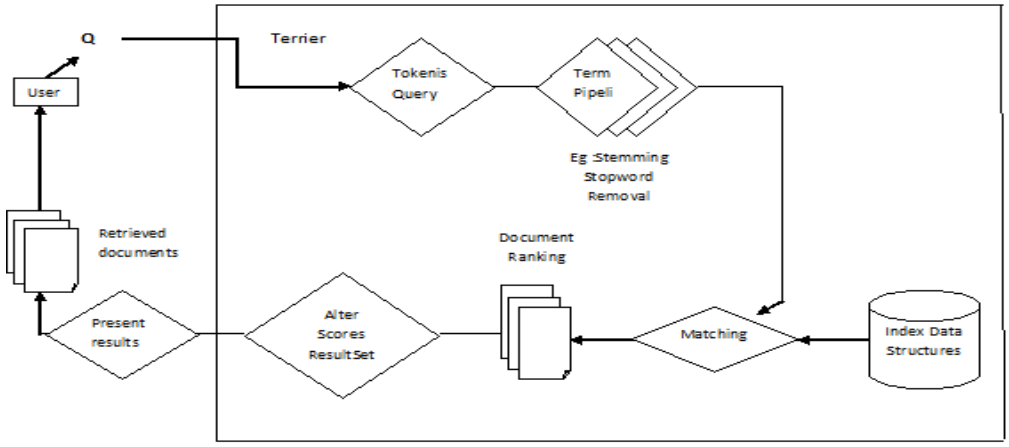


Figure III-6 : Le processus de recherche dans Terrier

III-3-2-2 Le langage java

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

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

Java se voit attribuer plusieurs qualités dont voici quelques unes :

- **Orienté objet** : la programmation orientée objets présente l'immense intérêt de faciliter la réutilisabilité des composants développés. Java est un langage orienté objet. Ecrire un programme revient à écrire une classe d'objets avec ses données et les méthodes qui permettent d'y accéder.
- **Simple :** java hérite une grande partie de la syntaxe du langage C++, et dans le but d'écrire des codes facilement et sans erreurs, il en a été dépouillé de tous les mécanismes complexes, redondants ou devenus inutiles, tels que la gestion des pointeurs, la gestion de la mémoire (la libération de la mémoire en particulier n'est plus à la charge du développeur mais est générée de manière automatique par ramasse-miettes intégré), l'héritage multiple,…etc.

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

III-3-2-3 NetBeans

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

GUI (Graphical User Interface) qui nous permet d'éditer, compiler et exécuter un programme écrit en langage java.

III-3-2-2 Outil TEXT-NSP

Le **p**aquet de **s**tatistiques de **N**gram (NSP) est une collection de modules de Perl qui facilitent en analysant Ngrams dans des fichiers texte. Nous définissons un Ngram comme ordre de 'n' tokens (terme) ; marques qui se produisent dans une fenêtre au moins de 'n' Tokens marques dans le texte ; ce qui constitue un 'n'; token; peut être défini par l'utilisateur. Les modules sous le texte : : NSP : : Mesure les mesures d'instrument de l'association qui sont employées pour évaluer si la Cooccurrence des mots dans un Ngram est purement par hasard ou statistiquement significatif.

III-4 Résultats et expérimentation

Cette section est consacrée à la présentation l'évaluation des performances de notre approche. Nous avons évalué les trois points suivant:

- La recherche simple la RI classique avec l'implémentation le modèle de recherche dirichlet (TS).
- Expansion de requêtes basée sur les termes simples (EQ_TS).
- La recherche basée sur mots composés (TC).

III-4-1Résultats obtenus avec la recherche simple (TS)

L'objectif de l'ensemble des tests d'évaluations basé sur la recherche simple est de déterminer la meilleure valeur de paramètre **μ** du modèle de recherche dirichlet.

Nous avons varié la valeur de ce dernier (μ) de 100 à 5000 avec un pas de 100, les résultats obtenus sont représentés dans le tableau suivant :

Les valeurs de μ	MAP	Les valeurs de μ	MAP
100	0,1918	2600	0,2124
200	0,2092	2700	0,2122
300	0,2137	2800	0,2113
400	0,2148	2900	0,2103
500	0,2168	3000	0,2098
600	0,2195	3100	0,2095
700	0,2165	3200	0,2087
800	0,2169	3300	0,2084
900	0,2198	3400	0,2081
1000	**0,2227**	3500	0,2078
1100	0,2225	3600	0,2074
1200	0,2225	3700	0,2067
1300	0,2222	3800	0,2064
1400	0,2212	3900	0,2061
1500	0,2211	4000	0,2058
1600	0,2204	4100	0,2055
1700	0,2199	4200	0,2051
1800	0,2195	4300	0,2039
1900	0,2193	4400	0,2036
2000	0,219	4500	0,2032
2100	0,2184	4600	0,203
2200	0,2167	4700	0,2027
2300	0,2164	4800	0,2025
2400	0,2131	4900	0,2022
2500	0,2124	5000	0,2018

Tableau III.1. Précision moyenne en faisant varier le coefficient μ du modèle de dirichlet

D'après les résultats du tableau où en varie la constante μ de modèle Dirichlet de 100 à 5000 avec un saut de 100 indiquent que μ=1000 donne la meilleur précision qui égale à 0.2227.

III-4-2 Résultats obtenus avec l'expansion de la requêtes basée sur les termes simples

Dans ce point nous intéressons à ajoute des termes simple ou reformulation de la requête d'utilisateur. Pour cela en a besoin des information(données) suivant :

μ	Nombre de document	Nombre de document	Saut (pour les documents et termes)
1000	50	50	5

μ fixé à 1000 déduit dans TS, le nombre de documents et de termes. Ces deux dernières nous les avons variés de 5 à 50 le saut 5. Pour chaque document nous varions le nombre de termes dans le but de trouver la meilleure précision d'expansion de requête, pour quelle nombre document et terme. Les résultats sont représentés dans le tableau en dessous.

Nombre de document	Nombre de terme	MAP
5	5	0,2223
	10	0,2183
	15	0,2180
	20	0,2183
	25	0,2183
	30	0,2178
	35	0,2178
	40	0,2172
	45	0,2172
	50	0,2173
10	5	0,2254
	10	0,2260
	15	0,2267
	20	**0,2299**
	25	**0,2299**
	30	0,2297
	35	0,2238
	40	0,2272
	45	0,2237
	50	0,2238
15	5	0,2216
	10	0,2207
	15	0,2218
	20	0,2224
	25	0,2224
	30	0,2164
	35	0,2164
	40	0,2160
	45	0,2160
	50	0,2160
20	5	0,2199
	10	0,2168
	15	0,2174
	20	0,2188
	25	0,2140
	30	0,2138
	35	0,2138
	40	0,2140
	45	0,2139
	50	0,2138
25	5	0,2169
	10	0,2105
	15	0,2087
	20	0,2111
	25	0,2118
	30	0,2116
	35	0,2089
	40	0,2088
	45	0,2090
	50	0,2087

Nombre de document	Nombre de terme	MAP
30	5	0,2085
	10	0,2066
	15	0,2004
	20	0,2046
	25	0,2058
	30	0,2057
	35	0,2057
	40	0,2058
	45	0,2057
	50	0,2058
35	5	0,2078
	10	0,2034
	15	0,2009
	20	0,2015
	25	0,2036
	30	0,2004
	35	0,2036
	40	0,2036
	45	0,2036
	50	0,2036
40	5	0,2050
	10	0,2029
	15	0,1980
	20	0,1990
	25	0,1968
	30	0,1972
	35	0,1971
	40	0,1971
	45	0,1973
	50	0,1973
45	5	0,2089
	10	0,2017
	15	0,2000
	20	0,1978
	25	0,1983
	30	0,1988
	35	0,1988
	40	0,1987
	45	0,1988
	50	0,1989
50	5	0,2039
	10	0,2018
	15	0,2006
	20	0,2005
	25	0,2017
	30	0,2017
	35	0,2021
	40	0,2021
	45	0,2022
	50	0,2023

Tableau III.2. Précision moyenne de la recherche avec l'expansion de requête

Selon le tableau III-4 l'expansion de la requête améliore les résultats obtenu tel que ; le nombre de document égal à 10 et le nombre de terme égal à 20 donne la meilleure amélioration avec la précession égale à 0.2299, une amélioration de +3.23 par rapport à la recherche simple.

III-4-3 Résultats obtenus avec la recherche terme composée (TC)

L'objectif de la recherche avec des termes composée est réduire le 'ambiguïté des termes, c'est l'approche proposée. Les paramètres exigés sont : **μ**=1000 et nous varions la valeur de **α** de la formule (III-3) entre 0 et 1 avec le pas de 0.1 pour trouver la meilleure précision moyenne.

μ	Les valeurs de Alpha	Map
1000	0	0,2431
	0,1	0,2458
	0,2	0,2487
	0,3	**0,2493**
	0,4	0,2487
	0,5	0,2449
	0,6	0,2403
	0,7	0,2381
	0,8	0,2359
	0,9	0,2289
	1	0,2196

Tableau III.3. Précision moyenne en faisant varier le coefficient α

A l'observation des valeurs de α, nous déduisons que la valeur idéale de la MAP est calculé pour α=0.3, MAP=0.2493.

Nous remarquons qu'il y a une amélioration visible en passant de recherche simple(TS) à recherche avec mot composé(TC).

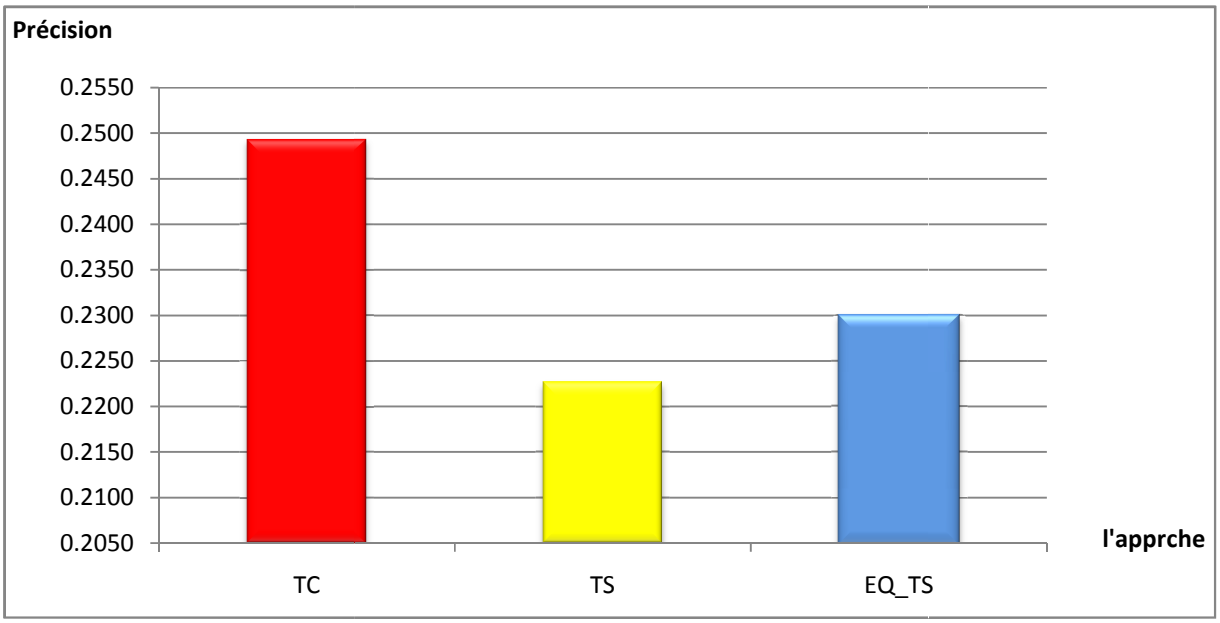


Figure III-7 comparaison les résultats de la recherche TS, TC et RQ_TS

Comme pouvons le remarquer, notre approche améliore les résultats du la recherche comme suit :

1- Premièrement le modèle utilisant les mots composée donne une amélioration de l'ordre de +11.94 par rapport à la recherche simple ce qui implique que l'utilisation des mots composées apporte un plus pour la RI.
2- Deuxièmement, le modèle utilisant l'expansion de requête améliore le modèle de base sur les mots simple ; l'amélioration est de l'ordre de +3.36 ; ce qui indique que l'expansion de requête peut apporter une amélioration pour la RI.

III-4-4 Analyse requête par requête

Pour avoir une vision plus claire des résultats obtenus précédent ;nous avons analysé les résultats requête- par requête. Les résultats sont donnés dans le tableau suivant.

N° requête	Map TS	Map EQ_TS	Map TC	le taux TS/ EQ_TS	le taux TS/ TC	N° requête	Map ts	Map EQ_TS	Map TC	le taux TS/EQ_TS	taux TS/TC
451	0,6039	0,6256	0,6058	3,59%	0,31%	491	0,006	0,0074	0,0057	23,33%	-5,00%
452	0,1537	0,2202		43,27%	-100,00%	492	0,2366	0,4383		85,25%	-100,00%
453	0,3399	0,3502		3,03%	-100,00%	493	0,0074	0,2888		3802,70%	-100,00%
454	0,4225	0,4658		10,25%	-100,00%	494	0,02219	0,2305	0,276	938,76%	1143,80%
455	0,3667	0,4447	0,3866	21,27%	5,43%	495	0,2628	0,3639		38,47%	-100,00%
456	0,0079	0,0014		-82,28%	-100,00%	496	0,2849	0,0842		-70,45%	-100,00%
457	0,1061	0,0645	0,1092	-39,21%	2,92%	497	0,3808	0,5605		47,19%	-100,00%
458	0,0264	0,0264		0,00%	-100,00%	481	0,3934	0,127	0,3872	-67,72%	-1,58%
459	0,2158	0,1717	0,19	-20,44%	-11,96%	482	0,0937	0,0093		-90,07%	-100,00%
460	0,148	0,2262		52,84%	-100,00%	483	0,0229	0,088	0,0229	284,28%	0,00%
461	0,7075	0,706	0,7079	-0,21%	0,06%	484	0,1324	1		655,29%	-100,00%
462	0,0753	0,0314		-58,30%	-100,00%	485	1	1,5664	1	56,64%	0,00%
463	0	0		-	-	486	0,4602	0,0728		-84,18%	-100,00%
464	0	0,0083		-	-	487	0	0,0315		-	-
465	0,0092	0,5152		5500,00%	-100,00%	498	0,0762	0,0951		24,80%	-100,00%
466	0,2627	0,1996		-24,02%	-100,00%	499	0,6772	0,0385	0,4133	-94,31%	-38,97%
467	0,17	0,0083	0,1531	-95,12%	-9,94%	500	0,0888	0,2053		131,19%	-100,00%
468	0,0122	0,0314		157,38%	-100,00%	501	0,047	0,0291	0,0529	-38,09%	12,55%
469	0,2468	0,2009	0,2585	-18,60%	4,74%	502	0,1289	0,4784		271,14%	-100,00%
470	0,1656	0,0493		-70,23%	-100,00%	503	0,1422	0,5059	0,1365	255,77%	-4,01%
471	0,053	0,096	0,1412	81,13%	166,42%	504	0,4788	0,1699		-64,52%	-100,00%
472	0,1001	0		-100,00%	-100,00%	505	0,3589	0,1496	0,3567	-58,32%	-0,61%
473	0	0	1	-	-	506	0,1276	0,1633		27,98%	-100,00%
474	0,0009	0,2745		30400,00%	-100,00%	507	0,2327	0,3538	0,231	52,04%	-0,73%
475	0,1998	0,3982	0,2024	99,30%	1,30%	508	0,2039	0,5833		186,07%	-100,00%
476	0,4256	0,0086		-97,98%	-100,00%	509	0,3856	0,384	0,3684	-0,41%	-4,46%
477	0,0084	0,0096	0,0055	14,29%	-34,52%	510	0,5622	0,1494		-73,43%	-100,00%
478	0,0128	0,1516		1084,38%	-100,00%	511	0,3632	0,1058	0,2716	-70,87%	-25,22%
479	0,3048	0,0026	0,4283	-99,15%	40,52%	512	0,1663	0,2324		39,75%	-100,00%
480	0,0023	0,4521		19556,52%	-100,00%	513	0,1008	0,214		112,30%	-100,00%
481	0,3934	0,127	0,3872	-67,72%	-1,58%	514	0,2061	0,0958		-53,52%	-100,00%
482	0,0937	0,0093		-90,07%	-100,00%	515	0,2022	0,0339	0,2208	-83,23%	9,20%
483	0,0229	0,088	0,0229	284,28%	0,00%	516	0,1007	0,1741		72,89%	-100,00%
484	0,1324	1		655,29%	-100,00%	517	0,0312	0,2112	0,0397	576,92%	27,24%
485	1	1,5664	1	56,64%	0,00%	518	0,1676	0,0725		-56,74%	-100,00%
486	0,4602	0,0728		-84,18%	-100,00%	519	0,1419	0,0217	0,1264	-84,71%	-10,92%
487	0	0,0315		-	-	520	0,1072	0,3916		265,30%	-100,00%
488	0,077	0,1678		117,92%	-100,00%	521	0,0208	0,4707	0,0446	2162,98%	114,42%
489	0,0765	0,0016		-97,91%	-100,00%	522	0,3189	0,1906		-40,23%	-100,00%
490	0,2093	0,2521		20,45%	-100,00%	523	0,4619	0,1395		-69,80%	-100,00%

N° requête	Map ts	Map EQ_TS	Map TC	le taux TS/ EQ_TS	le taux TS/ TC
527	0,4002	0,3087	0,5695	-22,86%	42,30%
528	0,5186	0,6844		31,97%	-100,00%
529	0,3504	0,0033	0,335	-99,06%	-4,39%
530	0,6086	0,2225		-63,44%	-100,00%
531	0,0159	0,1096	0,062	589,31%	289,94%
532	0,2654	0,0233		-91,22%	-100,00%
533	0,1156	0,0508	0,1745	-56,06%	50,95%
534	0,0148	0,2996		1924,32%	-100,00%
535	0,0446	0,0044	0,0578	-90,13%	29,60%
536	0,1872	0,3333		78,04%	-100,00%
537	0,0419	0,0613	0,0366	46,30%	-12,65%
538	0,325	0,1034		-68,18%	-100,00%
539	0,0927	0,2992	0,0469	222,76%	-49,41%
540	0,1294	0,0375		-71,02%	-100,00%
541	0,2906	0,005	0,2704	-98,28%	-6,95%
542	0,0366	0,5854		1499,45%	-100,00%
543	0,007	0,1894	0,007	2605,71%	0,00%
544	0,6335	0,0888		-85,98%	-100,00%
545	0,1473	0,2022	0,1948	37,27%	32,25%
546	0,108	0,375		247,22%	-100,00%
547	0,1999	0,3144	0,1948	57,28%	-2,55%
548	0,625	0,1647	1	-73,65%	60,00%
549	0,3146		0,2787	-100,00%	-11,41%
550	0,1169			-100,00%	-100,00%

Tableau III.4.Résultats obtenus avec l'analyse requête par requête avec la recherche Simple, expansion de requête et notre approche.

Une fois la lecture de tableau III-4 faite nous déduisons les résultats suivants :

- Avec la recherche basée sur l'expansion de requête:
  - 54 requêtes ont été améliorées ;
  - 38 requêtes ont été dégradées ;
  - 3 requêtes ont été restées constantes ;
  - 5 requêtes ont été ignorés ;
- Dans la recherche simple et recherche avec terme composée :
  - 21 requêtes ont été améliorées ;
  - 15 requêtes ont été dégradées ;
  - 3 requêtes ont été restées constantes ;
  - 58 requêtes ont été ignorés ;

Avec le nombre de requête utiliser dans l'évaluation est cent (100) requêtes

Le graphe de la figure III-2 illustre les résultats d'une manière plus clairs.

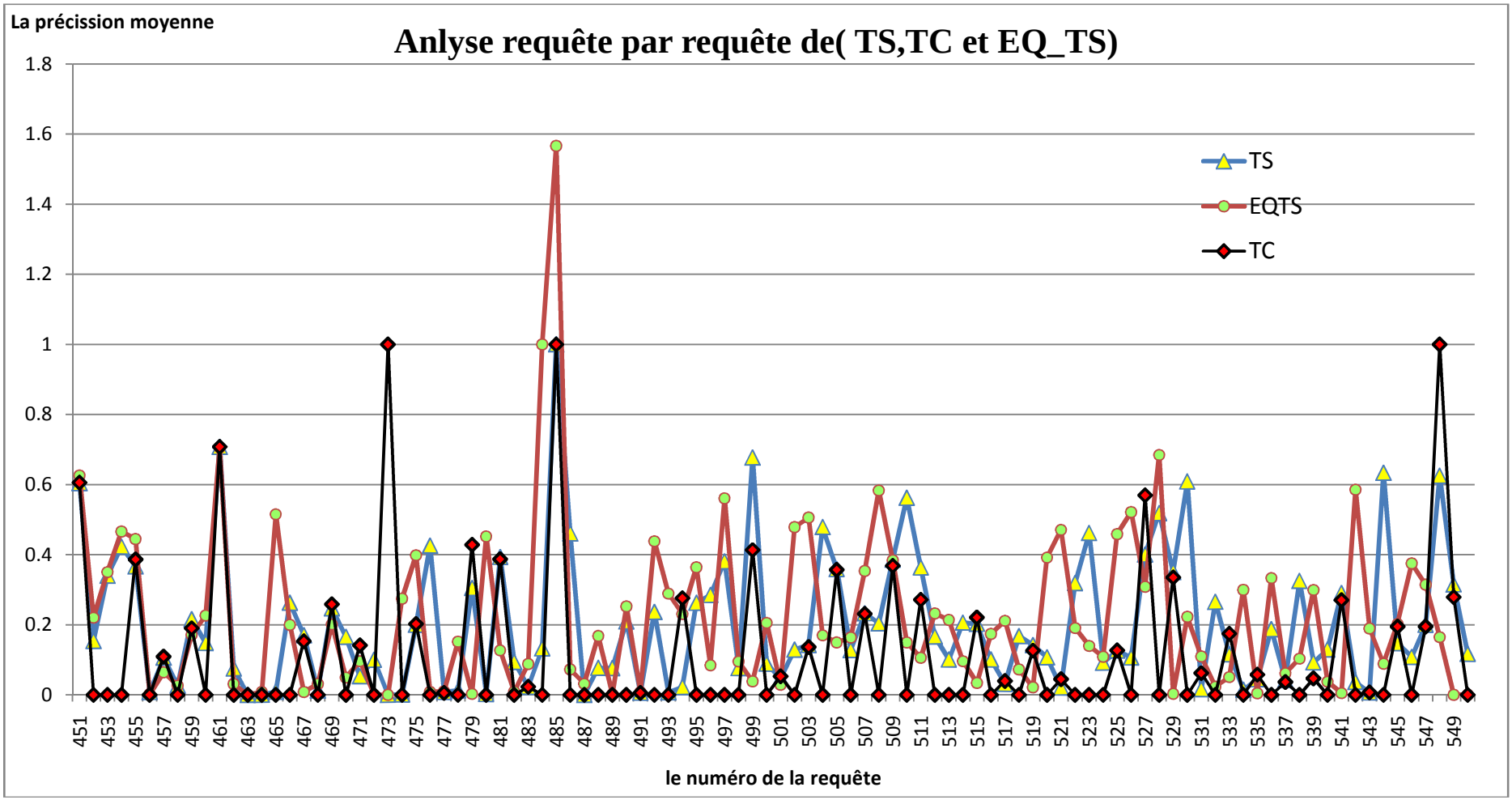


Figure III-8 Comparaison l'analyse requête par requête de la recherche simple, expansions de requête et terme composée

III4-5 Analyse des résultats basé sur le type de la requête

Dans ce qui suit nous analysons les résultats obtenus selon le type de la requête. Nous avons, défini trois(3) type de requête selon leurs précision moyenne (MAP) :

- les requêtes ambigües si la MAP <0,05 ;
- les requêtes moyennes si la 0,05 < MAP < 0,2 ;
- les requêtes améliores si la MAP > 0,2.

Les constantes prise pour distingue entre les différents type de la requête (pour les test de recherche. Le tableau ci-dessous montre les résultats obtenus

N° requête	Map TS	amilioration EQ_TS	teaux EQ_TS/TS	amilioration TC/TS	teaux	teaux (TC/EQ_TS)
463	ambigüe		75%		8%	8%
464	ambigüe	+				
473	ambigüe			+, ++		
487	ambigüe	+				
474	ambigüe	+				
480	ambigüe	+				
491	ambigüe	+				
543	ambigüe	+				
493	ambigüe	+				
456	ambigüe					
477	ambigüe	+				
465	ambigüe	+				
468	moyenne	+	67%		25%	19%
478	moyenne	+				
534	moyenne	+				
531	moyenne	+		+		
521	moyenne	+		+		
494	moyenne	+		+, ++		
483	moyenne	+				
458	moyenne					
517	moyenne	+		+		
542	moyenne	+				
537	moyenne	+				
535	moyenne			+, ++		
501	moyenne			+, ++		
471	moyenne	+		+, ++		
462	moyenne					
498	moyenne	+				
489	moyenne					
488	moyenne	+				
500	moyenne	+				

524	moyenne	+				
539	moyenne	+				
482	moyenne					
472	moyenne					
516	moyenne					
513	moyenne	+				
457	moyenne			+ , + +		
526	moyenne	+				
520	moyenne	+				
546	moyenne	+				
533	moyenne			+, ++		
550	moyenne					
525	moyenne	+		+		
506	moyenne	+				
502	moyenne	+				
540	moyenne					
484	moyenne	+				
519	moyenne			,++		
503	moyenne	+		, ++		
545	moyenne	+		+		
460	moyenne	+				
452	moyenne	+				
470	moyenne					
512	moyenne	+				
518	moyenne					
467	moyenne			,++		
536	moyenne	+				
475	moyenne	+		+		
547	moyenne	+				
515	claire	+	35%	+,++	20%	38%
508	claire	+				
514	claire	+				
490	claire	+				
459	claire			+,++		
507	claire	+				
492	claire	+				
469	claire			+ ,++		
466	claire					
495	claire	+				
532	claire					
496	claire					
541	claire			, ++		
479	claire			+, ++		

549	claire			,++		
522	claire					
538	claire					
453	claire	+				
529	claire			+, ++		
505	claire			, ++		
511	claire			, ++		
455	claire	+		+		
497	claire	+				
509	claire	+				
481	claire			,++		
527	claire			, ++		
454	claire					
476	claire					
486	claire					
523	claire					
504	claire			, ++		
528	claire	+				
510	claire					
451	claire	+				
530	claire					
548	claire			+, ++		
544	claire					
499	claire			, ++		
461	claire			+,++		
485	claire	+				

Tableau III.5.la comparaison les résultats par rapport au type de la requête

Dans cette section la comparaison se base sur le type de la requête, pour chaque approche le nombre de requêtes ambigües, moyennes et claires.

L'analyse des résultats montre que :

1) Le nombre de requêtes ambigües améliorées avec le modèle basé sur TC est de **1** au modèle basé sur TS est de **12**, le taux d'amélioration est de l'ordre de **8%.**
2) Le nombre de requêtes moyennes amélioré avec le modèle basé sur TC est de **12** au modèle basé sur TS **48**, le taux d'amélioration est d'ordre **25%**.
3) Le nombre de requêtes claires amélioré avec le modèle basé sur TC est de **8** au modèle basé sur TS **40,** le taux d'amélioration est d'ordre **20%.**
4) Le nombre de requêtes ambigües améliorées avec le modèle basé sur EQ_TS est de **9** au modèle basé sur TS est de **12**, le taux d'amélioration est de l'ordre de **75%.**
5) Le nombre de requêtes moyennes amélioré avec le modèle basé sur TC est de **32** au modèle basé sur TS **48**, le taux d'amélioration est d'ordre **67%**.
6) Le nombre de requêtes claires amélioré avec le modèle basé sur TC est de **14** au modèle basé sur TS **40,** le taux d'amélioration est d'ordre **35%.**

Dans le tableau en dessous en résument les résultats.

Type de la requête	Nombre de requête	Recherche simple (TS)	Expansion de requête	Taux	Recherche composée		Taux	
					TS	EQ_TS	TS	EQ_TS
Ambigüe	100	12	9	75%	1	1	8%	8%
Moyenne		48	32	67%	12	9	25%	19%
Claire		40	14	35%	8	15	20%	38%

Tableau III-6 amélioration des requêtes pars type dans le modèle TC et EQ-TS

III-5 Conclusion

Dans ce chapitre nous avons décrit et évalué notre approche qui combattre l'ambigüité et la disparité des termes dans la requête d'utilisateur. A partir les résultats d'expérimentations obtenus nous déduisons les point suivants :

- L'utilisation des mots composés améliore les résultats de la recherche d'information d'une manière substantielle.
- L'utilisation des l'expansion de requêtes améliore aussi les résultats de la RI.

Conclusion générale

Conclusion Générale

Il existe plusieurs travaux qui tentent d'améliorer les performances d'un système de recherche d'information basé sur les mots simples.

Notre approche dans ce mémoire s'inscrit dans les travaux qui améliorent la performance de recherche d'information, nous étudions l'ambiguïté et a disparité des termes de recherche d'information. Les solutions proposés sont : expansion de la requête qui permet de reformulé la requête de utilisateur et le terme composé comme une unité d'indexe dans le modèle de langue mixte.

Nous avons expérimenté et évalué les approche proposée sur la plateforme terrier1.2 avec l'implémentation le modèle de langue mixte pour la recherche avec mot composée et le modèle Dirichlet pour la recherche avec terme simple et l'expansion, la collection de test TRC(WT10g) et les requêtes (Topic451-550).

La première solution se base sur la expansion de requête d'utilisateur, pose le problème de disparité des termes, nous choisissons la technique l'expansion de requêtes automatique .L'amélioration relative à cette démarche a permis d'une part, de se focaliser sur le sens dominant de ces requêtes, d'améliorer la qualité des requêtes traduites et donc d'améliorer la qualité des résultats recensés

La deuxième solution proposée. Cette représentation du contenu sémantique des documents est implémenté dans le but de remédie le problème d'ambiguïté. L'extraction des mots composés est basée sur l'utilisation de la relation de cooccurrence entre termes.

Evaluation les deux solutions donnent des résultats très importante, dans chaque solution des améliorations remarquable. Les valeurs de la MAP s'améliorent d'une MAP à un autre la MAP, de la recherche classique par rapport à la recherche expansion terme simple d'une part.

d'autre part la MAP recherche classique (terme simple) par rapport à la recherche avec terme composé.

Le but d'un système de recherche d'information est donc de renvoyer l'information pertinente vis-à-vis du besoin en information de l'utilisateur. D'après notre approche, nous avons constaté que:

- ✓ *L'utilisation de l'expansion de requête améliore les résultats de la recherche.*
- ✓ *L'utilisation des mots composés améliore également les résultats de la recherche de manière substantielle.*

En concluent avec une approche très efficace ajoutée dans les approches d'amélioration de système de recherche d'information

.

ANNEX

Annexe

1- Environnement de Développement Intégré Java

a) Définition

Les IDE sont des programmes qui regroupent un ensemble d'outils pour le développement de logiciels.

De façon générale, un IDE contient un éditeur de texte, un compilateur, des outils automatiques de fabrication, et très souvent un débogueur. Il existe des IDE pour de nombreux langages, cependant il est très courant qu'un IDE soit conçu pour un seul langage de programmation. Il est également possible qu'un IDE dispose d'un système de gestion de versions et différents outils pour faciliter la création des interfaces graphiques

b) Fonctionnalités requises & besoins

Afin d'être le plus efficace possible lors du développement de l'application SubJects, il est nécessaire d'énumérer les différents critères à partir desquels sera réalisé le choix de l'IDE.

- La solution utilisée devra, au mieux, répondre aux besoins suivants : Possibilité de déploiement d'applications Web ;
- Rapidité de fonctionnement ;
- Léger au lancement ;
- Compilation possible du projet ;
- Gestion de plusieurs projets ;
- Débogueur précis ;
- Visualisation aisée de la JavaDoc ;
- Interfaçage avec un gestionnaire de versions ;
- Logiciel simple et facilité d'utilisation**.**

c) Outils existants

Les besoins maintenant exprimés, il a fallut réaliser une présélection des IDE afin de ne pas se disperser avec un comparatif trop large. Les environnements retenus sont donc NetBeans et Eclipse. Cela principalement du fait que ce sont les IDE java connues pour être les plus performantes et les plus largement utilisées.

d) NetBeans

Cet IDE a été créé à l'initiative de Sun Microsystems. Il présente toutes les caractéristiques indispensables à un environnement de qualité, que ce soit pour développer en Java, Ruby, C/C++ ou même

PHP.

NetBeans est sous licence OpenSource, il permet de développer et déployer rapidement et gratuitement des applications graphiques Swing, des Applets, des JSP/Servlets, des architectures J2EE, dans un environnement fortement personnalisable.

L'IDE NetBeans repose sur un noyau robuste, la plateforme NetBeans, que vous pouvez également utiliser pour développer vos propres applications Java, et un système de plugins performant, qui permet d'avoir un IDE modulable.

A coté de la version complète de l'IDE NetBeans, il existe différentes déclinaisons qui se concentrent sur une plateforme ou un langage précis (Java ME, Java : SE + ME + EE, Ruby, C/C++, PHP).

NetBeans contient, en plus du support pour CVS et SubVersion, un support pour ClearCase, mais aussi pour Mercurial.

Il permet également de déployer des applications Web, non seulement vers Tomcat et Glassfish qui sont livrés avec le "Pack Web", mais aussi vers JBoss, WebSphere 6.1, WebLogic 9.

NetBeans détient un support de développement d'applications Web avec des améliorations pour l'édition des JSP, la gestion serveur et le support des dernières versions de Tomcat.

Enfin cet IDE possède un débogueur de grande qualité ainsi qu'une interface graphique améliorée.

2- Le langage programmation java

A) Qu'est-ce que la technologie Java

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts !

La dernière version de Java comprend d'importantes améliorations en matière de performances, de stabilité et de sécurité pour les applications Java exécutées sur votre ordinateur. L'installation de cette mise à jour gratuite garantit que les applications Java sont toujours exécutées de manière sécurisée et efficace.

B) Qu'est-ce que le plug-in Java ?

Le plug-in Java est un composant de l'environnement JRE. Ce dernier permet aux applets écrites en langage de programmation Java d'être exécutées dans différents navigateurs. Le plug-in Java n'est pas un programme autonome et ne peut pas être installé séparément.

c) Java Virtual Machine" et "JVM

La Java Virtual Machine n'est qu'une partie du logiciel Java, liée à l'interaction avec le Web. Elle est incluse dans le téléchargement du logiciel Java et permet l'exécution des applications Java.

Quand un ordinateur veut lancer un programme java écrit sur un autre ordinateur, il charge le fichier bytecode de ce programme et le transmet à sa propre JVM. La JVM traduit alors le bytecode dans le langage machine de son propre système et lance le résultat.

3- Outil Text-NSP :

Le **p**aquet de **s**tatistiques de **N**gram (NSP) est une collection de modules de Perl qui facilitent en analysant Ngrams dans des fichiers texte. Nous définissons un Ngram comme ordre de ‘n’ tokens (terme) ; marques qui se produisent dans une fenêtre au moins de ‘n’ Tokens marques dans le texte ; ce qui constitue un ‘n’quot ; token&quot ; peut être défini par l'utilisateur.

Les modules sous le texte : : NSP : : Mesure les mesures d'instrument de l'association qui sont employées pour évaluer si la Cooccurrence des mots dans un Ngram est purement par hasard ou statistiquement significatif .

A) count.pl

Le programme count.pl prend un certain nombre documents de texte en clair ou d'annuaires de tels dossiers et compte tout le nombre de ngrams aussi bien leurs totaux marginaux. Il fournit la capacité de définir ce qu'une marque peut employer des expressions régulières (par l'intermédiaire du --option symbolique). Un ngram est un ordre commandé des marques de n,et sous ce plan les marques peuvent être presque quelque chose, y compris l'espace a séparé des ficelles, des caractères, etc En outre, des ngrams peuvent se composer de non adjacent

marques dues au --option de fenêtre qui laisse utilisateurs pour spécifier le nombre de marques dans lesquelles un ngram doit se produire.

Le compte est fait utilisant hache dans le Perl qui est mémoire intensive. En conséquence, le NSP fournit également le programme de huge-count.pl et les diverses autres utilités de huge-*.pl qui effectuent la fonctionnalité de count.pl utilisant l'espace d'unité de disque dur plutôt que la mémoire. Ceci peut mesurer à des quantités de texte beaucoup plus grandes, bien qu'habituellement prenant plus de temps dans le processus.
Par défaut count.pl traite des ngrams en tant qu'ordres commandés des marques ; la niche est distincte du chien de maison. Cependant, il se peut que l'ordre ne fasse pas toujours la matière, et un utilisateur peuvent simplement vouloir savoir si deux mots Co-se produisent. Dans ce cas le programme de combig.pl ajuste des comptes de count.pl pour refléter un compte non commandé, où le chien de niche et de maison sont considérés les mêmes

en conclusion, find-compounds.pl permet à un utilisateur de spécifier un dossier des expressions pluritermes déjà connues (comme des noms de lieu, des idiomes, etc.) et puis d'identifier toutes les occurrences de ceux dans un corpus avant de courir count.pl.

B) statistic.pl

Le noyau du NSP est un large éventail de mesures d'association qui peuvent être employées pour identifier les ngrams, en particulier les bigrams et les trigrams intéressants. Les mesures sont organisées en familles qui partagent les caractéristiques communes (qui sont décrites en détail dans la documentation de code source). Ceci tient compte d'une exécution orientée objectivement qui favorise l'héritage de la fonctionnalité commune parmi ces mesures. Notez que toutes les mesures d'information réciproque sont soutenues pour des trigrams, et que le rapport de Loglikelihood est soutenu pour 4 grammes.

C) rank.pl

Une expérience naturelle est de comparer la production de statistic.pl pour la même entrée utilisant différentes mesures d'association. rank.pl prend comme entrée la production de statistic.pl pour deux mesures différentes, et calcule le coefficient de corrélation luxuriant de l'homme armé d'une lance entre elles. Généralement les mesures dans la même famille se corrèlent plus étroitement les uns avec les autres qu'avec des mesures d'une famille différente. un exemple tmi et le ll aussi bien que les matrices et le jaccard diffèrent par seulement des termes constants et produisent donc le rang identique. Il est souvent intéressant d'entreprendre

des études exploratoires avec des mesures multiples, et la corrélation luxuriante peut aider à identifier quand deux mesures sont très semblables ou différentes.

Les commande principale pour utilisée TEXT-NSP :

Pour forme la liste des ngram :

- Perl count.pl Ngram n fichier1.txt fichier.txt

Pour forme la liste des ngram avec la fréquence :

- Perl count.pl frequency suil fichier1. Txt fichier.txt

Ou:

Fichier 1 le fichier de sortie

Fichier le fichier d'entré.

Le contenue de fichier de sortie est :

```
84627552
0<>00<>291776 736093 429982
00<>0<>141035 429696 736575
```

4- Terrier

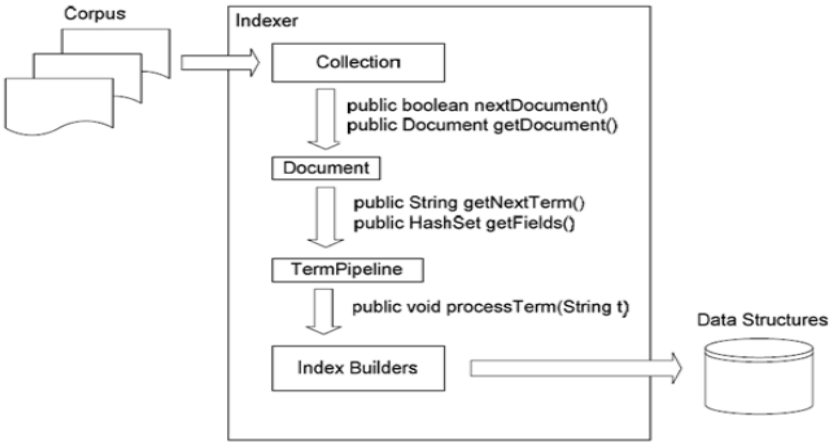
Terrier est une plate-forme dédiée à la recherche d'information. Elle implémente les différents modules intervenant dans le processus de RI classique et offre en plus un cadre pour l'évaluation des résultats de recherche pour différentes applications (Ounis et al., 2006). Terrier a été largement éprouvée (Middleton et al., 2007). Le choix de cette plate-forme est dû aussi à sa capacité à traiter de grandes collections de documents telles que les collections TREC.

A) Architecture de Terrier SIR

L'architecture de la plate-forme Terrier distingue les deux phases classiques : l'indexation et la recherche.

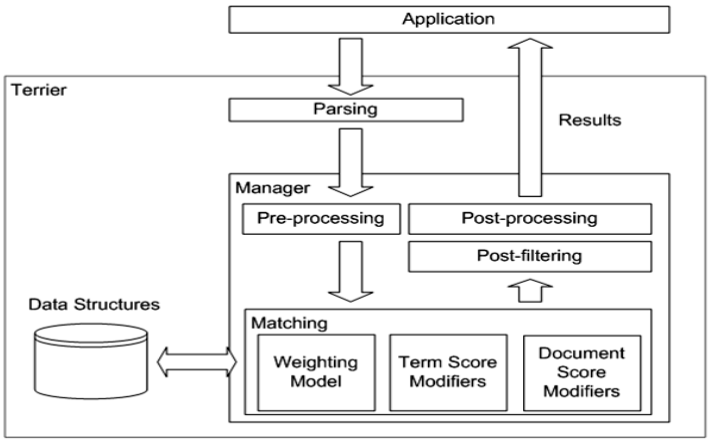
- **Indexation**

Dans la figure (1) en présentent les différentes parties principals de processus d'indexation



Un corpus documentaire est fourni en entrée au module d'indexation.
Les documents de la collection passent par un ensemble de prétraitements tels que la tokenisation. Les tokens sont ensuite injectés dans une chaîne de traitement TermPipLines, à savoir le StopWords PipeLine pour l'élimination des mots vides de sens, ou encore les Stemming pipeline et qui dépendent de la langue en question. La phase d'indexation conduit à la construction de l'index (Data structures).

Recherche



Une application, comme par exemple le Terrier de bureau ou les applications Trec Terrier, émet une requête au cadre Terrier.

- ❖ Dans la première étape , la requête sera analysée et une instanciation d'un objet de la requête (Query) aura lieu.
- ❖ Ensuite, la requête sera remis à la composante Manager. Le gestionnaire de processus tout d'abord le pré-rprocessus, en l'appliquant à la TermPipeline configuré.
- ❖ Après le Pre-processing de la requête sera remis à la composante de Matching. La composante de Matching est responsable de l'initialisation du modèle approprié de pondération(WeightingModel), TermScore modificateurs (TermScoreModifiers), et DocumentScoreModifiers. Une fois tous ces éléments ont été instancié le calcul de documents scores par rapport à la requête aura lieu.
- ❖ Ensuite, le Post Processing et PostFiltering a lieu. En PostProcessing, le ResultSet peut être modifié en aucune façon - par exemple, QueryExpansion élargit la requête, puis les appels Matching à nouveau pour générer un classement (ranking) amélioré des documents. Postfiltrage est plus simple, permettant documents à être soit inclus ou exclus - ce qui est idéal pour les applications interactives où les utilisateurs veulent restreindre le domaine des documents en cours de récupération.
- ❖ Enfin, le ResultSet sera retourné à l'organe d'application

Dans les deux partie précédentes en arrivent à déterminée globalement les deux partie principale de terrie.

Bibliographique

Bibliographique

[01]: Rami HARRATHI : thèse doctorat en informatique « Recherche d'information conceptuelle dans les documents semi-structurés » -L'Institut Nationale des Sciences Appliquées de Lyon2010

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

[03]: http://www.uhb.fr/urfist/Supports/RechInfoInit/RechInfo3Problematique.html

[04]: Lynda Tamine thèse doctorat en informatique « optimisation de requetes dans un systeme de recherche d'information *approche basee sur l'exploitation de techniques avancees de l'algorithmique genetique* » l'universite paul sabatier

[05]: R. Baea-Yates and B. Riberto-Neto. Modern Information Retrieval. New-York : ACP Press, Addison-Wesley, 1999.

[6] Rachid Arezki « Recherche adaptative de documents textuels» thèse de doctorat en informatique 2005

[07]: P. Ingwersen. Information Retrieval Interaction. Taylor Graham, London, 1992.

[08]: S. Mizzaro. Relevance : the whole history. Journal of the American Society for Information Science (JASIS), 48(9) :810–832, 1997.

[09]: P. Borlund and P. Ingwersen. Measures of relative relevance and ranked half-life : performance indicators for interactive ir. In International ACM-SIGIR conference, pages 24–28, 1998

[10] : C. Cleverdon. Evaluation tests of information retrieval systems. Journal of Documentation, 26 :55–67, 1970

[11] : S.P Harter. Psychological relevance and information science. American Society for Information Science (JASIS), 43(9) :602–615, 1992.

[12] : T. Saracevic. Relevance reconsidered. in information science : Integration in perspectives. In Conference on Conceptions of Library and Information Science, pages 201–218, 1996

[13] : G. Salton and C. Buckley On the use of spreading activation methods in automatic information retrieval. *11th ACM-SIGIR Conference*. p. 147-160, 1988

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

[15] : Porter, M. An algorithm for suffix stripping. Program, Vol. 14(3), pp. 130-137, 1980.

[6] : SOULE-DUPUY C., « Systèmes de recherche d'informations : mécanismes d'indexation et d'interrogation », Thèse de doctorat de l'Université Paul Sabatier, n°612, Toulouse III, février 1990

[17] : 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.

[18] : Williams, H., Zobel, J. Compressing Integers for Fast File Access. Computer Journal 42, pp. 193-201, 1999

[19]: Witten, I., Moffat, A., and Bell, T. Managing Gigabytes: Compressing and Indexing Documents and Images, Van Nostrand Reinhold, New York, 1994

[25]: Robertson S. E., Walker S. On relevance weights with little relevance information. In Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pages 16–24. ACM Press.

[26]: Singhal A., Mitra M., Buckely C. Learning routing queries in a query zone. In Proceedings of the 20th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Philadelphia, Pennsylvania, United States, July 27 - 31, 1997

Bibliographique

[27] : Ricardo B Y., Berthier R N. Modern information retrieval, ACM (Association for Computing Machinery)

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

[20] : Ricardo B Y., Berthier R N. Modern information retrieval, ACM (Association for Computing Machinery) .

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

[23] : L. Zadeh. Fuzzy sets. Information and Control, 8 :338–353, 1965.

[24] : G. Salton. Automatic text processing : the transformtion, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc., 1989

[28] : R. Krovetz. “Homonymy and polysemy in information retrieval”. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (A CL-97}, pages 72-79, 1997

[29] : M. Hammache Arezki ; « Recherche d’Information : un modèle de langue combinant mots simples et mots composés »,thèse doctorat 2013 Université Mouloud Mammeri de Tizi-Ouzou

[30] : Martin, W. J. R., AI, B. P. F., and van Strenkenburg, P. J. G. On the Processing of Test Corpus: Froni Textual Data to Lexicographical Information. In Lexicography. Principles and Practice, R. R. K. I-Tartmann (cd.), Academic Press, London, 1983, pp.77-87.

[31] : Fagan, J. L. Experiments in Automatic Phrase Indexing For Document Retrieval:A Comparison of Syntactic and Non-Syntactic Methods. *Ph.D. thesis, Department of Computer Science, Cornell University*, Ithaca, NY, 1987.

[32] : Khoo, C., Myaeng, S., and Oddy, R. Using Cause-Effect Relations in Text to Improve Information Retrieval Precision. *Information Processing and Management* 37, pp. 119-145, 2001.

[33] : Luhn, H. P. A Business Intelligence System. *IBM Journal Research and Development* (2:4), pp. 314-319, 1958

[34] : Baziz, M., Boughanem, M., Aussenac-Gilles, N. Chrisment, C. Semantic Cores for Representing Documents in IR. *Dans 20th ACM Symposium on Applied Computing*, Santa Fe, New Mexico, USA, ACM Press, New York, NY, USA, pp. 1011 - 1017, 2005.

[35] : Bartell, B.T., Cottrell, G.W., Belew, R.K. Automatic combination of multiple ranked retrieval systems. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 173 – 181, 1994.

[36] : Manning, D., Schütze, H. Foundations of Statistical Natural Language Processing. *MIT Press*, 2000.

[37] : Fagan, J. L. Experiments in Automatic Phrase Indexing For Document Retrieval:A Comparison of Syntactic and Non-Syntactic Methods. *Ph.D. thesis, Department of Computer Science, Cornell University*, Ithaca, NY, 1987. 67. Fagan, J.L. The Effectiveness of Non syntactic Approach to Automatic Phrase Indexing for Document Retrieval. *Journal of the american Socely for Information Science* 40:2, pp.115-132, 1989

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

[39] : A) P. Bailey, N. Craswell, et D. Hawking, 2003. Engineering a Multipurpose Test Collection for Web Retrieval Experiments. Information Processing & Management 39(6), 853–871.
b) P. Bailey, A. P. de Vries, N. Craswell, et I. Soboroff, 2007. Overview of the TREC 2007 Enterprise Track. In E. M. Voorhees et L. P. Buckland (Eds.), TREC, Volume Special Publication 500-274. National Institute of Standards and Technology (NIST).

Bibliographique

[40] : Smail M., «Vers des systèmes évolutifs de recherche d'information : un état de l'art » Technique et Science Informatiques, vol. 17, n°10, 1998

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

[42] : Ihadjaden M., Conception, réalisation et évaluation d'un système de recherche et de catégorisation automatique d'information textuelle sur Internet, Thèse de l'université ParisIV, 1994

[43]: Gerard Salton and Michael J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA,1986.

[44] D. Abberley, D. Kirby, S. Renals, and T. Robinson. The THISL broadcast news retrieval system. In Proc. ESCA Workshop on Accessing Information In Spoken Audio, pages 19{24, Cambridge, 1999.

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

[46) Olga Vectomova and Ying Wang. A study of the e_ect of term proximity on query expansion. Journal of Information Science, 32(4) :324{333, July 2006.

[47] : Carpineto, C. and Romano, G. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv. 44*, 1, Article 1 (January 2012), 50 pages.

[48] : M.J Blosseville, G. Hébrail, M.G Monteil & N. Pénot : Automatic Classification : Natural Langage Processing, Statical Analysis and Expert Techniques Used together, Conference on Research and Development in Information Retrieval (SIGIR), pp 51- 58, 1992

[49] : Lynda these doctora « Tamine Optimisation de requetes dans un Systeme de recherche d'information *Approche basee sur l'exploitation de techniques Avancees de l'algorithmique genetique »* l'UNIVERSITE PAUL SABATIER DE TOULOUSE 2002

[50] : 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

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

[52] : R. Attar & S. Franenckel : *Local Feedback in Full Text Retrieval Systems.* Journal of the ACM, 397-417, 1977

[53] : J. Xu & W.B. Croft : *Query Expansion Using Local and Global Document Analysis.* In Proc. ACM SIGIR Annual Conference on Research and Development, Zurich, 1996

[54] : J. Xu, J. Broglio, and B. Croft. The design and implementation of apart of speech tagger for english. Technical report, University of Massachusetts, Amherst, MA, USA, 1994

[55] : 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

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

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

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

[59] C. Lundquist, D. Grossman & O. Frieder : Improving Relevance Feedback in the Vector Space Model. In The Proceedings of the 6th ACM Annual Conference on Information Knowledge Management (CIKM'97)

Bibliographique

[60]: A. Singhal, G. Salton , M. Mitra, C. Buckley : Document Length Normalisation, Rapport de recherche, 1995

[62] : C.J. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In SIGIR ’92 : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, pages 77–88, New York, NY, USA, 1992. ACM Press.

[63]: G. Salton, C. Yang, and C. Yu. A Theory of term Importance in Automatic Text Analysis. Information Processing and Management, 24 :513–523, 1975.

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

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

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

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

[68]: C. Lundquist, D. Grossman & O. Frieder : *Improving Relevance Feedback in the Vector Space Model.* In The Proceedings of the 6th ACM Annual Conference on Information Knowledge Management (CIKM’97)

[69]: A. Singhal, G. Salton , M. Mitra, C. Buckley : *Document Length Normalisation,* Rapport de recherche, 1995