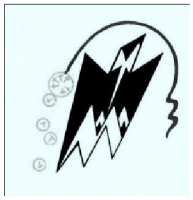
République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mouloud Mammeri Tizi-Ouzou
Faculté Génie électrique et d'informatique
Département d'Informatique
Option : Conduite de Projet Informatique

2ème ANNEE MASTER



MEMOIRE DE FIN D'ETUDE

Expansion de requêtes basée sur un modèle de langue mixte

Dirigé et proposé par :

Mr HAMMACHE Arezki.

Réalisé par :

Melle BOUALEM Chahinaze.

Melle LARID Nadia.

Promotion 2012/2013

Remerciements

Je remercie Dieu tout Puissant de m'avoir permis de mener à terme ce projet qui est pour moi le point de départ d'une merveilleuse aventure, celle de la recherche, source de remise en cause permanente et du perfectionnement perpétuelle.

Je tiens à remercier mon promoteur Mr HAMACHE Arezki, pour m'avoir transmis tout son savoir, pour m'avoir guidé et aidé durant le long de ce projet, d'avoir toujours été disponible et de bonne humeur, d'avoir été à l'écoute de toutes nos interrogations auxquelles il a su répondre avec une grande maitrise du sujet.

Je profite de cette tribune pour remercier les personnes qui de passage, ont pu m'apporter leur contribution, que ce soit au niveau des idées qu'à celui des conceptions. Qu'elles trouvent ici l'expression de mes sincères reconnaissances.

Enfin je remercie les membres du jury qui ont bien voulu accepter, et ce malgré, leur lourdes responsabilités pour procéder à l'évaluation de ce modeste travail.

Je dédie ce modeste travail :

A ma très chère mère affable, honorable et aimable : Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a jamais cessé de m'encourager et de prier pour moi.
A mon très cher père, Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être ;

Ce travail est le fruit de vos sacrifices que vous avez consentis pour mon éducation et ma formation ;

A mes deux chers frère « Boudjema et Karim » Mes deux ange Gardien et mes fidèles compagnons dans les moments les plus délicats ;

A mes deux chères sœurs « Lynda et Samia » en Témoignage de l'attachement, de l'amour et de l'affection que je porte pour elles;

A tous les membres de ma famille, petits et grands
Veuillez trouver dans ce modeste travail l'expression de mon affection ;

A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur ;

A toutes personnes m'ayant aidée de près ou de loin ;

Nadia

Je dédie ce modeste travail :

A mon très cher père, Rien au monde ne vaut les efforts fournit jour et nuit pour mon éducation et mon bien être ;

A ma très chère mère affable,honorable et aimable :Tu représentespour moi le symbole de labontéparexcellence,lasourcedetendresseet l'exemple du dévouement qui n'a jamais cessé de m'encourager et de prier pour moi.

Ce travail est le fruit de vos sacrifices que vous avez consentis pour mon éducation et ma formation ;

Mon cher petit frère « Slimane » *présent dans tous mes moments d'examens par son soutien moral et ses belles surprises sucrées.*
Je te souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité.

A mes chères sœurs «taous, kamilia, yasmina et tinhinane » en Témoignage de l'attachement, de l'amour et de l'affection que je porte pour elles;

A tous les membres de ma famille,petits et grands
Veuillez trouver dans ce modeste travail l'expression de mon affection ;

A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur ;

A toutes personnesm'ayant aidée de près ou de loin ;

Chahinaze

Sommaire

Introduction générale

Chapitre I : La recherche d'information

I.1Introduction ... 01
I.2Recherche d'information RI ... 01
I.3Système de recherche d'information SRI ... 01
I.4Processus générale de la RI ... 02
I.4.1Requête ... 03
I.4.2 Document ... 03
I.4.3 La pertinence ... 03
1.4.4 Indexation ... 04
1.4.4.1 Mode d'indexation ... 04
1.4.4.1.1 Analyse lexicale ... 04
1.4.4.1.2Élimination des mots vides ... 04
1.4.4.1.3La lemmatisation ... 05
1.4.4.1.4La pondération ... 05
1.4.5L'appariement requête-document ... 06
1.4.6 La reformulation de requête ... 07
1.4.6.1La Reformulation manuelle ... 07
1.4.6.2La Reformulation interactive ... 07
1.4.6.3 La Reformulation automatique ... 07
1.5Les principaux modèles de RI ... 07
1.5.1Modèle booléen (Boolean Model) ... 08
1.5.2Modèle vectoriel (VectorSpace Model) ... 09
1.5.3Modèle probabiliste (Probabilistic Model) ... 10
1.5.4Modèle de langue ... 11
1.6Évaluation de Système de recherche d'information ... 12
1.6.1 Les mesures précision-rappel ... 12
1.6.2 Autre mesures de performance ... 14
1.6.3 Corpus de test ... 14
1.6.3.1 Les collections TREC ... 14
1.7 Conclusion ... 15

Chapitre II : Reformulation de requête

II.1Introduction 16
II.2La reformulation de la requête de l'utilisateur 16
II.2.1Définition 16
II.2.2le processus de Reformulation 17
II.2.3 Les outils de base 17
II.2.3.1 La classification 17
a) Classification par choix d'attracteurs de groupe 17
b) Classification hiérarchique 17
c) Classification basée sur la pertinence des documents 18
II.2.3.2Thesaurus 18
II.2.4 La reformulation automatique 18
II.2.4.1Reformulation basée sur le contexte global 18
a) Utilisation d'un thesaurus manuel 19
b) Utilisation d'un thesaurus automatique basé sur la similarité 19
c) Utilisation d'un thesaurus basé sur le contexte 20
II.2.4.2 Reformulation basée sur le contexte local 21
II.2.5La reformulation par réinjection de pertinence 22
II.2.5.2 La réinjection de pertinence dans les modèles classique de RI 22
a) La réinjection de pertinence dans le modèle vectoriel 22
b) La réinjection de pertinence dans le modèle probabiliste 23
II.2.5.2La technique de réinjection de pertinence sans jugements de l'utilisateur 24
II.2.6Paramètre de performance de la reformulation de la requête 25
II.2.6.1Méthode de sélection des termes 25
II.2.6.2Langueur moyenne de requête 26
II.2.6.3Type de la collection 26
II.2.6.4Cooccurrence des mots 26
II.3Modèle de langue 29
II.3.1 Principe général des modèles de langue 29
II.3.2 Les techniques de lissage 31
II.3.3 Approches des modèles de langue 33
II.3.3.1 Le modèle de Ponte et Croft 34
II.3.3.2 Le modèle de Hiemstra 35
II.3.3.3Le ratio de vraisemblance 37
II.3.3.4Modèle basé sur l'entropie croisée 38
II.4 Expansion de requêtes dans le modèle de langue 39
II.4.1 Le modèle de Kullback-Leibler(KL-divergence) 39
II.4.2 Estimation du modèle de la requête dans le modèle de langue 40
A) Approches basé sur la traduction 40
B) Approche basée sur le retour de pertinence (feedback) 41
C) Approches basées sur la dépendance entre termes 42
II.5 Conclusion 46

Chapitre III : Approche à implémenter

III.1 Introduction 47
III.2Approche proposée 47
III.2.1Estimation de modèle de la requête 48
III.2.1.1Estimation de modèle de requête basé sur la mesure d'association PMI et DICE .49
A. Estimation des probabilités $P(t/Q)$et $P(T/Q)$ 49
B. Estimation de la probabilité $P(w/T)$ 50
C. Estimation de la probabilité $P(w/t)$ 51
III.2.2Estimation de modèle de document 53
III.3Conclusion.....54

Chapitre IV : Expérimentations et évaluation

IV.1Introduction.....56
IV.2 Architecture logicielle de notre approche 57
IV.3Environnement de développement 58
IV.2.1Le langage java.....58
A. Les caractéristiques de langage java.....58
B. Présentation de NetBeans.....59
IV.2.2La plateforme terrier 60
A. Architecture de terrier 61
A.1 Le processus d'indexation de Terrier 61
A.2 Le processus de recherche de Terrier 62
IV.2.3présentation de Ngram Statistics Package (Text ::NSP).....62
A .Le module count.pl 63
B. Le module statistic.pl 65
IV.4Données utilisés.....65
IV.5Les étapes de mise en œuvre de notre approche 66
IV.6Evaluation de l'approche d'expansion de requête basée sur un modèle mixte combinant les termes simples et les termes composés.....70
IV.6.1Paramètre de modèle de notre approche 70
IV.6.2Résultat d'évaluation 72
IV.6.3Résultat d'évaluation requête par requête.....76
IV.7Conclusion 88

Conclusion générale

Liste des figures

Figure 1 : Architecteur générale d'un Système de RI 02

Figure 2 **:** Exemple de rappel et de précision pour une requête 13

Figure 3 : Allure d'un courbe rappel et précision. 13

Figure 4 : Principe général des modèles de langue. 31

Figure 5 : réseaux des dépendances pour estimer le modèle basé sur la position de pertinence 42

Figure6 : modèles graphiques représentant la modélisation de pertinence (gauche) ;expansion latente de concept on utilisant des concepts de termes simples (milieu) ; et expansion latente de concept on utilisant deux concepts de termes (droits) pour une requête de trois termes 45

Figure7 : Schéma générale de notre application 58

Figure 8: l'interface principale de l'environnement NetBeans 6.5 61

Figure 9: Présentation du processus d'indexation de Terrier 62

Figure 10: Présentation du processus recherche de Terrier 63

Figure 11: Exemple d'un fichier output.txt 65

Figure 12 : Structure d'une requête 66

Figure 13 : extrait de contenu d'un document avec des termes composés (simples) 67

Figure 14 : extrait de contenu d'un document avec des termes simples 67

Figure 15 : extrait de contenu de résultat retourné par count.pl 70

Figure 16 : Interface de notre application 71

Figure 17 : Graphes représentants les résultats d'évaluation requête par requête pour les quatre recherches avec la collection AP88 77

Figure 18 : Graphes représentants les résultats d'évaluation requête par requête pour les quatre recherches avec la collection WSJ 78

Liste des tableaux

Tableau 1: la colinéarité des vecteurs documents et requête 09

Tableau 2 :Table de contingence pour mesurer le degré d'association entre deux mots 27

Tableau 3: comparaison des différents modèles on utilisant la collection AP88 73
Tableau 4: comparaison des différents modèles on utilisant la collection WSJ 75

Tableau 5: résultat d'évaluation de l'approche d'expansion de requête avec les mesures d'association PMI et Dice par la collection AP88 et WSJ 76

Tableau 6 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec le modèle mixte sans expansion avec AP88 80

Tableau 7 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche termes simples et de la recherche termes simples avec l'expansion avec AP88 81

Tableau 8 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche modèle mixte et de la recherche modèle mixte avec l'expansion avec AP88 82

Tableau 9 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche de modèle mixte avec expansion avec AP88 83

Tableau 10: Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec le modèle mixte sans expansion avec WSJ 85

Tableau 11 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple sans expansion et de la recherche termes simples avec expansion avec WSJ 86

Tableau 12 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche modèle mixte sans expansion et de la recherche modèle mixte avec expansion avec WSJ 88

Tableau 13 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple avec expansion et de la recherche modèle mixte avec expansion avec WSJ 89

Introduction générale

Introduction générale

Le développement rapide des technologies de l'information et de la communication nous ont confrontés à une très grande masse d'informations hétérogènes.

L'introduction de l'internet à, en effet, complètement bouleversé le monde de l'information documentaire en favorisant la multiplicité des ressources documentaires au grand public. L'explosion du web au milieu des années quatre-vingt-dix a propulsé la recherche d'information au premier plan et, désormais, la recherche d'information est devenue un phénomène socio-économique, politique et culturel. En dehors de l'application classiques (bibliothèques ; gestion électronique des documents, archives et serveurs bibliographiques, etc…), un nombre important d'applications fait appel aujourd'hui à des solutions de recherche d'information.

Cette expansion des systèmes de recherche d'information au grand public notamment des moteurs de recherche a entrainé à la fois une multiplication et une diversification des usagers et une hétérogénéité croissante des documents. Cette double évaluation n'a cependant pas modifié l'objectif fondamental de la recherche d'information : le repérage de l'information pertinente avec le maximum de précision.

Depuis leur introduction en Recherche d'Information (RI), les modèles de langue se sont distingués par leur efficacité et leur fondement mathématique solide, (Ponte et al.,1998), ((pont et croft) et al.,1998). Contrairement aux autres modèles classique de RI,les modèles de langue ne modélisent pas directement la notion de pertinence d'un document face à une requête. La pertinence d'un document vis-à-vis d'une requête est vue comme la probabilité que la requête soit générée par le modèle de langue du document appelée aussi score de pertinence.

La plus part des modèles de langue en RI se basant sur l'indépendance entre termes, Ce qui conduit au problème d'ambiguïté et disparité entre termes.

Pour solutionner le problème d'ambigüité, l'une des pistes les plus utilisée est l'utilisation des mots composés comme unité d'indexation.
Les termes composés permettent de construire des unités d'indexation non ambiguës et plus précises et peuvent par conséquent améliorer la précision de la RI.

Pour palier au problème de disparité de termes, la reformulation de requêtes est la méthode la plus utilisée, elle est une phase importante dans les systèmes de recherche d'information. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des mots permettant de mieux exprimer son besoin.
Une des techniques les plus répandues en RI est la reformulation par réinjection de la pertinence, communément appelée, Relevance Feedback (RF). Elle consiste à extraire à partir des documents jugés pertinents par l'utilisateur, les mots clés les plus expressifs et les ajouter à la requête.

L'objectif de ce travail est d'implémenter une approche de reformulation de requêtes basée sur un modèle de langue mixte combinant les mots simples et composés
Cette technique considère la requête comme un ensemble de mots composés et un ensemble de mots simples. Pour déterminer les mots d'expansion on additionne les poids des relations d'un mot candidat avec chacun des mots de la requête (simple, composé). Un mot candidat est choisi s'il est fortement en relation avec la plupart des mots de la requête. Cette technique est modélisée dans le cadre de modèle de langue.

Nous présentons dans le premier chapitre, les concepts de base de la recherche d'information, ainsi que les principales fonctionnalités d'un système de recherche d'information ; enfin nous présentons l'évaluation des SRI.

Dans le chapitre deux, nous présentons les techniques de reformulation de requêtes. Particulièrement nous présentons, le principe de ces techniques et les différentes catégories de ces techniques. Nous présentons ensuite l'implémentation de la reformulation de requêtes dans les modèles de RI classique. Enfin, l'utilisation de la reformulation de requêtes dans le modèle de langue.

Nous représentons dans le troisième chapitre, la partie contribution de ce mémoire en décrivant une nouvelle approche d'expansion de requête basée sur le modèle de langue mixte combinant les termes simples et les termes composés. Les mesures, PMI et DICE sont utilisées pour pondérer la relation entre les termes, dans l'objectif d'extraire les termes d'expansion.

Le dernier chapitre est consacré à la présentation de l'implémentation et l'évaluation de l'approche présentée dans le chapitre III.

Chapitre I :

Recherche d'information

I.1 Introduction

Le rapide développement des technologies de l'information durant ces dernières années a eu pour conséquence une prolifération de sources d'informations hétérogènes. Il devient de plus en plus difficile pour les utilisateurs de retrouver précisément ce qu'ils recherchent dans cette masse de données. Le problème qui se pose actuellement n'est plus tant la disponibilité de l'information mais la capacité d'accès et de sélection de l'information répondant aux besoins précis d'un utilisateur à partir des représentations qu'il perçoit, La RI est le domaine par excellence venu pour répondre à ce besoin.
Dans ce chapitre, nous allons définir les concepts de base de la RI et les systèmes de recherche d'information(SRI), puis aborder le processus de recherche d'information àsavoir l'indexation, l'appariementrequête-document et la reformulation de requête. Par la suite, nous définirons les différents modèlesde la RI et enfin nous présentons l'évaluation des SRI.

I.2 Recherche d'information RI

Abrégée en **RI** ou (**IR** : Information Retrieval en anglais). La recherche d'information concerne la représentation, le stockage, l'organisation et l'accès aux objets informationnels(Baesa 1999). Cette définition porte plutôt sur le système de recherche que sur le processus de recherche d'information. En revanche, la recherche d'information, telle qu'elle est définie par Fidel et coll. (2001, p.23), peut être interprétée dans un sens large qui consiste en des processus tels que l'identification du problème informationnel, l'analyse de besoin informationnel, la formulation de la requête, l'interaction avec le système, l'évaluation des résultats, et leur présentation. Cette dernière définition implique une dimension beaucoup plus large que celle du SRI.

I.3 Système de recherche d'information SRI

*Un **Système de Recherche d'Informations*** est un ensemble de logiciel assurant l'ensemble des fonctions nécessaires à la recherche. Il inclut un ensemble de procédures et d'opérations qui permettent la gestion, le stockage, l'interrogation, la recherche, la sélection et la représentation de cette masse d'informations [ZemirliW.Nesrine].

La figure suivante représente l'architecture d'un système de recherche d'information

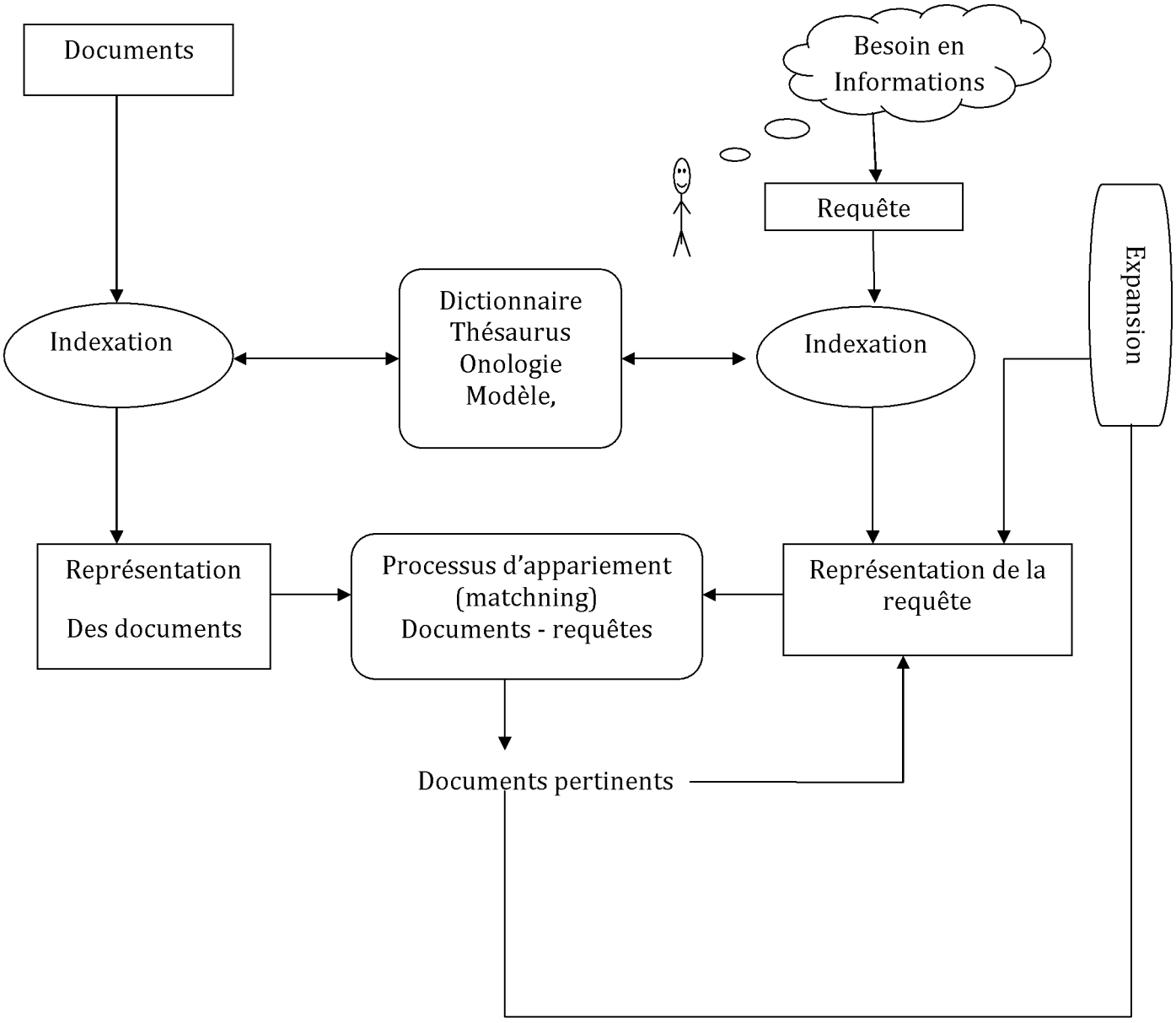


Figure 1 : Architecteur générale d'un Système de RI

I.4Processus générale de la RI :

Les différentes étapes du processus de RI, sont représentées schématiquement par le processus en U [Belkin 92]. La figure 1 illustre particulièrement :

- ✓ les notions de documents et de requêtes qui sont des conteneurs d'informations,
- ✓ les opérations d'analyse, d'indexation et d'appariement qui permettent globalement de traiter la requête dans le but de sélectionner des documents à présenter à l'utilisateur.

De la définition d'un SRI il ressort les concepts et processus suivants :

I.4.1 Requête

La requête constitue l'expression du besoin en information de l'utilisateur. Elle représente l'interface entre le SRI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête généralement est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique [ZemirliW.Nesrine].

I.4.2 Le document

Le document représente le conteneur élémentaire d'information, exploitable et accessible par le SRI. Ilpeut 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 un besoin en information exprimé par un utilisateur[NawelNassr].

I.4.3 La pertinence

Le but de la RI est de trouver seulement les documents pertinents. La notion de pertinence est très complexe. De façon générale, dans document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin. C'est sur cette notion de pertinence que le système doit juger si un document doit être donné à l'utilisateur comme réponse. Cette notion de pertinence peut être appréhendée à deux niveaux[BOUCHAM Souhila]:

- ***Le niveau utilisateur*** **:** à ce niveau, l'utilisateur a un besoin d'information dans sa tête, Et il espère obtenir les documents pertinents pour répondre à ce besoin. La relation entre le besoin d'information et les documents attendus est la relation de pertinence (idéale, absolue, …).

- ***Le niveau système*** : à ce niveau, le système répond à la requête formulée par L'utilisateur, par un ensemble de documents trouvés dans la base de documents qu'il possède. Remarquez que la requête formulée par l'utilisateur n'est qu'une description partielle de son besoin d'information. Beaucoup d'études ont montré qu'il est très difficile, voire impossible, de formuler une requête qui décrit complètement et précisément un besoin d'information. Du côté de document, il y a aussi un changement entre les deux niveaux : les documents que l'on peut retrouver sont seulement les documents inclus dans la collection de documents. On ne peut souvent pas trouver des documents parfaitement pertinents à un besoin. Il arrive souvent qu'aucun document pertinent n'existe dans la collection.

I.4.4 Indexation

L'indexation est une étape très importante dans le processus de RI, Elle consiste à déterminer et extraire les termes représentatifs du contenu d'un document ou d'une requête, qui couvrent au mieux leur contenu sémantique. La qualité de la recherche dépend en grande partie de la qualité de l'indexation [ZemirliW.Nesrine].

I.4.4.1 Mode d'indexation

L'indexation peut être manuelle, automatique ou semi-automatique

- *manuelle*: chaque document est analysé par un spécialiste du domaine correspondant ou par un documentaliste,

- *automatique*: chaque document est analysé à l'aide d'un processus entièrement automatisé,

- *semi-automatique*: le choix final reste au spécialiste du domaine correspondant ou documentaliste, qui intervient souvent pour établir des relations sémantiques entre mots-clés et choisir les termes significatifs.

Dans le cas d'une indexation automatique, elle passe par plusieurs étapes :

I.4.4.1.1 Analyse lexicale

Le lexique recense les mots et leurs différents sens, indépendamment des relations qu'ils établissent dans le discours. Lors des traitements automatiques, l'analyse lexicale est un facteur important (E. Laporte, dans Pierrel, 2000, p. 25, [53]) :*« Le niveau lexical du traitement des langues naturelles correspond aux traitements centrés sur la notion de mot. »*
Les ambiguïtés lexicales peuvent être levées à l'aide des dictionnaires électroniques, comme, par exemple, ceux élaborés par le LADL13 (Silberztein, 1993, [54]) ou ceux commercialisés par la société Memodata (Memodata, 1999, [55]).

I.4.4.1.2 Élimination des mots vides

Un des problèmes majeurs de l'indexation consiste à extraire des mots vides (pronoms personnels, préposition…).Les mots vides peuvent aussi être des mots athématique(les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le système mais ne le traitent pas, comme par exemple *contenir, appartenir*).
On distingue deux techniques pour éliminer les mots vides[HLA, 07] :

- L'utilisation d'une liste de mots vides (aussi appels « anti-dictionnaire » stoplist en anglais) ;
- L'élimination des mots dépassant un certain nombre d'occurrences dans la collection.

I.4.4.1.3 La lemmatisation

Les mots d'une langue peuvent être classés en deux catégories :
Les lemmes: formes canoniques (infinitif pour les verbes, singulier pour les noms, etc.) qui constituent en général les entrées dans un dictionnaire de cette langue, et les mots obtenus par flexion de ces lemmes: conjugaison d'un verbe, changement de genre ou de nombre, donc La lemmatisation consiste à remplacer un mot par son lemme.

Plusieurs techniques ont été utilisées pour réaliser cette tâche, entre autres : la troncature, qui consiste à tronquer les mots à X caractère (7 caractères pour la langue française), l'algorithme le plus connu est celui de Porter.

I.4.4.1.4 La pondération

La pondération permet d'affecter à chaque terme d'indexation une valeur qui mesure son importance dans le document où il apparaît.
Cette constatation a été faite par Zipf [ZemirliW.Nesrine]qui considère que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste qui représente des termes d'un texte donné.
Le pouvoir de discrimination des termes pour décrire le contenu des documents n'est pas identique pour tous les termes. Pour trouver les termes du document qui représentent le mieux son contenu sémantique, [Roberston 76] a défini la fonction de pondérationd'un terme dans un document connue sous le nom de ***Tf.Idf***, qui est reprise dans différentes versions par la majorité des SRI [Robertson 76], [Singhal 97] et [Sparck Jones 79].

- *Tf* (*termfrequency*) : cette mesure est proportionnelle à la fréquence du terme dans le document. L'idée sous-jacente est que plus un terme est fréquent dans un document, plus il est important dans la description de ce document. Le *Tf* est souvent exprimé selon l'une des déclinaisons suivantes :

  Tf : Utilisation brute.

$$0.5+0.5\ \frac{Tf}{Max(Tf)} \qquad I.1)$$

- *Idf*(*Inverse of Document Frequency*) : mesure l'importance d'un terme dans toute la collection. L'idée sous-jacente est que les termes qui apparaissent dans peu de documents de la collection sont plus représentatifs du contenu de ces documents que ceux qui apparaissent dans tous les documents de la collection. Cette mesure est exprimée selon l'une des déclinaisons suivantes :

$$Idf = log\left(\frac{N}{df}\right) \qquad I.2)$$

$$Idf = log\left(\frac{N - df}{df}\right) \qquad I.3)$$

- Où *df* est la proportion de documents contenant le terme et N le nombre total de documents dans la collection.
  La fonction de pondération de la forme ***Tf.Idf*** consiste à multiplier les deux mesures Tf et Idf. Une formule largement utilisée est la suivante :

$$\mathbf{Tf.Idf} = \left(0.5 + 0.5\frac{\mathbf{Tf}}{\mathbf{Max(Tf)}}\right) \times \mathbf{log}\left(\frac{\mathbf{N}}{\mathbf{df}}\right) \qquad \mathbf{I.4)}$$

Une normalisation de la mesure du *Tf.Idf* par rapport à la longueur des documents a été proposée par [Singhal 95].

$$Tf.Idf = \frac{Tf + log\,(\frac{N-df+0.5}{df+0.5})}{2.(0.25 + 0.75\,.\frac{dl}{\Delta d})} \qquad I.5)$$

Où :

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

En effet, lors des compagnes d'évaluation internationales, les termes appartenant aux documents longs apparaissent très fréquemment et emportent le poids sur les termes appartenant à des documents moins longs. Les documents longs auront alors plus de chance d'être sélectionnés [DeClaris 94].

I.4.5 L'appariement requête-document

Les SRI intègrent un processus de recherche/décision qui permet de sélectionner l'information jugée pertinente pour l'utilisateur. A cet effet, une mesure de similitude qui est appelée *Retrieval Status Value* ou *RSV*(correspondance) entre la requête indexée et les descripteurs des documents de la collection est calculée. Seuls les documents dont la similitude dépasse un seuil prédéfini sont sélectionnés par le SRI.
La fonction de correspondance est un élément clé d'un SRI, car la qualité des résultats dépend de l'aptitude du système à calculer une pertinence des documents la plus proche possible du jugement de pertinence de l'utilisateur.

Il existe deux types d'appariement :

1. *Appariement exact :* Le résultat est une liste de documents respectant exactement la requête spécifiée avec des critères précis. Les documents retournés ne sont pas triés.

2. *Appariement approché :* Le résultat est une liste de documents censés être pertinents pour la requête. Les documents retournés sont triés selon un ordre de mesure. Cet ordre reflète le degré de pertinence document/requête.

I.4.6La Reformulation de requête

Il est souvent difficile pour l'utilisateur de formuler son besoin exact en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent pas parfois. Retrouver des informations pertinentes en utilisant la seule requête initiale de l'utilisateur est aujourd'hui très difficile, et ce à cause du volume croissant des bases documentaires. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence du système, une étape de reformulation de la requête est souvent utilisée [Boubekeur,08].
La reformulation de requête peut être réalisée avec ou sans intervention de l'utilisateur.

I.4.6.1La Reformulation manuelle

Cette approche est associée aux systèmes de recherche booléens. On peut procéder à la Reformulation de requête en utilisant un vocabulaire contrôlé (thésaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes pour compléter sa requête [H.aliane].

I.4.6.2La Reformulation interactive

Dans une reformulation interactive, l'utilisateur joue un rôle actif. A l'inverse de la reformulation automatique, ici, ce sont le système et l'utilisateur qui sont, ensemble, responsables de la détermination et du choix des termes candidats à la reformulation. Le système joue un grand rôle dans la suggestion des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes. L'utilisateur examine cette liste et décide du choix des termes à ajouter dans la requête. C'est donc l'utilisateur qui prend la décision ultime dans la sélection des termes [H.aliane].

I.4.6.3La Reformulation automatique

La reformulation automatique de requêtes permet de générer une requête plus adéquate à la recherche d'information dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est de modifier la requête de l'utilisateurpar ajout de termes significatifs et/ou par réestimation de leur poids.

Cette reformulation intervient dans un processus plus général d'optimisation de la fonction de pertinence. Celle-ci a pour but de rapprocher la pertinence système de La pertinence utilisateur. Elle se présente comme une opération primordiale dans un SRI.

I.5Les principaux modèles de RI

La première fonction d'un système de recherche d'information est de mesurer la pertinence d'un document vis-à-vis d'une requête. Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de cette mesure de pertinence.
De façon générale, les modèles de RI peuvent être classés en trois modèles standard et les modèles de langue qui sont :

I.5.1 Modèle booléen (Boolean Model)

C'est le premier modèle utilisé en RI [Salton 71] ; il est basé sur la théorie des ensembles et l'algèbre de Boole [Gessler 93]. Le modèle booléen propose la représentation d'une requête sous forme d'une équation logique. Les termes d'indexation sont reliés par des connecteurs logiques *ET*, *OU* et *NON*.
L'approche booléenne consiste à trouver les documents qui ont exactement les mêmes termes qu'une requête construite par mots clefs. Les requêtes peuvent être affinées grâce aux opérateurs OR ou AND ou encore au moyen d'opérateurs comme NEAR .Ce type de recherche est la base des moteurs de recherche comme Altavista ou Google.

- **Les avantages du modèle booléen :**

- Le modèle est plus facile à implémenter et nécessite relativement peu de ressources [SALT90].
- Le langage de requête booléen est plus expressif que celui des autres modèles [CROF87].
- Ce modèle convient aux utilisateurs sachant exactement leurs besoins et en mesure de les formuler précisément avec le vocabulaire qu'ils maîtrisent parfaitement.

- **Les inconvénients du modèle booléen :**

- Il est difficile aux novices de formuler une requête combinant plusieurs opérateurs logiques, notamment pour les questions complexes. L'importance relative des mots clés ne peut pas être exprimée.
- Le classement des documents extraits par ordre de pertinence est difficile.
- La reformulation automatique des requêtes par la technique du RF est plus ardue.

I.5.2 Modèle vectoriel (VectorSpace Model)

Après le modèle booléen, le modèle qui a le plus influencé la recherche d'information est le modèle vectoriel qui a été créé au début des années 1970 par Gérard Salton et son équipe [SAL71], [SAL 83] dans cadre du projet SMART.
Dans ce type de modèle, les documents et les requêtes sont représentés dans un espace vectoriel engendré par l'ensemble des termes d'indexation $t_1, t_2,\ldots,t_N$
Où N est le nombre total de termes issus de l'indexation de la collection des documents.Chaque document est représenté par un vecteur :
$D_j=(d_{1j}, d_{2j},\ldots d_{ij}\ldots,d_{Tj})$.
Chaque requête est représentée par un vecteur :
$Q= (q_1, q2,\ldots q_i\ldots, q_T)$.

Où :

d_{ij} : Poids du terme t_i dans le document D_j
q_i : Poids du terme t_i dans le requête Q.

Les termes de poids nul représentent les termes absents dans un document alors que les poids positifs représentent les termes assignés.
La fonction de calcul du coefficient de similarité entre chaque document $D_j$et la requête est calculé sur la base d'une fonction qui mesure la colinéarité des vecteurs documents et requête. On peut citer notamment les fonctions suivantes :

Produit scalaire	$$RSV(Q,Dj = \sum_{i=1}^{T} qi * dij$$
Produit de jaccard	$$RSV(Q,Dj) = \frac{\sum_{i=1}^{T} qi * dij}{\sum_{i=1}^{T} qi^2 + \sum_{i=1}^{T} dij^2 - \sum_{i=1}^{T} qi * dij}$$
Mesure de cosinus	$$RSV(Q,Dj) = \frac{\sum_{i=1}^{T} qi * dij}{(\sum_{i=1}^{T} qi^2)^{1/2} * (\sum_{i=1}^{T} dij^2)^{1/2}}$$

Tableau1 : la colinéarité des vecteurs documents et requête

- **Les avantages du modèle vectoriel :**

  Il est possible d'assigner une pondération aux termes d'une requête.
  Le coefficient de similarité permet de :
  - classer les documents par ordre de pertinence,
  - déterminer le degré de similarité requête - document, document - document, phrase -phrase, etc...
  Certains résultats de recherche tendent à prouver que les systèmes de recherche vectoriels sontplus performants que les systèmes de recherche booléens [TURT94].

- **Les inconvénients du modèle vectoriel :**

  Ce modèle a les inconvénients suivants :
  -Les termes sont indépendants ; qui n'est pas toujours le cas,
  - Requêtes et documents sont essentiellement similaires alors que certains résultats produits par le calcul de similarité requête document ne reflètent pas la réalité.

I.5.3 Modèle probabiliste(Probabilistic Model)

Le modèle de recherche probabiliste utilise un modèle mathématique fondé sur la théorie de la probabilité. Le processus de recherche se traduit par calcul de proche en proche, du degré ou probabilité de pertinence d'un document relativement à une requête. Pour ce faire, le processus de décision complète le procédé d'indexation probabiliste en utilisant deux probabilités conditionnelles :

- $P(w_{ji}/Pert)$ **:** Probabilité que le terme t_i occure dans le document $D_j$sachant que ce dernier est pertinent pour la requête
- $P(w_{ji}/NonPert)$**:** Probabilité que le terme t_i occure dans le document *Dj* sachant que ce dernier n'est pas pertinent pour la requête.

Le calcul d'occurrence des termes d'indexation dans les documents est basé sur l'application d'une loi de distribution (type loi de poisson) sur un échantillon représentatif de documents d'apprentissage.

En posant les hypothèses que :

- La distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents
- Les variables « documents pertinents », « document non pertinent » sont indépendantes, la fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document *D*, notée *P (Pert/D)* [Risjbergen, 79] :

$$\boldsymbol{P(Pert/D) = \frac{P(D/pert) * p(pert)}{p(D)}} \qquad \boldsymbol{I.6)}$$

$$P(NonPert/D) = \frac{P(D/Nonpert) * p(Nonpert)}{p(D)} \qquad I.7)$$

L'ordre des documents est basé sur l'une des deux méthodes :

- Considérer seulement les termes présents dans les documents et requêtes.
- Considérer les termes présents et termes absents dans les documents et requêtes.

La similitude entre requête et document est calculée comme suit :

$$RSV(Q_K, D_J) = C \sum_{i=1}^{T} q_{Ki} d_{ji} + \sum_{i=1}^{T} f_{ij} q_{Ki} d_{ji} \log \frac{N.n_i}{n_i} \qquad I.8)$$

Où :

$$f_{ij} = \frac{tf_{ij}}{\max tf_j}$$

C : est une Constante.

De manière générale, le modèle probabiliste présente l'intérêt d'unifier les représentations des documents et concepts.

- **Les avantages du modèle probabiliste :**

Selon Savoy[SAV094],le modèle de recherche probabiliste est plus efficace que le modèle de recherche booléen, mais moinsperformant que le modèle de recherche vectoriel.

- **Les inconvénients du modèle probabiliste :**

Il n'existe pas de méthode d'estimation de la pertinence des termes avant toute extraction de document pertinent. Cette estimation se fait à posteriori.

I.5.4 Modèle de langue

Généralement, dans les modèles de langue, la pertinence d'un document face àune requête est en rapport avec la probabilité que la requête *Q* puisse être généréepar le modèle de langue du document *MD* (Ponte et al., 1998). Le score de pertinencedu document *D* face à une requête *Q* vue comme une suite de mots *Q = t1t2…tn* est déterminé par :

$$Score\ (Q, D) = P(Q|MD) = P\ (t1t2…tn|MD)\ I.9)$$

Pour estimer ce score, il faudrait que les mots $t_i$soient *indépendants* ainsi :

$$Score\ (Q, D) = \prod_{i=1}^{n} (t_i / M_d) \quad I.10)$$

Dans notre travail, nous utilisons le modèle de langue comme cadre pour implémenter notre approche, ce modèle est détaillé dans le chapitre II.

I.6 Évaluation de Système de recherche d'information

La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses idéales que l'utilisateur espère recevoir. Plus les réponses du système correspondent à celles que l'utilisateur espère, mieux est le système. L'évaluation constitue donc une étape importante lors de la mise en œuvre d'un modèle de recherche d'information puisqu'elle permet de paramétrer le modèle,d'estimer l'impact de chacune de ses caractéristiques et enfin de fournir des éléments de comparaison entre modèles.

I.6.1 Les mesures Précision/Rappel

Un système de recherche d'information est jugé sur sa capacité à distinguer entre les documents pertinents et non pertinents en réponse à une requête utilisateur. Un document donné peut être pertinent pour un utilisateur et ne pas l'être un autre.

Les mesures d'évaluation que nous détaillons ci-après ont été introduites dans le but d'évaluer au mieux les performances des SRI et de pouvoir par la suite les comparer.

Il est à noter que pour réaliser une comparaison valable, les mesures d'évaluation doivent être réalisées sur un même fond documentaire.

1. **La précision :** est la proportion des documents pertinents renvoyés par le système par apport à tous les documents restitués par le SRI. Autrement dit, la précision mesure la capacité du système à rejeter tous les documents non pertinents à une requête. La mesure précision est formulée ainsi :

$$Précision = \frac{Ra}{A}$$

Où :

Ra : est le nombre de document pertinents renvoyés par le système.
A : est le nombre de documents renvoyés par le système

2. **Le rappel :**mesure la capacité du système de retrouver tous les documents pertinents répondant à une requête.La mesure rappel est formulée ainsi :

$$Rappel = \frac{Ra}{R}$$

Où :

R *:* est le nombre de documents pertinents de la collection.

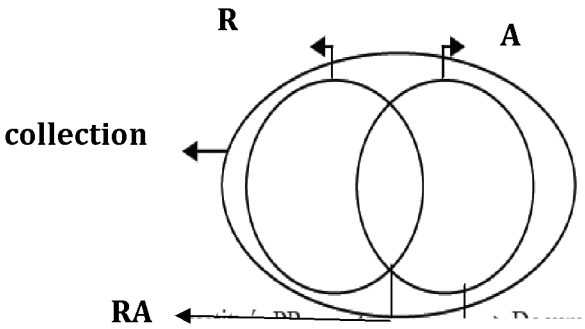


Figure 2 :Exemple de rappel et de précision pour une requête

- ✓ Le bruit représente les documents non pertinents retrouvés par le système, il est égale à : A-Ra
- ✓ Le silence constitue les documents pertinents non retrouvés par le système, il est égale à : R-Ra

La courbe de Précision-Rappel :

Dans le cas d'un système idéal, le taux de précision est égal au taux de rappel, c'est-à-dire que, tous les documents pertinents dans ce cas, et que ceux-ci, sont sélectionnés. On aurait donc une droite (à 1).

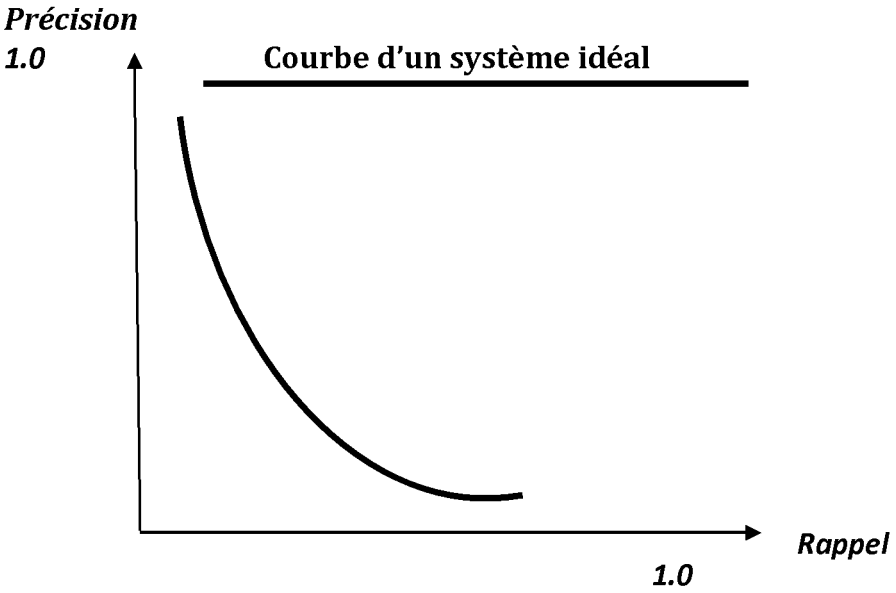


Figure3 : Allure d'un courbe rappel et précision [BAZ, 05]

L'inconvénient principal des mesures de rappel et de précision est qu'elles représentent des aspects différents de l'ensemble des documents retrouvés.

I.6.2 Autre mesures de performance

Il existe aussi d'autres mesures d'évaluation de performance des SRI telles que :

– *Le temps de réponse acceptable*: un SRI doit pouvoir fournir à l'utilisateur les documents correspondants à sa demande dans des temps très courts.
– *La présentation des résultatsclaire avec facilité d'utilisation* : capacité du système à comprendre les besoins de l'utilisateur et à mettre en valeur les documents correspondants à ceux-ci. Ceci est lié à l'interface avec l'utilisateur.
– *Le nombre total de documents pertinents retournés*, ou le rappel à 1000 documents : ces mesures permettant d'évaluer la performance globale du système au final. En fonction ou non du nombre de documents pertinents total.
– *Le rang du premier document pertinent*: cette mesure a été proposée pour prendre en compte la satisfaction de l'utilisateur qui chercherait un seul document pertinent (comme c'est éventuellement le cas pour les moteurs de recherche sur Internet).
– *La longueur de recherche*: elle est égale au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre n de documents pertinents.

I.6.3 Corpus de test

Pour arriver à une telle évaluation, on doit connaître 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:

- un ensemble de documents.
- un ensemble de requêtes.
- La liste de documents pertinents pour chaque requête (jugements de pertinence).

Pour qu'un corpus de test soit significatif, il faut qu'il possède un nombre de documents assez élevé. Pour la construction d'un corpus de test, les jugements de pertinence constituent la tâche la plus difficile. [Jian-Yun, 01].

Différentes collections de test sont utilisées en recherche d'information, parmiselles nous citons :

I.6.3.1 Les collections TREC

Le projet TREC est un programme international initié au début des années 90 par le NIST (National Institute of Standards and Technology) et du DARPA (Defense Advanced Reserach Projet Agency). Ce programme offre des moyens homogènes d'évaluation des systèmes de recherche d'information. Il est devenu la référence en recherche d'information pour diverses raisons. En effet, il a permis de définir les tâches en recherche d'information et de construire de larges collections de test.

Dans ce qui suit, nous allons définir les différents éléments qui constituent le projetTREC. Parmi ces éléments : les tâches, les participants, la source d'information etenfin la structure et le principe de construction de la collection TREC.

- **Tâches :**
  L'objectif est de permettre l'évaluation d'approches spécifiques en recherche d'information concernant le filtrage, le croisement de langues, la recherche dans de très large corpus (100 giga octet et plus), les modèles d'interactions.

- **Les participants** :
  25 groupes ont participé à la première édition de TREC en 1992 et 66 groupes de 16 pays différents ont également participé à TREC8.

- **Source d'information :**
  Les documents de la collection sont issus de la presse écrite en 1999 (Financial Time, Résumés de publication USDOE, SAN jose Mercury news, etc.). 5 CD-ROM contenant environ 5 giga octets de données textuelles et 450 besoins d'information (requêtes) sont déjà disponibles.

- **Structure et principe de construction de la collection :**
  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.

I.7 Conclusion

Ce premier chapitre a porté essentiellement sur l'étude des SRI de manière générale, nous avons présenté l'architecture de systèmes de recherche d'information (SRI) notammentl'appariement document/requête et les étapes essentielles d'une bonne indexation et enfin les différents modèles et stratégies utilisés lors de la mise en œuvre d'un SRI.

La reformulation de requête permet l'amélioration des performances des SRI et vu qu'elle représente le fond du sujet traité dans ce mémoire, nous allons présenter dans le chapitre suivant les principales approches développées en reformulation de requêtes particulièrement dans le cadre de modèle de langue.

Chapitre II :

Reformulation de requête

II.1 Introduction

Les performances d'un SRI, mesurées en général par la double mesure rappel précision, dépendent d'une part de l'efficacité du modèle de recherche mis en œuvre pour l'appariement des requêtes-documents, et d'autre part des requêtes formulées par l'utilisateur. En effet, l'utilisateur formule son besoin en information par une requête composée de ses propres mots clés et le choix de chaque terme à une influence directe sur l'ensemble des documents restitués par le système. Le plus souvent, l'utilisateur formule ses requêtes avec des termes qui lui sont propres, mais qui ne correspondent pas forcément à ceux utilisés pour indexer les documents pertinents des collections interrogées. Pour sélectionner le maximum de documents pertinents tout en limitant le bruit, il faudrait alors que l'utilisateur puisse choisir les termes utilisés comme index. Cette tâche s'avère difficile dans la mesure où il est impossible de connaitre le langage d'indexation utilisé et ou le nombre de termes indexés est généralement très grand. De plus, l'indexation et en particulier son exhaustivité, à également une incidence directe sur la qualité des réponses du système de recherche. De ce fait, retrouver les informations pertinentes en utilisant seulement la requête initiale de l'utilisateur est une opération quasi-impossible.

La reformulation de requête est une des stratégies qui permet d'améliorer la construction d'une requête. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin.

Comme nous l'avons vu en chapitre I, la reformulation de requête est une phase du processus de RI, dans ce chapitre, nous allons présenter ces différents techniques nous citons les outils de base ensuite nous détaillons les approches de reformulation de requête utilisées dans chacun des modèles standard (vectoriel et probabiliste) et dans les modèles de langue, comme notre travail s'insère dans le cadre de ce dernier modèle, nous présentons en détails les différents approches développées dans ce cadre.

II.2 La reformulation de la requête de l'utilisateur

II.2.1Définition

La reformulation de requêtes est proposée comme une méthode élaborée pour la recherche d'information s'inscrivant dans la voie de conception des SRI adaptatifs aux besoins des utilisateurs. C'est un processus permettant de générer une requête plus adéquate à la recherche d'information dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est de modifier la requête de l'utilisateur par ajout de termes significatifs et/ou ré estimation de leur poids.

La dimension de l'espace de recherche étant élevée, la difficulté fondamentale de la reformulation de requête est alors la définition de l'approche à adopter en vue de réduire l'espace de recherche par la détermination de [Efthimiadis, 1996] :

1. critères de choix des termes de l'expansion,
2. règles de calcul des poids des nouveaux termes,
3. hypothèse de base quant aux liens entre termes et documents.

II.2.2 le processus de Reformulation

- Chaque terme de recherche dans la requête initiale représente un mot clé sur lequel l'utilisateur veut ou non l'information. Le désir de l'utilisateur est représenté par les notions de « mot-clé positif » et « mot-clé négatifs » selon qu'il veut ou non de l'information sur un terme donné.
- Les termes initiaux de la recherche de l'utilisateur sont la meilleure indication de ses centres d'intérêt.
- Quelques termes de la base de connaissances peuvent être utiles.
- Le système ne doit jamais éliminer les mots-clés pour lesquels l'utilisateur a indiqué un intérêt.

II.2.3 Les outils de base

Les techniques de reformulation de requêtes ont généralement recours à l'utilisation de techniques de classification et du thesaurus.

II.2.3.1 La classification

La classification découpe l'espace des documents en sous-espaces homogènes appelés classes [Salton&MacGill, 1983] [Risjbergen, 1979] [Aboud, 1990]. 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.
Plusieurs stratégies de classification ont été proposées; nous présentons brièvement les techniques basées sur les attracteurs de groupes, similarité de documents et pertinence par rapport à une requête.

a) Classification par choix d'attracteurs de groupe

Le principe de classification consiste, dans ce cas, à choisir un document attracteur pour chaque groupe de documents. Un document est rattaché au groupe dont l'attracteur est le plus similaire. Dans [Blosseville& al, 1992], les pôles attracteurs sont déterminés préalablement par échantillonnage de documents classés par un expert. La même approche a été étudiée par Lewis [Lewis & Ringuette, 1994] en utilisant les arbres de décision. Les documents attracteurs peuvent également être choisis de manière aléatoire [Razouk, 1990] ou selon des algorithmes basés sur le contenu de la collection [Can & Ozkarahan, 1990].

b) Classification hiérarchique

Cette technique est basée sur le calcul d'une matrice de similitude entre documents [Salton&MacGill, 1983]. Dans la stratégie de classification avec un seul passage, on construit, à partir de la matrice de similitude, un graphe de classement où les nœuds représentent des documents. Deux sommets sont reliés par une arête si le degré de ressemblance entre documents correspondants est supérieur à un seuil établi. La décomposition du graphe obtenu en classes, utilise des techniques liées à la théorie des graphes; on citera notamment les définitions suivantes [Aboud, 1990] :

- Une classe est une composante connexe du graphe. Une classe forme un groupe de sommets dans lequel chaque sommet est connecté à tous les autres.

- Une classe est une étoile du graphe. Une étoile est un ensemble de sommets tel qu'il existe un sommet central connecté à tous les autres.

c) Classification basée sur la pertinence des documents

Une méthode de classification adaptative a été introduite dans [Yu& Chen, 1985]. A l'origine, on associe à chaque document une coordonnée aléatoire sur un axe réel; les coordonnées des documents pertinents pour une requête sont ensuite modifiées en vue de les rapprocher les uns des autres. Dans le but d'éviter la concentration de documents, le centroïde de ces documents est éloigné de celui de la collection. Raghavan&Deogun [Raghavan&Deogun, 1986] ont également développé une méthode de description de classes basée sur la description des documents pertinents aux requêtes. L'originalité de leur approche est de définir les classes de documents pertinents par fusions progressives de documents jugés pertinents mais éloignés des requêtes en cours. Les auteurs ont mis au point des heuristiques basées sur des calculs statistiques de distribution des termes dans la collection et dans les classes afin de maintenir un équilibre entre leurs tailles.

II.2.3.2 Thesaurus

Un thesaurus est un ensemble de termes formels, auxquels peuvent être associées des définitions permettant de représenter des connaissances. Ces définitions permettent de poser des contraintes sur l'utilisation des termes. Elles permettent ainsi d'effectuer des vérifications syntaxiques ou sémantiques. Autrement dit, elles permettent de préciser le contexte d'utilisation du terme, de guider l'association de certains termes avec d'autres termes et d'indiquer également des relations possibles entre termes.
Un thesaurus est une structure qui contrôle les complexités de la terminologie dans un langage. Il fournit aussi des relations conceptuelles idéales à travers la classification [Soergel 1997]. Plus particulièrement, dans un thesaurus multilingue, la relation d'équivalence inclue dans l'ensemble des termes choisis comme représentant du concept, la traduction et les synonymes de ces termes. La relation d'association permet d'améliorer la traduction des expressions. En effet, au lieu d'effectuer une traduction mot à mot en considérant les termes comme indépendants, il faut d'abord chercher à les relier par des relations d'associations pour obtenir la traduction exacte d'un concept multi terme.

II.2.4 La reformulation automatique

La reformulation automatique de requête induit un processus d'expansion et/ou repondération de la requête initiale en utilisant des critères de choix définis sans intervention de l'utilisateur. Ce type de reformulation peut être défini dans un contexte global, basé sur le thésaurus, ou alors local, basé sur les résultats de la recherche en cours.

II.2.4.1Reformulation basée sur le contexte global

Cette stratégie de recherche fait référence à l'exploitation d'informations préalablement établies dans la collection, et non dépendantes de la recherche en cours, en vue de réaliser la reformulation. Ceci fait alors appel essentiellement à l'utilisation de thesaurus.

a) Utilisation d'un thesaurus manuel

Le principe fondamental est d'ajouter à la requête initiale, les termes voisins définis Dans le thesaurus et sélectionnés par l'application d'un seuil et d'un algorithme de choix.

[Suy& Lang, 1994] proposent une expansion de requête basée sur l'utilisation du thesaurus manuel de Roget. La recherche d'information est effectuée selon les Principales étapes suivantes :

1. Expansion de requête en utilisant les liens sémantiques prédéfinis dans le thesaurus. Plus précisément, la requête utilisateur Q_k est étendue avec les termes de l'ensemble défini comme suit :

$$C^+ = \bigcup_{t_i} Q_k \qquad II.1)$$

Où :

$C^+_i = \{t_j \in C_i / (t_{kj} \neq 0) \cap (C_j = C_i)\}$, Ci : Catégorie de Roget du terme t_i.

En fait, on intègre à la requête utilisateur l'ensemble des termes qui traduisent la couverture sémantique de chacun de ses termes

2. Calcul de pertinence des documents selon un mécanisme d'activation propagation basé sur le modèle connexionniste.

b) Utilisation d'un thesaurus automatique basé sur la similarité

Qiu & Frei [Qiu & Frei, 1993] proposent une expansion de requête basée sur un thesaurus construit de façon automatique, modélisant des liens de similarité entre termes. A chaque terme t_i, on associe un descripteur vectoriel

$$\vec{t_i}(d_{i1} \ \ldots, d_{iN})$$

Avec, d_{ik} signifie le poids du document d_k par apport au terme t_i tel que k=1..n est le nombre de documents dans la collection.

La formule de pondération utilisée par Qui et Frei [Qui et Frei, 1993] pour calculer le poids d_{ik} est la suivante :

$$d_{ji} = \frac{(0.5+0.5*\frac{f_{ji}}{Max\, f_{ij}})}{\sqrt{\sum_{j=1}^{n}(0.5+0.5*\frac{f_{ji}}{Max\, f_{ij}})^2 itf_j^2}} \qquad II.2)$$

Où :

f_{ji} : fréquence de terme t_i dans le document D_j

itf_j : fréquence inverse du document D_j.

L'expansion de requête est alors effectuée selon les étapes suivantes :

1. Représentation sous forme vectorielle la requête initiale

$$\vec{Q_K} = \sum_{ti \in Qk} q_{ki} \vec{t_i} \qquad II.3)$$

2. Utiliser le thesaurus pour calculer la similarité entre chaque terme du thésaurus et la requête Q

$$Sim(\vec{Q_k}, \vec{t_j}) = Q_k t_j = \sum_{ti \in Q_k} q_{ki} C_{ij} \qquad II.4)$$

3. Ajouter à la requête les r top termes t_s sélectionnés par Sim (Q_k, Ks). A chaque terme ajouté t_a on utilisa un poids donné par :

$$q_{ai'} = \frac{Sim(Q_k, t_a)}{\sum_{ti \in Qk} q_{ki}} \qquad II.5)$$

Les expérimentations réalisées sur *3* collections de test standards montrent un accroissement de l'ordre de *20%* relativement à la Baseline [Yates & Neto, 1999].

Le modèle vectoriel généralisé est considéré comme une généralisation de cette technique, en ce sens que la principale différence est l'utilisation restreinte des *r* top termes pour l'expansion [Yates & Neto, 1999].

c) Utilisation d'un thesaurus basé sur le contexte

L'idée essentielle est d'étendre une requête par intégration de termes de même contexte que ceux qui la composent. Dans ce cadre, les approches différent principalement relativement au principe adopté pour la définition d'un contexte de mot.
Une expansion de requête basée sur l'utilisation d'un thesaurus organisé en classes définissant des contextes, a été proposée par [Carolyn & Yang, 1992]. Les travaux présentés proposent l'application d'un algorithme de classification pour l'organisation de documents ; les termes d'expansion sont issus d'un thesaurus constitué des termes de faible fréquence associés à chaque requête. La sélection des termes à ajouter, est basée sur le poids de la classe calculé par la formule :

$$W_c = \frac{w_{tc}}{|C|} \qquad II.6)$$

Où :

$|C|$: cardinale de la classe C.
w_{tc}: poids du terme t dans la classe C.

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…etc. Ces paramètres sont en outre très variables en fonction des collections interrogées [Yates & Neto, 1999].

L'expansion de requête proposée par [Gauch&Wang, 1996] est effectuée comme suit:

1. Construction préalable du thesaurus de contexte de termes ;
2. Calcul de similarités vecteur contexte – requête ;
3. Ajout des n top termes dont la valeur de similarité avec la requête est supérieure à un seuil déterminé.

Cependant, les résultats d'expérimentations n'ont pas montré un accroissement significatif de la précision lié à l'utilisation de la requête étendue.

L'approche proposée dans [Jing&Tzoukermann, 1999] est basée sur la distance contextuelle et la proximité morphologique entre termes. La principale motivation pour l'intégration de ces deux aspects dans le modèle, est que la corrélation basée sur la morphologie d'un mot fait augmenter le rappel alors que la corrélation basée sur le sens fait augmenter la précision.

II.2.4.2*Reformulation basée sur le contexte local*

Dans le cas de cette stratégie de recherche plus connue sous l'expression anglaise « adhoc 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 [Attar &Fraenkel, 1977].
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 [Xu &Croft, 1996] combine les atouts de l'analyse globale et analyse locale en procédant comme suit :

1. Identification des *n* tops passages de documents par appariement vectoriel avec la requête;
2. Pour chaque concept identifié, calculer :

$$\mathbf{Sim(Q_k,t_i)} = \prod_{t_j \in Q_k} \left(\delta + \frac{log\left(f\left(t_{i,t_j}\right) * idf_i^{idfj}\right)}{log(Ns)}\right) \qquad II.7)$$

Où :

N_s : nombre de top documents sélectionnés.
idf_j : fréquence inverse du terme t_j.
δ : constante.

II.2.5*La reformulation par réinjection de pertinence*

La reformulation de requête *RF* par injection de pertinence est la première technique développée pour s'adapter aux besoins de l'utilisateur. C'est L'une des stratégies de reformulation de requêtes est celle qui est dirigée par l'utilisateur.
Le principe de cette stratégie est de construire une nouvelle requête à partir de la structure des documents jugés par l'utilisateur : c'est ce que l'on appelle la réinjection de pertinence « relevance feedback » [Rochio 71] [Harman 92] [Boughanem98]. C'est un processus évolutif et interactif. Son principe fondamental est d'utiliser la requête initiale pour amorcer la recherche, puis modifier celle-ci à partir des jugements de pertinence et/ou de non-pertinence de l'utilisateur dans le but de ré pondérer les termes de la requête initiale, ou y ajouter (respectivement supprimer) d'autres termes contenus dans les documents pertinents (respectivement non pertinents).
La nouvelle requête obtenue à chaque itération de feedback, permet de corriger la direction de la recherche dans le sens des documents pertinents.
En effet, la simple comparaison du contenu de la requête et des documents de la base ne permet pas d'avoir tous les documents correspondant à une requête donnée.
Il reste toujours des documents pertinents non restitués, car ne contenant pas les termes de la requête.

II.2.5.1 La réinjection de pertinence dans les modèles classique de RI

a)La réinjection de pertinence dans le modèle vectoriel

Dans le modèle vectoriel, la requête et les documents sont représentés sous forme vectorielle, la réinjection de pertinence consiste à rapprocher le vecteur requête à ceux des documents pertinents et l'éloigner des documents non pertinents.
La nouvelle requête Q_{t+1} est construite grâce à la *formule de Rocchio* [Rocchio 71] dont l'idée est de dériver itérativement le vecteur requête optimal à partir d'opérations sur les vecteurs documents pertinents et les vecteurs documents non pertinents.
Le vecteur de la nouvelle requête est construit comme suit :

$$\boldsymbol{Q_{t+1} = \alpha.Q_t + \frac{\beta}{n}\sum n_p\, D_p^{(t)} - \frac{\gamma}{n_{np}}\sum n_{np}\, D_{np}^{(t)}} \qquad \boldsymbol{II.8)}$$

Où :

Q_{t+1} : vecteur de la nouvelle requête,
Q_t : vecteur de la requête initiale,
$D_p^{(t)}$(resp : $D_{np}^{(t)}$): vecteur d'un document pertinent (resp. non pertinent),
N_p(resp : n_{np}) : nombre de documents jugés pertinents (resp. non pertinents),
α, β, γ:constantes.

Il est également possible de simuler l'interaction d'un utilisateur en postulant que les dix premiers documents trouvés par une première recherche sont pertinents et les suivants sont non pertinents (*pseudo relevance feedback*).

b) La réinjection de pertinence dans le modèle probabiliste

Sur la base du modèle probabiliste, Robertson et Sparck-Jones [Robertson 76] ont développé une formule de pondération des termes basée sur la distribution des termes de la requête dans les documents jugés pertinents et les documents jugés non pertinents par l'utilisateur. Cette formule est la suivante :

$$w_i = \frac{\frac{r_i}{R - r_i}}{\frac{n_i - r_i}{(N - n_i) - (R - r_i)}} \qquad II.9)$$

Où :

w_i : poids du terme t_i dans la requête.

R :nombre de documents pertinents pour la requête.

r_i : nombre de documents pertinents contenant le terme t_i.

N : nombre de documents dans la collection.

n_i : nombre de documents contenant le terme t_i.

Croft [Croft 83] a défini une méthodologie de repondération en utilisant une version révisée de la formule de pondération de Sparck-Jones :

- Dans la phase recherche initiale la formule suivante est utilisée pour pondérer les termes **:**

$$w_{ijk} = (C + idf_i) f_{tk} \qquad II.10)$$

- Dans la phase reformulation le poids des termes est mesuré comme suit:

$$w_{ijk} = \left[C + log \frac{p_{ij}\ (1 - q_{ij})}{q_{ij}\ (1 - p_{ij})} \right] f_{tk} \qquad II.11)$$

Où :

w_{ijk} : le poids du terme t_i dans le requete Q_j et du document D_k,
idf_i : fréquence absolue du terme t_i dans la collection,
P_{ij} : probabilité que le terme t_i soit assigné à un ensemble de documents pour une requête Q_j,

$p_{ij} = \frac{r+0.5}{R+0.1}$ si $r > 0$;

$p_{ij} = 0.01$ si r=0.

Où :

Q_{ij} : probabilité que le terme t_i apparaisse dans un ensemble de documents non pertinents pour une requête Q_j :

$$q_{ij} = \frac{n - r + 0.5}{N - R + 10} f_{tk} = K + (1 - K).\frac{freq_{tk}}{max\ (freq_k)} \qquad II.12)$$

Où :

$freq_{tk}$: fréquence du terme dans le document K.

max $(freq_k)$: fréquence maximale d'un terme t_i dans le document

C, K : constantes.

II.2.5.2La technique de réinjection de pertinence sans jugements de l'utilisateur

L'approche alternative, appelée *pseudo* ou *Blind* RF emploie la technique de réinjection de pertinence automatiquement sans utiliser l'information issue des jugements de l'utilisateur.

Dans cette technique le système génère une liste trié de documents pour la premier requête initiale .Ensuite, il sélectionne un certain nombre de documents (petit ensemble) à partir de top documents du classement .Une nouvelle itération Rf sera exécutant on supposant que les documents sélectionnés sont pertinents. La nouvelle requête générée par le processus de reformulation de requête est utilisé pour produire une nouvelle liste triée de documents.

Le principe de base de *pseudo RF* est que : une itération feedback basée sur le premier document sélectionnés pondant l'exécution de la requête initiale, va engendre un meilleur classement pour les documents pertinents.

Selon Croft&Harper [Croft et Harper, 1979], cette technique souffre d'un problème majeur qui est appelé « *query drift* ». Ce problème apparaître lorsque l'ensemble des documents utilisés pour le feedback ne contient que peu de documents pertinents (ou ne contient pas de tout). Donc, cette technique ne fonctionne bien que dans le cas où la requête initiale permettre d'extraire de bons résultants (beaucoup de documents pertinents).

II.2.6 Paramètres de performance de la reformulation de la requête

Un nombre considérable d'expérimentations ont été effectuées sur les collections de documents pour l'évaluation de l'impact induit par la reformulation de requêtes sur le processus de recherche d'information. Ces expérimentations sont révélées en performance est en fonction d'un nombre considérable de paramètres.

Ces paramètres sont très dépendants de :

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

Nous évoquons dans ce qui suit les principaux paramètres :

II.2.6.1 Méthode de Sélection des termes

La sélection des termes ayant une valeur de poids importante revient à sélectionner les termes caractéristiques des documents pertinents avec une faible probabilité d'apparition dans les documents non pertinents.
Harman [79] a également démontré que la meilleure méthode de sélection des termes issus des documents pertinents devient inefficace au-delà de 20 à 40 termes ajoutés.
Croft et *al.* [77] et Robertson et *al.* [153] ont adopté une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer à chaque terme un nombre traduisant sa valeur.
Robertson propose la formule suivante pour calculer la valeur de sélection d'un terme :

$$\boldsymbol{selValue(i) = w_{ij} * (p_i - U_i)} \qquad \boldsymbol{II.13)}$$

Où :

w_{ij} : Poids du terme i dans la requête j,
p_i : La probabilité ($d_i = {}^{1}/_{D}$ *est pertinent*) ;
U_i : La probabilité $d_i = {}^{1}/_{D}$ *est non pertinent*.

Lundquist et al ont étudié une autre technique de tri des termes, Pour un terme *k,* les auteurs associent une valeur *pk x nidf* où *pk* est le nombre de documents dans l'ensemble des documents pertinents contenant le terme *k*, et *nidf* est une fréquence absolue inverse normalisée utilisant la normalisation telle que définie par Singhal.

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

Buckley et *al.* ont démontré que le taux de performance (Rappel-Précision) est davantage corrélé avec le nombre de termes ajoutés à la requête qu'avec le nombre de documents initialement retrouvés.

Cette idée est traduite par l'équation suivante :

$$RP(N) = A.\log(N) + B.\log(X) + C \qquad II.14)$$

Où :

RP(*N*) : la performance du système pour N documents restitués,
N :le nombre de documents restitués,
X : le nombre de termes ajoutés à la requête,
A, B, et C : des constantes telles que $B >> A > C$.

II.2.6.2 Longueur moyenne de requête

Les programmes d'évaluation considèrent généralement des requêtes composées de plusieurs sections qui permettent d'obtenir des représentations de requêtes variables, en fonction des sections considérées. Bien que la taille des requêtes soumises réellement aux systèmes puisse varier selon différents critères.

L'accroissement de performance de la relevance feedback est plus important lorsque les collections sont interrogées par les requêtes de longueur relativement petite.

II.2.6.3 Type de la collection

La relevance feedback est adaptée aux collections techniques, l'ensemble des documents pertinents est concentré plus réduit, produisant ainsi les meilleurs résultats.
Les travaux d'expérimentations effectués sur la reformulation de la requête, ont mis en évidence un nombre varié de paramètres qui conditionnent le taux de performance d'un SRI. Il en ressort que la nature et la valeur idéale de ces paramètres sont étroitement liées aux caractéristiques des bases des documentaires.

II.2.6.4 Cooccurrence des mots

La notion de cooccurrence fait référence au phénomène général par lequel des mots sont susceptibles d'être utilisés dans un même contexte [MS99]. Autrement dit, on considère qu'il y a cooccurrence lorsque la présence d'un mot dans un texte donné a une indication sur la présence d'un autre mot. Pour mieux comprendre cette définition, pensons à un exemple simple comme *«avion»* et *«aéroport»,* deux mots qui, selon toute vraisemblance, sont utilisés la plupart du temps dans un contexte commun. Afin de capturer la cooccurrence entre termes, plusieurs mesures d'associations sont utilisées, nous présentons ci-dessous les plus importantes :

- **Mesures d'association**

Dans le cadre de l'approche statistique du traitement de la langue naturelle, plusieurs façons d'attribuer un score d'association à une paire de mots qui ont été élaborées. Parmi ces mesures, certaines se basent sur de solides fondements théoriques, alors que d'autres relèvent plutôt du domaine de l'heuristique.

Certaines sont directement tirées de la discipline des statistiques alors que d'autres sont nées dans le domaine de la recherche d'information.

Généralement, les mesures d'association peuvent s'appliquer autant aux collocations qu'aux cooccurrences. À peu de choses près. Pour chacune des mesures qui seront présentées ici (et pour presque toutes les autres), le calcul se base sur les données du tableau 2, qui contient les fréquences d'apparition des paires de mots. C'est à partir des quatre valeurs présentes dans cette table que l'on va calculer le degré d'association entre deux mots : (*a*) le nombre de fois où les deux mots apparaissent ensemble, (*b*) le nombre de fois où le premier mot est présent sans que le deuxième ne le soit, (*c*) le nombre de fois où le deuxième mot est présent sans que le premier ne le soit, et (*d*) le nombre de fois où aucun des deux mots n'apparaît.

	Mot 2 présent	*Mot 2 absent*
Mot 1 présent	**A**	**B**
Mot 1 absent	**C**	**D**

Tableau 2 : Table de contingence pour mesurer le degré d'association entre deux mots

A. Test du χ^2

Une première façon d'assigner une valeur numérique au degré de cooccurrence entre deux mots est le test du χ^2 [MS99]. Cette mesure a été imaginée dans le cadre du problème classique en statistique qu'est la validation d'hypothèses.

L'idée générale de la mesure du χ^2 est de comparer les fréquences observées dans un corpus avec les fréquences attendues s'il y avait indépendance. Si la différence entre les deux est grande, on peut rejeter l'hypothèse d'indépendance. Basée sur le tableau 2, L'équation qui traduit cette notion est la suivante :

$$X^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \qquad II.15)$$

Où :

- i couvre les rangées du tableau 2 ;
- j couvre les colonnes du tableau 2 ;
- O_{ij} est la valeur observée pour la cellule (i , j) ;
- E_{ij} est la valeur attendue pour la cellule (i , j).

La statistique fait donc la somme des différences au carré entre les valeurs observées et attendues pour chaque cellule de la table, chaque différence étant normalisée par la valeur attendue. On peut la voir comme une sorte d'erreur quadratique moyenne. Les valeurs observées sont tout simplement calculées à partir des textes du corpus. Quant aux valeurs attendues, elles sont calculées à l'aide des probabilités marginales, c'est-à-dire à partir des totaux pour les lignes et pour les colonnes de la table, convertis en proportions. Par exemple, la probabilité attendue E_{11}, qui est la probabilité que le mot 1 et le mot 2 soient tous deux présents dans un texte, se calcule de la façon suivante : on multiplie la probabilité marginale que le mot 1 apparaisse avec la probabilité marginale que le mot 2 apparaisse, ainsi qu'avec le nombre de documents. Sachant que N représente le nombre total de documents, on peut faire les calculs suivants :

- Probabilité que le mot 1 apparaisse : $P_1 = (a + b) / N$
- Probabilité que le mot 2 apparaisse : $P_2 = (a + c) / N$
- Probabilité que le mot 1 et le mot 2 apparaissent tous les deux : $E_{11} = P_1 \cdot P_2 \cdot N$

Après quelques manipulations algébriques et simplifications, on arrive à l'expression suivante qui nous permet d'obtenir la valeur de χ^2 :

$$\boldsymbol{X^2 = \frac{N(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)}} \qquad \boldsymbol{II.16)}$$

B. Information mutuelle

L'information mutuelle (*«pointwise mutual information»* ou PMI) est une mesure qui est née de motivations provenant du domaine de la théorie de l'information. Elle mesure la quantité d'information apportée par la présence d'un mot au sujet de la présence d'un autre mot. L'équation suivante permet de calculer cette mesure et met en jeu, au numérateur, la proportion des documents où figurent les deux mots puis, au dénominateur, les proportions respectives des documents où figure un seul des deux mots.

$$\boldsymbol{I(mot1, mot2) = log_2 \frac{P(mot1 \& mot2)}{P(mot1)P(mot2)}} \qquad \boldsymbol{II.17)}$$

Intuitivement, il est facile de voir que plus deux mots ont tendance à apparaître ensemble, plus le score sera élevé. En effet, si les deux mots étaient totalement indépendants, le rapport des probabilités serait de 1, donnant donc une valeur finale de 0. À l'opposé, si les deux mots ont une forte tendance à apparaître ensemble, alors la probabilité au numérateur dépassera l'autre, faisant croître le score d'information mutuelle. Toujours à partir de la table de contingence et après certaines manipulations, on arrive à la façon simplifiée de calculer la PMI :

$$I(mot1, mot2) = log_2 \frac{Na}{(a+c)(a+b)} \qquad II.18)$$

L'information mutuelle est considérée comme étant une bonne mesure de l'indépendance entre deux mots. Sa valeur devient nulle si les deux mots sont parfaitement indépendants. Mais pour des paires de mots parfaitement dépendants (qui apparaissent toujours ensemble), la valeur de l'information mutuelle augmente plus ceux-ci sont rares, la raison étant que les probabilités au dénominateur diminuent. Ce phénomène est en fait un inconvénient lié à l'utilisation de cette mesure puisqu'il devient plus difficile de comparer les scores obtenus par des mots fréquents avec ceux obtenus par des mots rares.

C. Le coefficient de Dice

Une mesure également intéressante en termes d'évaluation de qualité est le coefficient de Dice (Smadja et al. (1996)), défini comme suit :

$$\mathbf{Dice(x,y)} = \frac{\mathbf{2 \times P(x,y)}}{\mathbf{P(x) + P(y)}} \qquad \mathbf{II.19)}$$

Qui devient :

$$\boldsymbol{Dice(x,y)} = \frac{\mathbf{2 \times nb(x,y)}}{\boldsymbol{nb(x) + nb(y)}} \qquad \boldsymbol{II.20)}$$

De faibles valeurs du coefficient indiquent des ruptures thématiques dans le texte alors que de fortes valeurs indiquent au contraire une cohérence thématique locale.
Cette méthode de segmentation n'est applicable qu'à des textes dans lesquels les termes significatifs des thèmes développés sont souvent réemployés tout au long du texte.

II.3 Modèle de langue

Une autre façon de modéliser les documents sous forme probabiliste réside dans l'utilisation des modèles de langue. Les modèles de langue fournissent une représentation d'un langage, Cette représentation peut être utilisée au sein de modèles de recherche d'information.

II.3.1 Principe général des modèles de langue

Par « modèle de langue », on désigne une fonction de probabilité *P* qui assigne une probabilité *P*(*s*) 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.

Considérons la séquence composée des mots suivants : $m_1, m_2, ..., m_n$. La probabilité $P(s)$ peut être calculée comme suit:

$$\boldsymbol{p(s) = \prod_{i=1}^{p} p(m_i \,|m_1 \dots m_{i-1})} \qquad \boldsymbol{II.21)}$$

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 $P(m_i/m_1...m_{i-1})$) à estimer, et ceci est souvent impossible de se 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 :

$$\boldsymbol{\prod_{i=1}^{p} p(m_i \,|m_1 \dots m_{i-1}) = \prod_{i=1}^{p} p(m_i|m_{i-n+1} \dots m_{i-1})} \qquad \boldsymbol{II.22)}$$

Par simplification des calculs, on fait l'hypothèse que seuls les n-1 mots précédents sont importants. Ainsi, on parle de modèle uni-grammes, bi-grammes ou tri-grammes :

Uni-gramme : $\boldsymbol{p(s) = \prod_{i=1}^{l} p(m_i)}$ $\qquad \boldsymbol{II.23)}$

Bi-gramme : $\boldsymbol{p(s) = \prod_{i=1}^{l} p(m_i|m_{i-1}) = \prod_{i=1}^{l} \frac{p(m_{i-1}m_i)}{p(m_{i-1})}}$ $\qquad \boldsymbol{II.24)}$

Tri-gramme : $\boldsymbol{p(s) = \prod_{i=1}^{l} p(m_i|m_{i-2}m_{i-1}) = \prod_{i=1}^{l} \frac{p(m_{i-2}m_{i-1}m_i)}{p(m_{i-2}m_{i-1})}}$ $\qquad \boldsymbol{II.25)}$

Pour estimer les probabilités, on utilise un corpus de documents représentatifs de la langue à modéliser. Si le corpus est suffisamment grand, il permet de faire l'hypothèse qu'il reflète, la langue en général et ainsi le modèle de langue correspond approximativement au modèle de langue pour le corpus. Le calcul de la probabilité d'un mot w dans un corpus C se base sur l'estimation de vraisemblance maximale du terme w, comme suit :

$$\boldsymbol{p_{ml}(w) = \frac{|w|}{\sum_{w_j \in c} w_j} = \frac{nombre\ d'occurences\ du\ terme\ w\ dans\ le\ corpus\ C}{le\ nombre\ de\ n-gramme\ de\ C}} \qquad \boldsymbol{II.26)}$$

- ✓ L'apprentissage du modèle de langue et l'évaluation de la probabilité d'appartenance d'un document à ce modèle (Figure 3). Ainsi, les paramètres du modèle de langue MLx s'estiment (2) en se basant sur les caractéristiques statistiques de langue extraites du corpus d'entraînement (1). A partir de ce modèle de langue MLx, la probabilité du document D d'appartenir à la langue X s'évalue par P(D | MLx) (3).

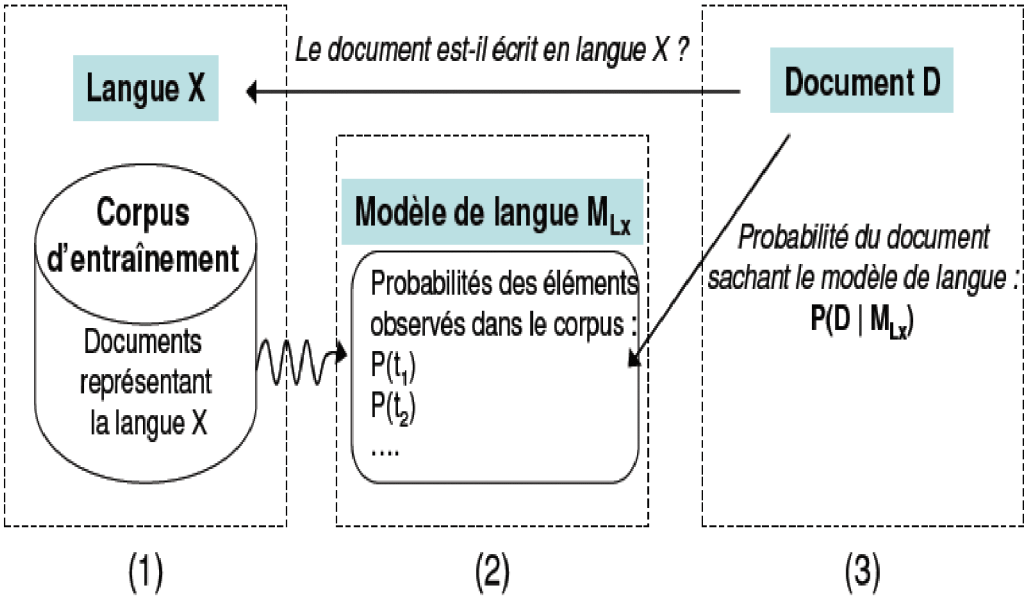


Figure 3 : Principe général des modèles de langue

II.3.2 Les techniques de lissage

Plus le corpus utilisé pour ces estimations est grand, plus on peut espérer obtenir des estimations de probabilité justes. Cependant, quelle que soit la taille du corpus d'entraînement, il ya toujours des mots ou des séquences de mots absents du corpus. Pour ces mots ou séquences de mots, leur estimation de probabilité est 0. La conséquence de cette probabilité nulle est qu'on attribuera une probabilité nulle à toute séquence de mots ou des phrases contenant un mot ou un n-gramme non rencontré dans le corpus. En d'autres termes, les modèles ainsi construits ne sauraient reconnaître que les phrases dont les n-grammes sont tous apparus dans le corpus. Ce sont donc des modèles très limités. Afin de généraliser les modèles, on voudrait assouplir cette attribution systématique de probabilité nulle aux mots ou séquences de mots non rencontrés. Cette procédure qui consiste à attribuer une probabilité non-nulle á ces éléments est appelée le lissage. Le lissage peut aussi être vu comme une façon d'éviter le surentraînement d'un modèle sur un corpus, et de doter du modèle d'une plus grande capacité de généralisation.

Le principe de lissage peut être résumé ainsi : au lieu de distribuer la totalité de masse de probabilité sur les n-grammes vus dans le corpus d'entraînement, on enlève une partie de cette masse et la redistribue aux n-grammes non vus dans le corpus. De cette façon, les n-grammes absents du corpus vont recevoir une probabilité non-nulle.
Sur la façon d'enlever et de redistribuer une partie de masse de probabilité, il y a une série de méthodes proposées dans la littérature, dont les plus courants ont présentées ci-dessous.

✓ **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) :

$$P_{ajouter-un}(\alpha|C) = \frac{|\alpha| + 1}{\sum_{\alpha_i \in V}(|\alpha_i| + 1)} \qquad II.27)$$

Si le corpus C ne contient qu'une petite partie des n-grams possibles (ce qui est souvent le cas), la plus grosse part de la masse de probabilité sera distribuée sur les n-grams non observés.

✓ **Lissage de 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 :

$$r^* = (r + 1)\frac{n_{r+1}}{n_r} \qquad II.28)$$

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

$$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_i \in C} |\alpha_i|} \qquad II.29)$$

- diminution de $f'(\propto)/f(\propto)$ du poids de∝, redistribué sur les ngrams non vus ;
- l'estimation Good-Turing pour les n-grams de grande fréquence est instable ;
- Recommandé pour les n-grams de faible fréquence.

✓ **Lissage Backoff(Katz)**

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 :

$$P_{Katz}(m_i|m_{i-1}) = \begin{cases} P_{GT}(m_i|m_{i-1}) Si\ m_{i-1}m_i \in C \\ \alpha(m_i)P_{Katz}(m_{i-1}) sinon \end{cases} \qquad II.30)$$

- la diminution de fréquence de l'estimation Good-Turing est redistribuée sur le modèle unigrams,

- $\alpha(m_{i-1})$ est un paramétre qui détermine la part de cette redistribution à m_i

$$\alpha(m_{i-1}) = \frac{1 - \sum_{m_j tq\ m_{i-1}m_j \in C} P_{GT}(m_j|m_{i-1})}{1 - \sum_{m_j} P_{ML}(m_j)} \qquad II.31)$$

✓ **Lissage par interpolation**

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 :

$$\mathbf{P_{JM}(m_i|m_{i-1}) = \lambda_{m_{i-1}} P_{ML}(m_i|m_{i-1}) + (1 - \lambda_{i-1}) P_{JM}(M_I)} \qquad \mathbf{II.32)}$$

$\lambda_{m_{i-1}}$Est un paramètre qui est établi par apprentissagedéterminé par processus EM (*Expectation Maximisation*).

✓ **Lissage Dirichlet**

Le lissage de Dirichlet est basé sur l'interpolation linéaire. Il permet l'augmentation des fréquences des n-grammes m_i dans un document. Présent comme suit :

$$P_\mu(m_i/D) = \frac{tf(m_i, D) + \mu P(m_i/C)}{\sum_{m_i} tf(m_i, D) + \mu} \qquad II.33)$$

Avec :

$tf(m_i, D)$Le nombre d'occurrence de n-grammes dans le document D.
$\sum_{m_i} tf(m_i, D)$=**|D|** la taille de document D.
μ est un paramètre appelé pseudo fréquence.
Le lissage de Laplace est un cas particulier de ce type de lissage.

II.3.3 Approches des modèles de langue

Dans les modèles traditionnels de RI, on tente de modéliser la relation de pertinence entre un document et une requête. Typiquement, dans le modèle probabiliste, on tente d'estimer la probabilité $P(R|D, Q)$ – la probabilité de pertinence d'un document face à une requête. Cette probabilité est calculée selon des méthodes paramétriques : on suppose que la distribution des mots suit une certaine norme (par exemple, distribution Poisson) parmi les documents pertinents (et non-pertinents). En fonction des distributions des mots parmi deux ensembles (pertinent et non-pertinent) de documents échantillons, on peut estimer les probabilités des mots pour la pertinence. La probabilité $P(R|D, Q)$ pourra alors être calculée.
Le principe des approches utilisant un modèle de langue est différent. On ne tente pas de modéliser directement la notion de pertinence dans le modèle; mais on considère que la pertinence d'un document face à une requête est en rapport avec la probabilité que la

requête puisse être générée par le modèle de langue du document. Ainsi, on considère qu'un document D incarne un sous-langage, pour lequel on tente de construire un modèle de langue M_D. Le score du document face à une requête Q est déterminé par la probabilité que son modèle génère la requête :

$$\mathbf{Score(Q, D) = P(Q|M_D)} \qquad \textit{II.34)}$$

On écrira aussi $P(Q|D)$ pour représenter la même probabilité dans les descriptions plus tard. De façon générale, une requête peut être vue comme une suite de mots : $Q = t_1 t_2 ... t_n$. Nous avons donc :

$$\textit{Score(Q, D) = P(}t_1 t_2 ... t_n|M_D\textit{)} \qquad \textit{II.35)}$$

Il est aussi possible de construire un modèle de langue pour le document $P(\bullet|M_D)$ et un autre pour la requête $P(\bullet|M_Q)$. Le score d'un document face à la requête peut être déterminé par une comparaison entre les deux modèles. C'est le principe utilisé dans la méthode d'entropie croisée.
Nous présentons dans ce qui suit les modèles les plus en vu :

II.3.3.1 Le modèle de Ponte et Croft

Le premier modèle à la RI basé sur la modélisation de langue est celui proposé par Ponte et Croft [Ponte98] ; L'intuition est qu'une requête n'est pas créée de façon aléatoire, mais l'utilisateur a une idée (un modèle) sur le document idéal qui peuvent apparaître dans D et de son modèle M_D. À partir de M_D, l'utilisateur choisit (génère) les termes pour constituer sa requête. Ainsi, la requête devrait être générée à partir d'un document pertinent ou de son modèle. La probabilité de cette génération est considérée comme le score du document :

$$\textit{Score (D,Q) = P(Q|}M_D\textit{)} \qquad \textit{II.36)}$$

Nous apportons quelques remarques ici :

Une requête peut être considérée comme une suite de mots : $Q = t_1, t_2, ... t_n$.

Comme dans la plupart des modèles de langues utilisées en RI, la simplification suivante a été faite : on suppose que les mots dans une requête sont indépendants.

Dans le modèle de Ponte et Croft, ils utilisent modèle le Bernoulli multiple, c'est-à-dire que non seulement les mots présents dans la requête, mais aussi ceux absents de la requête, sont pris en compte. Dans la plupart d'autres modèles, on utilise plutôt le modèle multinomial, où on ne considère que les mots présents dans la requête.
Supposons, sans perdre la généralité, que la requête Q est composée de l'ensemble de mots $t_1, t_2, ... t_n$, et que les mots $t_{n+1}, t_{n+2}, ... t_l$ sont absents de la requête. $P(Q|M_D)$ peut être ré-exprimée comme suit :

$$P(Q|M_D) = P(t_1 t_2 \ldots t_n|M_D) x P(\neg t1 \neg t2 \ldots \neg tn\ |M_D)$$

$$= \prod_{i=1}^{n} P(t_i|M_D) X \prod_{i=1}^{n} P(\neg t_i|M_D)$$

$$= \prod_{t_i \in Q} P(t_i|M_D) X \prod_{t_i \in Q} 1 - P(t_i|M_D)) \qquad II.37)$$

Ponte et Croft ont proposé une estimation de la probabilité $P_{PC}(t_i|M_D)$ d'une façon similaire à l'approche Backoff. Ils combinent un modèle de langue du document avec un modèle de langue du corpus comme suit :

$$P_{PC}(t_i|M_D) = \begin{cases} P_{ML}(t_i|D)^{1-R\hat{}(t_i,D)} X P_{avg}(t_i|D)^{1-R(t_i,D)} & Si\ tf(t_i|D) > 0 \\ P_{ML}(t_i|C) Sinon \end{cases} \qquad II.38)$$

où $tf(t_i, D)$ est la fréquence de t_i dans le document D

Dans cette formule, on note deux éléments principaux $P_{ML}(t_i|D)$ et $P_{ML}(t_i|C)$

Il y a aussi quelques autres éléments ajoutés pour tenir compte de certaines particularités de la RI : $P_{avg}(t_i)$ est la probabilité moyenne du terme t_i dans les documents qui le contiennent ; $R\hat{}(t_i, D)$ est une fonction de « risque »

Le fait de considérer les mots absents de la requête est intéressant : il permet de faire la différence entre un document qui couvre beaucoup de sujets et un autre document qui ne couvre que le sujet de la requête, donc l'aspect spécificité du document. Cependant, on peut facilement voir que si les termes qui n'apparaissent pas dans la requête sont nombreux (ce qui est le cas en général), le calcul devient très complexe.

II.3.3.2 Le modèle de Hiemstra

Hiemstra et al. [Hiemstra98] utilisent le même principe de génération de la requête par le document. La formule que Hiemstra propose est la suivante :

$$Score\ (D,Q) = P(D|Q) = P(D|t_1\ t_2 \ldots t_n)$$

$$= P(D) \frac{P(t1\ t2 \ldots tn|D)}{P(t1\ t2 \ldots tn)} \qquad II.39)$$

Il suppose que $P(t_1\ t_2 \ldots t_n)$ est un constante $1/C$, et que les mots dans sont indépendants. On a donc :

$$Score\ (D,Q) = cP(D) \prod_{t \in Q} P(t_i|D) \qquad II.40)$$

Pour P(t_i|D), Hiemstra utilise une approche d'interpolation :

$$\boldsymbol{P(t_i|D)= \propto P_{ML}(t_i|D) + (1-\propto)P_{ML}(t_i|C)} \qquad \boldsymbol{II.41)}$$

L'estimation de vraisemblance maximale de $P_{ML}(t_i|D)$ et $P_{ML}(t_i|C)$ sont respectivement Donnée comme suit :

$$\frac{\boldsymbol{tf(t_i,D)}}{\boldsymbol{|D|}} \text{et} \frac{\boldsymbol{df(t_i)}}{\boldsymbol{\sum_{t_j} df(t_j)}}$$

Où

$df(t_j)$est le nombre de documents contenant t_i,

V est le vocabulaire d'indexes.

Une fois on prend le logarithme, on obtient :

$$\boldsymbol{log\big(P(t_1 t_2 \dots t_n|D)\big) = log \prod_{t\in Q} P(t_i|D)}$$

$$\boldsymbol{= \sum_{i=1}^{n} log(\alpha P_{ML}(t_i|D) + (1-\alpha)P_{ML}(t_i|C))}$$

$$\boldsymbol{= \sum_{i=1}^{n} log\left(1 + \frac{\alpha P_{ML}(t_i|D)}{(1-\alpha)P_{ML}(t_i|C)}\right) + \sum_{i=1}^{n} log\big((1-\alpha)P_{ML}(t_i|C)\big)} \qquad \boldsymbol{II.42)}$$

On remarque que le dernier composant de cette formule ne dépend pas de document, mais seulement de la requête et le corpus C. Ainsi, c'est une constante que l'on peut ignorer pour la fin de classer les documents.

En ce qui concerne la valeur de α, la façon la plus simple consiste à déterminer une constante pour α. Mais en principe, ce paramètre peut bien dépendre du document ou de la requête. Ainsi, une façon plus sophistiquée consiste à estimer une valeur de α en utilisant un processus d'optimisation automatique telle la maximisation de l'espérance (EM).
La probabilité a priori d'un document *D* est estimée comme suit :

$$P(D)=\frac{|D|}{|C|}$$

En somme, on a :

$$\boldsymbol{\log \mathbf{Score}(Q,D) = \sum_{t\in Q} \log\left(1 + \frac{\alpha * \mathbf{tf}(t,D) \sum_{t_i\in V} \mathbf{df}(t_i)}{(1-\alpha)|D|\mathbf{df}(t)}\right) + \log\frac{|D|}{|C|}} \qquad \mathbf{II.43)}$$

Miller et al. [Miller98, Miller99] utilise une formulation similaire, à quelques détails près. La plus grande différence entre le modèle de Miller et celui de Hiemstra est la façon d'implanter le modèle.

Bien que Hiemstra et Miller aient tous utilisé le lissage par interpolation, leur intention initiale n'était pas pour traiter la claire semence de données, mais pour mieux modéliser la requête, dans laquelle certains mots peuvent être non-pertinents.

II.3.3.3 le ratio de vraisemblance

Ng [Ng99, Ng00] a proposé une autre variante de modèle de langue basée sur le ratio de vraisemblance. Au lieu d'estimer la probabilité de pertinence d'un document pour une requête, Ng propose d'utiliser la vraisemblance comme critère de classement.
L'idée qu'il avance est la suivante : Si la vraisemblance d'un document augmente après que la requête est soumis, le document a une plus forte probabilité d'être utile qu'un autre document dont la vraisemblance ne change pas ou décroît.

Ng propose la formule suivante pour mesurer cette quantité :

$$\boldsymbol{LR(D,Q) = \frac{P(D/Q)}{P(D)} = \frac{P(Q/D)}{P(Q)}} \qquad \boldsymbol{II.44)}$$

La probabilité $P(Q|D)$ et $P(Q)$ sont toutes les deux modélisées comme une distribution multinomiale. La requête Q est considérée comme une suite de mots indépendants. Ainsi, Ng utilise le lissage Good-Turing pour $P(Q)$:

$$\boldsymbol{P(Q/D) = \prod_{t \in Q} P(t/D) = \prod_{t \in Q} \alpha P_{ML}(t/D) + (\alpha + 1)\, P_{GT}\,(t/C)} \qquad \boldsymbol{II.45)}$$

La formule du score de document est donc la suivante :

$$\boldsymbol{Score(D,Q) = \sum_{t \in Q} log\left(\frac{\alpha P_{ML}(t/D) + (\alpha+1) P_{GT}\,(t/C)}{P_{GT}\,(t/C)}\right)} \qquad \boldsymbol{II.46)}$$

- Il est assez facile de comparer le modèle de Ng et celui de Hiemstra pour montrer leur grande similarité. En effet, le modèle de Ng peut être récrit comme suit :

$$\boldsymbol{Score(D,Q) = \sum_{t \in Q} log\left(1 + \frac{\alpha P_{ML}\,(t/D)}{(1-\alpha) P_{GT}\,(t)}\right) + \sum_{t \epsilon Q} log(1-\alpha)} \qquad \boldsymbol{II.47)}$$

Si on compare cette formule avec celle de Hiemstra, on peut voir facilement la similarité.

II.3.3.4 Le modèle basé sur l'entropie croisée

Une variante du modèle, basé sur le ratio de vraisemblance, est de représenter la RI comme une entropie croisée (ou écart d'entropie). Nous allons montrer comment nous pouvons effectivement passer de l'équation du ratio vers une forme entropique.
L'équation de vraisemblance est la suivant :

$$\boldsymbol{LR(D,Q) = \frac{P(D/Q)}{P(D)} = \frac{P(Q/D)}{P(Q)}} \qquad \boldsymbol{II.48)}$$

On suppose comme d'habitude, l'indépendance des termes, en considérant une fonction logarithmique et après quelques transformations, cette équation peut s'écrire de la façon suivante :

$$\boldsymbol{LR(Q,D) = log\frac{P(Q|M_D)}{P(Q|M_C)}} \qquad \boldsymbol{II.49)}$$

Où *P(Q|Mc)* est la probabilité générative de la requête étant donné un modèle de langage estimé sur un corpus *C*.

- Pour chaque terme de la requête, le ratio de vraisemblance mesure le rapport entre la probabilité d'observer une requête donnée étant donné un modèle de document sur la probabilité d'observer cette requête étant donné le modèle de la collection.

- Le modèle mesure de combien le modèle de document pourrait coder des événements du modèle de requête mieux que le modèle de corpus. En théorie de l'information, ceci est interprété comme une différence entre deux entropies croisées, exprimée comme suit :

$$\boldsymbol{NLR(Q|D) = H(X|C) - H(X|D)} \qquad \boldsymbol{II.50)}$$

où *X* est une variable aléatoire avec la distribution de probabilité $P(t_i)=P(t_i/M_Q)$ et *C* et *D* sont des fonctions de masse de probabilité représentant respectivement la distribution marginale (du corpus) et le modèle du document.

- L'entropie croisée permet donc de mesurer en moyenne l'écart entropique sur le fait que le modèle de document correspond bien la distribution probabiliste de la requête. D'autres formes entropie croisées ont été étudiées, on y trouve notamment celle basée sur l'information de Kullback-Leibler dans [lavrenko01].

  le principe de comparaison du modèle du document et le modèle de la requête est aussi utilisé dans le cadre de « minimisation de risque » [Lefferty01, Zhai02]. La décision pour retrouver un document est basée sur une fonction de perte comparant les deux modèles. Ils ont ensuite proposé un modèle de langue à deux étapes : une première étape pour lisser le modèle de document (pour le problème de claire

semence de données), et une deuxième étape pour tenir compte de la pertinence des termes dans la requête (le modèle de requête). Les techniques spécifiques utilisées pour les deux étapes sont respectivement le lissage Dirichlet et le lissage par interpolation.

II.4 Expansion de requêtes dans le modèle de langue

Depuis leur introduction en Recherche d'Information (RI), les modèles de langue se sont distingués par leur efficacité et leur fondement mathématique solide, (Ponte et al.,1998), (Song et al.,1998), Contrairement aux autres modèles de RI(vectoriel et probabiliste).

Les techniques d'expansion de requêtes peuvent être classées en tenant compte de plusieurs paramètres (Carpineto *et al.,* 2012) : La sélection des termes pour l'expansion on considérant chaque terme de la requête individuellement, ou la requête dans son ensemble. La représentation de la requête (document) est comme un ensemble de terme simple ou une représentation prenant en compte les relations entre termes.

Afin d'intégrer d'autres informations pour l'estimation de modèle de la requête, Lafferty et al [Lafferty et al] ont proposé une autre implémentation de la fonction de calcul de score basée sur le mesure de divergence de Kullback-Leibler(KL-divergence) entre le modèle de requête et celui de document.

II.4.1 Le modèle de Kullback-Leibler(KL-divergence)

Une autre formulation populaire de modèle de langue dans RI est estimé, est ainsi un modèle de la requête.

Le score est alors déterminé par(KL-divergence) entre les deux modèles :

$$\boldsymbol{Score(Q,D) = \sum_{t_i \in v} P(t_i|Q) X \log P(t_i|D)} \qquad \boldsymbol{II.51)}$$

Où

V : le vocabulaire de la collection.
M_Q : le modèle de langue pour la requête Q
M_D :le Modèle de langue pour le document D.

$P(t_i|Q)$Souvent est directement déterminé par l'Estimation de Maximum Linkelihood (ML) :

$$\boldsymbol{P(t_i|Q) = tf(t_i,Q) \Big/ \sum_{t_i} tf(t_i,Q)} \qquad \boldsymbol{II.52)}$$

Où $tf(t_i, Q)$est la fréquence de terme $\boldsymbol{t_i}$dans la requête Q,

De cette façon la somme peut être restreinte pour les termes de la requête seulement :

$$Score(Q,D) = \sum_{q_i \in v} P(q_i|Q) X \log P(q_i|D) \qquad II.53)$$

Dans la pratique, cette restreinte a l'avantage de réduire la complexité de processus d'évaluation de la requête.

Nous pouvons observer que les deux formules exigent toujours que les termes de la requête doivent apparaitre dans le document pour que ce dernier soit retrouvé. Mais avec le lissage du modèle de document, on peut augmenter la probabilité $\log P(q_i|D)$d'un termeq_i absent dans **D** de zéro à une petite valeur.

II.4.2 Estimation du modèle de la requête dans le modèle de langue

L'expansion de requête (Reformulation de la requête) est réalisée dans les modèles de langue avec diverses approches, on cite ci-dessous les plus en vues :

A) Approches basé sur la traduction

On considère que la RI est un processus de traduction particulière [Berger 99] :on considère qu'un besoin d'information est un message à transmettre. Ce message est d'abord formulé par une requête. Le processus de traduction est probable, plus le document peut être un document pertinent à la requête.

Dans cette approche Bai et al [Bai et al., 05]examinent comment un terme de la requête est important dans un document (le document est approprié à la requête).
D'une part si le terme de la requête est contenu dans le document, donc ce dernier à la requête à un certain degré. D'autre part, si un terme de la requête n'apparait pas dans un document ne signifie pas nécessairement que ce dernier n'est pas approprié. Il peut y avoir quelques autres termes dans le document qui sont fortement liées au terme de la requête, par exemple synonymes. Dans ce deuxième cas, le document peut encore être jugé approprié dans une certaine mesure par les relations entre termes.
Un utilisateur naïf de lissage par un modèle de collection a comme conséquence un grand modèle de requête (avec tous les termes ayant la probabilité différente de zéro), de ce fait augmentant le cout de la fonction de calcul entre la requête et les modèles de document.
La solution est de lisser le modèle original $P_{ML}(t_i|Q)$ de requête par une autre fonction $P_R(t_i|Q)$ de la probabilité déterminée en utilisant les relations entre termes :

$$P(t_i|Q) = \lambda P_{ML}(t_i|Q) + (1-\lambda) P_R(t_i|Q) \qquad II.54)$$

λ : est un paramètre de lissage.

L'expansion est basée sur des relations explicites entre termes, mettant ceci dans la formule 1) en utilisant le KL-divergence, la formule sera comme suit :

$$Score(Q,D) = \sum_{t_i \in v} P(t_i|Q) X \log P(t_i|D)$$

$$= \sum_{t_i \in v} [\lambda P_{ML}(t_i|Q) + (1-\lambda) P_R(t_i|Q)] X \log P(t_i|D)$$

$$= \lambda \sum_{q_i \in q} P_{ML}(q_i|Q) X \log P(t_i|D) + (1-\lambda) \sum_{t_i} P_R(t_i|Q) X \log P(t_i|D) \qquad II.55)$$

Où:

V est l'ensemble de vocabulaire de la collection,
Q la requête.

B) Approche basée sur le retour de pertinence (feedback)

Zhai et al **[**Zhai et al., 01**]** ont proposé un modèle basé sur l'approche de feedback qui peut être incorporé dans le cadre KL-divergence présenté dans [lafferty et al., 01]. Ils travaillent avec les documents de retour de pertinence, et estiment un modèle de requête qui peut être employé pour mettre à jour un modèle existant de la requête**.**
Dans ce cadre, ils ont proposé une approche basée sur la minimisation de la KL-divergence au-dessus des documents de feedback pour mettre à jour le modèle de langue de requête.

- ***La minimisation de la KL-divergence sur des documents de feedback***

Une stratégie différente pour estimer un modèle de requête basée sur des documents de feedback.
Soit $F = (d_1, d_2, \dots d_n)$ un ensemble de document de feedback.
La définition de KL-divergence entre le modèle de requêteθ et F l'ensemble des documents de feedback est comme suit :

$$D_e(\theta, F) = \frac{1}{|F|} \sum_{i=1}^{n} D(\theta || \hat{\theta}_{d_i}) \qquad II.56)$$

Cette formule présente la divergence moyenne entre la distribution empirique des mots de chaque document ($\hat{\theta}_{d_i}$).

Le modèle estimé de requête sera proche de chaque modèle de feedback,
Cependant, les documents de feedback partagent beaucoup de mots communs dus à la langue et les caractéristiques du domaine. En préférant un modèle qui encourt une plus grande langue pour le continent non pertinent.
Incorporant cette condition, la fonction de divergence d'un modèle de requête de feedback est le suivant **:**

$$D_e(\theta, F, C) = \frac{1}{|F|} \sum_{i=1}^{n} D(\theta || \hat{\theta}_{d_i}) - \lambda D(\theta || P(W|C)) \qquad II.57)$$

Où :

$\lambda \in [0,1]$ contrôle de poids sur le modèle de langue de collection, et
$P(W|C)$ est le modèle de langue de collection.

Réduire au minimum cette divergence est équivalent à maximiser l'entropie du modèle sous une contrainte de fréquence codée dans deuxième terme.

C'est très semblable à l'approche maximum d'entropie à l'évaluation de paramètre. En utilisant ce critère, l'estimation de $\theta_F^{\wedge} = \arg min_\theta D_e(\theta, F, C)$ est alors donnée par :

$$\boldsymbol{P(w|\theta_F^{\wedge}) \propto exp\ (\frac{1}{(1-\lambda)}\frac{1}{|F|}\sum_i log\ P(W|\theta^{\wedge}{}_{d_i}) - \frac{\lambda}{1-\lambda} log\ P(W|C))} \qquad \boldsymbol{II.58)}$$

Nous voyous que le modèle résultant assigne une probabilité élevée aux mots qui sont communs dans les documents de feedback, mais non commun salon le modèle de langue de collection.

Le paramètre $\boldsymbol{\lambda}$ contrôle poids sur le modèle de collection, quand $\boldsymbol{\lambda}$est mise à zéro, l'effet du modèle de langue de collection est ignoré, et nous avons alors un modèle de requête qui réduit au minimum la divergence sur des documents de feedback.
Avant d'exploiter $\boldsymbol{\theta_F^{\wedge}}$dans le modèle de recherche de KL-divergence, il faut d'abord l'interpoler avec le modèle originale de requête $\boldsymbol{\theta_Q^{\wedge}}$pour obtenir un modèle mise à jour $\boldsymbol{\theta_Q^{\wedge}}$, ensuite sélectionnant un document d par$\boldsymbol{D(\theta_Q^{\wedge}, ||\theta_d^{\wedge})}$.

C) Approches basées sur la dépendance entre termes

Lv et al [Lv et al., 09] présentent une approche qui permet de choisir efficacement parmi des documents de feedback les mots qui sont concentrés sur thème de requête, cette approche est basée sur l'exploitation de la position des termes dans des documents de feedback.
Ils ont proposé un modèle de pertinence basé sur la position des termes (PRM).

Le modèle de pertinence proposé(PRM) incorpore l'information de position du terme à l'estimation des modèles de feedback de sorte, qu'on puisse surpondérer des termes qui sont proches des termes de la requête dans le document de feedback.

PRM est une distribution polynômiale qui essaye de capturer la probabilité que le terme **w** est vu dans un document approprié. Cependant ; PRM estime la probabilité conditionnelle en termes de probabilité commune d'observer **w** avec la requête.

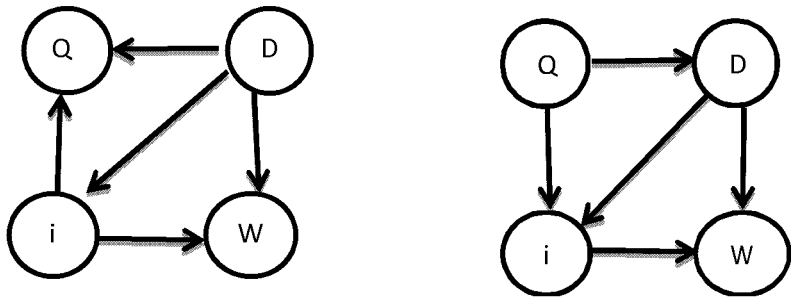


Figure.5 :réseaux des dépendances, pour estimer le modèle basé sur la position de pertinence.

Le modèle de requête basé sur la position des termes est défini comme suit :

$$P(w|Q) = \frac{P(w,Q)}{P(w)} \propto P(w,Q) = \sum_{D \in F} \sum_{i=1}^{|D|} P(w,Q,D,i) \qquad II.59)$$

Où :

i indique une position dans le document D ,

F est l'ensemble de documents de feedback.

Pour l'estimation du $P(w,Q,D,i)$les auteurs décrivent une méthode qui suppose que $w$est prélevé de même façon que, Qdans cette méthode, ils calculent la probabilité commune d'observer un mot en même temps que les mots de requête à chaque position.

La génération du mot $w$et celui de la requête Qsont indépendante, on a alors :

$$P(w,Q,D,i) \propto \frac{P(w,Q|D,i)}{|D|} = \frac{P(Q|D,i)P(w|D,i)}{|D|} \qquad II.60)$$

Remplaçant l'équation II.60) dans l'équation II.59), on obtient la formule suivante :

$$P(w|Q) \propto P(w,Q) \propto \sum_{D \in F} \sum_{i=1}^{|D|} \frac{P(Q|D,i)P(w|D,i)}{|D|} \qquad II.61)$$

$P(w|D,i)$Est la probabilité de prélèvement du mot wà la position $i$dans le document D

Pour une meilleure efficacité du modèle, ils ont simplifié le calcul de la probabilité $P(w|D,i)$comme suit :

$$P(w|D,i) = \begin{cases} 1.0 & si\ w\ apparait\ \text{à}\ la\ position\ i\ dans\ D \\ 0.0 & Sinon \end{cases}$$

Le terme $P(Q|D,i)$dans l'équation II.61) est le composante clé en estimant le modèle de pertinence. C'est la probabilité de Likelihood de la requête à la position $i$du document D

$P(Q|D,i)$Peut mesurer les poids relatif de positions dans le document : une position plus proche des mots de la requête génère plus probablement la requête, et on conséquence un terme qui se produit à cette position recevrait un poids plus élevé.

Avec la méthode d'approximation proposée dans [Yuanhua, 09], l'estimation suivante de $P(w|D,i)$est obtenu :

$$P(w|D,i) = \frac{c'(w,i)}{\sqrt{2\pi\sigma^2}} \qquad II.62)$$

Où$c'(w,i)$est la fréquence globale propagée de terme wà la position $i$des occurrences de $w$dans toutes les autres positions.

Suivant [Yuanhua, 09], l'estimation $c'(w,i)$ en utilisant la fonction de Gaussain Kernel :

$$c'(w,i) = \sum_{j=1}^{|D|} c(w,j)\, exp\left[\frac{-(i-j)^2}{2\sigma^2}\right] \qquad II.63)$$

Où :

iEtj sont les positions absolues des termes correspondants dans le document,

$|D|$Est la langueur du document,

$c(w,j)$Est la fréquence observée de wà la position j.

Le modèle ainsi proposé est ensuite lissé en utilisant le lissage de Jelinek-Mercer :

$$P_\lambda(w|D,i) = (1-\lambda)P(w|D,i) + \lambda P(w|C) \qquad II.64)$$

Où :

$\lambda \in [0,1]$Est un paramètre de lissage,

$P(w|C)$Est le modèle de langue de la collection,

Et Pour le calcul de la probabilité$P(Q|D,i)$, on utilise la formule suivante :

$$P(w|D,i) = \prod_{j=1}^{m} P_\lambda(q_j|D,i) \qquad II.65)$$

En remplaçant l'équation II.65) dans l'équation II.61), on obtient le modèle utilisé pour le calcul de pertinence.

- **Approches basées sur l'exploitation des relations entre termes de la requête**

Nous supposons que quand un utilisateur formule leur requêtes originales, ils ont un certain ensemble de concepts à l'esprit, mais pouvons seulement les exprimer un nombre restreint sous forme de requête. Nous traitons les concepts que l'utilisateur a à l'esprit, mais ne les avons pas explicitement exprimés en requête, en tant que concepts latents.

Ces concepts latents peuvent se composer d'un terme simple, des termes composés, ou d'une certaine combinaison des deux.
Dans les trois graphes suivants on représente l'exploitation des relations entre termes de la requêtes :

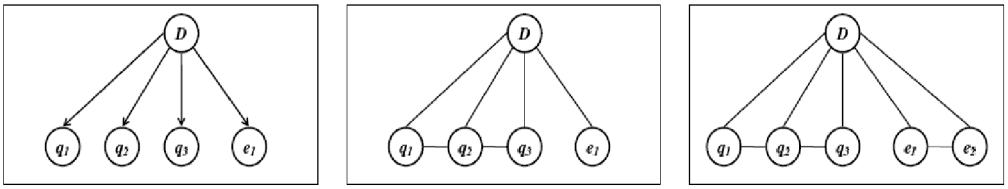


Figure 6 : « modèles graphiques représentant la modélisation de pertinence (gauche) ;expansion latente de concept on utilisant des concepts de termes simples (milieu) ; et expansion latente de concept on utilisant deux concepts de termes (droits) pour une requête de trois termes. »

Bien que ces deux exemples se servent de la prétention séquentielle de la dépendance (c.-à-d. dépendances entre les termes adjacentes de requête), il est important de noter que la requête originale et les concepts d'expansion peuvent employer n'importe quelle structure de l'indépendance.

Après que H soit construit, nous calculons $\mathbf{P}_{H}$, **(E|Q)**, unedistribution de probabilité au-dessus des concepts latents, selon :

$$P_{H,,}(E|Q) = \frac{\sum_{D \epsilon R} P_{H,}(Q,E,D)}{\sum_{D \in R} \sum_{E} P_{H,}(Q,E,D)} \qquad II.66)$$

Là où R est l'univers de tous les documents et E possibles est un certain concept latent qui peut se composer d'une ou plusieurs termes. Puisqu'il n'est pas pratique pour calculer cette addition, nous devons la rapprocher. Nous notons ce$P_{H,}(Q,E,D)$est susceptible d'être fait une pointe autour de ces documents D qui sont fortement rangés selon la requête$Q$. Par conséquent, nous rapprochons $P_{H,}(Q|E)$en additionnant seulement au-dessus d'un petit sous-ensemble de documents appropriés ou pseudo-appropriés pour la requête Q. Ceci est calculé comme suit :

$$P_{H,,}(E|Q) \approx \frac{\sum_{D \epsilon R_Q} P_{H,}(Q,E,D)}{\sum_{D \in R_Q} \sum_{E} P_{H,}(Q,E,D)}$$

$$\propto \sum_{D \in R_Q} exp\left[F_{QD}(Q,D) + F_D(D) + F_{QD}(E,D) + F_Q(E)\right] \qquad II.67)$$

Là où $\boldsymbol{R_Q}$ est un ensemble de documents appropriés ou pseudo-appropriés pour la requête $\boldsymbol{Q}$ et tous les ensembles de clique sont construits utilisant le $\boldsymbol{H}$.

Comme nous voyons, la contribution de probabilité pour chaque document dans $\boldsymbol{R_Q}$ est une combinaison des points de la requête originale pour le document. Pour la robustesse maximum, nous employons un ensemble de paramètres différent pour $\boldsymbol{F_{QD}(Q,D)}$et$\boldsymbol{F_{QD}(E,D)}$, qui nous permet de estimer le terme, commande, et dispositifs non commandés de fenêtre différemment pour la requête originale et le concept d'expansion de candidat.

II.5 Conclusion

Nous avons développé dans ce chapitre une vue détaillé des différents méthodes de reformulation de requêtes qui est une phase importante du processus de recherche d'information. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin.
Ensuite nous avons étudié la reformulation de requête dans les modèles traditionnels et dans le modèle de langue et comme notre travail s'insère dans le cadre de ce dernier, une grande partie lui a été consacrée.
Dans la reformulation de requête de l'utilisateur, un document est comme un ensemble de termes simples et des unités d'indexations plus complexes (termes composés);une nouvelle technique qui permet de combiner les termes simples et les termes composés. Pour une représentation plus détaillée du contenu des documents et des requêtes.

Dans ce contexte le chapitre suivant aura comme contribution la reformulation de requête qui combine les termes simples et les termes composés.

Chapitre III :

Approche à implémenter

III.1 Introduction

La plupart des modèles de langue développés pour la RI utilisent le principe de génération de la requête par un modèle de document.

Deux problèmes fondamentaux liés à l'utilisation des modèles de langage en RI sont la source de plusieurs études. Le premier problème concerne la clair semence de données (Data Sparseness) : qui consiste à attribuer la probabilité zéro à tout document ne contenant pas tous les termes, même s'il est pertinent. Pour y remédier la technique de lissage est utilisée, et permet d'attribuer une probabilité non nulle pour les termes non observés dans le document. Cette technique (Smoothing) peut jouer divers rôles, par exemple la combinaison de multiples sources d'information sur un document (informations sémantiques). Le second problème est lié à l'utilisation de modèle de langage uni -gramme qui ne prend pas en compte la dépendance entre les termes (document, requête). L'une des méthodes utilisées pour étendre le modèle uni-gramme est l'utilisation des termes composés comme unité d'indexation. En effet les termes composés permettent de construire des unités d'indexation non ambiguës et remédier au problème d'ambigüité des termes plus précis et peuvent par conséquent améliorer la précision de la RI.

L'autre problème engendré par l'utilisation des modèles uni-gramme est la disparité entre terme. L'expansion de requête est la solution plus utilisée pour pallier ce problème.

Nous proposons dans ce chapitre une méthode d'expansion de requête basée sur un modèle de langage mixte combinant les termes composés et termes simples Notre approche apporte alors des réponses aux deux problèmes cités précédent.

III.2 Approche proposée

Pour notre expansion on a choisi une Approche basé sur la dépendance des termes, d'abord on présente le formalisme de notre modèle. On considère une requête Q représentée dans le vocabulaire $V = \{T_1, \dots, T_m\} \cup \{t_1, \dots, t_n\}$contenant des termes composés $T_j$et des termes simplest_i,

Le terme composé Test un n-gramme formé par deux ou plusieurs termes simples adjacents non vides.

Pour le calcul de Score d'un document D Vis-à-vis d'une requête Q,on utilise Le modèle de Kullback-Leibler(KL-divergence) formulée ainsi :

$$\boldsymbol{Score(Q,D) = \sum_{w \in v} P(w|Q) X \log P(w|D)} \qquad \boldsymbol{III.1)}$$

Dans ce que suit-on définit l'estimation de modèle de la requête $\boldsymbol{P(w|Q)}$et de modèle de document $\boldsymbol{P(w|D)}$.

III.2.1 Estimation de modèle de la requête

On commençant par l'estimation de modèle de requête$\boldsymbol{P(w|Q)}$comme une mixture entre le modèle original de la requête$\boldsymbol{P_{org}(w/Q)}$ et le modèle considérant les mesures d'association PMI et DICE $\boldsymbol{P_{MA}(w/Q)}$ Ainsi, le nouveau modèle de la requête est exprimé comme suit:

$$\mathbf{P(w/Q)= \vartheta \times P_{org}(w/Q)+(1-\vartheta) \times P_{MA}(w/Q)} \qquad \textbf{III.2)}$$

Où :

$\vartheta \in [0,1]$, est un paramètre qui contrôle l'apport de modèle de la requête original par apport au modèle considérant les mesures d'associations.

La fonction de Score devient alors :

$$\boldsymbol{Score(Q,D) = \sum_{w \in V} \left(\vartheta \times P_{org}(w/Q) + (1-\vartheta) \times P_{MA}(w/Q)\right) log\, P(w/D)} \quad \boldsymbol{III.3)}$$

$$\boldsymbol{Score(Q,D) = \vartheta \times \sum_{w \epsilon Q} P_{org}(w/Q) + log\, P(w/D) + (1-\vartheta) \times \sum_{w \epsilon V} P_{MA}(w/Q)\, log\, P(w/D)} \qquad \boldsymbol{III.4)}$$

On note que le premier terme de la formule est une sommation à travers les termes de la requête (non pas à travers tous les termes de vocabulaire), car $\boldsymbol{P_{org}(w/Q) = 0}$ pour tout terme n'appartenant pas à la requête. Nous assumons que les termes reliés à la requête originale sont ceux qui apparaissent seulement dans les documents de réinjection DP. Nous ne considérons qu'un sous-ensemble de $Q \cup DP$, noté G;
Où : |G|=N, N est le nombre de termes à ajouter à la requête originale.

Par conséquent la formule III.4) devient ainsi :

$$\boldsymbol{Score(Q,D) = \vartheta \times \sum_{w \epsilon Q} P_{org}(w/Q)\, log\, P(w/D) + (1-\vartheta) \times \sum_{w \epsilon G} P_{MA}(w/Q)\, log\, P(w/D)} \qquad \boldsymbol{III.5)}$$

III.2.1.1 Estimation de modèle de requête basé sur Les mesures d'association PMI et DICE :

En considérant tous les termes de la requête (simple et composés), le modèle de la requête $\boldsymbol{P_{MA}(w/Q)}$ basé sur les mesures d'association PMI et DICE est exprimé ainsi :

$$\boldsymbol{P_{MA}(w/Q) = \sum_{t \in Q} P(w/t) \times P(t/Q) + \sum_{T \in Q} P(w/T) \times P(T/Q)} \qquad \boldsymbol{III.6)}$$

Où :

$\boldsymbol{P(w/t)}$: la probabilité de la mesure d'association d'un terme w sachant un terme simple t_i appartenant à la requête originale ;

$\boldsymbol{P(w/T)}$: la probabilité de la mesure d'association d'un terme w sachant un terme composé T_j appartenant à la requête originale ;

$\boldsymbol{P(t/Q)}$: le poids d'un terme simple dans la requête originale Q ;

$\boldsymbol{P(T/Q)}$: le poids d'un terme composé dans la requête originale Q.

A. Estimation des probabilités $\boldsymbol{P(t/Q)}$ et$\boldsymbol{P(T/Q)}$

Pour estimer ces deux probabilités nous supposons que la contribution d'un terme dans la requête initiale est liée à deux facteurs:

* la fréquence du terme dans la requête,
*le type du terme (simple ou composé).

Afin de calculer ce dernier facteur nous considérons qu'un terme composé ajoute plus de précision à la requête qu'un terme simple, nous lions cette apport à la taille du terme. Nous exprimons alors ces deux probabilités comme suit:

$$\boldsymbol{P(t/Q) = \frac{F(t,Q)}{|Q|}} \qquad \boldsymbol{III.7)}$$

De même :

$$\boldsymbol{P(T/Q) = \frac{F(T,Q) \times |T|}{|Q|}} \qquad \boldsymbol{III.8)}$$

Où :

$\boldsymbol{|T|}$ est la longueur de terme composé,
$\boldsymbol{|Q| = \sum_{t \in Q} F(T,Q) \times |T| + \sum_{t \in Q} F(T,Q)}$ est la longueur de la requête.

B. Estimation de la probabilité $P(w/T)$

L'estimation de la probabilité d'un terme $w$sachant un terme composé T appartenant à la requête originale est exprimée ainsi :

$$P(w/T) = \alpha P_{init}(w/T) + (1-\alpha)P(w/c(T)) \qquad \text{III.9)}$$

Où

$c(T)$ est l'ensemble des termes simples qui composent le terme T ;

α est facteur d'interpolation pour contrôler l'apport d'un terme composé par apport à ses termes composants.

Afin d'estimer la seconde partie de la formule, on propose une méthode similaire a cette développée dans le modèle de traduction (Berger et al., 1999). Par conséquent, on exprime cette probabilité comme suit :

$$P(w/T) = \alpha P_{init}(w/T) + (1-\alpha)\sum_{t\in T} P_{init}(w/t)P(t/T) \qquad III.10)$$

Où :

$P_{init}(w/T)$: La probabilité initiale de la mesure d'association d'un terme $w$sachant un terme composé T appartenant à la requête originale calculée par le formule III.17) ou III.18).

$P_{init}(w/t)$: la probabilité initiale de la mesure d'association d'un terme w sachant un terme simple qui appartient au terme composé Tappartenant à la requête originale calculée par le formule III.17 ou III.18).

$P(t/T)$ **:** la probabilité d'un terme simple $t$qui appartient au terme composé Tcalculée avec la formule III.12).

❖ **Estimation de la probabilité $P(t/T)$**

Pour cette estimation on pose l'hypothèse suivant :
« Les termes composants d'un terme composé ne sont pas de même importance » il peut avoir des termes plus important que d'autre comme par exemple, on trouve dans le terme composé <java script> le terme <java>est plus important que le terme <script>.
On considère intuitivement que la dominance d'un terme est déterminée par leur spécificité. On considère de l'estimer de la manière suivant :

$$imp(t) = {}^{N}/_{df(t)} \qquad III.11)$$

Où :

$\boldsymbol{t}$: un terme simple,

$\boldsymbol{N}$ est le nombre de documents ou le terme t apparait,

$\boldsymbol{df(t)}$ le nombre de documents dans la collection C.

Donc on calcule la probabilité $\boldsymbol{P(t/T)}$ comme suit :

$$\boldsymbol{P(t/T) = \frac{imp(t)}{\sum_{t,t_i \in T} imp(t_i)}} \qquad \boldsymbol{III.12)}$$

C. Estimation de la probabilité $\boldsymbol{P\ (w/t)}$

Pour calcule cette probabilité, on s'est basé sur l'intuition suivante :
on suppose que l'utilisateur l'lorsqu'il utilise un terme simple dans sa requête, il faut référencer généralement, à un ou plusieurs concepts (termes composé). Par exemple un utilisateur qui utilise le terme **système** dans sa requête peut faire référence au terme composé **système informatique**.
On propose donc de laisse la probabilité $P(^{\boldsymbol{w}}/_{\boldsymbol{t}})$en tenant compte de la probabilité de Mesure d'association « *MA* » de terme *w* dans l'ensemble des termes composés auxquels le terme $\boldsymbol{t}$ appartient.

La probabilité $\boldsymbol{P(^{w}/_{t})}$ est exprimée comme suit :

$$\boldsymbol{P(w/t) = \beta P_{init}(w/t) + (1 - \beta) P(w/C(t))} \qquad \boldsymbol{III.13)}$$

Où :

$\boldsymbol{P_{init}(w/t)}$:La probabilité initiale dela mesure d'associationd'un terme $\boldsymbol{w}$sachant un terme composé $\boldsymbol{t}$appartenant à la requête originale calculée par la formule III.17) ou III.18).

$\boldsymbol{C(t)}$ est l'ensemble des termes composés auxquels le terme $\boldsymbol{t}$appartient.

$\boldsymbol{\beta}$ est un facteur d'interpolation pour contrôler l'apport d'un terme simple par apport aux termes composés auxquels il appartient.

$\boldsymbol{P(w/C(t))}$ **:** La probabilité de la mesure d'association d'un terme $\boldsymbol{w}$sachant l'ensemble des termes composés auxquels le terme simple $\boldsymbol{t}$appartient calculée avec la formule III.14).

De même que la formule III.9), on utilise alors une méthode de translation [Berger et al. ,99].Donc la second partie de la formule est exprimée comme suit :

$$P(w/C(t)) = \sum_{t \in T} P_{init}(w/T)P(T/t) \qquad \textbf{III. 14)}$$

- ✓ T_l : terme composé ou le terme simple $t_i$appartient.

Estimation de la probabilité $P(T/t)$ **:** En appliquant le théorème de Bayes on obtient :

$$P(T/t) = \frac{P(t/T)P(T)}{P(t)} \qquad \textbf{III. 15)}$$

Tel que :

$P(t)$=1,

$P(t/T)$ est calculée en utilisant la formule III.12)

$P(T)$ est calculée ainsi :

$$P(T) = \frac{1/df(T)}{\sum_{Tm \in c(t)}(1/df(T_m))} \qquad \textbf{III. 16)}$$

Où $df(T)$est le nombre de documents contenant le terme composé T. C(t) est l'ensemble des termes composés auxquels le terme t appartient.

❖ **Estimation de la probabilité avec les mesures d'associations**

L'estimation de l'information mutuelle (PMI) de la manière suivant :

$$P_{MI}(w/w_j) = \log_2 \frac{P_{MI}(w, w_j)}{P_{MI}(w)P_{MI}(w_j)} \qquad \textbf{III. 17)}$$

L'estimation de coefficient de Dice est de la manière suivant :

$$P_{Dice}(w/w_j) = \frac{2 \times P_{Dice}(w, w_j)}{P_{Dice}(w) + P_{Dice}(w_j)} \qquad \textbf{III. 18)}$$

Où :

w , w_j ,$w_i \in V$

$P_{MI}(w, w_i)$ est la fréquence de l'information mutuelle (PMI) de couple (w, w_j)dans une fenêtrede taille F sur les documents retournés de la première recherche.

$P_{MI}(w)$ **:** est la fréquence de l'information mutuelle (PMI) de w dans une fenêtre de taille F sur les documents retournés de la première recherche.

$P_{MI}(w_j)$ est la fréquence de l'information mutuelle (PMI) de w_i dans une fenêtre de taille F sur les documents retournés de la première recherche.

III.2.2 Estimation de modèle de document

Après avoir estimé le modèle de requête, nous présentons maintenant l'estimation de modèle de la requête.
Nous supposons que le modèle de document $P(W|D)$ peut être estimé en utilisant deux modèles :Le Modèle de terme simple M_{Dt}et Le modèle de terme composé M_{DT}

Etant donnée la nouvelle requête Q_{new}(obtenu après expansion de requête), exprimé par des termes simple $t_i$et termes composésT_J. Les deux modèles de document sont estimés comme suit :

$$P(t_i|D) = P_{Dir}(t_i|M_{Dt}) \text{III. 19})$$

Et

$$P(T_j|D) = \omega P_{Dir}(T_j|M_{DT}) + (1-\omega)\prod_{t_k \in T} P_{Dir}(t_k|M_{D_t}) \qquad III.20)$$

Où:

$\omega \in [0,1]$ Paramètre de lissage,

$P_{Dir}(T_j|M_{DT})$ et $P_{Dir}(t_k|M_{D_t})$peut être évalué en utilisant n'importe quel modèle de langue uni-gramme. Dans ce cas le modèle de Dirichlet est utilisé.

Le modèle de terme simple est représenté comme suit :

$$P_{Dir}(t_i|M_{D_t}) = \frac{F(t_i, D_t) + \mu P(t_i|C_t)}{|D_t| + \mu} \qquad III.21)$$

Où:

$F(t_i, D_t)$ est la fréquence de terme simple$t_i$dans le document D_t ;

$P(t_i|C_t)$est le modèle de langue de collection de fond (la fréquence globale de terme est utilisé) ;

$|D_t|$ est la taille de document ;

μ est le paramètre de lissage.

Et de même manière on a :

$$P_{Dir}(T_j|M_{D_T}) = \frac{F(T_j, D_T) + \mu P(T_j|C_T)}{|D_T| + \mu} \quad III.22)$$

Où:

$F(T_j, D_T)$ **:** est la fréquence de terme composé**,** elle est calculée par compte de termeT_j**.**

$P(T_j|C_T)$ **:** est le modèle de langue de collection.

$|D_T| = \sum_{T \in C_T} F(T, D_T)$ est la taille de document représenté par des termes composés.

III.3 Conclusion

La reformulation de requête est une des stratégies qui permet d'améliorer la construction d'une requête. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin.

Dans ce chapitre, on a décrit une nouvelle approche pour la reformulation de requête de modèle mixte combinant les termes simples et les termes composés, et on a choisi les mesures d'association PMI et DICE, Qui permet d'extraire des nouveaux termes pour les ajouter à la requête originale.

Chapitre IV :

Expérimentation et Evaluation

IV.1 Introduction

Dans ce chapitre nous allons d'abord décrire l'environnement de développement utilisé pour implémenter notre approche, nous allons ensuite faire une petite description de l'architecture générale de notre application, puis, nous allons présenter notre expérimentation qui porte sur la reformulation de requête basée sur un modèle de langage mixte combinant les termes simples et les termes composé, ainsi que les tests et l'évaluation de notre application et enfin nous discutons les résultats obtenus.

IV.2Architecture logicielle de notre approche

Les étapes de notre approche sont structurées et illustrées dans la figure ci-dessous :

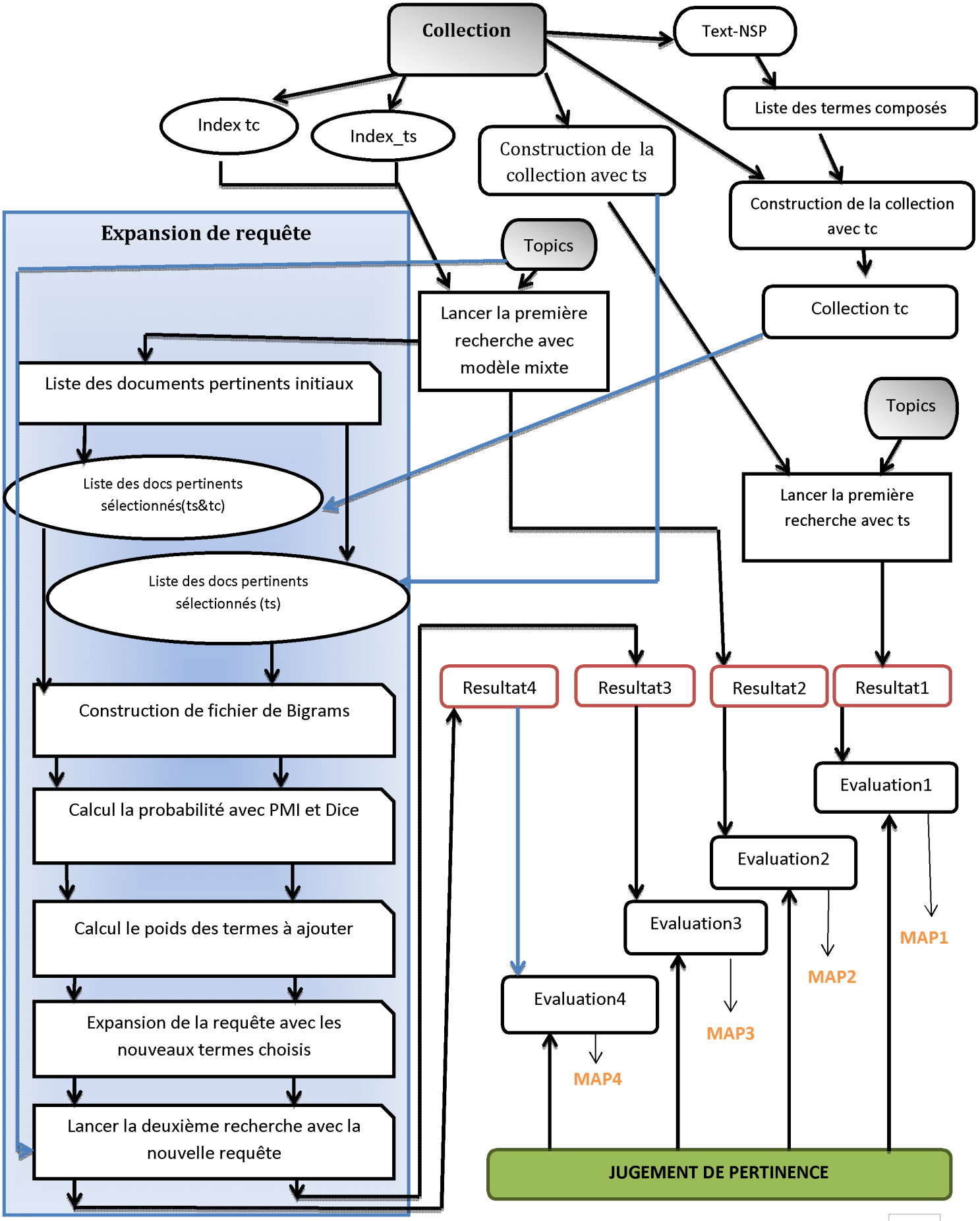

Collection
Text-NSP
Index tc
Index_ts
Construction de la collection avec ts
Liste des termes composés
Construction de la collection avec tc
Expansion de requête
Topics
Lancer la première recherche avec modèle mixte
Collection tc
Liste des documents pertinents initiaux
Topics
Liste des docs pertinents sélectionnés(ts&tc)
Lancer la première recherche avec ts
Liste des docs pertinents sélectionnés (ts)
Resultat4
Resultat3
Resultat2
Resultat1
Construction de fichier de Bigrams
Evaluation1
Calcul la probabilité avec PMI et Dice
Evaluation2
Evaluation3
Calcul le poids des termes à ajouter
MAP1
MAP2
Evaluation4
MAP3
Expansion de la requête avec les nouveaux termes choisis
MAP4
Lancer la deuxième recherche avec la nouvelle requête
JUGEMENT DE PERTINENCE

Figure7 : Schéma générale de notre application

ts : termes simples ;

tc : termes composés ;

Resultat1 : le résultat de recherche termes simples sans expansion ;

Resultat2 ; le résultat de recherche modèle mixte sans expansion ;

Resultat3 : le résultat de recherche termes simples avec l'expansion ;

Resultat4 : le résultat de recherche modèle mixte avec l'expansion de requête.

IV.3 Environnement de développement

Dans nos expérimentations on a utilisé la plateforme terrier qui a pour objectif de faciliter la recherche d'information et implémenter pour cela un certain nombre de fonctionnalités de recherche offre un large choix pour le développement de nouvelles applications et pour les testes en recherche d'information. Terrier est développé entièrement en java.

Notre approche fondée sur l'expansion de requête basée sur un modèle de langue mixte combinant les termes simples et les termes composés. Pour déterminer dans les documents les termes qui ont relation avec la plupart des termes de la requête et de calculer les mesures d'associations (PMI et DICE) entre terme on a utilisé Ngram Statistics Package (Text ::NSP).

IV.3.1 Le langage java

Java est unlangage orienté objet de programmation développé par *Sun Microsystems* dans les années 1990. Depuis lors, Java a gagné une énorme popularité en tant que langage informatique. Il permet d'exécuter des programmes au travers d'une machine virtuelle.

Java a des avantages significatifs par rapport aux autres Langues. Il est facile à utiliser et donc facile à écrire, compiler, déboguer. Par ailleurs, il est plus facile à apprendre par rapport aux autres langages de Programmation. Depuis Java est orienté objet, il permet de créer des programmes modulaires et des codes réutilisables.

A. Les caractéristiques de langage java

Le langage Java a été conçu à partir des langages objets *SmallTalk* et *C++*. Ses caractéristiques sont résumées ci-dessous :

- ***Simplicité***: d'une syntaxe familière aux programmeurs C et C++, il en a été dépouillé de tous les mécanismes complexes (gestion des pointeurs, gestion de la mémoire, l'héritage multiple, …).

- ***Orienté Objet*** : Tout est objet, Il faut concevoir ainsi. Contrairement au C++ qui autorise l'écriture du code des fonctions en dehors des classes, Java oblige le programmeur à définir les méthodes comme partie intrinsèque des classes.

- ***Orienté réseau et distribué***: Développement d'applications réparties. Java permet d'accéder facilement à des données distantes sur le réseau, et de réaliser des applications client/serveur grâce aux classes paquetages de communication (java.net).

- ***Multitâche (multi-thread)*** : Java permet de concevoir des applications capables d'effectuer plusieurs actions (*thread*) en même temps.

- ***Dynamique*** : Un programme Java charge les classes qu'il utilise (y compris les classes de base) en cours d'exécution.

- ***Robuste***: L'objectif de java est de permettre la réalisation de logiciels fiables. Les caractéristiques suivantes permettent d'atteindre cet objectif :typage fort, pas d'accès aux pointeurs (réduisant les risques d'erreurs),gestionnaire de la mémoire, mécanisme de gestion d'exception.

- ***Sécurisé*** : Eviter d'exécuter du code dommageable. De plus java possède des API destinés au cryptage, à la gestion de l'authentification, …

- ***Interprété, Indépendant des architectures matérielles*** : Le code produit par le compilateur n'est pas du code machine. C'est un code intermédiaire (*bytecode*) simple et rapide (plus rapide qu'un langage de script entièrement interprété) à traduire en langage machine par un interpréteur (*Java Virtual Machine - JVM*) gérant pour sa part les spécificités du système hôte.

B. Présentation de NetBeans

NetBeans est un projet open source fondé par Sun Microsystems en juin 2000 et. C'est un environnement de développement intégré (EDI), qui permet d'écrire, de compiler, déboguer et déployer des programmes. Il est écrit en Java et peut supporter différents autres langage de programmation, comme C,C++,XML,PHP,…etc. NetBeans est gratuit et disponible sous Windows, Linux, Solaris,...etc. Son installation nécessite l'installation de la JDK (*JavaDevelopment Kit*).

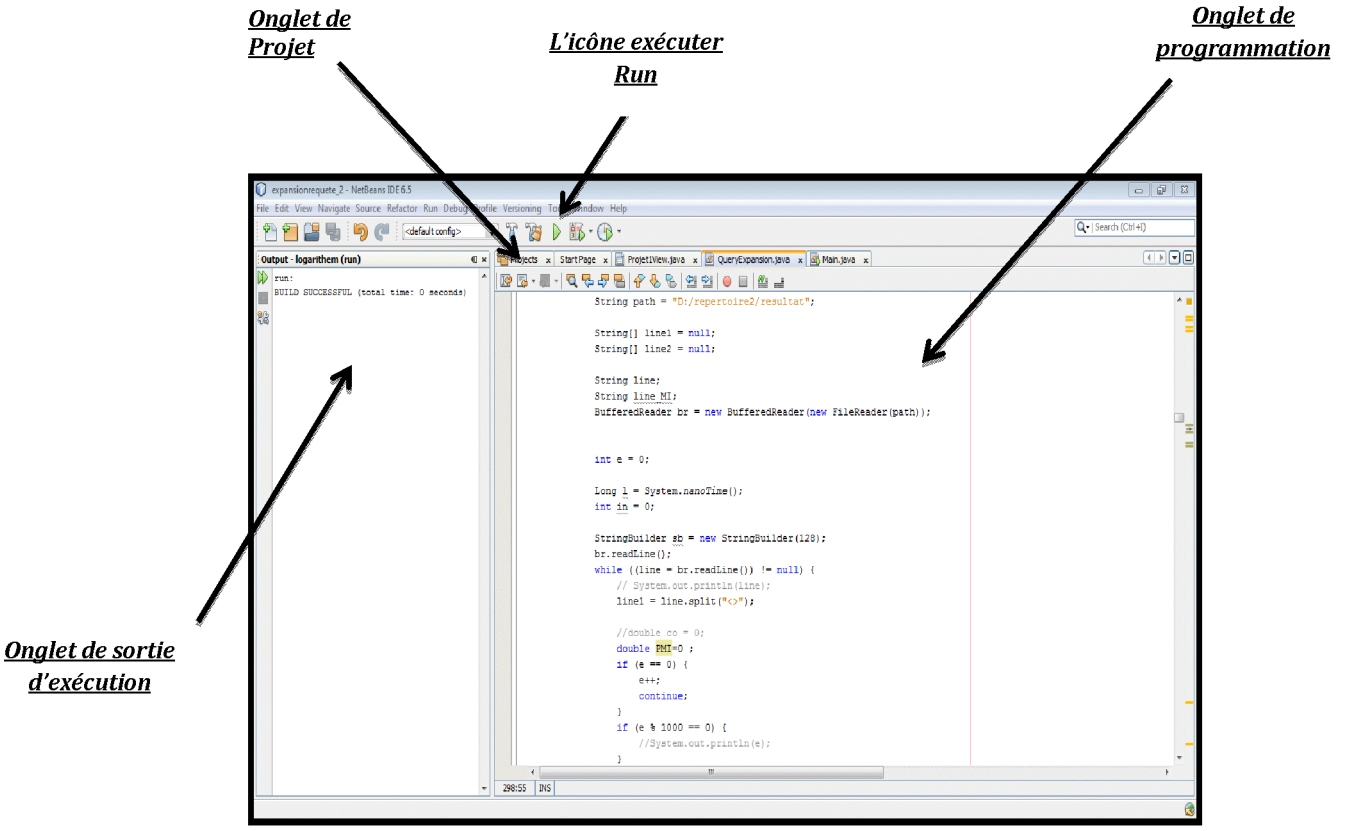


Figure 8 : l'interface principale de l'environnement NetBeans 6.5

IV.3.2 La plateforme terrier

Terrier ou TERabyte RetrIEveR est un moteur de recherche open-source développé par le département "Computing Science" de l'université de Glasgow.
Terrier 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 (Ouniset al.,2006). C'est une plateforme modulaire pour le développement rapide de la recherche d'information avec JAVA, offrant des fonctionnalités d'indexation et de recherche documentaire.

Terrier est livré avec une puissante preuve de concept d'application de recherche Google Desktop [Captures d'écran], et toutes les capacités de TREC, y compris la capacité d'index, de requête et d'évaluer les collections TREC standards, tels que AP, WSJ, WT10G, GOV et GOV2.

A. Architecture de terrier

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

A.1 Le processus d'indexation de Terrier

Pour l'indexation d'une collection de documents, Terrier se base sur l'indexation classique à 4 étapes (voir figues.9)

- Splitter la collection de document : consiste à parcourir l'ensemble du corpus et d'envoyer chaque document à l'étape suivant.
- L'extraction des termes « Tokenize Document » :qui consiste à parser chaque document reçu et en extraire les déffirents termes.
- Traitement des termes extraits : et qui consiste en l'élimination des mots vides et la lemmatisation des termes.
- Construction de l'index.

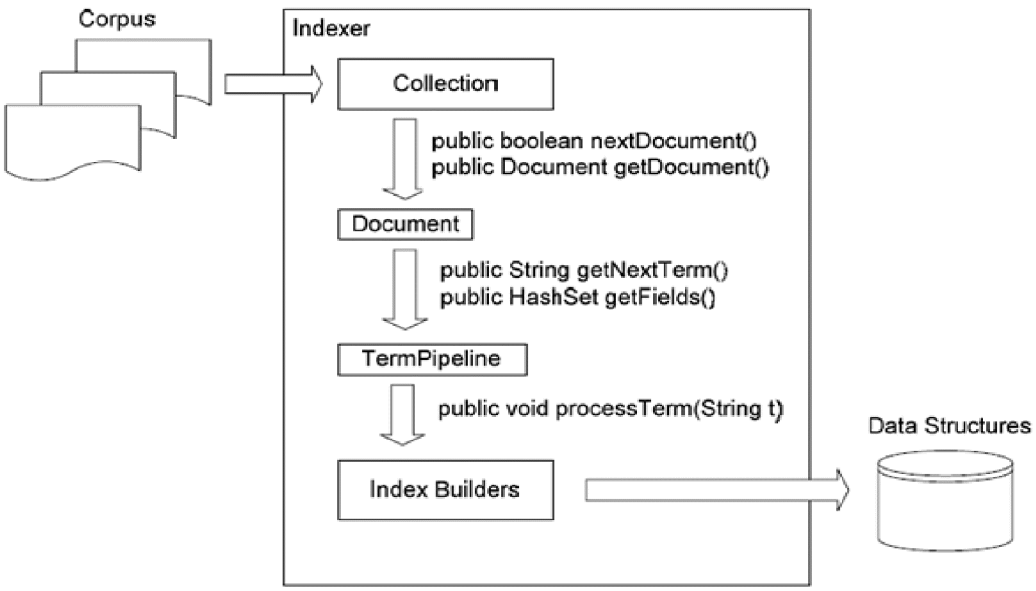


Figure 9: Présentation du processus d'indexation de Terrier

Le résultat consiste en un index constitué des structures de données suivant :

Inverted File, Lexicon File, Document Index (présentés en détail en Annexe).
Et les Modules Collection, Document, Index Builders font partie de l'API d'indexation de terrier(Ils sont présentés en détail dans Annexe).

A.2 Le processus de recherche de Terrier

Un des principaux objectifs de terrier est de faciliter la recherche d'information. Terrier implémente pour cela un certain nombre de fonctionnalités de recherche qui offre un large choix pour le développement de nouvelles applications et pour les tests dans RI. En effet, Terrier offre un grand choix de modèles de pondération, il propose aussi un langage de requête avancée. Une autre fonction de recherche très importante intégrée dans terrier est l'automatisation de l'expansion de requête. La figure 10 présente un aperçu du processus de recherche dans terrier.

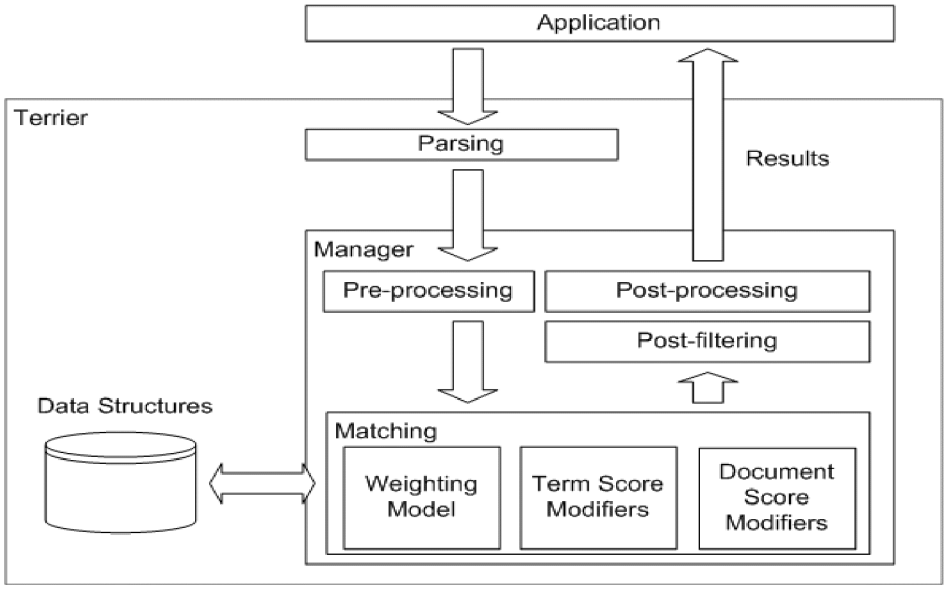


Figure 10 : Présentation du processus recherche de Terrier

Le processus de recherche de Terrier est constitué d'un ensemble de modules : Query, Manager, Matching (Présenté en détail dans Annexe).

IV.3.3 Présentation de NgramStatistics Package (Text :: NSP)

Ngram Statistics Package (Text ::NSP) est une collection de modules de Perl qui facilitent en analysant Ngrams dans des dossiers des textes. Il inclut un éventail de mesures d'association cela peut être employé pour identifier des collocations.

A. Ngrams

Nous définissons un Ngram comme ordre des marques de « n » qui se produisent dans une fenêtre au moins des marques de « n » dans le texte ; ce qui constitue la « marque »,elle peut être définie par l'utilisateur.

NSP utilise des expressions régulières de Perl pour définir les tokens dans un texte qui seront délimités par le symbole '<>'. Après avoir définit les tokens, les Ngrams seront construits en assemblant N tokens, Par exemple

'Operating<> system' est un Bigram dont les tokens sont 'Operating' et 'System'.NSP construit habituellement les Ngrams avec des tokens contigus, Comme il peut aussi les former avec des tokens non contigus, mais il faut d'abord définir une fenêtre de taille K qu'est une séquence de K tokens contigus, ou la valeur de K est supérieur ou égale à la valeur de N de Ngrams.
Un Ngram peut être formé de n tokens quelconque, et chacun de ces tokens doit appartenir à la fenêtre de taille K définie. Les n tokens qui apparaissent dans le Ngram doivent être exactement dans le même ordre dont ils se produisent dans la fenêtre.

On prend par exemple une partie d'un texte, ou les tokens apparaissent comme suit :

Measures to Protect the Atmosphere

- ***Les Bigrams formés des tokens précédent sont :***

Measures<>the
Measures<>Atmosphere
Measures<>Protect
to<>Protect
Measures<>to
Protect<>Atmosphere
the<>Atmosphere
to<>Atmosphere
Protect<>the
to<>the

- ***Les Trigrams formes des tokens précédent sont :***

Protect<>the<>Atmosphere
Measures<>to<>Protect
to<>Protect<>the

Le NSP est constitué de deux programmes qui permettent à l'utilisateur d'identifier et d'analyser Ngrams dans un texte.

A. Le module count.pl

Le programme count.pl est responsable de l'identification de Ngram et le calcul le nombre d'occurrence dans une fenêtre de taille K.
Count.pl prendre un fichier texte en entre et produit en sortie une liste de tout le Ngram avec leur fréquence.

- Par défaut la valeur de Ngram est égale à 2, mais elle peut être définit par l'utilisateur, par exemple :

Count.pl—ngram 3 output.txt input.txt

Telle que:

- ***--ngram 3*** pour spécifie qu'on cherche les Ngrams de taille 3.
- ***output.txt*** : le fichier résultat.
- ***input.txt*** : le fichier d'entre.

- La valeur de la fenêtre peut être aussi définit par l'utilisateur, par exemple :

Count.pl --window 3 output.txt input.txt

Telleque:

- ***--window 3*** pour spécifier la valeur de la fenêtre de taille 3.

Le contenu de fichier output.txt est comme suite :

```
10
        Measures<>the<>1 4 3
        Measures<>Atmosphere<>1 4 4
        Measures<>Protect<>1 4 2
        to<>Protect<>1 3 2
        Measures<>to<>1 4 1
        Protect<>Atmosphere<>1 2 4
        the<>Atmosphere<>1 1 4
        to<>Atmosphere<>1 3 4
        Protect<>the<>1 2 3
to<>the<>1 3 3
```

Figure11: Exemple d'un fichier output.txt

On a :
Dans la première ligne, le numéro 10 indique qu'il y a 10 Bigrams dans le fichier texte en entrée. Les autres lignes sont constitués des Bigrams et des liste de nombres :

- Le premier nombre de la liste dans chaque ligne désigne le nombre de fois le Bigramoccure dans le texte en entrée ;

- Le deuxième nombre dénote dans combien de Bigrams le première tokenoccure à la première position ;
- Le troisième nombre dénote dans combien de Bigrams le deuxième tokenoccure à la deuxième position.

Prenant comme exemple la deuxième ligne de fichier output.txt :

Measures<>Atmosphere<>1 4 4

- Le Bigram est constitué des deux tokens ***Measures*** et ***Atmosphere.***
- Les deux tokens apparaissent une fois ensemble dans le texte.
- Le token ***Measures*** apparait ***4*** fois à la première position dans le texte.
- Le token ***Atmosphere*** apparait ***4*** fois à la deuxième position dans le texte.

B. Le module statistic.pl

Le programme statistic.pl prend en entée la liste de Ngrams avec leurs fréquences calculée par count.pl pour calculer le score de chaque Ngram. Ces Ngrams sont placés dans un ordre descendant en fonction de leurs scores calculés. Le score calculé pour chaque Ngram permet de décider si ce dernier est ou pas un terme (le Ngram est significatif ou pas).

IV.4Données utilisés

Afin d'évaluer la performance de notre approche, nous avons utilisé une collection test qui est un ensemble de document; un ensemble de requête représente un ensemble de thèmes (topics) et des jugement de pertinence pour ces requêtes et ces documents :

- ***Collection :*** on a choisi :
  La collection AP88 contenant 79919 documents.
  La collection WSJ contenant 74520 documents.

- ***Topics251-300 :*** fichier contient 50topics, définissant les requêtes. Seul le titre des topics est utilisé.

Exemple d'une requête

```
<top>

<num> Number: 251
<title> Exportation of Industry

<desc> Description:
Documents will report the exportation of some part
of U.S. Industry to another country.

<narr> Narrative:
Relevant documents will identify the type of industry
being exported, the country to which it is exported; and as well
will reveal the number of jobs lost as a result of that exportation.

</top>
```

Figure12: Structure d'une requête.

- ***Liste des termes composés:*** construite à partir de la collection en faisant appel au module count.pl de Text-NSP-1.03. Un terme composé est retenu dans la liste si sa fréquence dans la collection est supérieure à 20.

- ***fichier de jugement de pertinence :***Une liste des documents pertinents et non pertinents pour chaque requête (réponse idéales).

IV.5 Les étapes de mise en œuvre de notre approche

❖ ***Etape1 : Construction de la collection***

A. Construction de collection avec termes composés et termes simples:

Cette étape permet de construire la collection avec des termes composés (termes simples) en utilisant la liste qui contient les termes composés.

Resultat1 : Collection avec des termes composés (termes simples).

Voici un extrait de contenu d'un document avec des termes composés (chaque terme est composé de deux termes simple concaténés avec un trait d'union entre ces derniers) et des termes simples.

0001 columbia_ampinsolvtakeov like thrift
big_lossblame_sharpsharp_deteriorjunk_bondbond_portfoliorichard_schmittschmitt_staf
fstaff_reportreport_wallwall_streetstreet_journal 04 02 90 wall_streetstreet_journal

Figure13:extrait de contenu d'un document avec des termes composés(simples).

B. Construction de collection avec termes simples :

Cette étape permet de construire la collection avec des termes simples.
Resultat1 : Collection avec des termes simples.

Voici un extrait de contenu d'un document avec des termes simples.

0001 insolvtakeov like thrift 04 02 90 a3 csvinsolvinsolv like expens
biggest alli revolution profit thrift virtucolumbia report
yesterdaimassivdeteriorcolumb

Figure14: extrait de contenu d'un document avec des termes simples.

❖ ***Etape 2 : Indexation de la collection :***

Cette étape a pour objectif d'identifier les termes (simples et composés) sensés représenter au mieux le contenu d'un document.
Deux types de termes sont successivement identifiés à l'issue de cette étape qui sont : Les termes simples et les termes composés.

- ***Indexation de la collection avec des termes simples :***

  Pour ce type d'indexation on a utilisé les classes BasicIndex et TRECIndexing existantes par défaut dans Terrier.

  Résultat : index termes simples.

- ***Indexation de la collection en termes composé :***

  Comme Terrier est un logiciel qui offre la possibilité d'extension sur le processus d'indexation ; on a ajouté une classe à l'application pour indexer les documents avec des termes composés.

  Résultat : index avec termes composés.

❖ **Etape 3 :Lancement de la première recherche :**

Cette étape consiste à lancer la première recherche, dont on a deux types :

- ***Recherche avec destermessimples :***

  Pour trouver les documents pertinents vis-à-vis de la requête originale, il faut d'abord affecter un poids aux termes de la requête pour indiquer leur importance dans le document en utilisant le modèle de langage Dirichlet.

  Résultat : obtention de résultat (fichiers .RES) pour l'ensemble de requête afin d'évaluer cette recherche.

- ***Recherche avec le modèle mixte :***

  Pour lancer cette recherche on a utilisé une autre classe qui traite la requête originale comme un ensemble de termes simples et des termes composés et donne un score au document qui vérifie la requête.
  L'une des caractéristiques principales de terrier est son extensibilité. Comme terrier ne possède pas un modèle de pondération qui prend enconsidération les termes composés, on a ajouté un modèle de

pondération qui permet d'assigner un score à chaque terme simple et terme composé de la requête dans le document.

Résultat : il crée un fichier résultat (fichier.RES)qui permet par la suite de calculer la précision moyenne(MAP).

❖ **Etape 4 :Expansion de requête**

Cette étape consiste à lancer la deuxième recherche, dont on a deux types :

- **Expansion de requête à base de modèle mixte**

La classe qui permet d'exécuter ce processus est Query Expansion dans Terrier (expansion automatique de requête) qui permet l'expansion de la requête avec des nouveaux termes extraits de N premiers documents pertinents, ainsi que la pondération des termes de la requête.
Comme notre approche se fond sur la reformulation de requête à base des termes simples et des termes composés, et comme Terrier est caractérisé par sa possibilité de modificationalors, on a modifié Query Expansion qu'est initialement intégré dans Terrier.

Pour faire l'expansion de requête on a suivi les étapes suivantes :

1. ***Récupération de l'ensemble de N documents pertinents :***

   Dans cette étape on récupère les documents pertinents retournés de la première recherche dont leurs contenus sont des termes simples et des termes composés, puis par la suite on a filtré ces documentsdans la collection construite avec des termes composés(termes simples) à l'étape1 ce qui donne par la suite des documents pertinents dont leur contenusest des termes simples ou composés.

2. ***Construction de fichier de Bigrams :***

   Après avoir récupéré l'ensemble des documents pertinents, on introduit ces derniers à Text-NSP (count.pl) qui retourne un fichier texte contenant tous les bigrams et leurs fréquences de cooccurrences.
   Voici un extrait de contenu de résultat retourné par count.pl :

```
Protect<>Atmosphere<>1 2 4
the<>Atmosphere<>1 1 2
Measures<>Protect<>1 2 2
Protect<>the<>1 2 2
to<>Protect<>1 2 2
Measures<>to<>1 2 1
to<>the<>1 2 2
```

Figure15: extrait de contenu de résultat retourné par count.pl

3. *Calcul de la cooccurrence entre terme par la mesure d'association*

Dans cette étape, on a utilisé le résultat retourné par Text_NSP (count.pl) pour calculer les probabilités de PMI/DICE de chaque Bi_gramme $\boldsymbol{P_{MI}(w, w_j)}$ **,** en utilisant la formule III.17) et$\boldsymbol{P_{Dice}(w, w_j)}$ la formule III.18) de chapitre III.

4. *Reformulation de requête avec des nouveaux termes :*

Avant la reformulation de requête originale il faut d'abord choisir les termes à ajouter :

- Choix des nouveaux termes :

Pour choisir les termes à ajouter on a utilisé la méthode présentée en chapitre III.

- **Expansion de requête avec les termes simples :**

Idem avec l'expansion de requête basé sur le modèle mixte, juste que dans la reformulation de requête, on a pris les termes simples sans les termes composés.

❖ **Etape5 : Lancement de la deuxième recherche avec la nouvelle requête :**

Une fois le modèle de la requête est construit, une nouvelle recherche est effectuée.

➢ **L'interface de notre application :**

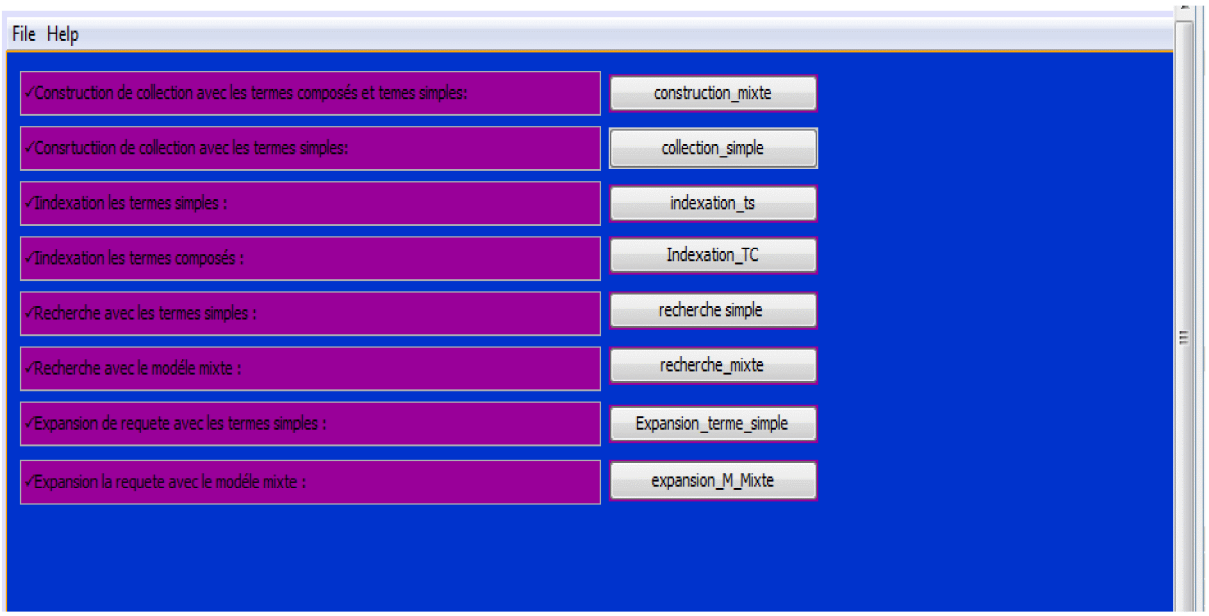


Figure 16 : Interface de notre application

IV.6 Evaluations de l'approche d'expansion de requêtes basée sur un modèlemixte combinant les termes simples et les termescomposés

Pour évaluer la performance de l'approche d'expansion de requête basé sur un modèle mixte combinant les termes simples et les termes composés définit dans le chapitre précédant , nous avons réalisé une série d'expérimentation dont le but est de comparer le résultat de recherche de documents pertinents avec la requête reformulé base sur un modèle mixte par apport à l'utilisation de la requête originale sans la plateforme terrier.

IV.6.1 Paramètres de modèle de notre approche :

On a estimé les différents paramètres de modèle présenté dans le chapitre précédent d'une manière empirique pour optimiser la valeur de MAP. Ces paramètres sont :

- Le paramètre μ du modèle de Dirichlet pour la recherche termes simples sans expansion est fixé à 2500.
- Les paramètres de modèle de document de modèle mixte sans expansion de requête:

- Le paramètre ω de la formule (III.20) qui contrôle l'apport de modèle de document de terme simple par apport au modèle de document des termes composés, La valeur de ce paramètre est fixé à 0.5
- Le paramètre μ de la formule (III.21) est fixé à 2500.
- Le paramètre μ de la formule (III.22) est fixé à 100.

♦ Les paramètres de modèle de document de modèle mixte avec l'expansion de requête :
- Le paramètre ω de la formule (III.20) est fixé à 0.5
- Le paramètre μ de la formule (III. 21) est fixé à 4300.
- Le paramètre μ de la formule (III.22) est fixé à 100.

♦ Le paramètre μ du modèle de Dirichlet pour la recherche termes simples avec l'expansion est fixé à 3100.

♦ Les paramètres de modèle de la requête à base des termes simples avec l'expansion de requête :

- Le paramètre ϑ de la formule (III.2) est fixé à 0.6
- Le paramètre β de la formule (III.13) est fixé à 0.3

♦ Les paramètres de modèle de la requête basée sur un modèle de langue mixte avec l'expansion de requête :

- Le paramètre ϑ de la formule (III.2) est fixé à 0.4
- Le paramètre β de la formule (III.13) est fixé à 0.6
- Le paramètre α de la formule (III.9) est fixé à 0.7

♦ Le paramètre **F** de la formule (III.17) et la formule (III.18) de la recherche des termes simples avec l'expansion, représentant la taille de la fenêtre pour estimer la fréquence de la PMI et DICE est fixée à 50.

♦ Le paramètre F de la formule (III.17) et la formule (III.18) de la recherche de modèle mixte avec l'expansion, représentant la taille de la fenêtre pour estimer la fréquence de la PMI et DICE. La valeur de ce paramètre est fixée à 20.

♦ Un terme composé est identifié si sa fréquence dans la collection est dépassé un seuil fixé à 25.

♦ Un terme simple est identifié si sa fréquence dans la collection est dépassé un seuil fixé à 20.

♦ Le nombre de documents d'où on extrait les termes à ajouter est fixé à 4.

IV.6.2 Résultats d'évaluation

Les tableauxsuivantsprésents les résultats obtenus pour l'ensemble de requêtes de tests on utilisant deux types de collection (AP88, WSJ). Les résultats montrent que notre approche d'expansion de requêtes basée sur un modèle mixte combinant les termes simples et les termes composés offre une précision moyenne plusimportante que les autres modèles de recherche.

- Résultats avec la collection AP88 :

Information	Recherche avec termes simple [A]	Recherche avant expansion avec modèle mixte [B]	Recherche après expansion de requête avec des termes simples [C]	Recherche après expansion de requête avec le modèle mixte [D]
Nombre de requêtes	48	48	48	48
Nombre de documents pertinents	1672	1672	1672	1672
Nombre de documents pertinents sélectionnés	722	753	783	763
Précision Moyenne	0.1729	0.1865	0.1943	0.2019

	B et A	C et A	D et B	D et C
Taux d'amélioration	+7.86%	+ 12.37%	+8.25 %	+3.91%

Tableau 3: comparaison des différents modèles on utilisant la collection AP88.

En analysant les deux tableaux, on remarque que :

A. Les précisions moyennes des recherches sont différentes :

A.1. les précisions avec la collection AP88 :

- La précision moyenne (MAP) de la recherche avec termes simples est : MAP=0.1729
- La précision moyenne (MAP) de la recherche avec termes composés est :

MAP = 0.1865

- La précision moyenne (MAP) de la recherche avec termes simples avec l'expansion est : MAP = 0.1943
- La précision moyenne (MAP) de la recherche avec termes composés avec l'expansion est : MAP = 0.2019.

B. Le nombre de documents pertinents retournés :

- Le nombre de documents pertinents retournés de la recherche avecmodèle mixte sans expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base des termes simples sans expansion.
- Le nombre de documents pertinents retournés de la recherche à base des termes simples avec expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base des termes simples sans expansion.
- Le nombre de documents pertinents retournés de la recherche à base de modèle mixte avec expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base de modèle mixte sans expansion.

D'après les résultats obtenus de la comparaison qu'est faite entre les recherches par rapport à leurs précisions, on peut dire que :

- La recherche à base des termes simples avec l'expansion est améliorée par rapport à la recherche à base des termes simples sans l'expansion, avec un taux d'amélioration de 12.37%
- La recherche à base de modèle mixte avec l'expansion est améliorée par rapport à la recherche à base de modèle mixte sans l'expansion, avec un taux d'amélioration de 8.25%
- La recherche à base de modèle mixte avec l'expansion est améliorée par rapport à la recherche à base des termes simples avec l'expansion, avec un taux d'amélioration de 3.91%

- Résultats avec la collection WSJ :

Information	Recherche avec termes simple [A]	Recherche avant expansion avec modèle mixte [B]	Recherche après expansion de requête avec des termes simples [C]	Recherche après expansion de requête avec le modèle mixte [D]
Nombre de requêtes	45	45	45	45
Nombre de documents pertinents	1064	1064	1064	1064
Nombre de documents pertinentssélectionnés	502	503	581	524
Précision moyenne	0.1549	0.1568	0.1557	0.1701

	B et A	C et A	D et B	D et C
Taux d'amélioration	+1.22%	+0.51%	+8.48%	+9.24%

Tableau 4: comparaison des différentsmodèles on utilisant la collection WSJ.

En analysant les deux tableaux, on remarque que :

C. Les précisions moyennes des recherches sont différentes :

- La précision moyenne (MAP) de la recherche avec termes simples est : MAP=0.1549
- La précision moyenne (MAP) de la recherche avec termes composés est : MAP = 0.1568
- La précision moyenne (MAP) de la recherche avec termes simples avec l'expansion est : MAP = 0.1557
- La précision moyenne (MAP) de la recherche avec termes composés avec l'expansion est : MAP = 0.1701.

D. Le nombre de documents pertinents retournés :

- Le nombre de documents pertinents retournés de la recherche à base des termes composés sans expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base des termes simples sans expansion.
- Le nombre de documents pertinents retournés de la recherche à base des termes simples avec expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base des termes simples sans expansion.
- Le nombre de documents pertinents retournés de la recherche à base de modèle mixte avec expansion est amélioré par rapport au nombre de documents pertinents retournés de la recherche à base de modèle mixte sans expansion.

D'après les résultats obtenus de la comparaison qu'est faite entre les recherche par rapport à leurs précisions, on peut dire que :

- La recherche à base des termes simples avec l'expansion est améliorée par rapport à la recherche à base des termes simples sans l'expansion, avec un taux d'amélioration de 0.51%
- La recherche à base de modèle mixte avec l'expansion est améliorée par rapport à la recherche à base de modèle mixte sans l'expansion, avec un taux d'amélioration de 8.48%
- La recherche à base de modèle mixte avec l'expansion est améliorée par rapport à la recherche à base des termes simples avec l'expansion, avec un taux d'amélioration de 9.24%

Le tableau suivant présent les résultats obtenus pour l'expansion avec les mesures d'association PMI et DICE.

Collection	Type de recherche	PMI	DICE
AP88	Termes simple avec expansion	0.1943	0.1722
	Modèle mixte avec expansion	0.2019	0.1696
WSJ	Termes simples avec expansion	0.1557	0.1557
	Modèle mixte avec expansion	0.1701	0.1392

Tableau 5: résultat d'évaluation de l'approche d'expansion de requête avec les mesures d'association PMI et Dice par la collection AP88 et WSJ.

IV.6.3 Résultat d'évaluation requête par requête :

Pour avoir une vision et une analyse plus claire et plus profonde on a effectué l'évaluation requête par requête dans les deux collections (AP88 et WSJ), les graphes suivant illustrent cette analyse :

1. Pour la collection AP88

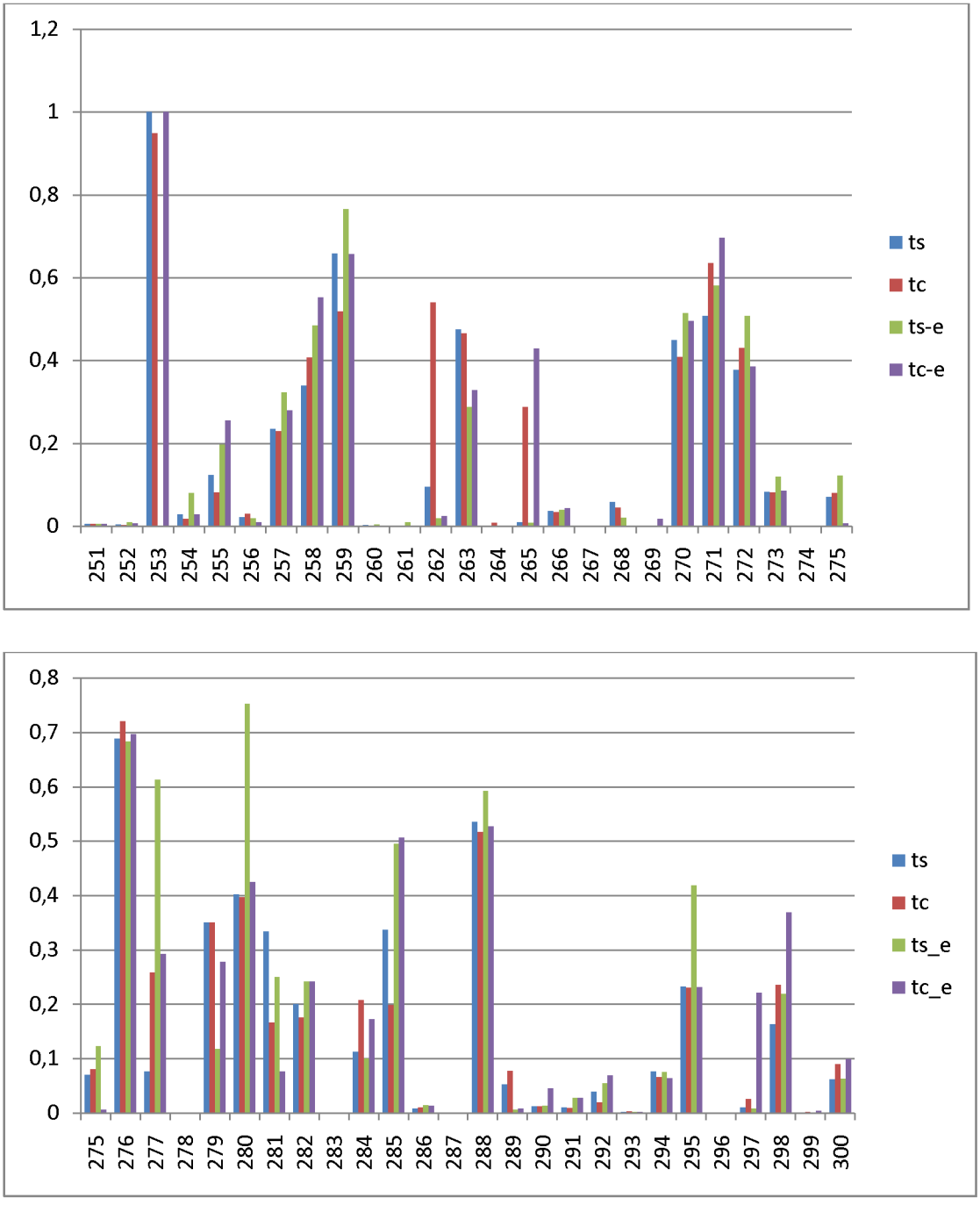
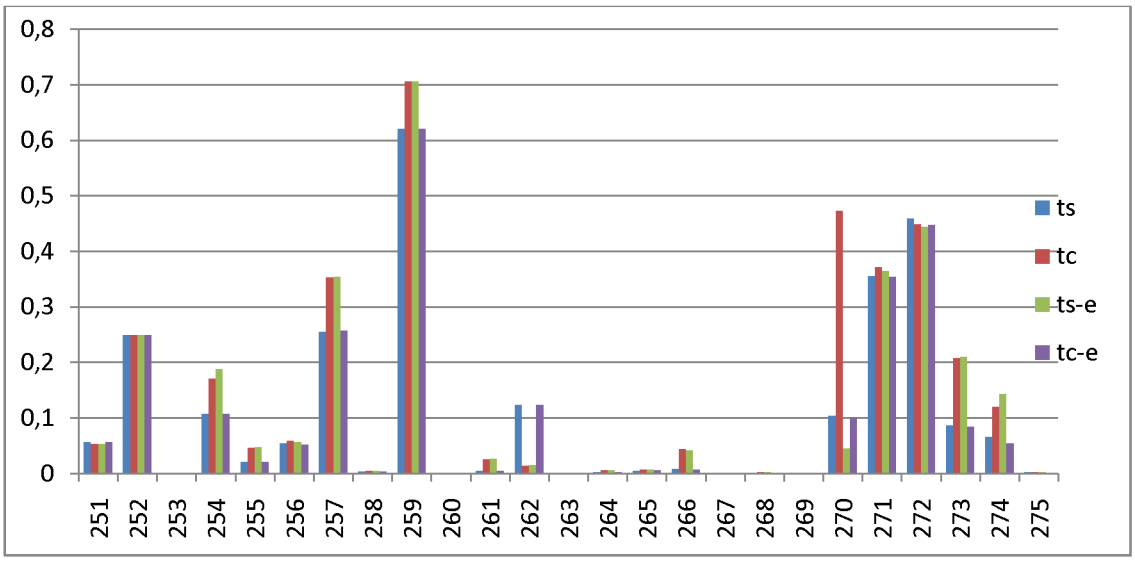


Figure 17 : Graphes représentants les résultats d'évaluation requête par requête pour les quatre recherches avec la collection AP88.

2. Pour la collection WSJ :



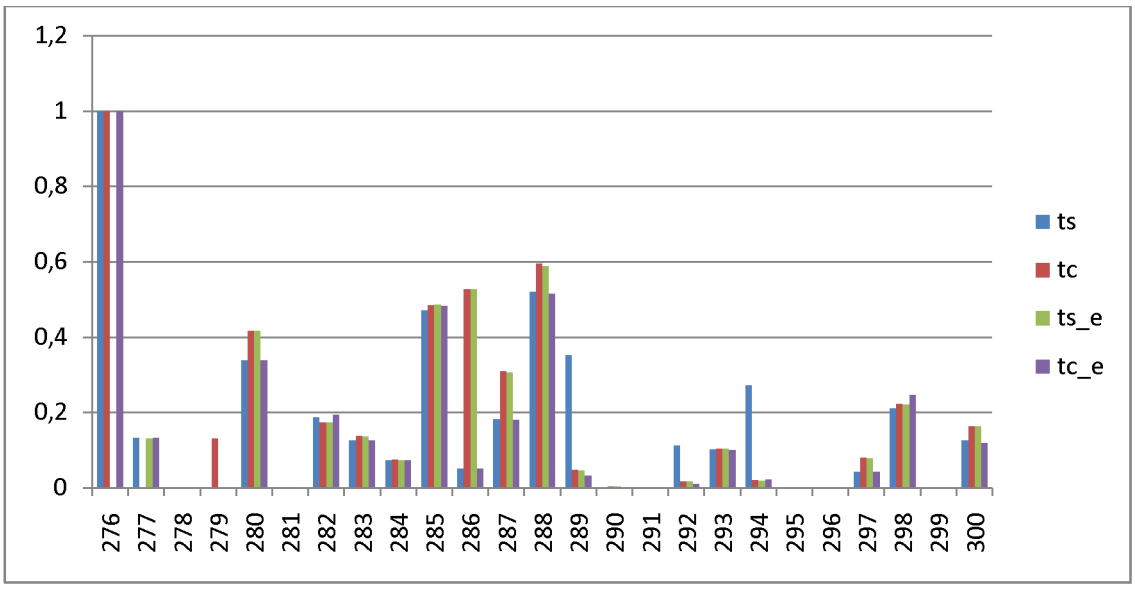


Figure 18 : Graphes représentants les résultats d'évaluation requête par requête pour les quatre recherches avec la collection WSJ.

ts : recherche termes simples sans l'expansion
tc : recherche modèle mixte sans expansion.
ts_e : recherche termes simples avec expansion,
tc_e : recherche modèle mixte avec l'expansion,

La comparaison entre les résultats d'évaluation requête par requête pour les quatre recherches sont représentés dans les tableaux suivants :

Requête	Recherche termes Simples	Recherche modèle mixte	Améliorations
251	0.0066	0.0057	-
252	0.0045	0.0038	-
253	1	0.9484	-
254	0.0292	0.019	-
255	0.1240	0.082	-
256	0.0227	0.0299	+
257	0.2359	0.2293	-
258	0.3398	0.4075	+
259	0.6589	0.5183	-
260	0.0035	0.001	-
261	0.0005	0.0007	-
262	0.0985	0.54	+
263	0.4758	0.466	-
264	0.0024	0.0085	+
265	0.0101	0.2886	+
266	0.0370	0.0346	-
267	0	0.0011	+
268	0.0585	0.0451	+
269	0.0017	0.0023	+
270	0.4499	0.4096	-
271	0.5082	0.6354	+
272	0.3779	0.341	-
273	0.0835	0.0819	-
274	0	0	0
275	0.0704	0.0806	+
276	0.6885	0.72	+
277	0.0765	0.2579	+
278	0	0	0
279	0.3500	0.3500	0
280	0.4018	0.3971	-
281	0.3333	0.1667	-
282	0.2005	0.1762	-
283	0	0	0
284	0.1124	0.2074	+
285	0.3367	0.1983	-
286	0.0084	0.01	-
287	0	0	0
288	0.5355	0.5164	-
289	0.0523	0.0772	+
290	0.0120	0.0120	0
291	0.0098	0.0095	-
292	0.0388	0.0201	-
293	0.0016	0.0027	+
294	0.0759	0.0662	-

295	0.2322	0.2306	-
296	0	0	0
297	0.0100	0.0261	+
298	0.1629	0.2357	+
299	0.0008	0.0023	+
300	0.0622	0.0896	+

Tableau 6 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec le modèle mixte sans expansion avec AP88.

Requête	Recherche termes simples	Recherche termes simples avec l'expansion	Améliorations
251	0,0066	0,0064	-
252	0,0045	0,0105	+
253	1	0	-
254	0,0292	0,0804	+
255	0,124	0,1972	+
256	0,0227	0,019	-
257	0,2359	0,3234	+
258	0,3398	0,4852	+
259	0,6589	0,7663	+
260	0,0035	0,004	+
261	0,0005	0,0095	+
262	0,0958	0,019	-
263	0,4758	0,2888	-
264	0,0024	0,0005	-
265	0,0101	0,0082	-
266	0,037	0,0394	+
267	0	0,0012	+
268	0,0585	0,0209	-
269	0,0017	0,0016	-
270	0,4499	0,5149	+
271	0,5082	0,5808	+
272	0,3779	0,5074	+
273	0,0835	0,1205	+
274	0	0	0
275	0,0704	0,1231	+
276	0.6885	0.6833	-
277	0,0765	0,613	+
278	0	0	0
279	0,35	0,1181	-
280	0,4018	0,7521	+
281	0,3333	0,25	-

282	0,2005	0,2416	+
283	0	0	0
284	0,1124	0,1005	-
285	0,3367	0,4949	+
286	0,0084	0,0143	+
287	0	0	0
288	0,5355	0,5925	+
289	0,0523	0,0066	-
290	0,012	0,0134	+
291	0,0098	0,0278	+
292	0,0388	0,0552	+
293	0,0016	0,0019	+
294	0,0759	0,0761	+
295	0,2322	0,4183	+
296	0	0	0
297	0,01	0,0081	-
298	0,1629	0,2197	+
299	0,0008	0,0002	-
300	0,0622	0,0629	+

Tableau 7 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche termes simples et de la recherche termes simples avec l'expansion avec AP88.

Requêtes	Recherche modèle mixte	Recherche modèle mixte avec expansion	Améliorations
251	0,0057	0,0054	-
252	0,0038	0,0068	+
253	0,9484	1	+
254	0,019	0,0294	+
255	0,082	0,255	+
256	0,0299	0,0099	-
257	0,2293	0,2798	+
258	0,4075	0,5533	+
259	0,5183	0,6574	+
260	0,001	0,0016	+
261	0,0007	0,0005	-
262	0,54	0,0244	-
263	0,466	0,3283	-
264	0,0085	0,0009	+
265	0,2886	0,4296	+
266	0,0346	0,044	+
267	0,0011	0	-

268	0,0451	0,0016	-
269	0,0023	0,0183	+
270	0,4096	0,4951	+
271	0,6354	0,697	+
272	0,431	0,3861	-
273	0,0819	0,0863	+
274	0	0	0
275	0,0806	0,0067	-
276	0.72	0.6968	-
277	0,2579	0,2929	+
278	0	0	0
279	0,35	0,2778	-
280	0,3971	0,4253	+
281	0,1667	0,0769	-
282	0,1762	0,2414	+
283	0	0	0
284	0,2074	0,1728	-
285	0,1983	0,5063	+
286	0,01	0,0139	+
287	0	0	0
288	0,5164	0,5268	+
289	0,0772	0,0087	-
290	0,012	0,045	+
291	0,0095	0,0276	+
292	0,0201	0,0695	+
293	0,0027	0,0018	-
294	0,0662	0,0645	-
295	0,2306	0,2312	+
296	0	0	0
297	0,0261	0,2208	+
298	0,2357	0,3689	+
299	0,0023	0,0037	+
300	0,0896	0,0994	+

Tableau 8 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche modèle mixte et de la recherche modèle mixte avec l'expansion avec AP88.

Requêtes	Recherche termes simples avec expansion	Recherche modèle mixte avec expansion	Améliorations
251	0.0064	0.0054	-
252	0.0105	0.0068	-
253	1	1	0
254	0.0804	0.0294	-
255	0.1972	0.2550	-
256	0.0190	0.0099	-
257	0.3234	0.2798	-
258	0.4852	0.5533	+
259	0.7663	0.6574	-
260	0.0040	0.0016	-
261	0.0095	0.0005	-
262	0.0190	0.0244	+
263	0.2888	0.3283	-
264	0.0005	0.0009	-
265	0.0082	0.4296	+
266	0.0394	0.0440	+
267	0.0012	0	+
268	0.0209	0.0016	+
269	0.0016	0.0183	+
270	0.5149	0.4951	-
271	0.5808	0.6970	+
272	0.5074	0.3861	-
273	0.1205	0.0863	+
274	0	0	0
275	0.1231	0.0067	-
276	0.6833	0.6968	+
277	0.0613	0.2929	+
278	0	0	0
289	0.1181	0.2778	+
280	0.7521	0.4253	+
281	0.2500	0.0769	-
282	0.2416	0.2414	-
283	0	0	0
284	0.1005	0.1728	+
285	0.4949	0.5063	-
286	0.0143	0.0139	-
287	0	0	0
288	0.5925	0.5268	-
289	0.0066	0.0087	+
290	0.0134	0.0450	+
291	0.0278	0.0276	+
292	0.0552	0.0695	-
293	0.0019	0.0018	-
294	0.0761	0.0645	-

295	0.4183	0.2312	-
296	0	0	0
297	0.0081	0.2208	+
298	0.2197	0.3689	+
299	0.0002	0.0037	+
300	0.0629	0.0994	+

Tableau 9 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche de modèle mixte avec expansion avec AP88.

Requêtes	Recherche termes simples	Recherche modèle mixte	Améliorations
251	0,0578	0,0543	-
252	0,25	0,25	0
253	0	0	0
254	0,1081	0,1715	+
255	0,022	0,0476	+
256	0,0558	0,0594	+
257	0,256	0,3534	+
258	0,0045	0,0055	+
259	0,6215	0,7065	+
260	0,002	0,0023	+
261	0,006	0,027	+
262	0,125	0,0149	-
263	0	0	0
264	0,0034	0,0065	+
265	0,0061	0,0079	+
266	0,0087	0,0449	+
267	0	0	0
268	0,0021	0,0033	+
269	0,0014	0,0013	-
270	0,1045	0,474	+
271	0,3564	0,372	+
272	0,4599	0,4493	-
273	0,088	0,2088	+
274	0,0664	0,1205	+
275	0,0032	0,0031	-
276	1	1	0
277	0,1343	0	-
278	0	0	0
279	0	0,1316	+
280	0,3382	0,4157	+
281	0	0	0
282	0,1882	0,1734	-

283	0,1268	0,1368	+
284	0,0741	0,0745	+
285	0,4715	0,4851	+
286	0,0526	0,527	+
287	0,1833	0,3093	+
288	0,52	0,5946	+
289	0,352	0,0482	-
290	0,0012	0,0036	+
291	0,0024	0,0023	-
292	0,113	0,0175	+
293	0,1022	0,1037	+
294	0,272	0,0197	-
295	0,001	0	-
296	0	0	0
297	0,0438	0,0792	+
298	0,2124	0,2222	+
299	0,0017	0,0016	-
300	0,1264	0,1639	+

Tableau 10: Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec le modèle mixte sans expansion avec WSJ.

Requêtes	Recherche termes simples	Recherche termes simples avec l'expansion	Améliorations
251	0,0578	0,0541	-
252	0,25	0,25	0
253	0	0	0
254	0,1081	0,1893	+
255	0,022	0,0479	+
256	0,0558	0,058	+
257	0,256	0,3551	+
258	0,0045	0,0052	+
259	0,6215	0,7065	+
260	0,002	0,0021	+
261	0,006	0,0278	+
262	0,125	0,0161	-
263	0	0	0
264	0,0034	0,0064	+
265	0,0061	0,0082	+
266	0,0087	0,0431	+
267	0	0	0
268	0,0021	0,0033	+

269	0,0014	0,0013	-
270	0,1045	0,046	-
271	0,3564	0,3654	+
272	0,4599	0,4449	-
273	0,088	0,2105	+
274	0,0664	0,1443	+
275	0,0032	0,0031	-
276	1	0	-
277	0,1343	0,1311	-
278	0	0	0
279	0	0	0
280	0,3382	0,4157	+
281	0	0	0
282	0,1882	0,1738	-
283	0,1268	0,1354	+
284	0,0741	0,0736	-
285	0,4715	0,4856	+
286	0,0526	0,527	+
287	0,1833	0,3067	+
288	0,52	0,5889	+
289	0,352	0,0459	-
290	0,0012	0,0035	+
291	0,0024	0,0023	-
292	0,113	0,0176	-
293	0,1022	0,1028	+
294	0,272	0,0183	-
295	0,001	0	-
296	0	0	0
297	0,0438	0,0787	+
298	0,2124	0,2213	+
299	0,0017	0,0016	-
300	0,1264	0,1625	+

Tableau 11 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple sans expansion et de la recherche termes simples avec expansion avec WSJ.

Requêtes	Recherche modèle mixte	Recherche modèle mixte avec expansion	améliorations
251	0,0543	0,0578	+
252	0,25	0,25	0
253	0	0	0
254	0,1715	0,108	-
255	0,0476	0,0222	-
256	0,0594	0,0533	-
257	0,3534	0,2587	-
258	0,0055	0,0044	-
259	0,7065	0,6214	-
260	0,0023	0,0015	-
261	0,027	0,006	-
262	0,0149	0,125	-
263	0	0	0
264	0,0065	0,0033	-
265	0,0079	0,0071	-
266	0,0449	0,0085	-
267	0	0	0
268	0,0033	0,002	-
269	0,0013	0,0013	0
270	0,474	0,1014	-
271	0,372	0,3551	-
272	0,4493	0,4487	-
273	0,2088	0,0857	-
274	0,1205	0,0558	-
275	0,0031	0,0025	-
276	1	1	0
277	0	0,1328	+
278	0	0	0
279	0,1316	0	-
280	0,4157	0,3384	-
281	0	0	0
282	0,1734	0,1941	+
283	0,1368	0,126	-
284	0,0745	0,0737	-
285	0,4851	0,4839	-
286	0,527	0,0526	-
287	0,3093	0,1807	-
288	0,5946	0,5155	-
289	0,0482	0,0334	-
290	0,0036	0,0009	-
291	0,0023	0,0022	-

292	0,0175	0,0119	-
293	0,1037	0,1017	-
294	0,0197	0,0235	+
295	0	0	0
296	0	0	0
297	0,0792	0,0435	-
298	0,2222	0,2464	+
299	0,0016	0,0024	+
300	0,1639	0,1205	-

Tableau 12 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche modèle mixte sans expansion et de la recherche modèle mixte avec expansion avec WSJ.

requêtes	Recherche termes simples avec expansion	Recherche modèle mixte avec expansion	Améliorations
251	0,0541	0,0578	+
252	0,25	0,25	0
253	0	0	0
254	0,1893	0,108	-
255	0,0479	0,0222	-
256	0,058	0,0533	-
257	0,3551	0,2587	-
258	0,0052	0,0044	-
259	0,7065	0,6214	-
260	0,0021	0,0015	-
261	0,0278	0,006	-
262	0,0161	0,125	-
263	0	0	0
264	0,0064	0,0033	-
265	0,0082	0,0071	-
266	0,0431	0,0085	-
267	0	0	0
268	0,0033	0,002	-
269	0,0013	0,0013	0
270	0,046	0,1014	+
271	0,3654	0,3551	-
272	0,4449	0,4487	+
273	0,2105	0,0857	-
274	0,1443	0,0558	-
275	0,0031	0,0025	-
276	0	1	+

277	0,1311	0,1328	+
278	0	0	0
279	0	0	0
280	0,4157	0,3384	-
281	0	0	0
282	0,1738	0,1941	+
283	0,1354	0,126	-
284	0,0736	0,0737	+
285	0,4856	0,4839	-
286	0,527	0,0526	-
287	0,3067	0,1807	-
288	0,5889	0,5155	-
289	0,0459	0,0334	-
290	0,0035	0,0009	-
291	0,0023	0,0022	-
292	0,0176	0,0119	-
293	0,1028	0,1017	-
294	0,0183	0,0235	+
295	0	0	0
296	0	0	0
297	0,0787	0,0435	-
298	0,2213	0,2464	+
299	0,0016	0,0024	+
300	0,1625	0,1205	-

Tableau 13 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple avec expansion et de la recherche modèle mixte avec expansion avec WSJ.

IV.7Conclusion :

L'évaluation est une étape importante dans le domaine de la recherche d'information du fait qu'elle permet de juger la pertinence des documents restitués par le système de [2]recherche d'information et donc d'évaluer les performances d'un SRI.

A la fin de ce chapitre,nous avons abordé les différentes expérimentations que nous avons réalisées, et cela, d'abord nous avons détaillé l'architecture générale, ensuite passanta la présentation de notre environnement de développement et la définition des différents outils utilisé,après à notre application et finalement les expérimentations et évaluations que nous avons réalisées , et cela confirme l'intérêt de l'approche d'expansion de requête de modèle mixte combinant les termes simples et les termes composés.

Conclusion générale

Conclusion générale

Notre travail se situe dans le cadre de la reformulation de requête qui est une phase importante dans les systèmes de recherche d'information. Cette technique permet de récrire la requête de l'utilisateur selon les informations retrouvées par la requête initiale. De manière générale ceci consiste, dans le cas notamment de la réinjection aveugle de la pertinence, d'extraire à partir des premiers documents pertinents retournés par le système, les mots clés important puis les rajouter à la requête initiale.

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

Nous avons vu que le problème majeur de la traduction de requêtes est l'ambigüité des termes sources. La pluparts des systèmes de recherches d'informations(SRI) utilisent des mots simples pour représenter des documents et des requêtes. Il est reconnu depuis longtemps que cette représentation est insuffisante à cause de l'ambigüité des mots isolés et du non prise en compte des relations entre les mots.

Naturellement il faut choisir un modèle de RI qui supporte une telle représentation. Notre choix s'est fixé sur les modèles de langue étant données les performances qu'ils ont montrées.

Les expérimentations et évaluations que nous avons réalisées en utilisant deux collections de TREC (AP88 et WSJ) sur la plateforme de recherche d'information Terrier, et deux mesures différentes pour pondérer la relation entre les termes(PMI/DICE) pour l'expansion, ont permet de valider notre approche. Plus précisément, les résultats obtenus mettent en exergue l'intérêt de l'approche d'expansion de requête à base de modèle mixte.

L'évaluation de notre approche a montré son impact positif sur les résultats de la recherche. Plus précisément, l'analyse des expérimentations révèlent des améliorations assez importantes de la précision moyenne (MAP) ce qui a confirmé la validité de cette approche.

Pour cela nous pensons que l'utilisation de modèle mixte pour reformuler la requête de l'utilisateur peut encore évoluer vers plus de performance.

Bibliographie

Bibliographies:

- ✓ Donald Metzler, W. Bruce Croft,"Latent Concept Expansion Using Markov Random Fields", Center for Intelligent Information Retrieval Department of Computer Science University of Massachusetts Amherst, MA 01003.
- ✓ Olga Vechtomova,Ying Wang, »A Study of the Effect of Term Proximity onQuery Expansion », *Olga Vechtomova, 200 University Avenue West, Waterloo, Canada.*
- ✓ YuanhuaLv, ChengXiang Zhai, » Positional Relevance Model for Pseudo-Relevance Feedback »,Department of Computer Science University of Illinois at Urbana-ChampaignUrbana, IL 61801.
- ✓ Seung-Hoon Na, In-Su Kang,Jong-Hyeok Lee,"Parsimonious Translation Models for Information Retrieval", Div. of Electrical and Computer Engineering POSTECH, AITrc.
  A. Rozenknop, « Modèles en Recherche d'Information »,Cours Master Recherche Paris 13Recherche et extraction d'information,
- ✓ Massih-RezaAmini, « Recherche et Extraction d'Information »,Université Pierre et Marie Curie ,l'aboratoire de paris 6.
- ✓ Jacques Le Maitre, « Recherche d'information »,Université du Sud Toulon-Var.
- ✓ Carmen Alvarez, Philippe Langlais et Jian-Yun Nie, « Mots composés dans les modèles de langue pour la recherche d'information, » RALI/IRO, Université de MontréalCP. 6128, succursale Centre-ville Montréal, Québec, H3C 3J7 Canada,19–21 avril 2004.
- ✓ Mohand Boughanem,Wessel Kraaij, Jian-Yun Nie, « Modèles de langue pour la recherche d'information », Institut de Recherche en Informatique de Toulouse, 118 route de Narbonne 31000 Toulouse Cedex 9, France ;TNO TPD, 2600 AD Delft, Pays bas ;DIRO, Université de Montréal, CP. 6128, succursale Centre-ville, Montréal, Québec, H3C 3J7 Canada.
- ✓ THESE en vue de l'obtention du Doctorat Toulouse « Reformulation de Requetés parRéinjection de Pertinence dans lesDocuments Semi-Structurés »en vue de l'Université Paul Sabatier Spécialité : INFORMATIQUE *P*ar Lobna HLAOUA, 14 Décembre 2007.
- ✓ MEMOIRE Présenté pour l'obtention du diplôme de Magister , Spécialité: Informatique , Option: *Spécification de logiciels et traitement de l'information,* Par MATAOUI M'hamed, « Reformulation de requêtes dans les systèmes de recherche d'information dans des documents XML « , Université M'hamed BOUGARA de BOUMERDES , 29/04/2007.
- ✓ Thèse en vue de l'obtention du Doctorat de l'université PAUL SABATIER Spécialité Informatique Par Mustapha BAZIZ « INDEXATION CONCEPTUELLE GUIDEE PAR ONTOLOGIE POUR LA RECHERCHE D'INFORMATION »,le 14 décembre 2005.
- ✓ Thèse en vue de l'obtention du Doctorat de l'Université Paul Sabatier Spécialité Informatique Par Nawel Nassr« Croisement de langues en recherche d'information : traduction et désambiguïsation de requêtes »le 09 décembre 20002.

- ✓ Théseen vue de l'obtention du Doctorat de l'université de TOULOUSE,*Nongdo Désiré Kompaoré* **«** *Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes :vers un processus de RI adaptatif »* **Le** *26 juin 2008.*
- ✓ ZemirliW.Nesrine, « Vers le développement d'un système de recherche d'information personnalisé intégrant le profil utilisateur », Année 2003/2004.
- ✓ InesBannour, Haïfa Zargayouna, « Une plate-forme open-source de recherche d'information sémantique »,France (2012).
- ✓ H. Aliane* , Z. Alimazighi **, R. O. Boughacha* , T. Djelliout, »Un Système de reformulationde requêtes pour la recherche d'information »,Centre de Recherche sur l'Information Scientifique et Technique , Alger, Algérie,Université des Sciences et de la Technologie Houari Boumedienne, Alger, Algérie.
- ✓ Leïla NAIT-BAHA, Agata JACKIEWICZ, Philippe LAUBLET, « Reformulation de requêtes et extraction de phrasespertinentes pour la collecte d'informations sur le Web », *Equipe LAngage, Logique, Informatique et Cognition (LaLIC)Centre d'Analyse et de Mathématiques Sociales (CAMS)UMR 17 du CNRS, EHESS, Paris-Sorbonne.*
- ✓ Thèse doctorat de l'université PAUL SABATIER, Spécialité INFORMATIQUE ParLynda Tamine, » OPTIMISATION DE REQUETES DANS UN SYSTEME DE RECHERCHE D'INFORMATION », le 21/12/ 2000.
- ✓ Thèse doctorat de l'université MOULOUD MAMMERI de tizi ouzou, par Lynda TAMINE, »Les systèmes de recherche d'information : Reformulation de requête et apprentissage base sur les algorithmes génétiques »,Le 14/02/1998.
- ✓ Jian-Yun Nie, » Le domaine de recherche d'information – Un survol d'une longue histoire »,Département d'informatique et recherche opérationnelleUniversité de Montréal.
- ✓ Hassan CHOUAIB, »Reformulation de requêtes dans un modèle de réseau possibiliste pour la recherche d'information »,Université Paul Sabatier, 2006.
- ✓ Jian-Yun Nie, Jean-Pierre Chevallet, Yves Chiaramella, « Vers la recherche d'informations à base de termes », Département d'Informatique et Recherche Opérationnelle, Université de Montréal, Equipe Modélisation et Recherche d'Information Multimédia (MRIM) du laboratoire de Communication Langagière et Interaction Personne-Système (CLIPS) de l'MAG).
- ✓ William B. Frakes and Ricardo Baeza. Information Retrieval: Data Structures and Algorithms, -Yates, Prentice Hall, 1992.
- ✓ Eric Gaussier, Christian Jacquemin, & Pierre, Zweigenbaum. Traitement automatique des langues et recherche d'information.
- ✓ In Eric Gaussier and Marie-HeleneStefanini, editors, Assistance intelligente a la recherche d'informations, chapter 2, pages 71-96. Hermes-Lavoisier, Paris, 2003.
- ✓ Salton, G., & McGill, M.. Introduction to Modern InformationRetrieval. McGraw-Hill, New York, 1983.
- ✓ E. Voorhees, "Using WordNet to Disambiguate Word Sensesfor Text Retrieval", Proceedings of the 16thAnnual Conference on Research andDevelopment in Information Retrieval, SIGIR'93, Pittsburgh, PA, 1993.
- ✓ Christophe GNAHO, INTRODUCTION AU LANGAGE JAVA.
- ✓ H. Mounier, Support de cours JavaStructures de données Notions en Génie Logiciel et Programmation Orientée Objet,Université Paris Sud.

- ✓ IadhOunis, Gianni Amati, VassilisPlachouras, Ben He, Craig Macdonald,and Douglas Johnson,"Terrier Information Retrieval Platform",University of Glasgow, Glasgow G12 8QQ, UK.
- ✓ Eric Gaussier," Recherche d'Information Généralités et Modèles »Université Grenoble 1 - Lab. Informatique Grenbole.
- ✓ TamásKiezer, MiklósErdélyi, » Introduction to Information Retrieval », IR architecture, document processing, indexing, weighting, University of Pannonia.
- ✓ InesBannour, Haïfa Zargayouna, » Une plate-forme open-source de recherche d'information sémantique », Laboratoire d'Informatique de l'université Paris-Nord.
- ✓ Craig Macdonald & Ben He, » Researching and BuildingIR applications using Terrier », Department of Computing Science University of Glasgow,ECIR 2008.
- ✓ Parantapa Goswami, Batch (TREC) Indexing and Retrieval using Terrier 2.2.1, Indian Statistical Institute, Kolkata.
- ✓ Thèse Magister, Farida Achemoukh,"Modèle de langage pour la recherché d'information», Université mouloud Mammeri de tiziouzou2006.
- ✓ Thèse Doctorat,ABDELKRIM BOURAMOUL, « Recherche d'information contextuelle et sémantique sur le web »,Université MENTOURI de Constantine,2010/2011.
- ✓ Thèse Master,Karima LOUNI,« Proposition & implémentationd'heuristiques pour identifier lanature des documents recherchés »,Université mouloud Mammeri de tizi ouzou ,2011/2012.
- ✓ Thèse Master, ARHAB Karima et GUEZOUI Faroudja, « Expansion de requête de l'utilisateur à base des termes composés », Université mouloud Mammeri de tizouzou ,2011/2012.

Webiographies:

- ✓ www.google.com
- ✓ http://www.findthatzipfile.com/search-45427224-hZIP/winrar-winzip-download-qrels.51-100.disk3.parts1-5.tar.gz.htm
- ✓ http://www.01net.com/telecharger/windows/Programmation/cgi_et_perl/fiches/42.html
- ✓ http://theses.ulaval.ca/archimede/fichiers/22376/ch05.html
- ✓ www.yahoo.fr
- ✓ http://search.cpan.org/dist/Text-NSP/
- ✓ www.terrier.org
- ✓ http://www.programmez.com/telechargements.php?id_logiciel=258

Annexe

La plateforme de RI Terrier 2.1 

Cet annexe est principalement consacré à la présentation (Version 2.1), une plateforme de RI de haute performance et évolutive qui permet le développement rapide et à grande échelle de nouvelles applications de recherche d'information.

Introduction :

Terrier (TeRabyteRetrIeveR) a été développe par le département informatique de l'université de Galasgow. C'est un logiciel open source entièrement écrit en Java. Il est utilisé avec succès pour la recherche Ad hoc, la recherche Web et la recherche inter-langage dans des environnements centralises et distribues.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requetés basées sur le Framework DFR (Divergence FromRandomness). Comme tous les moteurs de recherche Terrier possède les principales facettes suivantes :

- ***Indexation :*** Permet l'extraction des termes des différents documents du corpus (basic indexed unit).
- ***Recherche :*** Permet de générer des résultats aux requêtes formulées par les utilisateurs.

1. Installation de Terrier :

1.1. Préalablement :

Pour pouvoir utiliser Terrier il est nécessaire d'installer une JRE (1.5.0 ou plus). La JDK peuvent être téléchargé sur le site de java
http://www.oracle.com/technetwork/java/javase/dowloads/index.html

1.2. Installation :

Après avoir téléchargé Terrier version 2.1 sur la page d'accueil du projet Terrier www.terrier.org ,créer un nouveau répertoire et dézippé Terrier dans ce dernier.

2. La structure des répertoires de Terrier :

Terrier contient un ensemble de répertoires et ils sont structures comme suit :

- bin\ : Contient les scripts nécessaires pour démarrer Terrier.
- doc\ : Contient la documentation relative a Terrier.
- etc\ : Contient les fichiers de configurations de Terrier.
- lib\ : Contient les classes compilées de Terrier et les différentes librairies externes utilises par Terrier.
- licences\ : Contient les informations sur la licence des différents composants inclus dans Terrier.
- share\ : Contient la liste des mots vides (stop wordlist).
- scr\ : Contient le code source d Terrier.
- var/ : Il contient les deux fichiers :
  - index\ : Contient les structures de donnees : fichier inverse, fichier lexicon, index direct, document index.
  - results\ : Contient les resultats de la recherche.

3. Les applications de Terrier :

Terrier vient avec trois applications :

- **Groupe (TREC) Terrier :**Ceci te permet facilement de classer, rechercher, et évaluer des résultats sur des collections de TREC.
- **Terrier Interactif :** Ceci vous permet à de faire la récupération interactive. C'est une manière rapide d'examiner Terrier.
- **Terrier Desktop :**Une application de bureau de recherche d'échantillon.

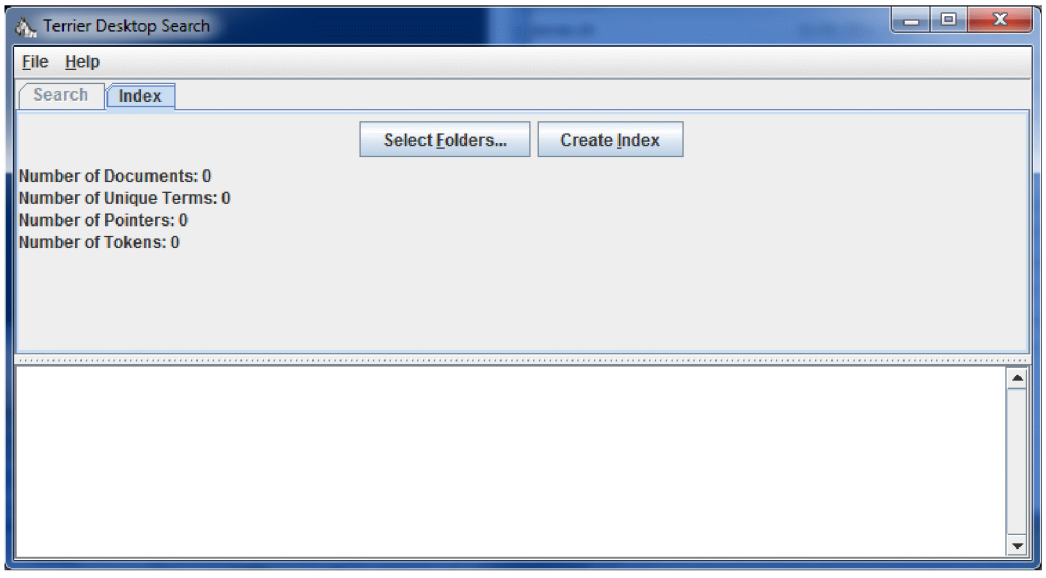


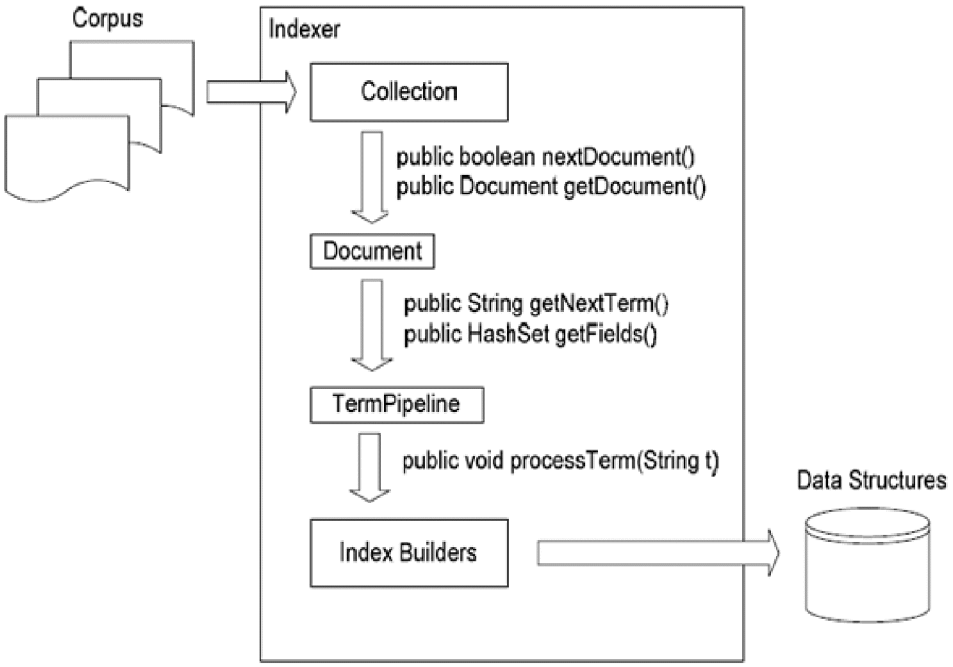
Figure1 : Présentation de l'interface Desktop Terrier.

4. L'API d'Indexation de Terrier :

L'indexation dans Terrier est devisée en quatre procédures :

1. Extraction de l'objet Document de la collection qui est générée par l'ensemble des corpus reçu en entrée par Terrier.
2. Parcourir chaque document de la collection pour extraire les termes ainsi que les informations relatives.
3. Traitement des termes extrait on utilisant TermPipelines (mots-vide, lemmatisation).
4. La construction de l'inde via l'utilisation de la classe Index Builder.

Le graphique ci-dessous donne une vue d'ensemble de l'interaction des composants principaux impliqués dans le processus d'indexation.



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

1. *Collection :*

Ce composant met en résume le concept le plus fondamental à l'indexation avec Terrier. Une Collection c'est-à-dire un ensemble de documents. la « collection » . C'est un objet qui représente le corpus, c'est-à-dire un ensemble de documents. Collection est une interface qui se trouve dans le package org.terier.indexing.
Utilisée généralement par terrier pour intégrer de nouvelles sources de données (nouveaucorpus) et cela en ajoutant une nouvelle classe Java qui implémente cette interface. Elle permet de parcourir un ensemble de documents et de renvoyer un objet document en faisantappel à sa méthode nextDocument().

Terrier offre plusieurs classes qui implémentent cette interface telle que SimpleFileCollection,SimpleMedlineXMLCollection, SimpleXMLCollection, TRECCollection,TRECUTFCollection, WARC018Collection et WARC09Collection.

2. *Document :*

C'est une interface qui se trouve dans le package org.terrier.indexing. Cette composanteenglobe le concept de Document. Les classes qui impl mentent cette interface s'occupe deparcourir les documents et dont extraire les différents termes. Terrier possède plusieursparseurs de documents, par exemple : HTMLDocument, FileDocument, MSExcelDocument, etc.

3. *TermPipeline :*

C'est une interface qui se trouve dans le packageuk.ac.gla.terrier.terms. Cette composante reçoit l'ensemble des termes extrait des documents. Cette étape subdivise en deux sous processus

3.1 Eliminations des mots vides (stopword-removal) :

Ce processus permet de supprimer les mots vides dans les documents tel que les pronoms,les connecteurs, lesespaces, etc. Le but est d'améliorer le résultat d'indexation.

3.2 La *normalisation (stemming) :*

Ce processus permet de retrouver les différentes formes d'un mot et les représenter sous formes unique, dont le but est d'avoir une forme suffisante à la représentation des différentes apparitions des autres formes.

Terrier utilise plusieurs méthodes de normalisation comme différentes variantes de l'algorithme de porter.

Apres les trois étapes successives précédentes, une étape de construction des structures d'index vient pour compléter le processus d'indexation

4. Indexer :

Cette composant est responsable de gérer le processus d'indexation, construire l'index et de son écriture et dans la structure de données appropriée.

Terrier offre deux types d'indexeur : BasicIndex, BlokIndex.

Terrier consiste en quatre structures de données :

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

Le tableau 1. Ci-dessus permet de représenter le contenu de ces différentes structures index :

Index Structure	***Contents***
Lexicon	Term Term id Document Frequency Term Frequency Byte offset in inverted file Bite offset in inverted file
Inverted Index	Document id Term Frequency Fields (#of fields bits) Blok Frequency [Block id]
Document Index	Document id Document Length Document Number Byte offset in direct file Bite offset in direct file
Direct Index	Term id Term Frequency Fields(#of fields bits) [Block id]

Tableau1: Présentation des différentes structures Index.

L'indexation en Terrier peut être configurée on changeant les propriétés approprié dans le fichier *ets\terrier.properties*. Chaque étage cité auparavant possède ces propres propriétés.

5. L'API de recherche de Terrier :

La recherche s'effectue sur les structures d'index précédemment crées et cela âpres analyse de la requête par le système. Cette opération vise a retourner des document triés selon l'ordre décroissant de leurs scores.

Terrie utilise trois composantes principales pour la recherche : Query, Manager, Matching.

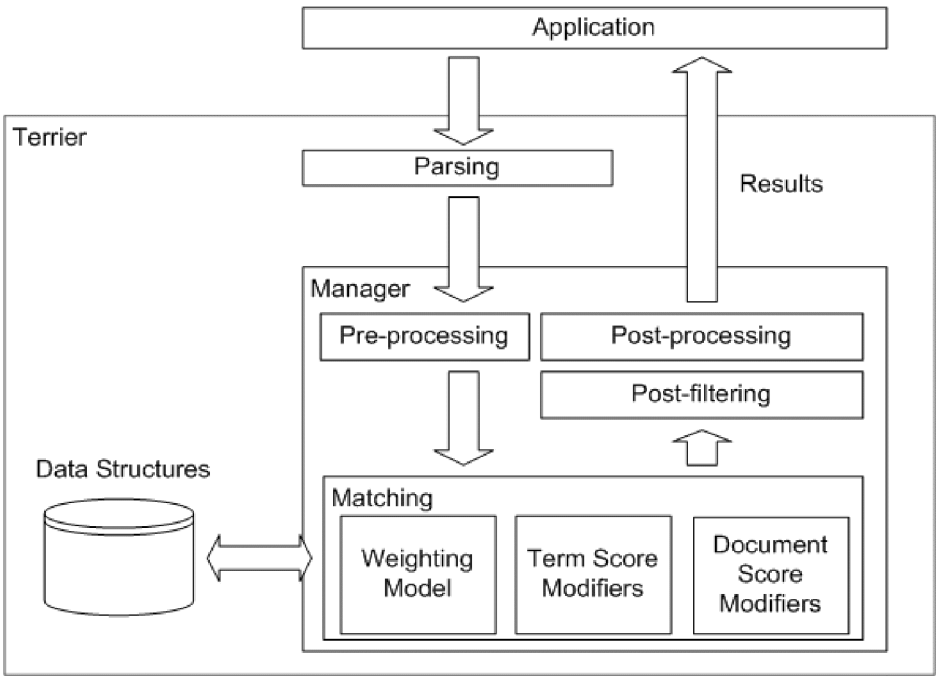


Figure 2. Présentation du processus de recherche de terrier

5.1 Query :

C'est l'entrée que l'application offre a Terrier (voir la Figure2) .Le module Matching est responsable de déterminer quel document correspond à une requête spécifique et attribut un score au document qui respect la requête.

Query est une classe abstraite,qui se trouve dans le package ac.gla.terrier.querying.parser et qui modélise les requêtes. Elle consiste soit en un sub-queries ou bien un query termes. Un objet Query est créé pour chaque requête.

Terrier offre un support pour chaque type de requêtes :

- SingleTermeQuery : modèle de requête avec un seul terme.
- MultitermeQuery : modèle de requête qui contient plus d'un terme.
- File Query : modèle de requête qualifié par un champ (field).

5.2 Manager :

C'est le module qui est chargé de la gestion de la recherche .Dans un premier niveau il lit chaque requête en utilisant le parseur approprié(l'analyse : qu'il s'agit de la reconnaissance des mots de la requête,elle se déroule de la même façon que pour les documents dans l'étape d'indexation. Puis il créait un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaires et en spécifiant un modèle de pondération puis il crée un fichier résultat (.Res) ou il associe à chaque requête les documents qui lui sont pertinents. Le résultat de chaque requête est donné par le second niveau de gestion.

Le second niveau de gestion est responsable de l'ordonnancement et de la coordination des opérations principales de haut niveau sur une seule requête.

Ces opérations sont : Pre-processing, Matching, Post-processing, Post-filtring

- **Pres-processing (pretraitement) :**

Cette étape est constituée des processus d'élimination des mots vides, de la lemmatisation et normalisation de termes. Ce prétraitement s'opère sur les termes précédemment extrais par l'analyse.

- **Matching :**

Le composant Matching est responsable de déterminer les documents correspondant à la requête spécifié et donne un score au document qui vérifie la requéte.il utilise l'interface *WeightingModele*pour assigner un score à chaque mot de la requête dans le document.

Terrier contient un nombre important de modèle de pondération .parmi ces modèles nous pouvons citer les suivants :

- Tf-IDF : ou la formule de TF est donnée par Robertson et IDF par Spark Jones.
- Lemur TF-IDF : le modèle proposé par Lemur.
- BM25 : le modèle probabiliste BM25.
- Hiemsta-LM : le modèle de langage proposé par Hiemstra.

Terrier dispose de plusieurs fonctions de modification de scores que ce soit pour les termes ou les documents, cela dans le but par exemple : d'accroitre la priorité d'un terme donné ou bien d'une phrase.

- **Post-processing /filtering (post-traitement/Filtrage) :**

Après l'étape précédente, nous obtenons un ensemble de documents avec leurs scores respectifs. A ce niveau deux autres processus sont disponibles pour améliorer les résultats, il s'agit du Post-traitement et Post-filtring.

❖ **Post-traitement:**

Ce traitement permet la modification du résultat de plusieurs manières, nous pouvons citer l'une d'elle.

✓ **La reformulation de la requête:**

Nous parlons dans ce cas de pseudo-relance feedback qui permet soit l'expansion de la requête avec l'ajout de nouveaux termes, soit la repondération des termes de la requête. Cette reformulation permet d'apporter de nouvelles informations, mais peut aussi engendrer du bruit. Terrier déploie différents modèles de reformulation de la requête, il permet aussi l'extension et l'ajout d'autres modèles.

Après à la fin de la reformulation, terrier doit à nouveau rappeler les fonctions d'appariement.

❖ **Post-Filtrage :**

Cette opération est plus simple que la précédente, elle permet soit l'inclusion ou l'exclusion d'un document. Elle est surtout utilisée pour les besoin de restriction de l'utilisateur pour un domaine particulier.

- **Resultat :**

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

6. Modification Terrier:

L'une des caractéristiques principales de terrier est,son extensibilité que ce soit sur ses processus d'indexation, dans ses modèles de pondération ou dans ses étapes de reformulation de la requête. Ces extensions peuvent aussi survenir dans le cas d'intégration de nouvelles collections de documents spéciales ou personnelles qui peuvent servir pour l'évaluation.

Terrier permet aussi la modification du code source pour intégrer tout changement nécessaire.Pour que toute modification soit prise en compte, il est nécessaire de recompiler tout le code source de Terrier.

7. Utilisation de Batch (TREC) Terrier :

Dans cette section nous décrivant l'utilisation de Batch(TREC) Terrier pour l'indexation, la recherche et l'évaluation sur le système d'exploitation Windows(XP) 7.

- ***Indexation :***

1. Allez au répertoire ou Terrier est installé ou utilisant la commande *cd* :

>>cd terrier 2.1

2. Initialisation de terrier pour l'indexation d'une nouvelle collection TREC:

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

3. Maintenant vous pouvez indexer la collection TREC.

>>.\bin\trec_terrier-i

Remarque :

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

- ***Recherche et Evaluation :***

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

1. Spécification du fichier de jugement de pertinence dans le fichier *etc\trec.qrels.*

```
#add the qrels files to use for evaluation

D:\Master2\terrier\qrel-wsj.txt
```

Figure3: Exemple d'un fichier trec.qrels

2. Spécification des fichiers contenant les requétes (topics file) dans le fichier *etc\trec.topics.list.*

```
#add the topics files to use for querying

D:\Master2\terrier\topics.51-100
```

Figure 4: Exemple d'un fichier trec.topics.list

3. Spécification du modèle de pondération (ex. TF_IDF) à utiliser dans le fichier *etc\trec.models.*

Apres que ses étapes soient faites la recherche ou l'évaluation peuvent être lancé sur Terrier.

4. Pour lancer la recherche dans Terrier il suffit d'exécuter la commande :
.\bin\trec_terrier-r

5. Les résultants de la recherche peuvent être évalués en exécutant la commande :
.\bin\trec_terrier-e