

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Génie Electrique et d'Informatique
Département d'Informatique



Mémoire



En vue d'obtention du diplôme de Master en Informatique
Option : Ingénierie des Systèmes d'information

Thème
***Implémentation et évaluation d'une méthode de
réordonnancement de documents se basant sur
la clarté du document.***

Proposé et dirigé par :

Réalisé par :

Mr A. HAMMACHE Mr HADDOUCHE Aghilas

Mr HAMMACHE Faredj

Promotion 2013/2014

Remerciements

L'achèvement de tout travail procure une grande satisfaction. Il est l'occasion de se remémorer les étapes passées et les personnes qui y ont contribué. Nous tenons à remercier très sincèrement notre promoteur Monsieur HAMMACHE Arezki pour avoir dirigé nos recherches, pour sa gentillesse, sa disponibilité, ainsi que pour son aide. Ses conseils et ses remarques constructives nous ont permis d'améliorer grandement la qualité de ce mémoire.

*Nous remercions également nos collègues du Département
d'informatique de l'Université d'UMMTO.*

*Finalement, nos remerciements vont à toute personne ayant contribué
de près ou de loin à l'aboutissement de ce modeste travail.*

Dédicaces

Je dédie ce travail :

A mes parents qui m'ont soutenu durant toute ma vie.

A vous, mes sœurs Ourdia et Sonia.

A vous mes oncles.

A vous mes grand pere.

A vous, Lilia, Faredj, Massinissa, Menad, Sofian, Khaled,

Hafidh et Lyazid avec qui je passe tous les bons moments de ma vie.

Aghilas

Dédicaces

Je dédie ce modeste travail :

A mes parents et à mes grands-parents.

A mes frères et sœurs.

A tous mes proches.

A tous mes amis(es).

A mon binôme Aghilas HADDOUCHE et à toute sa famille.

Faredj

Sommaire

Introduction générale.....	1
----------------------------	---

Chapitre I. La recherche d'Information

I.1. Introduction.....	3
I.2. La recherche d'information.....	3
I.2.1. Définition.....	3
I.2.2. Concepts de base de la recherche d'information.....	3
I.3. Notions de base.....	5
I.3.1. Collection de documents.....	5
I.3.2. Requête.....	5
I.3.3. Pertinence.....	5
I.4. Le processus d'indexation.....	6
I.4.1. L'analyse lexicale.....	7
I.4.2. L'élimination des mots vides.....	7
I.4.3. La normalisation.....	7
I.4.4. Le choix des descripteurs.....	8
I.4.5. La création de l'index.....	8
I.5. Les modèles de recherche d'information.....	9
I.5.1. Le modèle booléen.....	9
I.5.2. Le modèle vectoriel.....	10
I.5.3. Les modèles probabilistes.....	13
I.5.3.1. Le modèle probabiliste de base.....	13
I.5.3.2. Le modèle de langue.....	15
I.6. La reformulation de la requête.....	15
I.6.1. Reformulation par réinjection de la pertinence.....	17
I.6.2. La réinjection par pseudo feedback (réinjection aveugle).....	18
I.7. Evaluation des SRI.....	18
I.7.1 Les collections de test.....	19
I.7.2 Mesures d'évaluation de SRI.....	21
I.8. Conclusion.....	25

Chapitre II. Modèles de Langue pour la RI

II.1. Introduction.....	26
II.2. Les modèles de langue en linguistique informatique.....	26
II.2.1. Idée de base.....	26
II.2.2. Les techniques de lissage.....	28
II.2.2.1. Lissage de Laplace.....	28
II.2.2.2. Lissage de Good-Turing.....	29
II.2.2.3. Lissage par interpolation (Jelinek-Mercer).....	29
II.2.2.4. Lissage de Dirichlet.....	30
II.3. Modèles de langue et recherche d'information.....	30
II.3.1. Approches d'exploitation des modèles de langue en RI.....	31
II.3.1.1. Génération de la requête par le modèle de document (QueryLikelihoodModels).....	32
II.3.1.2. Génération de document à partir du modèle de la requête (Document Likelihood Model).....	34
II.3.1.3. Comparaison des modèles de requête et du document.....	35
II.4. Intégration des caractéristiques indépendantes du contenu de document dans le modèle de langue.....	36
II.4.1. La taille du document.....	36
II.4.2. La date de création du document.....	37
II.4.3. La structure des liens.....	38
II.4.4. L'hypothèse de recommandation.....	38
II.5. Conclusion.....	41

Chapitre III. Approche implémentée et évaluation

III.1. Introduction.....	42
III.2. Présentation de l'approche implémentée.....	42
III.2.1. La phase calcul du score de clarté du document.....	43
III.2.1.1. Définition de la clarté.....	43
III.2.1.2. Clarté de la requête.....	43
III.2.1.3. Clarté du document.....	43
III.2.2. La phase de réordonnancement.....	44
III.3. Exemple illustratif.....	44

III.4. les outils utilisés.....	45
III.4.1. Le système Terrier.....	45
III.4.2. Java.....	46
III.4.3. Netbeans.....	47
III.5. Interface de l'application.....	49
III.5.1. Création de l'index (bouton Indexation).....	49
III.5.2. La recherche (bouton Recherche).....	50
III.5.3. Bouton Prior.....	50
III.5.4. Bouton Test-Prior.....	50
III.6. Résultats d'évaluation.....	52
III.6.1. Les collections et les requêtes utilisées.....	52
III.6.2 Résultats obtenus.....	53
III.6.2.1. En utilisant le facteur clarté de documents.....	53
III.6.2.2. En utilisant le facteur taille de documents.....	56
III.7. Conclusion.....	59
Conclusion générale& perspectives.....	60

Bibliographie

Bibliographie

Annexes

Annexe : La plateforme terrier

Liste des figures

Figure I.1 Architecture générale d'un Système de Recherche d'Information.....	4
Figure I.2 Exemple d'un document TREC.....	20
Figure I.3 Exemple d'une requête TREC.....	21
Figure I.4 Courbe de rappel et précision.....	23
Figure II.1 : Modèle de Markov à deux états.....	34
Figure II.2 : Les pages Hubs et Autorités.....	40
Figure III.1 : Notre approche.....	42
Figure III.2 : Architecture de terrier.....	46
Figure III.3 : IinterfaceNetbeans IDE.....	48
Figure III.4 : Interface Terrier.....	49
Figure III.5 : Extrait du code qui calcule la clarté du document.....	50
Figure III.6 : La class Modif_taille.....	51
Figure III.7 : La class Modif_clarté.....	51
Figure III.8 : Exemple de requête (104) de la collection AP88.....	52
Figure III.9 : Graphe détaillant les résultats d'évaluation de la recherche en utilisant le facteur clarté de document.....	55
Figure III.10 : Graphe illustrant la précision moyenne de la recherche classique et la recherche en utilisant la taille de document.....	58

Liste des tableaux

Tableau I.1 Les mesures de similarité utilisées dans le modèle vectoriel.....	12
Tableau I.2 Exemple de calcul de rappel et de précision pour une requête.....	22
Tableau III.1 : résultat de l'exemple illustratif.....	44
Tableau III.2 : statistiques sur la collection et les topics utilisés.....	52
Tableau III.3 : Résultats globaux de l'évaluation de la recherche classique et de la recherche utilisant le modèle clarté de document.....	53
Tableau III.4 : Résultats d'évaluation en utilisant le facteur clarté de document et en variant le coefficient λ	53
Tableau III.5 : Résumé des résultats d'évaluation en utilisant le facteur clarté de document pour $\lambda=2$	55
Tableau III.6 : Résultats globaux de l'évaluation de la recherche classique et de la recherche utilisant le modèle taille de document.....	56
Tableau III.7 : Résultats d'évaluation en utilisant le facteur taille de document et en variant le coefficient λ	56
Tableau III.8 : Résumé des résultats d'évaluation en utilisant le facteur taille de document pour $\lambda=0,5$	58



Introduction générale

Introduction générale :

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

L'objectif d'un SRI est d'aiguiller la recherche dans le fond documentaire, en direction de l'information pertinente relativement à un besoin en information exprimé par une requête utilisateur.

Cependant dans la majorité des SRI, uniquement le contenu textuel est utilisé pour établir la correspondance entre les documents et les requêtes. D'autres caractéristiques indépendantes de la requêtes peuvent être utilisé pour l'ordonnancement a priori des documents, tel que le pagerank dans le web etc.

C'est dans ce contexte que s'insère notre travail; plus précisément à exploiter la caractéristique de la clarté du document vis-à-vis de la collection en supposant qu'un document claire est a priori pertinent.

L'organisation retenue pour la présentation de notre travail s'articule en trois chapitres. Dans le chapitre 1 nous décrivons deux points essentiels. Tout d'abord, nous donnons les concepts de base de la recherche d'information. On y trouve les notions de requête, de document, de collection et de pertinence et le processus d'indexation. Nous décrivons aussi les différents modèles de la recherche d'information en particulier le modèle booléen, le modèle vectoriel et le modèle probabiliste. Le deuxième point traité dans ce chapitre concerne l'évaluation des systèmes de recherche d'information. Au second chapitre nous décrivons les modèles de langue en linguistique informatique avec l'idée de base et les techniques de lissage et aussi les approches utilisés pour caractériser les documents. Etant le but et de réordonner des documents à l'aide

d'un facteur qui est le score de clarté, au troisième et dernier chapitre nous présentons notre approche qui est implémentée en utilisant la plateforme terrier, ainsi que nous présentons les outils de développement et enfin des résultats expérimentaux obtenus sur la collection de test TREC AP88.



Chapitre 1

La recherche d'Information

I.1. Introduction

La recherche d'informations ou (RI) est l'une des disciplines de l'informatique qui est née juste quelques années après l'invention des ordinateurs (les années 1940) du besoin de localiser les documents liés aux applications dans les bibliothèques qui faisait partie de ce qu'on appelait avant « automatisation des bibliothèques ». Le principe de ces applications consistait à établir des représentations de documents par la construction d'index afin d'en récupérer les informations.

Mais face aux quantités croissantes de l'information générée et gérée chaque jours, soit en local ou sur les différents réseaux (intranet, internet), l'accès à cette dernière devient de plus en plus exigeant en matière de temps – et qui dit temps, dit argent – et cela au détriment de l'efficacité et de la pertinence des résultats. Ce qui imposait alors le domaine de la RI comme la solution triviale remédiant aux problèmes suscités.

I.2. La recherche d'information

I.2.1. Définition

La recherche d'information peut se définir comme l'ensemble méthodes et procédures ayant pour objet de retrouver dans des fonds documentaires des documents pertinents répondant à une requête précise.

I.2.2. Concepts de base de la recherche d'information

Le rôle d'un Système de Recherche d'Information (SRI) est de mettre en œuvre des techniques et des moyens permettant de retourner les documents pertinents d'une collection en réponse à un besoin en information d'un utilisateur, exprimée par un langage de requêtes qui peut être le langage naturel, une liste de mots clés ou un langage booléen.[3]

Afin d'atteindre cet objectif, un processus d'indexation des documents de la collection est effectué. Il permet de construire une représentation synthétique des documents, appelée index. Lorsque l'utilisateur formule sa requête un processus

similaire est effectué sur la requête. Il consiste à analyser la requête et établir une représentation interne. Puis, le système établit une correspondance entre la représentation de la requête et la représentation des documents (index) pour sélectionner et présenter les documents qui répondent le mieux au besoin de l'utilisateur (les documents pertinents).

Le SRI s'appuie sur des modèles de RI pour établir cette correspondance entre les documents et la requête.

L'architecture générale d'un SRI illustrée par la figure I.1 fait ressortir des éléments constitutifs tels que : le document, le besoin en information, la requête et la pertinence, ainsi que trois principales fonctionnalités : l'indexation, la recherche et la reformulation de la requête.

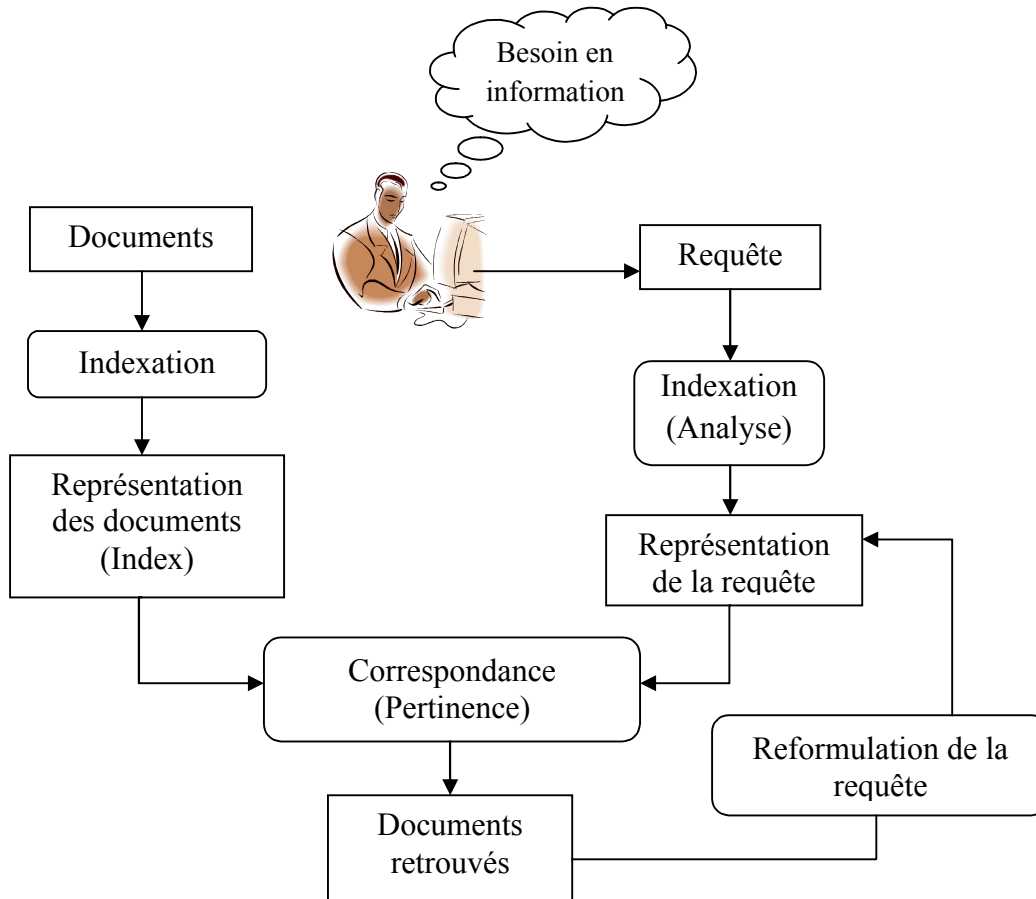


Figure I.1 Architecture générale d'un Système de Recherche d'Information.

I.3. Notions de base

I.3.1. Collection de documents

C'est le fond informatif dont le contenu est accessible, compréhensible et exploitable par l'utilisateur, or un rassemblement de granules documentaires (unité d'information renvoyer à l'utilisateur). Un granule de document peut représenter soit tout ou une partie d'un document qui est retournée en réponse à une requête de l'utilisateur.

I.3.2. Requête

La requête représente l'interface entre l'utilisateur et le SRI, elle exprime le besoin d'information d'un utilisateur. Elle peut être exprimée selon différent langages de requêtes : le langage naturel, le langage à base de mot clés ou le langage booléen. Le langage le plus utilisé est le langage naturel.

Pour répondre au besoin en information de l'utilisateur un Système de Recherche d'Information (SRI) met en évidence un certain nombre de processus qui accomplissent la mise en correspondance des informations renfermées dans un fond documentaire d'une part, et celles exprimées par les utilisateurs d'autres part.

I.3.3. Pertinence

La pertinence est une notion fondamentale et cruciale dans le domaine de la RI. Cependant, la définition de cette notion complexe n'est pas simple, car elle fait intervenir plusieurs notions[6][35]. Basiquement, elle peut être définie comme la correspondance entre un document et une requête ou encore comme une mesure d'informativité du document à la requête.

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

La pertinence Système[14] est souvent présentée par un score attribué par le SRI afin d'évaluer l'adéquation du contenu des documents vis-à-vis de celui de la requête. Ce type de pertinence est objectif et déterministe.

Pertinence utilisateur[22][35] [45] quant à elle, se traduit par les jugements de pertinence de l'utilisateur sur les documents fournis par le SRI en réponse à

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

I.4. Le processus d'indexation

Pour que la recherche d'information se réalise avec des coûts acceptables, il convient d'effectuer une opération fondamentale sur les documents de la collection. Cette opération est nommée indexation[3] [33].. Elle consiste à associer à chaque document une liste de mots clés appelée aussi descripteur, susceptible de représenter au mieux le contenu sémantique des documents.

La finalité de l'indexation est donc de produire une représentation synthétique des documents, formé de termes, ces termes peuvent être extraits de trois manières :

Manuelle : chaque document de la collection est analysé par un spécialiste du domaine ou un documentaliste. L'indexation manuelle assure une meilleure précision dans les documents restitués par le SRI en réponse aux requêtes des utilisateurs[38].

Néanmoins, cette indexation présente un certain nombre d'inconvénients liés notamment à l'effort et le prix qu'elle exige (en temps et en nombres de personnes).

De plus, cette indexation est subjective, qui est liée au facteur humain, différents spécialistes peuvent indexer un document avec des termes différents. Il se peut même arriver qu'un spécialiste indexe différemment un document, à différents moments.

Semi-automatique : la tâche d'indexation est réalisée ici conjointement par un programme informatique et un spécialiste du domaine[24]. Le choix final des descripteurs revient à l'indexeur humain. Dans ce type d'indexation un langage d'indexation contrôlé est généralement utilisé.

Automatique : dans ce cas, l'indexation est entièrement automatisée. Elle est réalisée par un programme informatique et elle passe par un ensemble d'étapes pour créer d'une façon automatique l'index. Ces étapes sont : l'analyse lexicale, l'élimination des mots vides, la normalisation (lemmatisation ou radicalisation), la sélection des descripteurs, le calcul de statistiques sur les descripteurs et les documents (fréquence d'apparition d'un descripteur dans un document et dans la collection, la taille de chaque document, etc.) et enfin la création de l'index et éventuellement sa compression.

I.4.1. L'analyse lexicale

Elle permet de convertir un texte de document en une liste de termes. Un terme est un groupe de caractères constituant un mot significatif[20]. L'analyse lexicale permet de reconnaître les espaces de séparation des mots, les chiffres, les ponctuations, etc.

I.4.2. L'élimination des mots vides

Les mots vides (article, proposition, conjonction, etc.) sont des mots non significatifs dans un document, car ils ne traitent pas le sujet du document.

On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste préétablie de mots vides (aussi appelée *antidictionnaire* ou *stop-list*),
- L'élimination des mots ayant une fréquence qui dépasse un certain seuil dans la collection.

L'élimination des mots vides réduit la taille de l'index, ce qui améliore le temps de réponse du système. Cependant, elle peut réduire le taux de rappel, en réponse à des requêtes bien spécifiques (par exemple, la requête *be or not to be*).

I.4.3. La normalisation

La normalisation consiste à représenter les différentes variantes d'un terme par un format unique appelé lemme ou racine. Ce qui a pour effet de réduire la taille de l'index. Plusieurs stratégies de normalisation sont utilisées : la table de

correspondance, l'élimination des affixes (l'algorithme de Porter[53]), la troncature, l'utilisation des N-grammes.[65]

L'inconvénient majeur de cette opération est qu'elle supprime dans certains cas la sémantique des termes originaux, c'est le cas par exemple des termes *derivate/derive*, *activate/active*, normalisés par l'algorithme de Porter.

I.4.4. Le choix des descripteurs

Elle consiste à déterminer le type d'unités élémentaires pour représenter les documents. On parle aussi de descripteur. L'objectif est d'avoir une représentation des documents permettant une moindre perte d'information sémantique possible. On distingue plusieurs types de descripteurs.[4]

- **Les mots simples** : les mots simples du texte de document en éliminant les mots vides,
- **Les lemmes** ou les racines des mots extraits.
- **Les N-grammes** : qui sont une représentation originale d'un texte en séquence de N caractères consécutifs. On trouve des utilisations de bi-grammes et trigrammes dans la recherche d'information.
- **Les mots composés** : groupes de mots ou expression (phrase en anglais) sont souvent plus riches sémantiquement que les mots qui les composent pris séparément. Par exemple, le mot composé "imprimante laser" est plus précis que "imprimante" et "laser" pris isolément. Cet argument a conduit à leur large utilisation en RI.

I.4.5. La création de l'index

Au terme du processus d'indexation, un ensemble de structure de données sont créés. Ces dernières permettent un accès efficace à la représentation des documents. Le fichier inverse est la structure de données la plus utilisée[3] [33], il enregistre pour chaque descripteur les identificateurs des documents qui le contiennent et sa fréquence dans chacun de ces documents.

Généralement, les structures de données sont compressées avant d'être enregistrées sur le disque, ce qui permet de réduire la taille de l'index. Parmi les

méthodes de compression utilisées on peut citer la méthode Elias Gamma [203] qui opère au niveau bit requérant ainsi beaucoup d'opérations pour la compression et la décompression. D'autres méthodes plus efficaces, opérant au niveau octet ont été proposées dans.[43]

D'autres caractéristiques sur un document, permettant de calculer la pertinence a priori d'un document indépendamment de toute requête, peuvent être calculées et stockées a ce stade.[17]

I.5. Les modèles de recherche d'information

Un modèle de RI fournit une interprétation théorique de la notion de pertinence. Plusieurs modèles de RI ont été proposés dans la littérature, ils s'appuient sur des cadres théoriques différents, théorie des ensembles, algèbre, probabilités, etc. Globalement, on distingue trois principales catégories de modèles : modèles booléens, modèles vectoriels et modèles probabilistes.[18]

I.5.1. Le modèle booléen

Les premiers SRI développés sont basés sur le modèle booléen, même aujourd'hui beaucoup de systèmes commerciaux (moteurs de recherche) utilisent le modèle booléen. Cela est dû à la simplicité et à la rapidité de sa mise en œuvre. Le modèle booléen est basé sur la théorie des ensembles et l'algèbre de Boole. Dans ce modèle, un document d est représenté par un ensemble de mots-clés (termes) ou encore un vecteur booléen. La requête q de l'utilisateur est représentée par une expression logique, composée de termes reliés par des opérateurs logiques : ET (\wedge), OU (\vee) et SAUF (\neg).

L'appariement (RSV) entre une requête et un document est un appariement exact, autrement dit si un document implique au sens logique la requête alors le document est pertinent. Sinon, il est considéré non pertinent. La correspondance entre document et requête est déterminée comme suit :

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

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

- Les documents retournés à l'utilisateur ne sont pas ordonnés selon leur pertinence.
- La représentation binaire d'un terme dans un document est peu informative, car elle ne renseigne ni sur la fréquence du terme dans le document ni sur la longueur de document, qui peuvent constituer des informations importantes pour la RI.
- Il est difficile pour les utilisateurs de formuler de bonnes requêtes. Par conséquent, l'ensemble des documents trouvés est souvent trop grand, pour les requêtes courtes, ou complètement vide dans le cas de requêtes longues.
- Ce modèle ne supporte pas la réinjection de pertinence.
- Les tests effectués sur des collections d'évaluation standards de RI ont montré que les systèmes booléens sont d'une efficacité de recherche inférieure.

Afin de remédier à certains problèmes de ce modèle, des extensions ont été proposées, parmi lesquelles on trouve : le modèle booléen basé sur la théorie des ensembles flous[26] [37], le modèle booléen étendu.[42]

I.5.2. Le modèle vectoriel

Le modèle vectoriel de base a été introduit par Salton[41], concrétisé dans le cadre du système SMART. Ce modèle se base sur une formalisation géométrique. En effet, les documents et les requêtes sont représentés dans un même espace, défini par un ensemble de dimensions, chaque dimension représente un terme d'indexation. Les requêtes et les documents sont alors représentés par des vecteurs, dont les composantes représentent le poids du terme d'indexation considéré dans le document (la requête). Formellement, si on a un espace T de termes d'indexation de dimension n , $n = t_1, t_2, \dots, t_n$. Un document d_i est représenté par un vecteur $(w_{i1}, w_{i2}, \dots, w_{in})$. Une requête q par un vecteur $(w_{q1}, w_{q2}, \dots, w_{qn})$.

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

Le modèle vectoriel offre des moyens pour la prise en compte du poids de terme dans le document. Dans la littérature, plusieurs schémas de pondération ont été proposés. La majorité de ces schémas prennent en compte la pondération locale et la pondération globale.[31]

La pondération locale permet de mesurer l'importance du terme dans le document. Elle prend en compte les informations locales du terme qui ne dépendent que du document. Elle correspond en général à une fonction de la fréquence d'occurrence du terme dans le document (noté tf pour *term frequency*), exprimée ainsi :

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

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

Quant à la pondération globale, elle prend en compte les informations concernant le terme dans la collection. Un poids plus important doit être assigné aux termes qui apparaissent moins fréquemment dans la collection. Car les termes qui apparaissent dans de nombreux documents de la collection ne permettent pas de distinguer les documents pertinents des documents non pertinents (i.e. peu utile pour la discrimination). Un facteur de pondération globale est alors introduit. Ce facteur nommé *idf* (*inverted document frequency*), dépend d'une manière inverse de la fréquence en document du terme et exprimé comme suit :

$$idf = \frac{1}{\log(n_i)} \quad (I.3)$$

Où n_i est la fréquence en document du terme considéré, et N est le nombre total de documents dans la collection.

Les fonctions de pondération combinant la pondération locale et globale sont référencées sous le nom de la mesure $tf*idf$. Cette mesure donne une bonne approximation de l'importance du terme dans les collections de documents de taille homogène. Cependant, un facteur important est ignoré, la taille du document. En effet, la mesure $(tf*idf)$ ainsi définie favorise les documents longs, car ils ont

tendance à répéter le même terme, ce qui accroît leur fréquence, par conséquent augmentent la similarité de ces documents vis-à-vis de la requête.

Pour remédier à ce problème, des travaux ont proposé d'intégrer la taille du document dans les formules de pondération, comme facteur de normalisation.[39] [46].

L'appariement document-requête dans le modèle vectoriel, consiste à trouver les vecteurs documents qui s'approchent le plus de vecteur de la requête. Cet appariement est obtenu par l'évaluation de la distance entre les deux vecteurs. Plusieurs mesures de similarité ont été définies[47], dont les plus courantes sont décrites dans le Tableau I.1 ci-dessous.

Mesures	Formules
Le produit scalaire	$(q, d) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$
La mesure de cosinus	$(q, d) = \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$ $= \frac{\sum_{i=1}^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$
La mesure de Dice	$(q, d) = \frac{2 \times \sum_{i=1}^n q_i d_i}{\sum_{i=1}^n q_i^2 + \sum_{i=1}^n d_i^2}$
La mesure de Jaccard	$(q, d) = \frac{\sum_{i=1}^n q_i d_i}{\sum_{i=1}^n q_i^2 + \sum_{i=1}^n d_i^2 - \sum_{i=1}^n q_i d_i}$

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

Le modèle vectoriel caractérisé par sa prise en compte du poids des termes dans les documents, permet de retrouver des documents qui répondent partiellement à une requête. De plus, ce modèle offre un moyen facile pour classer les résultats d'une recherche, qui est basée sur la similarité potentielle entre documents et

requête. L'inconvénient majeur de modèle vectoriel est qu'il repose sur l'hypothèse de l'indépendance des termes d'indexation, or ces termes dans les documents sont souvent sémantiquement liés.

Plusieurs variantes du modèle vectoriel ont été proposées, pour remédier à cette limitation, c'est-à-dire prendre en compte la dépendance entre termes d'indexation. Parmi elles, on trouve, le modèle vectoriel généralisé[51], le modèle LSI (Latent SemanticIndexing) [5][19]et le modèle connexionniste[8] [27].

I.5.3. Les modèles probabilistes

I.5.3.1.Le modèle probabiliste de base

Le modèle probabiliste est fondé sur la théorie des probabilités[54]. Il trie les documents selon leur probabilité de pertinence vis-à-vis d'une requête. La fonction de classement (tri) de ce modèle est exprimée ainsi :

$$(q, d) = \frac{(q, d)}{(q, d)}, \quad (I.4)$$

L'idée de base de cette fonction est de sélectionner les documents ayant à la fois une forte probabilité d'être pertinents et une faible probabilité d'être non pertinents à la requête.

Où (q, d) et (q, \bar{d}) : la probabilité qu'un document d soit pertinent (vis-à-vis de la requête q) (respectivement non pertinent(\bar{d})).

En appliquant la formule de bayes pour les deux probabilités on obtient :

$$(q, d) = \frac{(q, d) \cdot (d, q)}{(q, d)} \quad (I.5)$$

$$(q, \bar{d}) = \frac{(q, \bar{d}) \cdot (\bar{d}, q)}{(q, \bar{d})}, \quad (I.6)$$

Où :

(d, q) est la probabilité de choisir le document d , on considère qu'elle est constante ;

(d, q) indique la probabilité que d fait partie des documents pertinents pour la requête ;

$(p_{ij} | D_i)$ indique la probabilité que D_i fait partie des documents non pertinents pour la requête Q_j ;

$(p_{ij} | D_i)$ et $(p_{ij} | \bar{D}_i)$ indiquent respectivement la probabilité de pertinence et de non pertinence d'un document quelconque (avec $(p_{ij} | D_i) + (p_{ij} | \bar{D}_i) = 1$) qui sont fixes.

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

$$(D_i, Q_j) = \frac{(p_{ij} | D_i)}{(p_{ij} | \bar{D}_i)}, \quad (I.7)$$

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

$$(p_{ij} | D_i) = \prod_{t \in T_j} p_{it} | D_i, \cdot \prod_{t \notin T_j} (1 - p_{it} | D_i), \quad (I.8)$$

$$(p_{ij} | \bar{D}_i) = \prod_{t \in T_j} p_{it} | \bar{D}_i, \cdot \prod_{t \notin T_j} (1 - p_{it} | \bar{D}_i), \quad (I.9)$$

Où

$p_{it} | D_i$, Indique la probabilité d'apparition du terme t sachant que le document appartient à l'ensemble des documents pertinents et $p_{it} | \bar{D}_i$, indique la probabilité d'apparition du terme t sachant que le document appartient à l'ensemble des documents non pertinents.

En posant $p_{it} = p_{it} | D_i$, $\bar{p}_{it} = p_{it} | \bar{D}_i$, et $p_{it} = 1$ pour les termes qui n'apparaissent pas dans la requête, et après simplification, le calcul du score de correspondance entre un document D_i et une requête peut être exprimé ainsi :

$$(D_i, Q_j) = \sum_{t \in T_j} \log \left[\frac{p_{it}}{\bar{p}_{it}} \right] \quad (I.10)$$

Afin de classer les documents avec cette formule, il faut estimer les valeurs des deux probabilités p_{it} et \bar{p}_{it} . En l'absence de collection (documents) d'apprentissage ; on peut attribuer la valeur fixe à p_{it} comme par exemple 0.5 ; comme elles peuvent être estimées à l'aide de l'avis de l'utilisateur sur les résultats d'une première recherche (réinjection de pertinence).

I.5.3.2. Le modèle de langue

Les modèles statistiques de langue sont exploités avec beaucoup de succès dans divers domaines : la reconnaissance de la parole[25], la traduction automatique[11] [34], la recherche d'information[23] [28], etc.

L'utilisation des modèles de langue en RI remonte à 1998[36]. Le principe de ce modèle consiste à construire un modèle de langue pour chaque document, soit $P(d)$, puis de calculer la probabilité qu'une requête q puisse être générée par le modèle de langue du document, soit $P(q|d)$. [10] [36] Le modèle de langue utilisé est souvent le modèle uni-gramme, la probabilité $P(t_i|d)$ est alors exprimée ainsi :

$$P(q|d) = \prod_{i=1}^n P(t_i|d) \quad (I.11)$$

$P(t_i|d)$ Peut être estimée en se basant sur l'estimation maximale de vraisemblance (maximum likelihood estimation). Elle est donnée par :

$$P(t_i|d) = \frac{f(t_i, d)}{\sum_{t \in V} f(t, d)} \quad (I.12)$$

Où $f(t, d)$ est la fréquence du terme t dans le document d .

Pour remédier au problème posé par les mots de la requête absents dans le document, qui ont pour effet d'avoir la probabilité $P(t_i|d)$ nulle ; des techniques de lissage (smoothing) sont utilisées, dont le lissage de Laplace (ajouter-un), le lissage de Good-Turing, le lissage

Backoff, le lissage par interpolation, etc[59]. Leur principe consiste à assigner des probabilités non nulles aux termes, qui n'apparaissent pas dans les documents.[10]

I.6. La reformulation de la requête

La reformulation de la requête est un processus permettant la construction d'une nouvelle requête, plus à même de représenter les besoins en information de l'utilisateur. Elle est souvent opérée par ajout et/ou réévaluation des poids des termes de la requête initiale.

Les techniques de reformulation de la requête peuvent être classées en tenant en compte de plusieurs paramètres [12]:

- Les sources de données utilisées pour l'expansion de requête ;

- La méthode de sélection des termes d'expansion : la relation de cooccurrence, les mesures d'information, les techniques de classification, etc.
- La sélection des termes d'expansion en considérant chaque terme de la requête individuellement, ou la requête dans son ensemble ;
- La représentation de la requête (document) comme un ensemble de mots simples (sac de mots) ou une représentation prenant en compte les relations de proximité entre termes ;
- Le type de terme d'expansion (mot simple ou mot composé).
- L'intervention de l'utilisateur dans le processus d'expansion de la requête (automatique, manuelle, semi-automatique).

Différentes sources de données sont utilisées pour reformuler la requête initiale. Elles peuvent être [33]: (1) des ressources externes, telles que les ontologies, les thésaurus et la relation de cooccurrence entre termes dans la collection. Les méthodes basées sur ces sources sont dites méthodes globales. (2) Les documents résultants de la première recherche; les méthodes basées sur ces sources sont dites méthodes locales. Ces méthodes sont également connues sous le nom de réinjection de pertinence. La réinjection de pertinence a montré son efficacité avec différents modèles de la RI [30] [32] et a affiché de meilleurs résultats que les méthodes globales.

D'autres sources de données ont été utilisées, on peut citer les textes d'ancre [16], les logs des moteurs de recherche, les liens dans les documents Wikipedia [2] et les FAQs.

D'autres études utilisent des informations complémentaires fournies par l'utilisateur pour l'extension de la requête telles que, les balises, les images et les catégories. [48]

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

I.6.1. Reformulation par réinjection de la pertinence

Ces méthodes impliquent que l'utilisateur doit sélectionner les documents qu'il considère pertinents à partir des résultats issus de sa requête initiale. Ce jugement de pertinence de l'utilisateur est ensuite exploité pour reformuler la requête initiale en modifiant le poids des termes qu'elle contient et/ou en ajoutant de nouveaux termes considérés utiles pour retrouver des documents pertinents. La technique de réinjection de pertinence a été mise en place à l'origine dans le modèle vectoriel. Rocchio[40] a proposé le modèle de reformulation de requête suivant :

$$Q = \frac{1}{|R|} \sum_{r \in R} r + \frac{\lambda}{1+\lambda} \frac{1}{|R'|} \sum_{r' \in R'} r' \quad (I.13)$$

Où :

Q est le vecteur de la nouvelle requête (reformulée) ;

Q' est le vecteur de la requête originale ;

R est l'ensemble des vecteurs r des documents jugés pertinents par l'utilisateur ;

R' est l'ensemble des vecteurs r' des documents jugés non pertinents par l'utilisateur ;

λ , sont les paramètres de la reformulation.

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

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

$$w_{qj} = \frac{r_j + \lambda w'_{qj}}{r + \lambda} \quad (I.14)$$

Où : r représente le nombre de documents pertinents ;

r' est le nombre de documents pertinents contenant le terme q_j ;

n_j est le nombre de documents contenant le terme q_j ;

n est le nombre total de documents dans la collection.

I.6.2. La réinjection par pseudo feedback (réinjection aveugle)

Ces méthodes de reformulation nommées aussi, pseudo-réinjection de pertinence (ou blind) sont effectuées de manière automatique. Elles se basent sur l'hypothèse que les documents les mieux classés (les premiers) sont considérés comme pertinents. Le système utilise alors les premiers documents pour reformuler la requête.

La variante de la formule de Rocchio pour la réinjection automatique de la requête est exprimée par la formule suivante :

$$= \cdot + \frac{1}{N} \sum_{i=1}^N \epsilon \quad (I.15)$$

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

Plusieurs travaux[9] [21] ont tenté d'évaluer l'impact de la pseudo-réinjection, en variant le nombre de nombre de termes à rajouter à la requête. Ils montrent que la performance du système est obtenue lorsque la requête est construite entre 20 et 40 termes.

I.7. Evaluation des SRI

Dès l'apparition des premiers SRI, la pratique d'évaluation desdits systèmes est apparue ; les premières évaluations datent de 1953.[29]

L'évaluation des SRI est abordée selon deux angles différents. L'un est dit « paradigme système », qui vise à évaluer les performances du système essentiellement en termes de qualité des documents retournés par le système, c'est-à-dire leur pertinence vis-à-vis des besoins en information des utilisateurs.

L'autre est dit « paradigme usager », qui est centré sur la satisfaction de l'utilisateur, et non sur les performances intrinsèques du système, en modélisant le comportement des utilisateurs en situation de recherche.

Nous présentons ci-dessous seulement l'approche basée « système », la plus utilisée dans le domaine de la RI. Elle se base sur deux éléments essentiels à savoir : des collections de test et des mesures d'évaluation.

I.7.1 Les collections de test

Une collection (ou corpus) de test constitue le moyen d'évaluation des SRI. Elle est généralement composée d'un ensemble de documents, d'un ensemble de requêtes et des jugements de pertinence associés à ces requêtes. L'évaluation d'un SRI consiste à comparer les résultats retournés par ce dernier par rapport aux jugements de pertinence. Des mesures d'évaluation, décrites dans la section suivante, sont utilisées pour effectuer cette comparaison.

Les collections de test sont le résultat de projets d'évaluation qui se sont multipliés depuis les années 1970, on peut citer la collection CACM1, la collection CISI2, la campagne CLEF3 et la campagne TREC4.

La campagne TREC constitue à ce jour la campagne de référence dans le cadre de l'évaluation des systèmes de recherche d'information et cela depuis son lancement en 1992. L'objectif de cette campagne est de proposer une plate-forme qui réunit des collections de test, des tâches spécifiques et des protocoles d'évaluation pour chaque tâche afin de mesurer les différentes stratégies de recherche.[7]

Les tâches proposées se sont diversifiées d'une campagne à une autre (à raison d'une par an) ; parmi les tâches proposées dans TREC 2012 on peut citer : recherche d'information sur le web, recherche d'information médical, recherche d'information dans les micros blogs, recherche d'information contextuelle, etc.

La taille des collections augmente au fil des années, passant de 2 Go dans TREC 1 à 25 To dans TREC 2011. Chaque collection est composée d'un certain nombre de documents, allant de quelques milliers à plusieurs millions. Les documents sont codés à l'aide de SGML dans un format spécifique TREC. La figure I.2 illustre un exemple d'un document TREC.

```

<DOC>
<DOCNO> WSJ920324-0113 </DOCNO>
<DOCID> 920324-0113. </DOCID>
<HL> Venture of Kimbaco</HL>
<DATE> 03/24/92 </DATE>
<SO> WALL STREET JOURNAL (J), PAGE C9 </SO>
<CO> H.TSI </CO>
<MS> FINANCIAL (FIN) </MS>
<IN> ALL BANKS, BANKING NEWS AND ISSUES (BNK)
SECURITIES (SCR) </IN>
<NS> JOINT VENTURES (JVN) </NS>
<RE> FAR EAST (FE)
HONG KONG (HK)
PACIFIC RIM (PRM)
SOUTH KOREA (SK)
</RE><LP>
NEW YORK -- South Korean merchant banking firm
Kimbaco said it joinedwith Hong Kong brokerage house
Peregrine Securities to form a newinvestment firm
Kimbaco Peregrine Capital Ltd.The firm will seek out
cross-border transactions and direct
investmentopportunities in Asia, with special
emphasis on U.S.-Korean ventures.
</LP><TEXT>
</TEXT>
</DOC>

```

Figure I.2 Exemple d'un document TREC.

Chaque collection TREC a généralement 50 à 100 requêtes correspondantes. Une requête TREC est structurée comme suit: un identifiant de requête unique TREC, un titre, une description plus détaillée du besoin en information et une rubrique qui explique dans quelles circonstances un document doit être jugé pertinent ou non pertinent pour une requête. Un exemple d'une requête TREC est montré dans la figure I.3. Dans la plupart de nos expérimentations, nous n'utilisons que le champ titre des requêtes, car ceux-ci sont similaires aux besoins en information des utilisateurs sur le web.

```
<top>
<num> Number: 562
<title> world population growth
<desc> Description:
What is the outlook for world population
growth?
<narr> Narrative:
Relevant documents include projections of
anddiscussion of world population growth.
Growth ofindividual nations' populations
is relevant, butdata on states within the
U.S. is not relevant.
</top>
```

Figure I.3 Exemple d'une requête TREC.

I.7.2 Mesures d'évaluation de SRI

Le principal objectif d'un système de recherche d'information est de restituer à l'utilisateur tous les documents pertinents et de rejeter tous les documents non pertinents. Cet objectif est évalué à l'aide de différentes mesures d'évaluation[170]. On présente ci-dessous les plus utilisées.

- **La précision** : est le rapport du nombre de documents pertinents restitués par le système

(SP) sur le nombre total de documents restitués (R), exprimée ainsi :

$$\dot{e} = -\frac{1}{2} \quad (\text{I.16})$$

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

$$= \quad (I.17)$$

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

- **Le bruit** : la mesure d'évaluation bruit est une notion complémentaire à la précision, elle est définie par $B = I - P$ où P est la précision du SRI.
- **Le silence** : la mesure d'évaluation silence est une notion complémentaire au rappel, elle est définie par $S = I - R$ où R est le rappel du SRI.

- **Courbe de Rappel-Précision** : un système idéal devrait retourner tous les documents pertinents et que les documents pertinents ; c'est à dire un taux de précision et de rappel égal à 100%. Cette situation ne se produit pas dans un système réel car le taux de précision et de rappel sont antagonistes. En effet, Lorsque la précision augmente, le rappel diminue et inversement. Ainsi, pour mesurer les performances d'un système il faut utiliser les deux mesures conjointement. Cela est réalisé en calculant la paire des mesures (taux de rappel, taux de précision) à chaque document restitué.

Nous considérons par exemple une requête pour laquelle il existe six (6) documents pertinents dans le corpus. Le Tableau I.2 illustre le calcul de la précision et de rappel pour les dix (10) premiers documents retournés par un SRI. La lettre (P) précise que le document est pertinent.

Rang du document renvoyé	Pertinence	Rappel	Précision
Doc1	P	0,17	10
Doc2		0,17	,5
Doc3	P	0,33	0,66
Doc4		0,33	0,5
Doc5	P	0,5	0,6
Doc6		0,5	0,5
Doc7	P	0,67	0,57
Doc8		0,67	0,5
Doc9		0,67	0,44
Doc10	P	0,83	0,5

Tableau I.2 Exemple de calcul de rappel et de précision pour une requête.

La figure I.4 illustre la courbe de rappel et précision correspondante aux résultats du Tableau I.2. Pour rendre la courbe lisible on ne garde que la précision calculée à chaque point de rappel (c'est à dire à chaque document pertinent restitué).

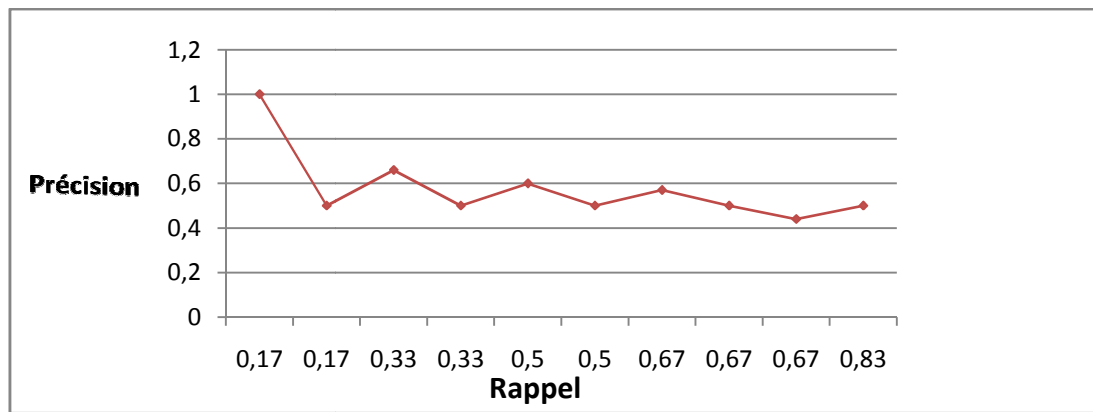


Figure I.4 Courbe de rappel et précision.

La courbe ci-dessus permet d'évaluer les performances du système pour la requête considérée. Afin d'évaluer le système pour un ensemble de requêtes, on calcule la moyenne des précisions à chaque niveau de rappel. Comme les niveaux de rappel ne sont pas unifiés pour l'ensemble des requêtes, on retient généralement 11 points de rappel standards de 0 à 1 avec un pas de 0.1.

Les valeurs de précision sont calculées par une interpolation linéaire. Pour deux points de rappel, i et j , $i < j$, si la précision au point i est inférieure à celle au point j , on dit que la précision interpolée à i égale la précision à j .

Cette interpolation est encore discutable, mais présente un intérêt dans l'évaluation de systèmes de recherche d'information. Elle permet entre autre de construire des courbes décroissantes plus simple à comparer.[3]

- Les mesures alternatives

- **La précision exacte** : notée aussi R-précision, elle est calculée sur les « R » premiers documents retournés par le système, sachant que la requête admet « R » documents pertinents.
- **La précision moyenne non interpolée (MAP)**: la précision moyenne non interpolée (AverageMeanPrecision) est calculée en deux étapes. D'abord on calcule la précision moyenne pour une requête donnée (APq), ainsi pour chaque document pertinent retrouvé on calcule sa précision ($pr(di)$) qui est égale au nombre de documents pertinents

retrouvés sur le rang de ce document ; pour les documents retrouvés non pertinents leur précision est égale à zéro.

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

$$P_q = \frac{1}{N_q} \sum_{r=1}^{N_q} p(r) \quad (I.18)$$

Avec

$$p(r) = \frac{1}{N_q} \sum_{d=1}^{N_q} \frac{1}{r} \quad (I.19)$$

Où r dénote le rang du document d qui a été retrouvé et qui est pertinent pour la requête, N_q est le nombre de documents pertinents retrouvé au rang r et N est le nombre total de documents pertinents pour la requête q .

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

$$P = \frac{1}{Q} \sum_{q=1}^Q P_q \quad (I.20)$$

Où P_q dénote la précision moyenne pour la requête « q » et Q représente le nombre de requêtes considérées.

- La R-Précision :

La R-précision est un paramètre pour observer le comportement d'un système pour chaque requête individuellement, elle est formalisée comme suit :

$$R = \frac{1}{N} \sum_{d=1}^N \frac{1}{r} \quad (I.21)$$

N : Documents Pertinents Retournés.

R : Le rang de la précision.

I.8.Conclusion

Dans ce chapitre nous avons passé en revue les principaux concepts de la RI. Nous avons, particulièrement, introduit des notions de base, telles que le besoin en information, la requête, le document et la pertinence. Nous avons aussi décrit les processus de base de la RI, à savoir l'indexation et la reformulation de la requête. Ensuite, nous avons étudié les différents modèles de la RI. Enfin, l'évaluation des systèmes de recherche d'information est traitée.

Parmi les modèles étudiés dans ce chapitre, nous nous intéressons au modèle de langage comme modèle utilisé dans notre travail pour la combinaison des sources d'information sur un document.

Dans le chapitre suivant, nous détaillons le modèle de langue, ainsi que les différentes caractéristiques indépendantes de la requête utilisées pour le classement des documents.



Chapitre 2

Modèles de Langue pour la RI

II.1.Introduction

Depuis leur première utilisation en recherche d'information[36] (RI), les modèles statistiques de langue ont acquis une grande popularité, en raison de leur simplicité, efficacité et performance. Ces modèles ont été aussi appliqués avec succès dans différentes tâches de la RI, telles que la RI sur le web[55] la RI distribuée, la recherche d'expert.

Un des atouts de cette nouvelle classe de modèles de RI, concerne leur fondement théorique, basé sur la théorie des probabilités. De plus, ces modèles offrent la possibilité de combiner différentes informations (évidences) sur un document web pouvant être liées à la requête tel que le contenu du document ou non liées à la requête telles que la structure du document, la structure des liens, la date de création d'un document, etc.

II.2. Les modèles de langue en linguistique informatique

Les premiers travaux qui ont porté sur l'utilisation des modèles de langue pour la recherche d'information sont inspirés de l'application avec succès des techniques statistiques de modélisation de langue en informatique linguistique. En effet, les deux domaines ont en commun plusieurs caractéristiques, entre autres, ils manipulent (possèdent) de grandes masses de texte, ce qui permet d'entraîner aisément les modèles statistiques pour des finalités différentes.

II.2.1. Idée de base

La modélisation statistique de langue est une distribution de probabilités sur toutes les séquences possibles « » ou autres unités linguistiques dans une langue. Elle peut être vue comme un « processus génératif » qui consiste à estimer la probabilité (|) de générer une séquence de mots « » à partir du modèle de la langue .

Supposons que $w_1 w_2 \dots$ est une séquence de mots, la probabilité de génération de la séquence « » est formulée ainsi :

$$P(w_1, w_2, \dots) = \prod_{i=1}^{\infty} P(w_i | w_1, \dots, w_{i-1}) \quad (\text{II.1})$$

Dans cette formulation le terme w_i ne dépend que de ses $i-1$ prédécesseurs immédiats, c'est une simplification qui est faite pour réduire le nombre de paramètres à estimer, dans ce cas le modèle est dit modèle *n-gramme*.

Dans le cas où $n=1$ on parle du modèle *uni-gramme*. On considère dans ce modèle que les mots de la séquence sont générés indépendamment les uns des autres, la formule précédente devient alors :

$$P(w_1, w_2, \dots) = \prod_{i=1}^{\infty} P(w_i) \quad (\text{II.2})$$

Quand $n=2$ le modèle de langue est dit *bi-gramme*. Il est estimé en utilisant des informations sur la cooccurrence de paires de mots, c'est-à-dire chaque mot dépend seulement de son prédécesseur. La probabilité $P(w_i | w_{i-1})$ est alors formulée ainsi :

$$P(w_1, w_2, \dots) = \prod_{i=1}^{\infty} P(w_i | w_{i-1}) \quad (\text{II.3})$$

Dans le cas où $n=3$ on parle du modèle *tri-gramme*.

En pratique, ce sont ces trois modèles qui sont les plus utilisés.

Tous ces modèles nécessitent le calcul de la probabilité d'un *n-gramme* $P(w_1, w_2, \dots, w_n)$ ce calcul se base sur l'estimation de vraisemblance maximale (MLE), qui revient à calculer la fréquence relative d'un *n-gramme* dans le corpus d'entraînement, soit $\frac{1}{N}$.

Cette probabilité est estimée comme suit :

$$P(w_1, w_2, \dots, w_n) = \frac{1}{\sum_{e \in \mathcal{E}} 1} = \frac{1}{N} \quad (\text{II.4})$$

Où $| \quad |$ est la fréquence d'occurrence du n-gramme (du mot) dans le corpus, est un *n-gramme* de la même longueur que et le dénominateur $| \quad |$ correspond à la taille du corpus (c'est-à-dire le nombre total d'occurrences de n-grammes).

Cependant, quelle que soit la taille du corpus d'entraînement utilisé pour l'estimation de ces probabilités, il est fréquent que beaucoup de n-grammes n'apparaissent pas dans le corpus.

Ceci entraîne l'attribution d'une probabilité nulle pour toute séquence contenant ces n-grammes.

Pour remédier à ce problème (clairsement de données) et rendre les modèles plus généraux, des techniques de lissage sont utilisées. Le principe de ces techniques consiste à prélever une quantité de probabilités associées aux n-grammes observés [10] dans le corpus d'entraînement et de la redistribuer sur les n-grammes non observés. De cette façon, les n-grammes absents du corpus vont recevoir une probabilité non nulle.

II.2.2. Les techniques de lissage

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

II.2.2.1. Lissage de Laplace

Cette méthode consiste à ajouter la fréquence un (1) à tous les n-grammes, appelé aussi ajouter-un. La probabilité du n-gramme est estimée ainsi :

$$(\quad) = \frac{| \quad |}{\sum_{\epsilon \quad | \quad |} | \quad |} = \frac{| \quad |}{| \quad |} \quad (II.5)$$

Où est le nombre de n-grammes (distincts) et $| \quad |$ est la taille du corpus.

L'inconvénient de cette méthode est qu'une grande masse de probabilité est distribuée sur les n-grammes non observés dans le corpus, d'où la faible participation des n-grammes observés dans la définition du modèle.

II.2.2.2. Lissage de Good-Turing

Cette méthode permet l'ajustement de la fréquence « » d'un n-gramme () en une fréquence dite corrigée « * », exprimée ainsi :

$$* = (+ 1) \times \frac{f}{f+1} \quad (\text{II.6})$$

Où est le nombre de n-grammes de fréquence « » dans la collection d'apprentissage. Ainsi, pour tout n-gramme () l'estimation de sa probabilité devient alors :

$$() = \frac{f^*}{\sum_{\epsilon} f^*} \quad (\text{II.7})$$

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

II.2.2.3. Lissage par interpolation (Jelinek-Mercer)

Ce type de lissage consiste à combiner le modèle de langue considéré avec un ou plusieurs modèles de références estimés sur d'autres corpus d'apprentissages.

Typiquement, dans le cas de collection de documents, on pourrait par exemple estimer le modèle de document en le combinant avec le modèle de la collection.

Dans ce cas, le modèle de document est exprimé ainsi :

$$(|) = (1 - \alpha) (|) + \alpha (|) \quad (\text{II.8})$$

Les modèles (|) et (|) sont estimés selon le maximum de vraisemblance.

II.2.2.4. Lissage de Dirichlet

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

$$P(w_i | d) = \frac{\frac{1}{|D|} + \frac{1}{|d|}}{\frac{1}{|D|} + \frac{1}{|d|}} P(w_i | d) + \frac{\theta}{|D| + \theta} P(w_i | D) = \frac{\frac{1}{|D|} \times \frac{1}{|d|} + \theta \times \frac{1}{|D|}}{\frac{1}{|D|} + \frac{1}{|d|} + \theta} P(w_i | D) \quad (II.9)$$

$$\text{Avec } P(w_i | D) = \frac{c(w_i, D)}{|D| + \theta} \quad (II.10)$$

Où $|D|$ est la taille du document (le nombre d'occurrences de mots) $c(w_i, D)$ est la fréquence du mot w_i dans D et θ est un paramètre appelé pseudo fréquence.

Plusieurs études en RI ont montré que le choix de la méthode de lissage a un grand impact sur les performances du SRI. Zhai et lafferty[52] ont expérimenté plusieurs techniques de lissage en utilisant le modèle de langue uni-gramme et ils ont rapporté que la méthode de lissage de Dirichlet donne de meilleurs résultats que les autres méthodes de lissage. Dans notre travail nous utilisons cette méthode de lissage.

II.3. Modèles de langue et recherche d'information

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

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

II.3.1. Approches d'exploitation des modèles de langue en RI [1]

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

1. Génération de la requête par le modèle de document (QueryLikelihoodModels) : cette catégorie correspond à ce que nous avons expliqué ci-dessus. Un modèle de langue est associé à chaque document. Les documents sont alors classés selon leurs probabilités de génération de la requête, soit $(D | Q)$.
2. Génération de document par le modèle de la requête (Document LikelihoodModels) : cette approche procède dans le sens inverse. Ainsi, un modèle de langue est associé à la requête, les documents sont alors classés selon leurs probabilités que leur contenu soit généré par le modèle de la requête, soit $(D | Q)$.
3. Similarité document-requête : dans cette catégorie, on considère à chaque document et à chaque requête est associé un modèle de langue. Les documents sont alors ordonnés selon la similarité de leurs modèles avec celui de la requête. Cette similarité est estimée en calculant l'entropie croisée ou la KL (Kullback-Leibler).

Nous présentons ci-dessous quelques modèles développés implémentant ces différentes catégories d'approches.

II.3.1.1. Génération de la requête par le modèle de document (QueryLikelihoodModels)

Ponte et Croft[36] furent les premiers à proposer un modèle de langue pour la RI.

L'intuition derrière leur modèle est que l'utilisateur est supposé avoir une idée des termes qui sont présents dans le document pertinent ou modèle de ce document. À partir de là, l'utilisateur génère la requête en utilisant ces termes. Ainsi, la requête est censée fournir des indices (des termes) associés au modèle du document (les documents recherchés).

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

$$(q, d) = (q | d) \quad (\text{II.11})$$

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

$$(q | d) = \prod_{t \in q} (t | d) \times \prod_{t \notin q} (1 - (t | d)) \quad (\text{II.12})$$

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

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

La probabilité $(t | d)$ est calculée par une méthode non paramétrique qui utilise la probabilité moyenne d'apparition du terme t dans les documents qui le contiennent $(t | C)$ et un facteur de risque pour un terme observé dans le document (t, d) . Par contre, la probabilité d'un terme dans la collection est utilisée pour les termes

qui n'apparaissent pas dans le document. Le calcul de cette probabilité est exprimé ainsi :

$$P(w_i | D) = \frac{P(w_i | C) \times P(C)}{P(w_i)} \quad (II.13)$$

Ponte et croft ont obtenu une amélioration de +8.74% de performance (précision moyenne) par rapport au modèle utilisant le schéma de pondération \times d'Okapi sur des collections TREC.

Dans le modèle proposé par Hiemstra[23], la requête est considérée comme une séquence de

termes, $Q = (q_1, q_2, \dots, q_n)$. Dans ce modèle on ne considère que les mots présents dans la requête, une distribution multinomial est utilisée. Le score d'un document vis-à-vis d'une requête (la probabilité de générer une requête sachant le modèle de document D) est donné par la formule suivante :

$$P(Q | D) = P(D) \prod_{i=1}^n P(q_i | D) \quad (II.14)$$

Où $P(D)$ est la probabilité a priori d'un document.

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

$$P(q_i | D) = \lambda P(q_i | D) + (1 - \lambda) P(q_i | C) \quad (II.15)$$

Où λ est le poids d'interpolation qui varie entre 0 et 1. Ce paramètre peut être considéré constant ou il peut être estimé d'une manière sophistiquée en utilisant un processus d'optimisation automatique tel que l'algorithme de maximisation d'espérance (EM). Le calcul des deux probabilités $P(q_i | D)$ et $P(q_i | C)$ est réalisé selon une estimation de vraisemblance maximale, exprimée ainsi :

$$P(q_i | D) = \frac{C_i}{N} \text{ et } P(q_i | C) = \frac{O_i}{\sum_{\epsilon} O_i} \quad (II.16)$$

Où $f_{t,d}$ est la fréquence du terme t dans le document d , N_d est le nombre de documents contenant le terme t et V est le vocabulaire d'index.

Miller et al [56] ont proposé un modèle similaire à celui de Hiemstra. La différence entre les deux modèles réside dans leur implémentation. Miller utilise un modèle de Markov caché à deux états, illustré par la figure 3.1 suivante.

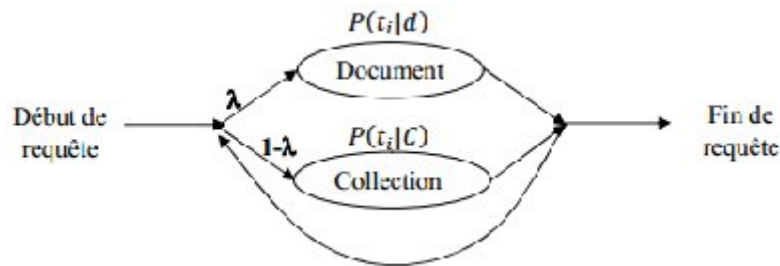


Figure II.1 : Modèle de Markov à deux états.

Ce modèle correspond à la formule suivante :

(II.17)

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

—————(II.18)

Où $f_{t,C}$ est la fréquence du terme t dans le corpus et N_C est le nombre de termes dans le corpus et V est le vocabulaire d'index.

II.3.1.2. Génération de document à partir du modèle de la requête (Document Likelihood Model)

Au lieu de modéliser la RI comme processus de génération de la requête, Lavrenko et Croft ont proposé de modéliser explicitement le modèle de pertinence. Ils ont en effet proposé d'estimer ce modèle à partir du modèle de la requête sans utilisation de données d'entraînement, en faisant le parallèle avec la

modélisation de la pertinence proposée dans le modèle probabiliste classique. Ils considèrent en effet que pour chaque requête, il existe un modèle permettant de générer le sujet (thème) abordé par la requête, c'est ce que les auteurs appellent le modèle de pertinence, soit $P(q|t)$.

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

$$P(t|q) = \sum_{d \in D} \frac{P(t|d)P(d|q)}{\sum_{d \in D} P(d|q)} = \sum_{d \in D} P(t|d)P(d|q) \quad (\text{II.19})$$

Où D dénote l'ensemble de documents feedback.

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

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

II.3.1.3. Comparaison des modèles de requête et du document

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

Lafferty et Zhai[28] ont proposé un cadre de minimisation de risque basé sur la théorie de décision bayésienne. Dans ce cadre la requête et les documents sont modélisés par des modèles statistiques. Le modèle de la requête modélise entre autres les préférences et les besoins des utilisateurs, le modèle du document

permet de modéliser le processus de génération de document et de capturer les préférences de l'auteur.

La similarité entre documents et requête est mesurée par la fonction de divergence de Kullback-Leibler (KL) entre le modèle de la requête (θ) et celui du document (ϕ) [28], exprimée comme suit :

$$D_{KL}(\theta \parallel \phi) = - \sum_{v \in V} \theta_v \log \frac{\theta_v}{\phi_v} \quad (II.20)$$

Où V est le vocabulaire d'index.

Cette approche peut être vue comme la combinaison de certains aspects des deux approches précédentes. Les expérimentations menées par les auteurs sur des collections TREC ont montré l'utilité de cette méthode. De plus, selon la fonction de risque (perte) choisie, les différents modèles de la RI peuvent s'avérer comme des cas spécifiques de cette approche.

II.4. Intégration des caractéristiques indépendantes du contenu de document dans le modèle de langue

Nous présentons dans ce qui suit les différentes caractéristiques utilisées pour estimer la probabilité de pertinence a priori.

II.4.1. La taille du document

Dans ce cas, la probabilité de pertinence a priori est proportionnelle à la taille du document, exprimée ainsi :

$$P(r) = \frac{|d|}{|C|} \quad (II.21)$$

Où $|d|$ est la taille du document et $|C|$ est la taille de la collection.

L'intuition de l'utilisation d'une telle caractéristique est qu'un document plus long tend à contenir plus d'informations et par conséquent, il est plus probable d'être pertinent. Les résultats obtenus avec l'utilisation de cette caractéristique ont été mixtes et cela selon la collection utilisée [57].

Parapar et al [58] ont proposé d'estimer cette probabilité P_c en utilisant la taille compressée d'un document. La formule utilisée est exprimée ainsi :

$$P_c = \frac{C}{\sum C} \quad (II.22)$$

Où C (est) la taille en octets du document compressé (zippé) divisée sur la taille originale en octets du document. Ce nouveau facteur a été évalué et comparé au facteur taille originale du document en utilisant quatre collections TREC. Les résultats présentés montrent que la taille compressée d'un document donne des améliorations de précision (MAP) allant de +0,4% à +3,1% par rapport à l'utilisation de la taille originale du document.

II.4.2. La date de création du document

D'autres travaux utilisent l'intuition suivante : « les documents récents tendent à être plus pertinents que les documents anciens », pour estimer la probabilité a priori d'un document.

Li et Croft[59] ont proposé un modèle de langue qui permet d'intégrer la notion de « temps » dans l'évaluation de la pertinence d'un document vis-à-vis d'une requête, où ils assignent une plus grande probabilité de pertinence pour les documents ayant une date de création récente. Ainsi, ils expriment la probabilité de pertinence a priori d'un document sachant sa date de création, comme une distribution exponentielle, exprimée comme suit :

$$P(d | q) = \frac{1}{\sum d} e^{-\lambda d} \quad (II.23)$$

Où d est la date la plus récente dans toute la collection (exprimée en mois) et λ est la date de création du document.

Les évaluations réalisées sur un ensemble particulier de requêtes montrent que l'incorporation de la notion de temps en utilisant la distribution exponentielle est bénéfique pour la RI.

II.4.3. La structure des liens

L'intuition derrière l'utilisation de la structure des liens est que les documents populaires ou les plus cités tendent à être plus pertinents. La méthode la plus simple d'exploitation de la structure des liens est l'utilisation du nombre de liens entrants. La probabilité de pertinence a priori est alors exprimée comme suit :

$$P(d) = \frac{C(d)}{\sum C(d)} \quad (\text{II.24})$$

Où $C(d)$ est le nombre de liens entrants dans le document d .

D'autres facteurs plus sophistiqués peuvent être utilisés comme : le HITS, le PageRank, etc.

II.4.4. L'hypothèse de recommandation

Elle stipule que si une page est pointée par un lien alors cette page est recommandée par la page qui l'a référencé. Ainsi, une page avec beaucoup de liens entrants est une page fortement recommandée (populaire, autorité) et donc susceptible d'être mieux classée. Plusieurs algorithmes et méthodes de « Ranking » se sont basés sur cette hypothèse, pour le calcul de l'importance de la page. Nous décrivons ci-dessous quelques algorithmes représentatifs de cette classe.

- PageRank

L'algorithme de PageRank (PR) développé par Brin et Page en 1998 est à l'origine du moteur de recherche Google. Cet algorithme assigne une valeur de popularité (un score, un PR) à toutes les pages du web indépendamment de toute requête.

La mesure PageRank est une distribution de probabilité sur les pages. Elle mesure en effet la probabilité PR, pour un utilisateur navigant au hasard, d'atteindre une page donnée. Elle repose sur un concept très simple : un lien émis par une page A vers une page B est assimilé à un vote de A pour B. Plus une page reçoit de votes,

plus cette page est considérée comme importante. Le PageRank se calcule de la façon suivante :

$$P(i) = (1 - \alpha) \times \frac{1}{N} + \alpha \sum_{j \in \text{In}(i)} \frac{P(j)}{\text{Out}(j)} \quad (\text{II.25})$$

Où :

$P(i)$ est le PageRank de la page i ;

N est le nombre total de pages web indexées ;

α est un paramètre fixé à 0.85 ;

$\text{In}(i)$ est le nombre de liens sortant de la page i , et $\text{Out}(j)$ est le nombre de pages qui pointent la page j .

Le score PageRank est utilisé afin de réordonner la liste des pages retournées.

L'algorithme de PageRank n'améliore pas significativement les résultats des stratégies qui ignorent les hyperliens, selon les évaluations faites lors des campagnes de TREC.[61] [62].

Plusieurs extensions de PageRank ont été proposées, tels que le PageRank personnalisé[60], qui consiste à incorporer les centres d'intérêts des utilisateurs dans le calcul de PageRank.

- HITS

Kleinberg a proposé un algorithme dit HITS (HyperlinkInducedTextSearch) [63] pour identifier les documents autorités au moment de la recherche.

L'algorithme est basé sur l'analyse de la matrice d'adjacence des pages retournées pour une requête donnée. Les pages web concernant le sujet de la requête peuvent être soit des Hubs ou des Autorités. Les pages Autorités contiennent de l'information sur le sujet. Par contre, les pages Hubs pointent les pages Autorités. Ces deux types de pages ont une relation de renforcement mutuelle entre elles, ainsi : un bon Hub est une page qui pointe beaucoup de bonnes Autorités, et une

bonne Autorité est une page qui est pointée par beaucoup de bons Hubs. La Figure II.2 ci-dessous illustre la relation entre ces deux types de pages.

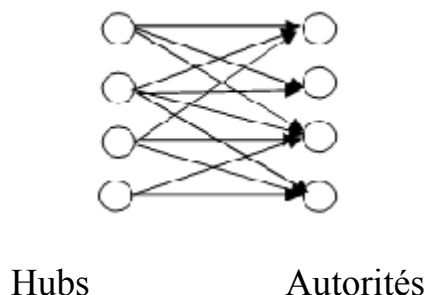


Figure II.2 : Les pages Hubs et Autorités.

L'algorithme HITS calcule deux scores pour une page : le score d'Autorité et le score de Hub. L'évaluation de ces deux scores se fait au moment de l'interrogation, selon les deux formules ci-dessous, traduisant la relation de renforcement mutuelle entre les deux types de pages :

$$\begin{aligned}
 &= \rightarrow \\
 &= \rightarrow
 \end{aligned}$$

Où h_i et a_i représentent respectivement le score d'Autorité et de Hub de la page « i », \rightarrow indique l'ensemble des pages qui pointent la page « i » et \rightarrow indique l'ensemble des pages pointées par la page « i ».

Le calcul des scores h_i et a_i se fait d'une manière itérative. Il se termine lorsque la convergence des scores s'opère, ces scores sont alors utilisés pour classer les documents.

Plusieurs autres variantes ont été proposées élargissant les méthodes présentées précédemment, comme l'algorithme SALSA.[64]

II.5. Conclusion

Nous avons présenté dans ce chapitre les modèles de langue ainsi que les caractéristiques (évidences) d'un document, indépendantes du contenu de document qui ont été utilisées pour estimer la probabilité a priori d'un document.

Plusieurs caractéristiques (évidences) d'un document, indépendantes du contenu de document ont été utilisées pour estimer la probabilité a priori d'un document. Or, des caractéristiques qui évaluent la cohérence du contenu textuel d'un document tel que la clarté du document a été introduite pour estimer la probabilité a priori d'un document.

Nous décrivons dans le chapitre suivant notre approche de réordonnancement de document à l'aide du score de clarté ainsi que les résultats obtenus de l'évaluation.



Chapitre 3

Approche implémentée et évaluation

III.1.Introduction

Le travail présenté dans ce mémoire se situe dans le contexte de la recherche d'informations et plus particulièrement l'exploitation de la caractéristique déclarée de document.

Nous avons présenté dans le chapitre précédent les différentes approches utilisées pour caractériser un document, dans ce chapitre notre objectif est de présenter une nouvelle approche dont l'objectif est de réordonner les documents à l'aide d'un facteur qui est le score de clarté de document.

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

III.2. Présentation de l'approche implémentée

Nous décrivons dans ce qui suit (Figure III.1) les différentes étapes que nous avons suivies pour la mise en œuvre de notre approche.

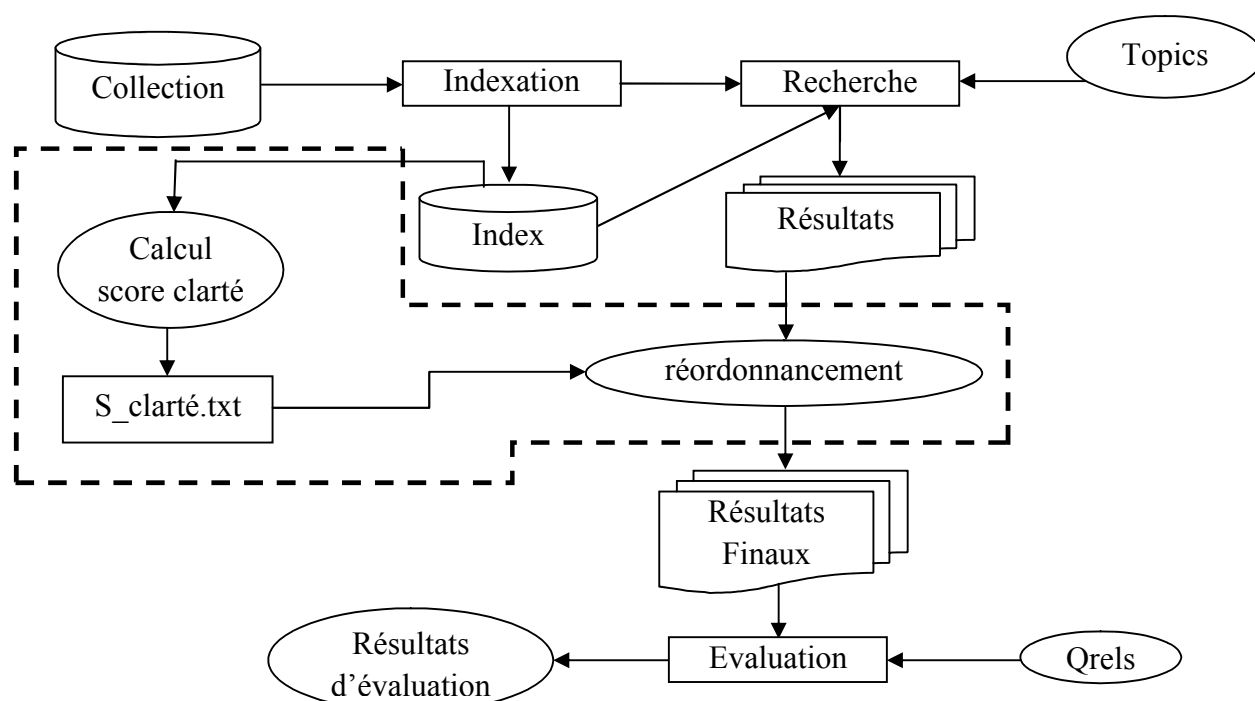


Figure III.1 : Notre approche.

[] : Etape du réordonnancement à l'aide du score de clarté.

La Figure III.1 illustre les étapes de notre approche, jumelant la recherche classique (indexation, recherche et évaluation) et l'étape de réordonnancement (encadré en pointillé) à l'aide du score de clarté.

III.2.1. La phase calcul du score de clarté du document

III.2.1.1. Définition de la clarté

C'est une mesure qui indique la proportion de ressemblance entre deux distributions de probabilité.

III.2.1.2. Clarté de la requête

La mesure de clarté a été initialement proposée pour la requête afin d'estimer le degré de correspondance entre la requête et la collection, elle a été ensuite utilisée pour déterminer quelle requête nécessitant l'expansion. C'est-à-dire une requête claire ne nécessite pas d'expansion.[66]

La clarté de la requête est formalisée comme suit :

$$C_r = \sum_{i=1}^m (p_i | q) \log \frac{(p_i | q)}{(p_i | C)} \quad (III.1)$$

Où : m est le nombre de terme des m premiers documents retournés.

$(p_i | q)$ est la probabilité de terme p_i dans la collection q .

III.2.1.3. Clarté du document

En ce qui nous concerne nous employons cette mesure (clarté) au niveau du document afin de réordonner les documents retrouvées pour une requête.

- **Intuition** : L'intuition derrière l'utilisation de la clarté du document est la suivante : « un document claire dans la liste des documents de la collection est un bon document donc il faut le privilégié lors du réordonnancement ».

- **Formalisation** : Nous formalisons la clarté de document comme suit :

$$C_d = \sum_{i=1}^n (p_i | d) \log \frac{(p_i | d)}{(p_i | C)} \quad (III.2)$$

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

$(p_i | C)$ est la probabilité de terme p_i dans l'ensemble des documents de la collection (noté C).

III.2.2. La phase de réordonnancement

Elle consiste à réordonner les documents en utilisant un facteur qui est le score de clarté définis précédemment.

Le réordonnancement est réalisé comme suit :

$$S_{\text{réordonnancement}}(D) = S_{\text{initial}}(D) + \alpha \cdot S_{\text{clarté}}(D) \quad (\text{III.3})$$

Où : α est un facteur $\in [0,1]$

$S_{\text{initial}}(D)$ est le score obtenu dans la première recherche.

$S_{\text{réordonnancement}}(D)$ est le score après réordonnancement.

Pour comparer notre approche (Score clarté de document) nous avons utilisé un autre facteur qui est la taille du document formalisé comme suit :

$$S_{\text{clarté}}(D) = S_{\text{initial}}(D) + \frac{1}{|D|} \quad (\text{III.4})$$

Où : $|D|$ est la taille du document.

III.3. Exemple illustratif

Soit une collection C contenant trois (03) documents.

$$D_1 = \{(t_1, 4), (t_2, 6), (t_3, 8), (t_4, 5)\}$$

$$D_2 = \{(t_1, 5), (t_8, 9), (t_9, 4), (t_{10}, 7)\}$$

$$D_3 = \{(t_1, 2), (t_2, 6), (t_6, 5), (t_8, 3)\}$$

Soit la requête Q qui contient deux (02) termes.

$$Q = (t_1, t_2)$$

Après le calcul du score initial ($tf * idf$) et le score final (score initial + score de clarté) nous avons obtenus les classements suivants :

Document	Score initiale	Classement initial	Taille	Score clarté	Score finale	Classement final
D_1	0.43	2	23	0.37	0.8	1
D_2	0.2	3	25	0.13	0.33	3
D_3	0.5	1	16	0.096	0.59	2

Tableau III.1 : résultat de l'exemple illustratif

D'après les résultats obtenus dans le tableau ci-dessus nous remarquons que le classement des documents a changés après l'ajout du score de clarté, à titre d'exemple le document D_1 a vu son rang passé de 2 à 1.

III.4. les outils utilisés

Nous avons implémenté notre application sous netbeans en utilisant la librairie Terrier qui est un open source écrit en java donc ce dernier est imposé à l'utilisation, dans la section suivante nous présentons les outils utilisés :

III.4.1. Le système Terrier

Terrier est une plate-forme dédiée à la recherche d'information. Elle implémente les différents modules intervenant dans le processus de RI classique et offre en plus un cadre pour l'évaluation des résultats de recherche pour différentes applications.

Terrier a été largement éprouvé. Le choix de cette plate-forme est dû aussi à sa capacité à traiter de grandes collections de documents telles que les collections TREC.

L'architecture de la plate-forme Terrier distingue les deux phases classiques : l'indexation et la recherche. Un corpus documentaire est fourni en entrée au module d'indexation. Les documents de la collection passent par un ensemble de prétraitements tels que la tokenisation. Les tokens sont ensuite injectés dans une chaîne de traitement

TermPipelines, à savoir le StopWords Pipeline pour l'élimination des mots vides desens, ou encore les Stemming pipeline et qui dépendent de la langue en question. La phase d'indexation conduit à la construction de l'index (Data structures).

La phase de recherche comprend le Manager, un module qui interagit avec l'application, réalise la mise en correspondance à travers les calculs des pondérations (selon le schéma de pondération (Weighting Model) choisi : PL2, BM25, etc.) ainsi que les scores des documents. Le résultat renvoyé à l'utilisateur, est la liste des documents jugés pertinents et leurs scores respectifs.

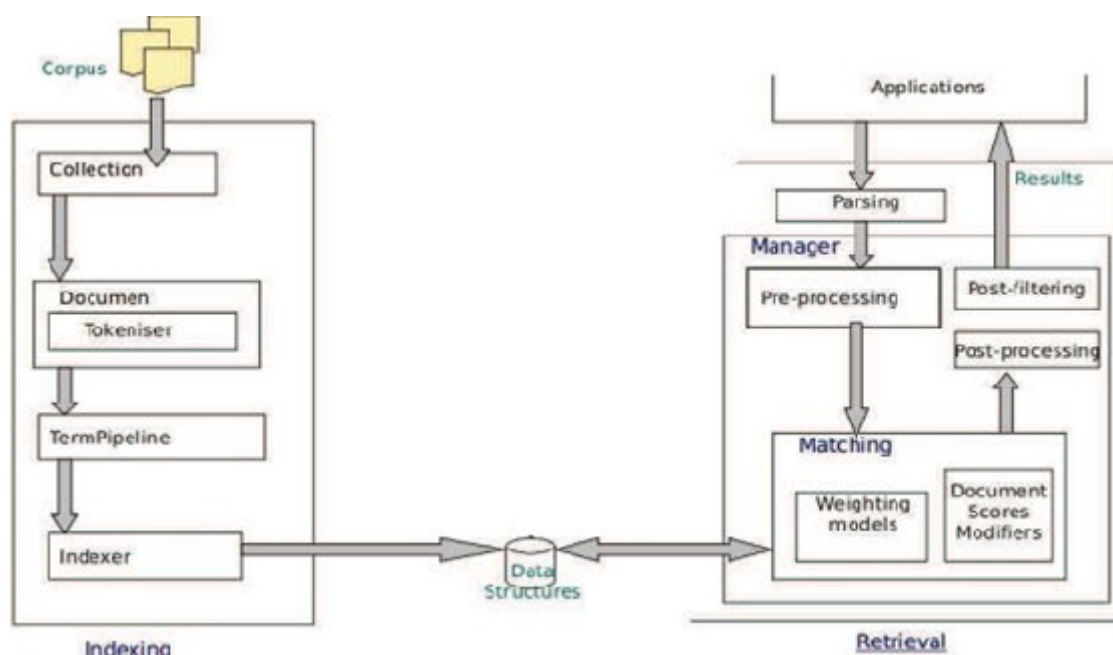


Figure III.2 : Architecture de terrier

III.4.2. Java

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.

Le langage Java reprend en grande partie la syntaxe du langage C++, très utilisée par les informaticiens. Néanmoins, Java a été épurée des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que les pointeurs et références, ou l'héritage multiple contourné par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

Java permet de développer des applications client-serveur. Côté client, les applets sont à l'origine de la notoriété du langage. C'est surtout côté serveur que

Java s'est imposé dans le milieu de l'entreprise grâce aux servlets, le pendant serveur des applets, et plus récemment les JSP (JavaServer Pages) qui peuvent se substituer PHP, Asp et Asp.net.

Java a donné naissance à un système d'exploitation (JavaOS), à des environnements de développement (eclipse/JDK), des machines virtuelles (MSJVM, JRE) applicatives multiplate-forme (JVM), une déclinaison pour les périphériques mobiles/embarqués (J2ME), une bibliothèque de conception d'interface graphique (AWT/Swing), des applications lourdes (Jude, Oracle SQL Worksheet, etc.), des technologies web (servlets, applets) et une déclinaison pour l'entreprise (J2EE). La portabilité du bytecode Java est assurée par la machine virtuelle Java, et éventuellement par des bibliothèques standard incluses dans un JDK.

Cette machine virtuelle peut interpréter le bytecode ou le compiler à la volée en langage machine. La portabilité est dépendante de la qualité de portage des JVM sur chaque OS.

III.4.3. Netbeans

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

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

L'IDE NetBeans repose sur un noyau robuste, la plateforme NetBeans, que vous pouvez également utiliser pour développer vos propres applications Java, et un système de plugins performant, qui permet d'avoir un IDE modulable. À côté de la version complète de l'IDE NetBeans, il existe différentes déclinaisons qui se concentrent sur une plateforme ou un langage précis (Java ME, Java : SE + ME + EE, Ruby, C/C++, PHP). NetBeans contient, en plus du support pour CVS et SubVersion, un support pour ClearCase, mais aussi pour Mercurial.

Il permet également de déployer des applications Web, non seulement vers Tomcat et Glassfish qui sont livrés avec le "Pack Web", mais aussi vers JBoss, WebSphere 6.1, WebLogic 9. NetBeans détient un support de développement d'applications Web avec des améliorations pour l'édition des JSP, la gestion serveur et le support des dernières versions de Tomcat.

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

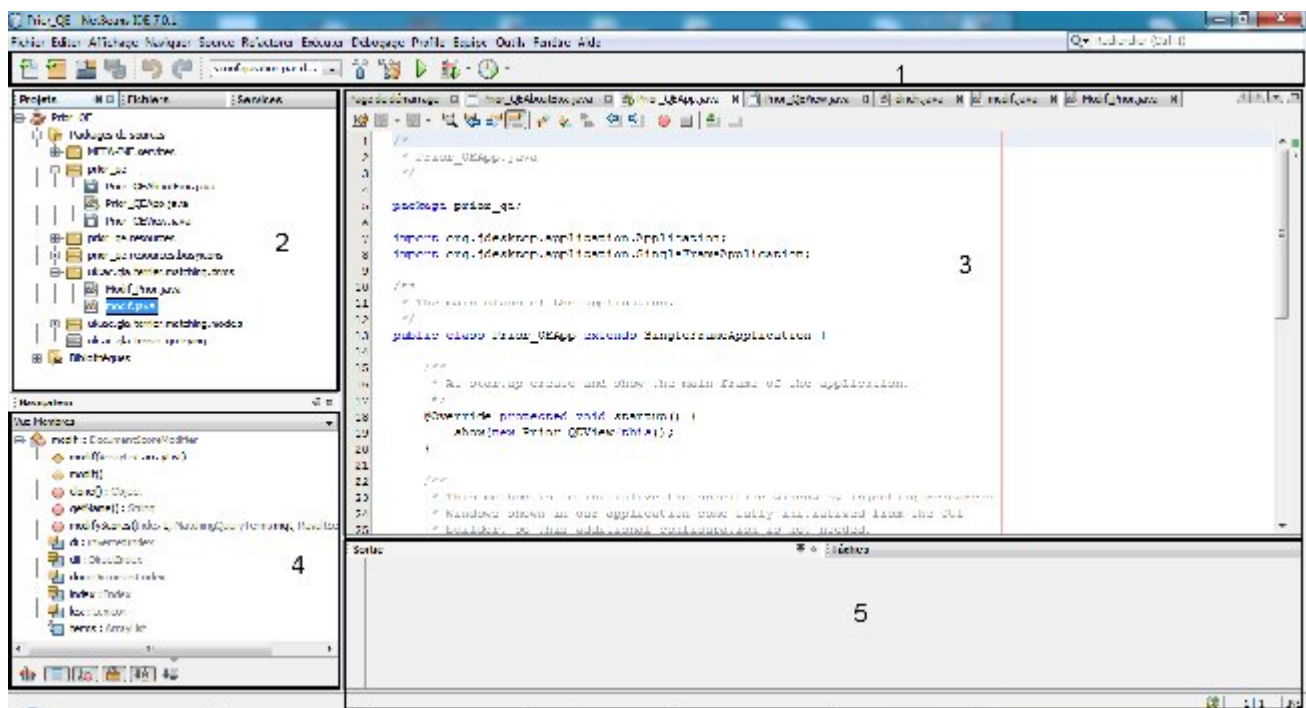


Figure III.3 :Interface Netbeans IDE.

- Dans l'encadré 1 : La barre d'outils de Netbeans IDE.
- Dans l'encadré 2 : Comporte les dossiers qui constituent notre projet.
- Dans l'encadré 3 : La feuille pour éditer, dans notre exemple c'est la feuille de modif.java.
- Dans l'encadré 4 : Panel navigateur
- Dans l'encadré 5 : la console, pour afficher les résultats de l'exécution.

III.5. Interface de l'application

L'application développée à l'interface suivante, elle contient quatre boutons chacun effectue une tâche déterminée, nous présentons ci-dessous la fonction remplie par chaque bouton :

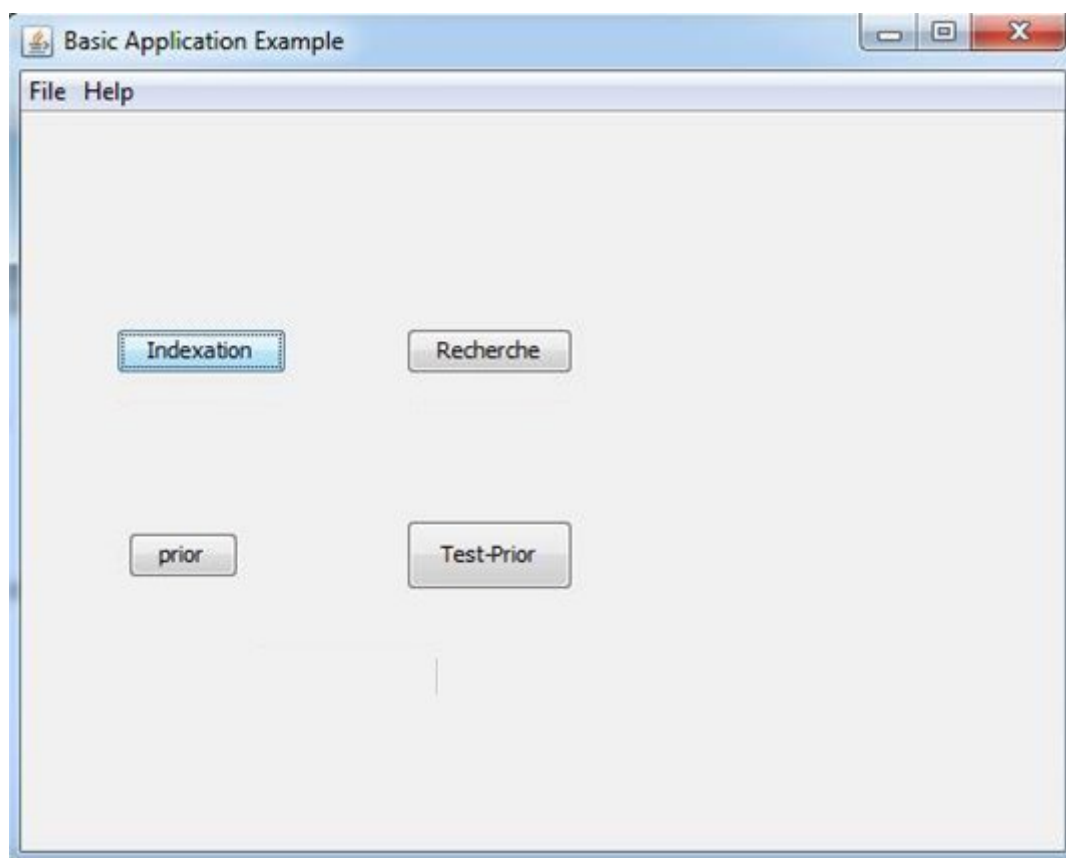


Figure III.4 : Interface Terrier

III.5.1. Création de l'index (bouton Indexation)

Après la configuration des propriétés comme le chemin de l'index (D:/Prior_QE/terrier/var/index) et du fichiers collection.spec qui regroupe les différents chemins des documents de la collection, et nous créons l'index à l'aide des deux principales classes suivantes :

- la class TRECCollection qui lit le fichier collection.spec et ouvre aussi le premier fichier de la collection à traiter.
- la class TRECIndexing qui crée le fichier direct et le fichier inverse à l'aide de plusieurs méthodes tel que *createDirectFile()*, *createInvertedFile()*, *index()*.

III.5.2. La recherche (bouton Recherche)

Après la spécification des différentes propriétés comme le modèle de pondération utilisé, dans notre cas nous avons utilisé le modèle Dirichlet vu dans la section II.2.2.4, nous effectuons la recherche en utilisant :

- La classe TRECQuerying : qui initialise l'index inversé, le lexique et les structures d'index du document.
- La méthode processQueries() : Appartient à la classe TRECQuerying, elle exécute la mise en correspondance utilisant le modèle de pondération spécifié. Il analyse le fichier des requêtes, crée le fichier "résultats.res", et pour chaque requête, obtient les documents pertinents, les scores, et met les résultats dans le fichier de résultat.

III.5.3. Bouton Prior

Permet de calculer et de retourner les valeurs de clarté des documents de la collection. La Figure III.5 montre un extrait du code.

```
for (int i=0; i< effDocuments;i++)//boucle sur l'ensemble des docs de la
collection
{
    String doc_id=doi.getDocumentNumber(i);//identificateur du document
    double sclarte=0.0;//declaration de la variable score de clarté
    int[][] terms = dii.getTerms(i);//l'index du doc i
    for(int j=0;j<terms[0].length;j++)//parcourir l'index du doc i
    {
        int tf=terms[1][j];// tf du terme j du doc i
        int taille_doc=doi.getDocumentLength(i);//taille du doc i
        int TF=oh.get(terms[0][j]);//fréquence global du terme i dans la collection
        double f=(double)tf/taille_doc;
        double c=(double)TF/index.getCollectionStatistics().getNumberOfTokens();
        sclarte+=f*Math.log(f/c);//score de clarté
    }
    valeur_clarté.put(i, sclarte) ;//sauvegarde de score de clarté
    if(i%1000==0)
        System.out.println(i);}
}
```

Figure III.5 : Extrait du code qui calcule la clarté du document

III.5.4. Bouton Test-Prior

Permet d'effectuer une recherche comme dans le deuxième cas (b) sauf que là nous utilisons la propriété matching.dsms qui fournit l'interface et les classes pour modifier les scores des documents après qu'un score a été attribué aux documents de la première recherche, dans notre cas nous utilisons les classes `modif_taille.java` utilisée pour la formule de la taille du document et `modif_clarte.java` utilisé pour la formule de la clarté du document, et nous présentons dans les deux figures ci-dessous un extrait des deux interfaces.

```

public class modif implements DocumentScoreModifier
{
protected ArrayList terms = null;
Index index = Index.createIndex();//Méthode de fabrique pour créer un index.
InvertedIndex di = index.getInvertedIndex();//Renvoie l'index inversé à
utiliser.
DocumentIndex doi = index.getDocumentIndex();//Retourne l'index des documents
associé.
DirectIndex dii=index.getDirectIndex();//Retourne l'index direct associé.
Lexicon lex = index.getLexicon();
public boolean modifyScores(Index I, MatchingQueryTerms mqt, ResultSet r)
{
double[] scores = r.getScores();//retourne les scores des documents après la
recherche pour les affecter à la valeur de type double 'scores'
int[] docids = r.getDocids();//retourne les ids des docs pour les affecter à la
valeur int 'docids'
double prior[] =new double[docids.length];
for(int j=0;j<docids.length;j++)//parcourir les doc de la collection
{
prior[docids[j]]=Math.log(1+(double)doi.getDocumentLength(docids[j]));//calcul
du score taille de document
}
for(int i=0;i<scores.length;i++)
{double delta=1;
scores[i] = scores[i] + landa*prior[i];//modification du score[réordonnancement]
}
return true;
}}

```

Figure III.6 : La class Modif_taille.

```

public class Modif_Prior implements DocumentScoreModifier {
protected ArrayList terms = null;
Index index = Index.createIndex();
InvertedIndex di = index.getInvertedIndex();//Renvoie l'index inversé à
utiliser.
DocumentIndex doi = index.getDocumentIndex();//Retourne l'index des
documents associé.
DirectIndex dii = index.getDirectIndex();//Retourne l'index direct associé.
Lexicon lex = index.getLexicon();//retourne le lexicon associé
Map<String,String> mp=new HashMap();
public Modif_Prior(ArrayList arrayList) {
throw new UnsupportedOperationException("Not yet implemented"); }
public boolean modifyScores(Index I, MatchingQueryTerms mqt, ResultSet r) {
double[] scores = r.getScores();//retourne les scores initiaux
int[] docids = r.getDocids();//retourne l'identifiant des documents
double min=0;
double prior[] = new double[docids.length];
mp=prior_qe.Prior_QEView.get_map();
double s=prior_qe.Prior_QEView.somme;
double
taille_c=index.getCollectionStatistics().getNumberOfTokens();//renvoi le nbr de
termes de la collection
for (int j = 0; j < docids.length; j++) { //parcour les documents
de la collection
prior[docids[j]] = if(min> prior[docids[j]]) min= prior[docids[j]]; }
for (int i = 0; i < scores.length; i++) { //parcour les scores initiaux
double landa = 1; //initialisation du coefficient
scores[i] = scores[i] + landa*prior[i]; //modification des
scores initiaux[réordonnancement] return true; }}

```

Figure III.7 : La class Modif_clarté.

III.6. Résultats d'évaluation

III.6.1. Les collections et les requêtes utilisées

Pour évaluer les différentes formules proposées dans notre approche, nous nous appuyons sur la collection de test TREC. La collection est AP88 (Associated Press 1988). Nous avons utilisé 50 requêtes dans un fichier nommé Topic.101-150.

La figure ci-dessous montre un exemple de requête utilisée sur la collection *AP88*.

Cette requête est composée de trois champs, le champ « *title* » est utilisé pour construire la requête. La requête construite par notre système est «catastrophic Health insurance ».

```
<top>
<head> Tipster Topic Description
<num> Number: 104
<dom> Domain: Law and Government
<title> Topic: Catastrophic Health Insurance
<desc> Description:
Document will enumerate provisions of the U.S. Catastrophic Health
InsuranceAct of 1988, or the political/legal fallout from that
legislation.
<smry> Summary:
Document will enumerate provisions of the U.S. Catastrophic Health
InsuranceAct of 1988, or the political/legal fallout from that
legislation.
<narr> Narrative:
A relevant document will detail the content of the U.S. medicare act of
1988 which extended catastrophic illness benefits to the elderly, with
particular attention to the financing scheme which led to a firestorm
of protest and a Congressional retreat, or a relevant document will
detail the political/legalconsequences of the catastrophic health
insurance imbroglio and subsequent efforts by Congress to provide
similar coverages through a less-controversialmechanism.
<con> Concept(s):
1. Catastrophic Coverage Act of 1988, Medicare Part B, Health Care
Financing Administration
2. catastrophic-health program, catastrophic illness, catastrophic
care, acute care, long-term nursing home care
3. American Association of Retired Persons, AARP, senior citizen,
National Committee to Preserve Social Security and Medicare
<fac> Factor(s):
<nat> Nationality: U.S.
</fac>
<def> Definition(s):
</top>
```

Figure III.8 : Exemple de requête (104) de la collection AP88

La table suivante montre quelques statistiques sur la collection et les Topics utilisés :

Collection	Taille	#document	topics
AP88	237Mo	79919	101-150

Tableau III.2 : statistiques sur la collection et les topics utilisés.

III.6.2 Résultats obtenus

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base à partir des quels nous allons construire un repère. Ces résultats seront ensuite comparés à ceux obtenus après le réordonnancement à l'aide du score de clarté en appliquant les mêmes paramètres de recherches.

Pour évaluer les différents modèles nous avons utilisé la MAP (Précision moyenne) définie dans la section I.7.2.

III.6.2.1. En utilisant le facteur clarté de documents

- Résultats globaux

Pour les premiers résultats d'évaluation obtenus nous avons utilisé deux mesures, la Précision Moyenne (MAP) et la R-précision en utilisant leurs formules respectives (1.20) et (1.21) :

Nom du modèle	MAP	R-Précision
Modèle de base (dirichlet)	0.0574	0.0634
Modèle clarté de document	0.0555	0.0622

Tableau III.3 :Résultats globaux de l'évaluation de la recherche classique et de la recherche utilisant le modèle clarté de document.

Nous remarquons dans le tableau ci-dessus que les meilleurs résultats d'évaluation, précision moyenne et r-précision, sont obtenus en utilisant le modèle de recherche classique et que les résultats ne sont pas améliorés en intégrant le facteur clarté de document.


- Résultats requête par requête en variant le coefficient λ


	$\lambda=0,1$	$\lambda=0,2$	$\lambda=0,3$	$\lambda=0,4$	$\lambda=0,5$	$\lambda=0,6$	$\lambda=0,7$	$\lambda=0,8$	$\lambda=0,9$	$\lambda=1$	Recherche classique
101	0,0425	0,0428	0,0425	0,044	0,044	0,0438	0,044	0,0433	0,0432	0,0432	0,0425
102	0,1608	0,1615	0,1612	0,1584	0,1584	0,1557	0,1548	0,1545	0,1536	0,1536	0,1595
103	0,0127	0,0126	0,0126	0,0126	0,0125	0,0125	0,0125	0,0125	0,0125	0,0125	0,0127
104	0,0377	0,0377	0,038	0,0381	0,0382	0,0382	0,0381	0,0381	0,0379	0,0377	0,0378
105	0	0	0	0	0	0	0	0	0	0	0
106	0,0869	0,0875	0,0875	0,0875	0,0875	0,0875	0,0875	0,0874	0,0874	0,0874	0,0869
107	0,0131	0,0132	0,0132	0,0133	0,0138	0,0146	0,0146	0,0146	0,0146	0,0146	0,0131
108	0,0172	0,0172	0,0154	0,0154	0,0154	0,0154	0,0153	0,015	0,015	0,0149	0,0176
109	0	0	0	0	0	0	0	0	0	0	0
110	0,0314	0,0313	0,0309	0,031	0,0308	0,0314	0,0317	0,0321	0,032	0,0318	0,0312

111	0,1353	0,1345	0,1339	0,1331	0,1324	0,132	0,1314	0,131	0,1307	0,1293	0,1364
112	0,0207	0,0208	0,0207	0,0206	0,0206	0,0205	0,0204	0,0206	0,0204	0,0203	0,0207
113	0,026	0,0276	0,0275	0,0275	0,0293	0,0293	0,0313	0,0313	0,0313	0,0313	0,026
114	0,041	0,041	0,041	0,0409	0,0408	0,0405	0,0405	0,0402	0,0411	0,0397	0,0412
115	0,0101	0,0101	0,0101	0,0101	0,0101	0,0101	0,0101	0,01	0,0094	0,01	0,0101
116	0	0	0	0	0	0	0	0	0	0	0
117	0,0335	0,0337	0,0343	0,0342	0,0347	0,0344	0,0343	0,0343	0,034	0,0335	0,0333
118	0,0082	0,008	0,0081	0,008	0,008	0,0078	0,0078	0,0078	0,0078	0,0077	0,0084
119	0,0094	0,0099	0,0102	0,0102	0,0105	0,0105	0,0105	0,0105	0,0105	0,0103	0,0102
120	0,0015	0,0015	0,0015	0,0015	0,0014	0,0014	0,0014	0,0014	0,0013	0,0013	0,0015
121	0	0	0	0	0	0	0	0	0	0	0
122	0,0042	0,0042	0,0043	0,0043	0,0043	0,0043	0,0044	0,0044	0,0045	0,0045	0,0042
123	0,0111	0,0111	0,0111	0,0111	0,0114	0,0111	0,0114	0,0114	0,0115	0,0115	0,011
124	0,0874	0,0876	0,0879	0,088	0,0879	0,0881	0,0879	0,0874	0,087	0,087	0,0875
125	0,0049	0,0049	0,0049	0,0049	0,0049	0,0049	0,0049	0,0053	0,0053	0,0049	0,0049
126	0,0296	0,0293	0,029	0,0293	0,0289	0,0289	0,0289	0,0285	0,0282	0,0281	0,0296
127	0,0069	0,0069	0,0069	0,0067	0,0066	0,0066	0,0066	0,0063	0,0063	0,0061	0,007
128	0,0013	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0013
129	0,0192	0,0192	0,0184	0,0184	0,0178	0,0173	0,0173	0,0173	0,0173	0,0173	0,0207
130	0,0776	0,0767	0,0772	0,077	0,0768	0,0753	0,075	0,0741	0,0739	0,0729	0,0777
131	0,0503	0,0502	0,0501	0,0499	0,05	0,0498	0,0497	0,0501	0,05	0,0499	0,0504
132	0,078	0,078	0,0779	0,0775	0,0775	0,0775	0,0774	0,0767	0,0761	0,0754	0,0766
133	0,8304	0,8304	0,8304	0,8304	0,8304	0,8125	0,7986	0,7986	0,7986	0,7986	0,8542
134	0,7576	0,7576	0,7576	0,7576	0,75	0,75	0,7436	0,7436	0,7436	0,7436	0,7576
135	0,0209	0,0208	0,0207	0,0206	0,0205	0,0205	0,0204	0,0204	0,0203	0,0203	0,0209
136	0	0	0	0	0	0	0	0	0	0	0
137	0,0026	0,0026	0,0026	0,0026	0,0026	0,0027	0,0027	0,0027	0,0027	0,0028	0,0025
138	0,0272	0,0273	0,0272	0,0271	0,0274	0,0274	0,0273	0,0273	0,0272	0,0268	0,0271
139	0,0114	0,0114	0,0115	0,0116	0,0116	0,0108	0,0109	0,0109	0,011	0,011	0,0113
140	0,0022	0,0022	0,0022	0,0022	0,0023	0,0023	0,0023	0,0024	0,0024	0,0025	0,0021
141	0,0144	0,0144	0,0144	0,0138	0,0137	0,0136	0,0132	0,0135	0,0138	0,0138	0,014
142	0,0076	0,0079	0,0078	0,0078	0,0078	0,0078	0,0078	0,0075	0,0075	0,0076	0,0079
143	0,0009	0,0009	0,0008	0,0008	0,0009	0,0009	0,0009	0,0009	0,0009	0,0008	0,0009
144	0,0011	0,0011	0,0011	0,0012	0,0012	0,0012	0,0013	0,0013	0,0014	0,0014	0,001
145	0,0061	0,0061	0,006	0,006	0,0057	0,0057	0,0057	0,0057	0,0058	0,0058	0,0061
146	0,0401	0,0397	0,041	0,0411	0,0407	0,041	0,0407	0,0407	0,0421	0,0414	0,0389
147	0,0019	0,0019	0,0019	0,0016	0,0016	0,0016	0,0016	0,0014	0,0012	0,0012	0,0019
148	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0016	0,0015	0,0015	0,0016
149	0,0017	0,0016	0,0016	0,0012	0,0012	0,0012	0,0012	0,0012	0,0012	0,0012	0,0017
150	0,0018	0,0018	0,0018	0,0018	0,0018	0,0018	0,0016	0,0016	0,0016	0,0017	0,0018

Tableau III.4 : Résultats d'évaluation en utilisant le facteur clarté de document et en varient le coefficient λ .

 : Précision moyenne stable (égale).

 : Précision moyenne dégradés.

 : Précision moyenne améliorés.

En utilisant le facteur clarté de document pour réordonner les documents selon leurs scores de clarté nous remarquons une forte amélioration par rapport aux résultats d'évaluation de la recherche de base sur trois ensembles de requêtes, premièrement, une amélioration de 62,5% sur les requêtes allant de 104 jusqu'à 107, deuxièmement une amélioration de 80% pour les requêtes (122),(123) et (124), troisièmement une amélioration de 82,5% pour les requêtes (137), (138), (139) et (140).

Le tableau suivant fournit une statistique sur le nombre d'amélioration, d'égalité et de dégradation de la précision moyenne selon le type de la requête.

Valeur	Dégradation	Egalité	Amélioration	Global
Requêtes_amb(MAP<0,08)	13	15	16	44
Requêtes_cl(0,08<MAP<0,45)	1	0	3	4
Requêtes_cl(MAP>0,45)	1	1	0	2
Résultat global	15	16	19	50

Tableau III.5 : Résumé des résultats d'évaluation en utilisant le facteur clarté de document pour $\lambda=0,2$.

Nous avons aussi analysé nos résultats selon le type de la requête. Trois types de requêtes sont identifiés :

- Requêtes ambiguës ayant un score map inférieur ($<$) à 0,08 (Requêtes_amb).
- Requêtes claires ayant un score map supérieur à ($>$) 0,45 (Requêtes_cl).
- Requêtes moyenne ayant un score map entre 0,08 et 0,45 (Requêtes_moy).

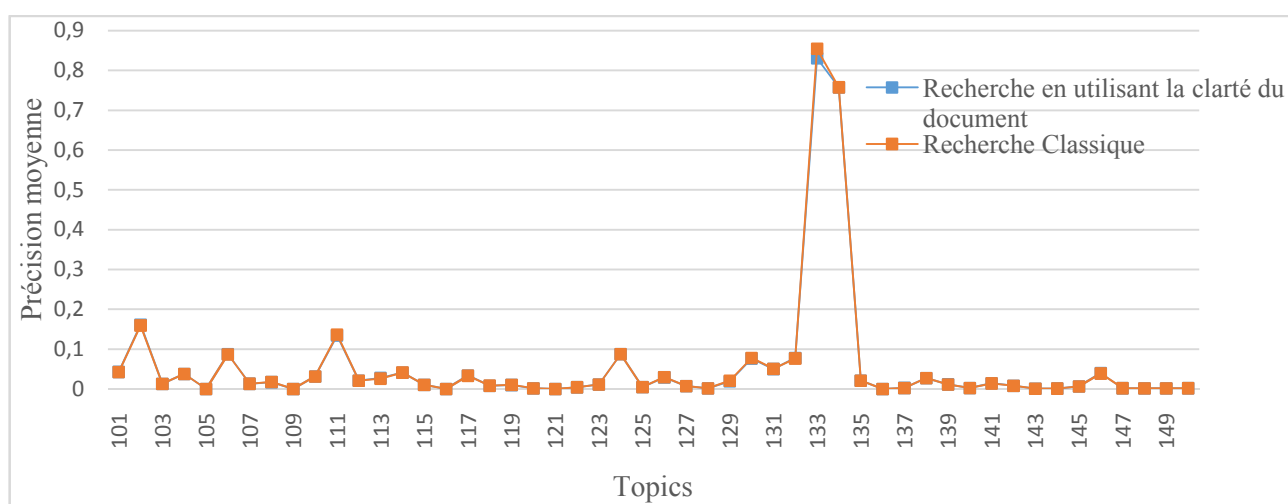


Figure III.9 : Graphe détaillant les résultats d'évaluation de la recherche en utilisant le facteur clarté de document.

Le graphe ci-dessus donne une vue sur les résultats de la précision moyenne obtenus lors de l'évaluation de la recherche classique et de la recherche en utilisant le modèle clarté de document avec le coefficient $\lambda=0,2$.

III.6.2.2. En utilisant le facteur taille de documents

- Résultats globaux

Les résultats obtenus dans le tableau suivant sont obtenus à l'aide des formules (1.20) et (1.21).

Nom du modèle	MAP	R-Précision
Modèle de base	0.0574	0.0634
Modèle taille de document	0.0512	0.0603

Tableau III.6 :Résultats globaux de l'évaluation de la recherche classique et de la recherche utilisant le modèle taille de document.

Nous remarquons dans le tableau ci-dessus que les meilleurs résultats d'évaluation sont obtenus avec le modèle de recherche classique avec une précision moyenne égale à 0,0574 et la R-précision égale à 0,0634.

- Résultats requête par requête en variant le coefficient λ

Le Tableau III.4 présente les résultats d'évaluation obtenus avec le modèle de base et le modèle taille de document en faisant varier le coefficient λ pour ce dernier de 0,1 à 1.

	$\lambda=0,1$	$\lambda=0,2$	$\lambda=0,3$	$\lambda=0,4$	$\lambda=0,5$	$\lambda=0,6$	$\lambda=0,7$	$\lambda=0,8$	$\lambda=0,9$	$\lambda=1$	Recherche classique
101	0,0425	0,0432	0,0429	0,0422	0,0432	0,042	0,0404	0,0407	0,0396	0,0394	0,0425
102	0,1595	0,157	0,1526	0,1497	0,1477	0,1484	0,1453	0,1435	0,1412	0,142	0,1595
103	0,0127	0,0127	0,0127	0,0118	0,0117	0,0101	0,0102	0,0096	0,0096	0,0097	0,0127
104	0,0378	0,0365	0,0365	0,0356	0,035	0,0344	0,0332	0,0311	0,029	0,0278	0,0378
105	0	0	0	0	0	0	0	0	0	0	0
106	0,0869	0,0875	0,0658	0,0658	0,0618	0,0588	0,057	0,0486	0,048	0,0412	0,0869
107	0,0131	0,0141	0,014	0,014	0,014	0,0144	0,0136	0,0137	0,0121	0,0123	0,0131
108	0,0176	0,0158	0,0155	0,0155	0,0156	0,0147	0,0146	0,0147	0,0147	0,014	0,0176
109	0	0	0	0	0	0	0	0	0	0	0
110	0,0312	0,0327	0,0328	0,035	0,0374	0,0386	0,0408	0,0427	0,0444	0,0456	0,0312
111	0,1364	0,1351	0,134	0,131	0,129	0,1281	0,1242	0,122	0,1188	0,1141	0,1364
112	0,0207	0,02	0,0196	0,019	0,0187	0,018	0,0175	0,0166	0,0162	0,0161	0,0207
113	0,026	0,0312	0,0312	0,0311	0,033	0,033	0,0365	0,0365	0,0365	0,0399	0,026
114	0,0412	0,0416	0,0421	0,0416	0,0416	0,0413	0,0411	0,0402	0,0413	0,0397	0,0412
115	0,0101	0,0095	0,0095	0,0107	0,0094	0,0094	0,0082	0,0083	0,0078	0,0068	0,0101
116	0	0	0	0	0	0	0	0	0	0	0

117	0,0333	0,0323	0,0335	0,0345	0,035	0,0357	0,0363	0,0333	0,0355	0,0365	0,0333
118	0,0084	0,0082	0,0084	0,0084	0,0086	0,0086	0,008	0,0083	0,0076	0,0076	0,0084
119	0,0102	0,01	0,01	0,0097	0,0095	0,0093	0,0093	0,009	0,0083	0,0078	0,0102
120	0,0015	0,0016	0,0016	0,0017	0,0017	0,0018	0,0017	0,0015	0,0014	0,0014	0,0015
121	0	0	0	0	0	0,0001	0,0001	0,0001	0,0001	0,0001	0
122	0,0042	0,0042	0,0043	0,0044	0,0044	0,0036	0,0036	0,0036	0,0037	0,0037	0,0042
123	0,011	0,0122	0,0123	0,0129	0,0132	0,0127	0,0134	0,0137	0,0146	0,0146	0,011
124	0,0875	0,0863	0,085	0,0825	0,0805	0,0797	0,0777	0,0703	0,0652	0,0637	0,0875
125	0,0049	0,0049	0,0053	0,0049	0,0048	0,0044	0,004	0,004	0,0036	0,004	0,0049
126	0,0296	0,0294	0,029	0,0286	0,0287	0,0286	0,0288	0,0283	0,028	0,0279	0,0296
127	0,007	0,0064	0,0064	0,0057	0,0055	0,0055	0,005	0,0046	0,0044	0,0041	0,007
128	0,0013	0,0016	0,0016	0,0023	0,0023	0,0027	0,0026	0,0019	0,0016	0,0016	0,0013
129	0,0207	0,0173	0,0168	0,0155	0,0145	0,0138	0,0135	0,0118	0,0106	0,009	0,0207
130	0,0777	0,0768	0,076	0,0772	0,077	0,0772	0,0769	0,0777	0,0783	0,0785	0,0777
131	0,0504	0,0513	0,053	0,0621	0,063	0,0687	0,069	0,069	0,0696	0,0656	0,0504
132	0,0766	0,0773	0,0754	0,0723	0,0677	0,0634	0,0592	0,0567	0,0535	0,0494	0,0766
133	0,8542	0,8125	0,8125	0,775	0,7611	0,7361	0,7361	0,7183	0,7183	0,7183	0,8542
134	0,7576	0,7436	0,7333	0,7333	0,7292	0,7255	0,7292	0,7292	0,7292	0,7292	0,7576
135	0,0209	0,0211	0,0204	0,0183	0,0188	0,0191	0,0196	0,02	0,0195	0,0197	0,0209
136	0	0	0	0	0	0	0	0	0	0	0
137	0,0025	0,0026	0,0026	0,0027	0,0027	0,0028	0,0029	0,0024	0,0024	0,0025	0,0025
138	0,0271	0,0266	0,0261	0,0255	0,0245	0,0232	0,0221	0,0209	0,0195	0,0187	0,0271
139	0,0113	0,0117	0,0115	0,0117	0,0118	0,0116	0,0111	0,0112	0,0114	0,0119	0,0113
140	0,0021	0,0023	0,0024	0,0025	0,0025	0,0025	0,0024	0,0024	0,0024	0,0023	0,0021
141	0,014	0,0147	0,0179	0,02	0,0201	0,0199	0,0211	0,0209	0,0197	0,0197	0,014
142	0,0079	0,0075	0,0078	0,0075	0,0073	0,0076	0,0077	0,0081	0,0088	0,0093	0,0079
143	0,0009	0,0009	0,0008	0,0008	0,0008	0,0008	0,0008	0,0008	0,0009	0,0009	0,0009
144	0,001	0,0011	0,0012	0,0012	0,0012	0,0014	0,0014	0,0014	0,0013	0,0012	0,001
145	0,0061	0,0059	0,006	0,0062	0,0062	0,0063	0,0059	0,0063	0,0054	0,0054	0,0061
146	0,0389	0,0415	0,0405	0,0409	0,0411	0,0394	0,0392	0,0403	0,0395	0,0396	0,0389
147	0,0019	0,0016	0,0014	0,0014	0,0014	0,0014	0,0012	0,0012	0,001	0,001	0,0019
148	0,0016	0,0014	0,0014	0,0013	0,0013	0,0013	0,0013	0,0013	0,0012	0,0011	0,0016
149	0,0017	0,0012	0,0012	0,0012	0,0008	0,0008	0,0008	0,0008	0,0008	0,0008	0,0017
150	0,0018	0,0016	0,0015	0,0015	0,0017	0,0017	0,0015	0,0017	0,0015	0,0016	0,0018

Tableau III.7 : Résultats d'évaluation en utilisant le facteur taille de document et en varient le coefficient λ .

: Précision moyenne stable (égale).

: Précision moyenne dégradés.

: Précision moyenne améliorés.

Nous remarquons qu'il y a aucune amélioration avec $\lambda=0.1$, mais pour λ varient entre 0,2 et 1 nous remarquons une forte amélioration de la précision moyenne pour

les requête allant de 137 jusqu'à 146 et une légère amélioration pour les requête allant de 101 jusqu'à 135 et une dégradation pour les requête (147), (148), (149) et 150.

Le tableau suivant fournit une statistique sur le nombre d'amélioration, d'égalité et de dégradation de la précision moyenne selon le type de la requête.

Valeur	Dégradation	Egalité	Amélioration	Global
Requête_amb (MAP<0,08)	20	5	19	44
Requête_moy (0,08<MAP<0,45)	4	0	0	4
Requête_cl (MAP>0,45)	2	0	0	2
Résultat global	26	5	19	50

Tableau III.8 : Résumé des résultats d'évaluation en utilisant le facteur taille de document pour $\lambda=0,5$.

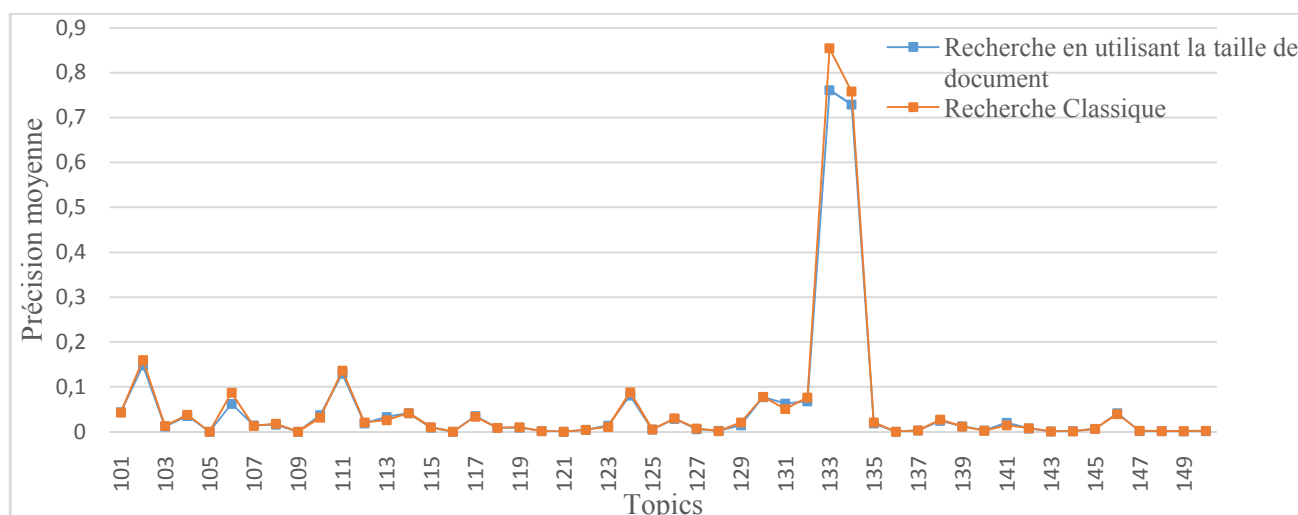



Figure III.10 : Graphe illustrant la précision moyenne de la recherche classique et la recherche en utilisant la taille de document.

Le graphe ci-dessus donne une vue sur les résultats de la précision moyenne obtenus lors de l'évaluation de la recherche classique et de la recherche en utilisant le modèle taille de document avec le coefficient $\lambda=0,5$.

III.7. Conclusion

Nous avons proposé dans ce chapitre une formule de réordonnancement des documents en se basant sur le score de clarté de document, cette formule a été testée et expérimenté sur la collection de test AP88 en utilisant la plateforme terrier. Les résultats des tests, ont révélé que la prise en compte de la clarté d'un document donne une meilleure performance par rapport au modèle utilisant la taille de document mais en la comparant à la recherche classique ce dernier est plus abouti que le modèle utilisant le réordonnancement à l'aide du score de clarté.



Conclusion générale & perspectives

Conclusion générale & perspectives

Le travail développé dans ce mémoire s'inscrit dans le cadre de la recherche d'information et plus précisément dans l'étude de l'impact du facteur clarté du document et nous nous sommes intéressés au réordonnancement des documents de la collection.

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 réordonnancement sont utilisées pour optimiser la sélection des documents.


Dans notre approche nous avons proposé d'améliorer le processus de sélection des documents en se basant sur le facteur de clarté des documents avec le score initial des documents obtenus dans la première recherche.

Pour valider les formules proposées, nous les avons évalués en utilisant la collection de test TREC AP88, sous la plateforme de RI "*Terrier*".

L'évaluation de notre approche a montré son impact positif sur les résultats de la recherche pour certaines requêtes. Plus précisément, l'analyse des expérimentations révèlent certaines améliorations de la précision moyenne (MAP). Mais en vue global nous ne constatons pas une amélioration des résultats d'évaluations par rapport au modèle classique.

Il est également à noter que nos formules sont applicables sans avoir de restrictions sur des collections de documents précises.

Comme perspectives à notre travail nous pouvons citer l'exportation de notre approche sur des collections plus volumineuses, telle que la collection (.GOV).



Références Bibliographiques

- [1]. Hammache, A. Recherche d'Information : un modèle de langue combinant mots simples et mots composés. *Thèse de Doctorat en Informatique de l'Université Mouloud Mammeri de Tizi-Ouzou*, 2011.
- [2]. Arguello, J., Elsas, J.L., Callan, J., Carbonell, J. G. Document representation and query expansion models for blog recommendation. *Proceedings of the 2nd International Conference on Weblogs and Social Media*.
- [3]. Baeza-Yates, R., Ribeiro-Neto, B. A. Modern Information Retrieval. *Pearson Education Ltd.*, Harlow, UK, 2nd edn, 2011.
- [4]. Baziz M., Indexation Conceptuelle Guidée Par Ontologie Pour La Recherche d'Information. *Thèse de Doctorat en Informatique de l'Université Paul Sabatier de Toulouse*, 2005.
- [5]. Berry, M.W., Dumais, S.T., O'Brien, G. W. 1995. Using linear algebra for intelligent information retrieval. *SIAM Rev.* 37(4), pp. 573-595, 1995.
- [6]. Borlund, P. Measures of relative relevance and ranked halflife: performance indicators for interactive ir. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 24–28, 1998.
- [7]. Boughanem, M. Outils de validation en recherche d'information. La campagne d'évaluation TREC, 2003. <http://inforsid2003.loria.fr/resumeConfRI.pdf>, 2003.
- [8]. Boughanem, M. Les Systèmes de Recherche d'Information : d'un modèle classique à un modèle connexionniste. *Thèse de Doctorat de l'Université Paul Sabatier*, 1992.
- [9]. Boughanem, M., Chrismont, C., Soule-Dupuy, C. Query modification based on relevance back-propagation in ad hoc environment. *Information Processing and Management*, 35, pp. 121–139, 1999.
- [10]. Boughanem, M., Kraaij, W., Nie, J.-Y. Modèles de langue pour la recherche d'information. Les systèmes de recherche d'informations, pages 163–182. *Hermès-Lavoisier*, 2004.
- [11]. Brown, P., DellaPietra, S., DellaPietra, V., and Mercer, R. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), pp. 263-311, 1993.
- [12]. Carpineto, C., Romano, G., «A Survey of Automatic Query Expansion in Information Retrieval». *ACM Computing Surveys*, Vol. 44, No. 1. 2012.
- [13]. Cho, J., Garcia-Molina, H., Page, L. The anatomy of efficient crawling through url ordering. *Computer Networks and ISDN Systems*, 30(1–7), pp. 161–172, 1998.
- [14]. Cleverdon, C. Progress in documentation. Evaluation of information retrieval systems. *Journal of Documentation*, 26, pp. 55–67, 1970.
- [15]. Cutts, M. Spotlight keynote. *In Proceedings of Search Engine Strategies*, 2012.

- [16]. Dang, V. and Croft, B. W. Query reformulation using anchor text. *Proceedings of the third ACM International conference on Web search and data mining*, pp. 41–50, 2010.
- [17]. Das, A. & Jain, A. Indexing the World Wide Web: The journey so far. *IGI Global*, chap. 1, 1–28, 2012.
- [18]. Dominich, S. Mathematical Foundations of Information Retrieval. *Kluwer Academic Publishers*, Dordrecht, Boston, London, 2001.
- [19]. Dumais, S. Latent Semantic Indexing (LSI). *Proceeding of TREC-3*, 1994.
- [20]. Fox, C. Lexical analysis and stoplists, Frakes W B, Baeza-Yates R (eds) *Prentice Hall*, New jersey, pp. 102–130. 1992.
- [21]. Harman, D. Relevance feedback and other query modification techniques. In *Information Retrieval: Data Structures and Algorithms*, William B. Frakes and Ricardo Baeza-Yates, editors, *Prentice Hall*, Englewood, Cliffs, NJ, pp. 241–263, 1992.
- [22]. Harter, S. Psychological relevance and information science. *Journal of the American Society for Information Science* (JASIS), 43:602–615, 1992.
- [23]. Hiemstra, D. A linguistically motivated probabilistic model of information retrieval. In Nicolaou, C., and Stephanides, C., editors, *Research and Advanced Technology for Digital Libraries - Second European Conference, ECDL '98, Proceedings*, number 1513 in *Lecture Notes in Computer Science*. Springer Verlag, pp. 569–584, 1998.
- [24]. Jacquemin, C., Daille, B., Royanté, J., and Polanco, X. In vitro evaluation of a program for machine-aided indexing. *Inf. Process. Manage.* 38, 6, pp. 765–792. 2002.
- [25]. Jelinek, F. Statistical Methods for Speech Recognition. *MIT Press*, Cambridge, MA, 1998. Références bibliographiques 142.
- [26]. Kraft, D. H. and Buehl, D. A. Fuzzy sets and generalized Boolean retrieval systems. *International Journal on Man-Machine Studies*, 19: pp. 49–56, 1983.
- [27]. Kwok, K.L. A neural network for probabilistic information retrieval. *International ACM SIGIR Conference*.
- [28]. Lafferty, J., Zhai, C. Document language models, query models, and risk minimization for information retrieval. In W.B. Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds.). *Proceedings of the 24th annual international ACM-SIGIR conference on research and development in information retrieval*, pp. 111–119, 2001.
- [29]. Lancaster, F. *Information Retrieval Systems: characteristics, testing, and evaluation*, John Wiley, New York, 1979.
- [30]. Lavrenko, V., & Croft, W. B. Relevance-based language models. In W.B. Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds.). *Proceedings of the 24th Annual International ACM-*

SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana, pp.120-127, 2001.

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

[32].Lv, Y., Zhai. C. A comparative study of methods for estimating query language models with pseudofeedback. *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1895-1898, 2009.

[33].Manning, D., Raghavan, P. AndSchute, H. Introduction to Information Retrieval. *Cambridge University Press*, 2008.

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

[35].Mizzaro, S. Relevance, the whole (hi) story. *Journal of the American Society for Information Science*, 48, pp.810–832, 1997.

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

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

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

[39].Robertson, S.E., S. Walker. On relevance weights with little relevance information. *Proceedings of the20th annual international ACM SIGIR conference on Research and development in informationretrieval*, pp. 16–24, 1997.

[40].Rocchio, J.J. Relevance Feedback in Information Retrieval, in *The Smart System Experiments in Automatic Document Processing in Automatic Document Processing*. *Editor Prentice-Gall*. pp. 313-323, 1971.

[41].Salton, G. The smart Retrieval System: Experiments in Automatic Document Processing. *Prentice-Hall*,1971.

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

[43].Salton, G., McGill, M. Introduction to Modern Information Retrieval. *McGraw-Hill Int. Book Co*, 1984.

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

- [45].Saracevic, T. Relevance reconsidered. *Conceptions of Library and Information Science*, pp. 201–218, 1996.
- [46].Singhal, A., Salton, G., Mitra, M., Buckley, C. Document length normalization. *Information Processing and Management*, 32(5), pp. 619–633, 1996.
- [47]. Van Rijsbergen, C. J. Information retrieval. London: Butterworth, 1979.
- [48].Weerkamp, W., Balog, K., and Meij, E. J. A generative language modeling approach for ranking entities. *Advances in Focused Retrieval*, 2009.
- [49].Williams, H., Zobel, J. Compressing Integers for Fast File Access. *Computer Journal* 42, pp. 193–201, 1999.
- [50].Witten, I., Moffat, A., and Bell, T. Managing Gigabytes: Compressing and Indexing Documents and Images, *Van Nostrand Reinhold*, New York, 1994.
- [51].Wong, S., Ziarko, W., Wong, P. Generalized vector space model in information retrieval. *Proceedings of the 8th ACM SIGIR Conference on Research and Development in information retrieval*, New York, pp.18–25, 1985.
- [52]. Zhai, C., Lafferty, J. A study of smoothing methods for language models applied to ad hoc information retrieval. In W.B. Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds.), *Proceedings of the 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 334–342, 2001.
- [53].Porter, M. An algorithm for suffix stripping. *Program*, Vol. 14(3), pp. 130–137, 1980.
- [54].Robertson, S. E. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4), pp. 294–304, 1977.
- [55].Kraaij, W., Westerveld, D., Hiemstra, D. The Importance of Prior Probabilities for Entry Page Search. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 27–34, 2002.
- [56].Miller, D.R.H., Leek, T., Schwartz, R.M. A hidden markov model information retrieval system. *Hearst et al.*, pp. 214–221, 1999.
- [57].Hauff, C., Azzopardi, L. Age dependent document priors in link structure analysis. *European Conference in Information Retrieval*, pp 552–554, 2005.
- [58].Parapar, J., E.Losada, D., Barreiro, A. Compression-Based Document Length Prior for Language Model. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 652–653, 2009.
- [59].Li, X., Croft, W.B. Time-based language models. *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 469–475, 2003.

[60].Haveliwala, T.H. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEETransactions on Knowledge and Data Engineering*, Volume 15(4), pp. 784–796, 2003.

[61]. Hawking, D., Craswal, N. Overview of the TREC-2001 web track". *Proceeding of TREC-10, NISTpublication #500-250*, Gaithersburg, pp. 61-67, 2002.

[62]. Hawking, D., Craswal, N. Overview of the TREC-2002 web track". *Proceeding of TREC-11, NISTpublication #500-251*, Gaithersburg, pp. 86-95, 2003.

[63].Kleinberg, J.M. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM* 46, 5, pp. 604–632, 1999.

[64].Lempel, R., Moran, S. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect.*Proceedings of the 9th international World Wide Web conference on Computer networks: the internationaljournal of computer and telecommunications networking*, pp. 387-401, 2000.

[65]. Adamson, G., Boreham, J. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Journal of Information Storage and Retrieval* vol. 10, no. 7-8, pp. 253-260, 1974.

[66].Steve Cronen-T ownsend, W. Bruce Croft. Quantifying Query Ambiguity. *University of Massachusetts, Amherst, MA 01003*

Annexe

La plateforme de RI Terrier 2.1

1. Introduction :

Terrier, *terabyte retriever*, est un projet initialisé par l'université de Glasgow en 2000, dans le but de fournir une plateforme flexible pour le développement rapide des applications de recherche d'information. Terrier est un produit Open source écrit en Java, qu'il a été mis à disposition du général public depuis novembre 2004 sous la licence de MPL.

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

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

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

Recherche : permet de générer des résultats aux requêtes formulées par l'utilisateur.

2. Installation de terrier :

Pour pouvoir utiliser terrier il est nécessaire d'installer une JRE (1.5.0 ou plus).

La JRE ou la JDK peuvent être téléchargés sur le site de Java. Puis aller sur le terrier pour le télécharger <http://www.terrier.org>

Ensuite dézipper la copie téléchargée dans le répertoire que vous souhaitez installer terrier

3. la structure des répertoires de terrier :

Les répertoires de terrier sont structurés comme suite :

- Bin\ : contient les scripts nécessaires pour démarrer terrier.
- Doc\ : contient la documentation relative à Terrier.
- Etc\ : contient les fichiers de configuration de Terrier.
- Lib\ : contient les classes compilées de Terrier et les différentes librairies externes utilisées par Terrier.
- Sare\ : contient la liste des mots vides (stop wordlist).
- Src\ : contient le code source de Terrier.
- Var\index : contient les structures de donnée : fichier inverse, fichier lexicon, index direct, document index.
- Var\result : contient les résultats de la licence des différents composants inclus dans Terrier.

4. les applications de Terrier :

Terrier offre application :

Batch (TREC) Terrier : permet l'indexation, la recherche et l'évaluation des résultats d'une collection TREC.

Interactive Terrier : permet une recherche interactive en exécutant le script `interactive_terrier`. C'est un moyen facile de tester Terrier.

Desktop Terrier : c'est une interface graphique pour la plateforme de RI Terrier.

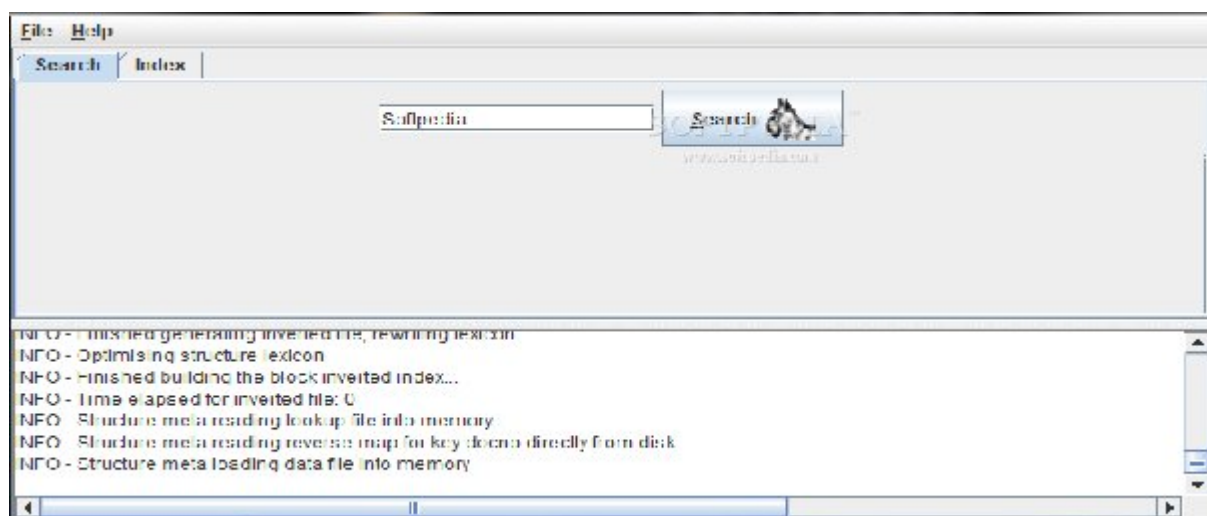


Figure 1 : Présentation de l'interface Desktop Terrier.

5. L'API d'indexation de Terrier :

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

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 en extraire les termes ainsi que les informations relatives.
3. Traduction des Termes extraits en utilisant *TermPipelines* (mots vides, lemmatisation).
4. La construction de l'index via l'utilisateur de la classe *Indexer*.

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

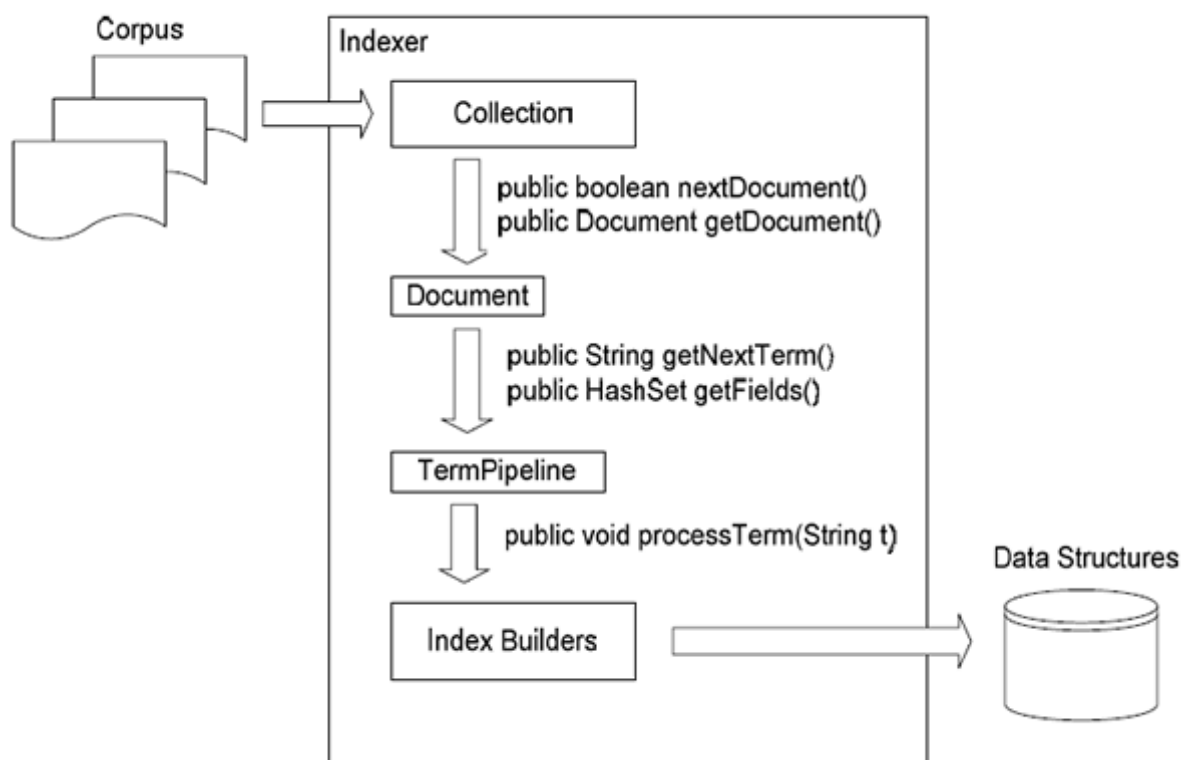


Figure2 : présentation du processus d'indexation dans Terrier.

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

5.1 Collection :

Cette composante englobe le concept le plus fondamental de l'indexation avec Terrier qui est 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 `uk.ac.gla.terrier.indexing`. Utilisée généralement par Terrier pour intégrer de nouvelles source de donnée (nouveau corpus) et cela en ajoutant une nouvelle classe java qui implémente cette interface, dans notre cas nous avons utilisé la classe `TRECCollection` qui permet de traiter les collections de type TERC (à savoir AP88 et WSJ9092). Elle permet de parcourir un ensemble de document et de renvoyer un objet document en faisant à sa méthode `nextDocument ()`.

Terrier offre plusieurs classes qui implémentent cette interface telle que `SimpleFileCollection`, `TRECUTFCollection`, `Simple XMLCollection`, `TRECCollection`

5.2 Document :

C'est une interface qui se trouve le package `uk.ac.terrier.indexing`. Cette composante englobe le concept de document. Les classes qui implémentent cette interface s'occupe de parcourir les documents et dont extraire les différents Termes. Terrier possède plusieurs parseurs de documents, par exemple : `HTMLDocument`, `fileDocument`, `MSExelDocument`, etc.

5.3 Term pipeline :

C'est une interface qui se trouve dans le package `uk.ac.gla.terrire.terms`. Cette composante reçoit l'ensemble des termes extraient des documents. Elle est considérée comme étant un pipeline qui traite et transforme chaque terme.

5.4 indexer :

Cette composante est chargée de la gestion du processus d'indexation. Entre autre la construction de l'index et de son écriture dans la structure de donnée appropriée.

Terrier offre deux types d'indexeur : `basickIndexer`.

5.5 Les structures d'Index :

Terrier 2.1

L'index de terrier est composé de quatre structures de donnée principales que sont :

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

Structure d'Index	Contenu
lexicon	Term : nom du terme. Term id : identificateur du terme. Document Frequency : fréquence en document pour le terme. Termfrequency : fréquence globale dans la collection. Byte offset il inverted file : pour avoir la position dans le fichier inverse.
Inverted Index	Document id : identificateur du document d'appariement. TermFrequency : la fréquence du terme dans le document. Fields (# of fields bits) : les champs dans lesquels le terme apparait sont codés en utilisant un bite Set. Block Frequency : fréquence dans une fenêtre. [Block id] : l'identificateur du bloc dans le document ou le terme apparait.
Document Index	Document id : identificateur de document. Document Lengh : la longueur de document. Document Number : numéro de document. Byte offset in direct file : pour avoir la position

Terrier 2.1

	dans le fichier direct.
Direct Index	Term id : identificateur du terme. Termfrequency : la fréquence du terme. Fields (# of fields bits) Block frequency : fréquence dans une fenêtre. [Block id] : identificateur du bloc.

Tableau 1 : présentation des structures d'Index.

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

6. L'API de recherche de Terrier :

Terrier utilise trois composants principaux pour la recherche : Query, Manage, Matching qui sera décrit ci-dessous :

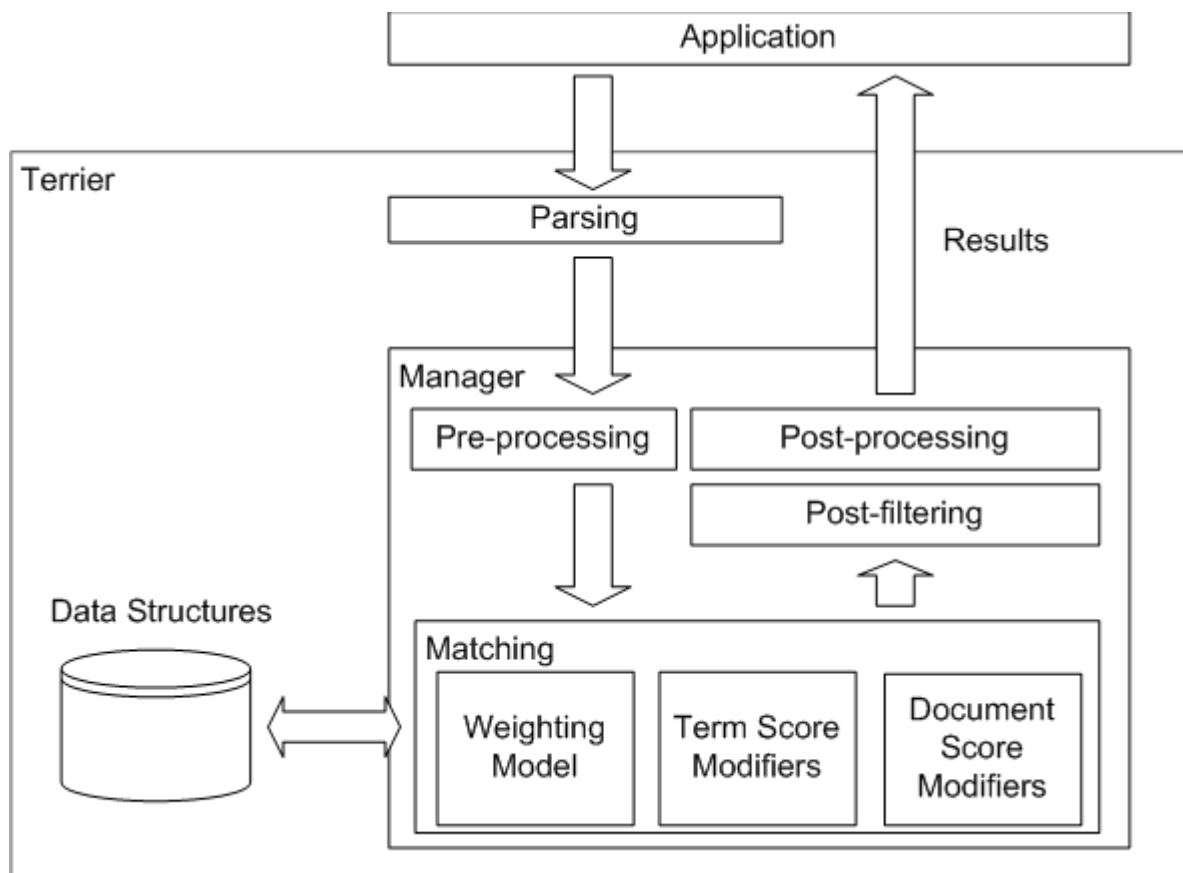


Figure 3 : présentation du processus de recherche de Terrier.

6.1 Query :

C'est l'entrée que l'application offre à terrier. Le module *Matching* est responsable de déterminer quel document correspond à une requête spécifique et attribue un score au document qui respecte la requête.

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

6.2 Manage :

C'est le modèle qui est chargé de la gestion de la recherche. Dans un premier niveau il lit chaque requête en utilisant le parseur approprié. Puis il crée un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaires et en spécifiant un modèle de pondération. Puis il crée un fichier résultat (*result file*) 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-filtrage.

6.3 Matching :

Le composant *Matching* est responsable de déterminer les documents correspondant à la requête spécifiée et donne un score au document qui vérifie la requête. Il utilise l'interface *weightingmodels* pour assigner un score à chaque mot de la requête dans le document. Terrier offre plusieurs classes qui implémentent cette interface parmi eux (TF-IDF, BM25).

- *Document Score Modifier* : Modifie le score des documents en fonction des propriétés du document.
- *Term Score Modifiers* : modifie le score des documents en fonction de la position des termes.

Post-processing

Après l'application de *termScoreModifers* ou *documentScoreModifers*, l'ensemble de documents recherchés (retourné par l'opération de *Matching*) sera modifié en appliquant le *Post-processing* ou le *post-fltering*.

Le *post-processing* est approprié pour implémenter des fonctionnalités, qui apportent un changement à la requête originale. Un exemple de *post-processing* est l'expansion de requête (EQ), puis exécute une autre fois le *Matching* avec cette nouvelle requête. Un autre exemple possible de *post-processing* est l'application de *clustering*.

Post-filtrring :

Le *post filtrring* est l'étape finale du processus de recherche de terrier, ou une série de filtres peut enlever les documents déjà recherchés, qui ne satisfont pas une condition donnée, par exemple, dans le contexte d'un moteur de recherche Web, un *post filtre* peut être utilisé pour réduire le nombre de documents recherchés depuis le même site Web, afin d'augmenter la diversité dans les résultats.

7. modification de Terrier :

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

8. Utilisation de batch (TREC) Terrier :

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

8.1 Indexation :

1. Aller dans le répertoire où Terrier est installé et utilisons la commande `cd` :

```
>>>cd master\terrier
```

Tel que Master contient la copie téléchargée de terrier

2. initialiser Terrier (supprimer le contenu du dossier *index* dans *var*) pour effectuer l'indexation d'une nouvelle collection TREC on utilisant l'option suivant :

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

3. maintenant nous pouvons indexer la collection TREC en utilisant l'option `-i` (indexer) comme suit :

```
>>.\bin\trec_terrier-i
```

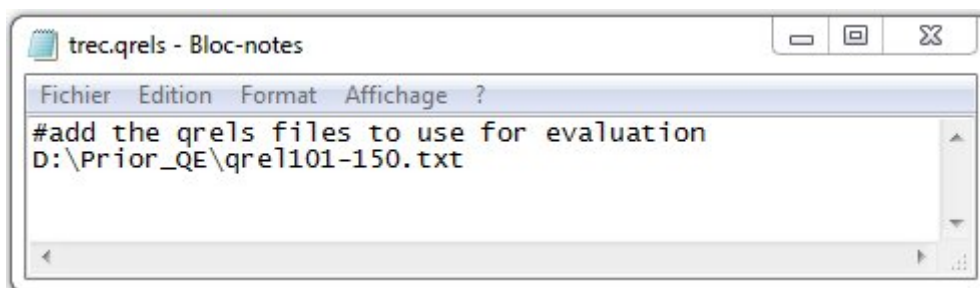
Remarque :

Pour indexer une collection autre qu'une collection TREC, il est nécessaire d'apporter quelque modification au fichier *terrier.properties*.

8.2 Recherche :

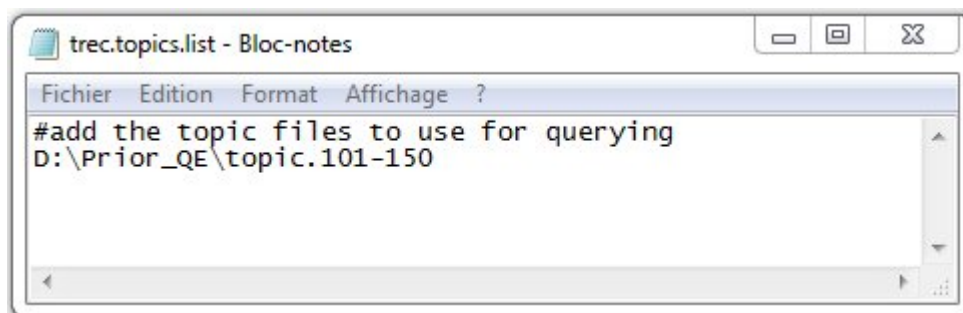
Avant toute recherche et évaluation il est nécessaire de réaliser les trois étapes suivantes :

1. spécification du fichier de jugement de pertinence (dans notre cas *6.txt*) dans le fichier *etc\trec.qrels*.



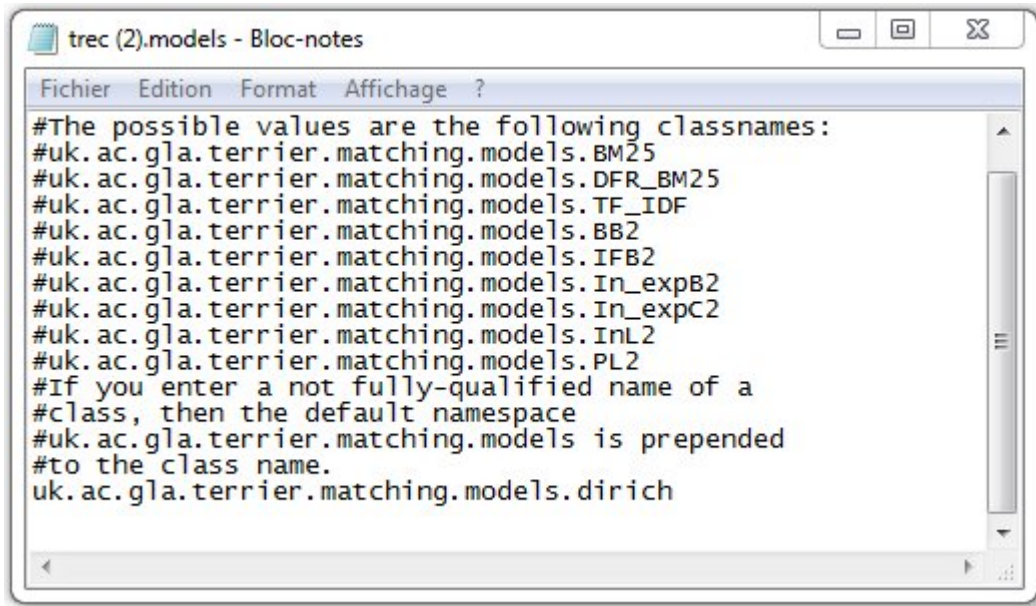
Tel que *6.txt* est le fichier de jugement de pertinence.

2. spécification du fichier contenant les requêtes (*topics.251-300*) dans le fichier *etc\trec.topics.list*.



3. spécification du modèle de pondération à utiliser dans le fichier etc\terc.models.

Pour cela il suffit de décocher le modèle choisir comme suit :



Maintenant on peut lancer la recherche ou l'évaluation.

4. pour lancer la recherche dans Terrier on utilise l'option-r (retrieval) comme suit :

.\bin\terc_terrier-r

5. Les résultats de la recherche peuvent être évalués en exécutant la commande :

.\bin\terc_terrier-e