

Chapitre I

Recherche d'Information

Chapitre II

Reformulation de Requête

Chapitre III

*Implémentation et
évaluation d'une
approche de sélection
de document d'expansion*

Introduction

La recherche d'information (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 faisaient 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,...etc.), l'accès à cette dernière devient de plus en plus exigeant en matière de temps 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 su-cités.

I.1. La recherche d'information (RI) [Urfist, 2004]

I.1.1. Définition

- D'après (AFNOR, 79), *La recherche d'information (RI) est un ensemble de méthodes et de procédures ayant pour objet extraire d'une collection de documents, les informations voulues. Dans un sens plus large, la RI est toute opération ayant pour but la collecte, la recherche et l'exploitation de l'information en réponse à une question sur un sujet précis.*
- Un Système de Recherche d'Information (SRI) permet de retrouver à partir d'une collection de documents les documents pertinents répondant à une requête d'utilisateur.

I.1.2. Notions de base

A partir de la définition de la RI, on peut extraire les concepts de base suivant :

A. 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.

B. 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 différents langages de requêtes: le langage naturel, le langage à base de mots 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'autre part.

C. Notion de pertinence

Elle est l'objet de tout système de recherche d'information. C'est une notion fondamentale en RI et elle est subjective car elle dépend de l'utilisateur, mais reste toutefois mal définie. Plusieurs définitions lui sont attribuées parmi celles-ci on retient [Nie, 2004]: La pertinence est :

- ✓ la correspondance entre un document et une requête, une mesure d'informativité du document à la requête;
- ✓ un degré de relation (chevauchement, relativité, ...) entre le document et la requête;
- ✓ une mesure d'utilité du document pour l'utilisateur.

La notion de pertinence est une notion subjective. La relation qu'elle induit est non intrinsèque et par conséquent difficile à formaliser et à automatiser. La pertinence peut être perçue selon deux niveaux : le niveau système et le niveau utilisateur.

❖ **Le niveau utilisateur** : A ce niveau, l'utilisateur a un besoin mental en information et espère obtenir les documents pertinents pour répondre à ce besoin, donc la pertinence utilisateur correspond à l'ensemble des jugements de pertinence que produit l'utilisateur qui utilise le système (le document est jugé pertinent par l'utilisateur).

❖ *Le niveau système*: A ce niveau, le système répond à la requête formulée par l'utilisateur, par un ensemble de documents trouvés (les documents sont jugés pertinents par le système) dans la base documentaire qu'il possède.

I.2. Le système de recherche d'information

I.2.1. Définition

C'est un outil logiciel et donc un ensemble de programmes informatiques qui a pour but la mise en relation des informations contenues dans le corpus documentaire d'une part, et le besoin de l'utilisateur d'autre part afin de sélectionner le maximum de documents pertinents à la requête de l'utilisateur(et le minimum de documents non-Pertinents).

I.2.2. Fonctionnement

De manière générale, la recherche dans un SRI consiste à comparer la représentation interne de la requête aux représentations internes des documents de la collection. La requête est formulée, par l'utilisateur, dans un langage de requêtes mais elle sera transformée en une représentation interne équivalente, lors d'un processus d'interprétation. Un processus similaire, dit indexation, permet de construire la représentation interne des documents de la base documentaire. Le processus de recherche consiste alors à mettre en correspondance et à calculer le degré d'appariement des représentations internes des documents et de la requête. Les documents qui correspondent au mieux à la requête, ou documents dits pertinents, sont alors retournés à l'utilisateur, dans une liste ordonnée par ordre décroissant de degré de pertinence lorsque le système le permet. Afin d'améliorer les résultats de la recherche, le système peut être doté d'un mécanisme d'amélioration et de raffinement de la requête par reformulation.

I.3. Processus de Recherche d'Information

Le fonctionnement général d'un SRI est donné au travers du processus de recherche communément appelé processus en U présenté en figure I.1

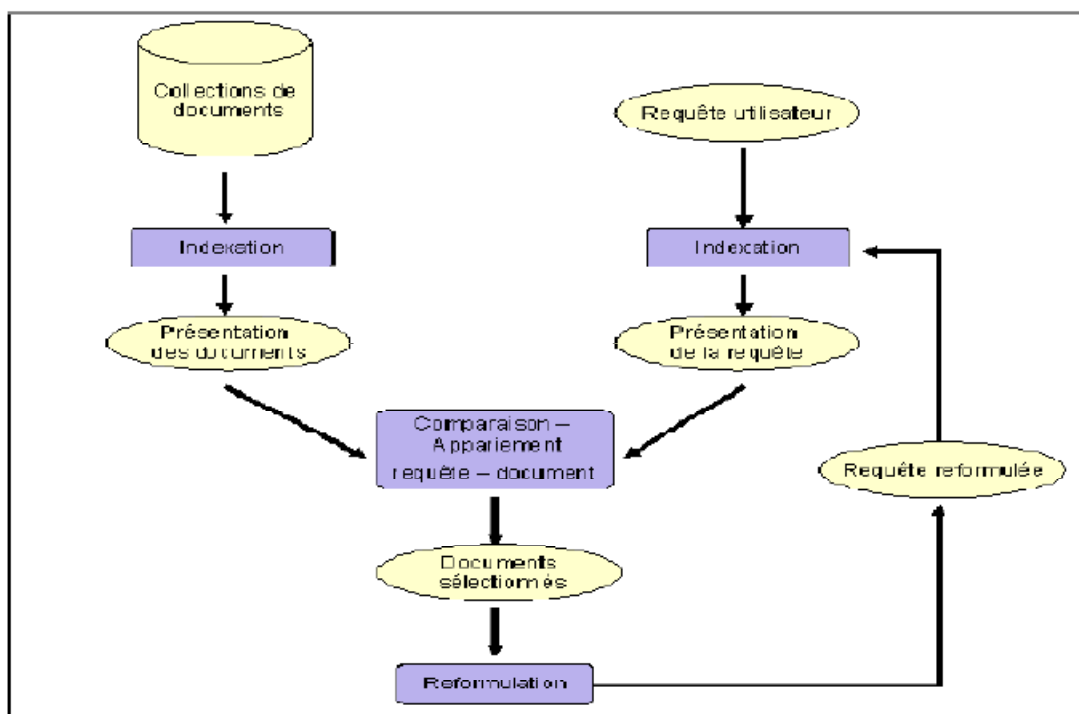


Figure I.1 : Processus de Recherche d'Information. [Hlaoua, 07]

Ce processus fait ressortir trois mécanismes de base :

I.3.1. Le processus d'indexation

Quelques fois dit processus d'interprétation pour les requêtes, c'est une étape primordiale sur la base documentaire. Cette étape consiste à analyser chaque document de la collection afin de créer un ensemble de mots-clés : on parle de l'étape d'indexation. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet ainsi de créer une représentation des documents dans le système. Son objectif est de trouver les concepts les plus importants du document (ou de la requête), qui formeront le descripteur du document. L'indexation peut être :

A. Indexation Manuelle

Chaque document est analysé par un spécialiste du domaine ou par un documentaliste .Elle permet d'assurer une meilleure pertinence dans les réponses apportées par le SRI. Elle présente toutefois plusieurs inconvénients : deux indexeurs différents peuvent présenter des termes différents pour caractériser un même document. Les agents doivent avoir une connaissance minimale des contenus des documents pour pouvoir choisir les informations

qui les caractérisent sinon l'indexation risque d'être erronée. De plus, le temps nécessaire à sa réalisation est très important.

B. Indexation Semi-automatique

Dans le cas d'une indexation semi automatique le choix final revient au spécialiste ou au documentaliste, qui intervient souvent pour choisir d'autres termes significatifs. Dans ce cadre les indexeurs utilisent un thesaurus ou une base terminologique, qui est une liste organisée de descripteurs (mots clés) obéissant à des règles terminologiques propres et reliés entre eux par des relations sémantiques.

C. Indexation Automatique

Le processus d'indexation est entièrement informatisé, elle présente l'avantage d'une régularité de l'indexation, le document aura toujours le même index.

Dans les systèmes de recherche d'informations, l'indexation automatique est organisée en trois étapes : *l'Extraction, la sélection et la pondération des termes etc.* L'objectif final est de définir pour chaque document ses termes d'indexation

I.3.1.1. Les étapes du processus d'indexation automatique.

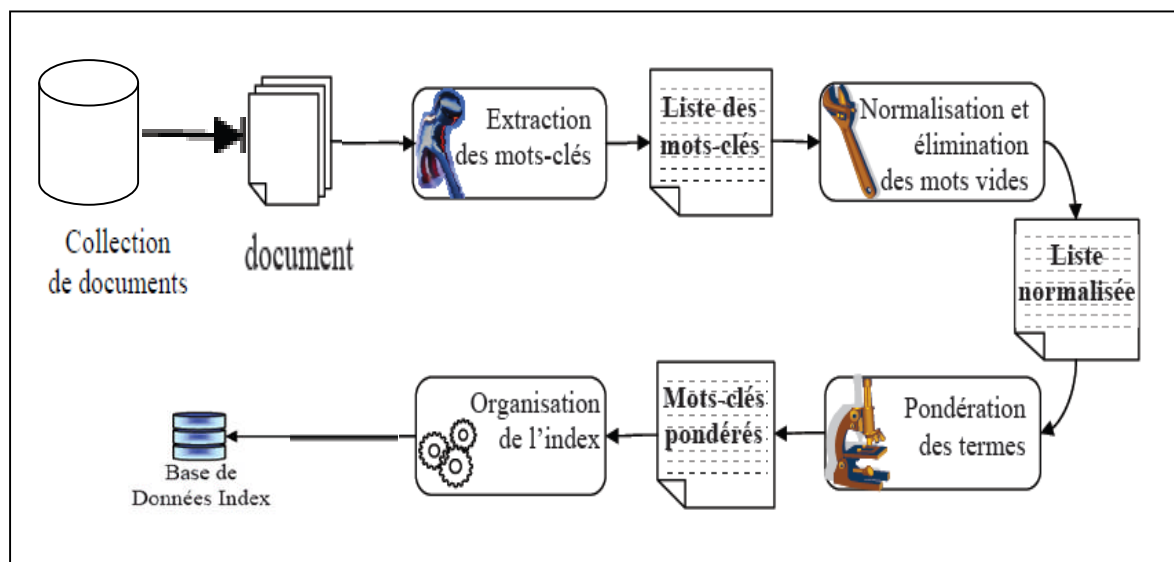


Figure I.2 : Étapes du Processus d'indexation automatique.

A. L'extraction des mots-clés (terme) du document

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

B. La normalisation (radicalisation ou lemmatisation) des mots-clés du document

Pour des raisons grammaticales, les documents utilisent différentes formes d'un mot, alors cette étape consiste à réduire les mots à leur forme canonique, à leur racine : toutes les formes d'un verbe par exemple sont regroupées à l'infinitif, tous les mots au pluriel sont ramenés au singulier, etc. L'objectif de la lemmatisation est de rendre l'ensemble des formes des mots de la même famille représentées par un seul mot pour toute la famille. C'est la forme commune entre eux, qui est la forme de base (radical, par exemple flexible pour flexible, flexiblement, flexibilité, etc.)

C. L'élimination des mots vides

L'une des étapes dans le processus d'indexation permettant d'améliorer la fiabilité d'un SRI au sens de qualité logiciel (temps d'exécution) et de performance, est l'élimination des mots vides (pronoms personnels, prépositions, etc.). Ce sont des mots qui ne traitent pas le sujet d'un document. L'élimination de ces mots permet de réduire l'index, on gagne alors en espace mémoire, mais aussi le non traitement des mots vides fait gagner un SRI en temps d'exécution. On distingue deux techniques pour filtrer les mots vides :

- L'utilisation d'une liste prédéfinie de mots vides aussi appelée anti-dictionnaire ou stop-list,
- Le comptage du nombre d'apparition d'un mot dans un document de la collection : On supprime les mots ayant une fréquence qui dépasse un certain seuil et qui deviennent alors vraisemblablement des mots vides.

D. La pondération des mots-clés

La pondération est l'une des fonctions fondamentales en RI. Elle est la clé de voûte de la majorité des modèles et approches de RI proposée depuis le début des années 1960. Elle permet de définir l'importance qu'a un terme dans un document donné, elle est également utilisée pour filtrer l'index résultant du processus d'indexation (c'est-à-dire : éliminer les index dont le poids est inférieur à un certain seuil). Le poids d'un terme dans un document

traduit l'importance de ce terme dans ce document. Cette mesure est souvent calculée en se basant sur les propriétés statistiques suivantes :

- ***Les statistiques sur le texte***

La majorité des modèles et approches proposés en RI se base sur des considérations et interprétations plus statistiques que linguistiques, pour par exemple identifier les mots importants du document, ou encore mesurer la pertinence d'un document vis-à-vis d'une requête. Ainsi, dans le cas de l'indexation, les approches statistiques supposent explicitement ou implicitement que les données textuelles des documents suivent une certaine distribution statistique. Ces propriétés sont fondamentales pour, par exemple, assigner une importance à un terme dans le document. Alors nous présentons quelques distributions utilisées pour représenter les données textuelles dans le contexte de la RI.

- ***Loi de Zipf*** : Les premières études sur la distribution des termes ont été effectuées par George Zipf [Zipf, 1949]. Zipf a réalisé plusieurs études sur l'utilisation des termes dans le contenu des documents d'une collection. Il a observé que les termes suivent plusieurs lois empiriques, représentées par une règle basée sur le principe à moindre effort (Principle of Least Effort).

L'une des lois utilisant le principe à moindre effort est désignée par la loi de Zipf. Elle décrit la distribution de la fréquence des termes dans une collection. Si on dresse une liste (Figure I.3) de l'ensemble des mots différents d'un texte quelconque, classés par ordre de fréquences décroissantes, on constate que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste. Formellement, ceci peut être traduit par la formule suivante :

$$\text{fréquence} \times \text{rang} = \text{constante}$$

Zipf explique la courbe hyperbolique de la distribution des termes par ce qu'il appelle le principe à moindre effort ; il considère qu'il est plus facile pour un auteur d'un document, de répéter certains termes que d'en utiliser des nouveaux. La relation entre la fréquence et le rang des termes permet de sélectionner les termes représentatifs d'un document. La sélection de termes discriminants par la loi de Zipf consiste à éliminer respectivement les termes de fréquences très élevées car ne sont pas discriminants entre le document et la collection, et les termes de fréquences très faibles (exemple, ceux qui apparaissent dans

très peu de documents), car ils sont rarement utilisés dans une requête. En utilisant cette approche, la taille du langage d'indexation d'une collection peut être réduite considérablement.

La loi de Zipf est plutôt intéressante pour caractériser les termes dans une collection, mais moins intéressante si on veut distinguer les documents d'une collection. Exemple des SRI qui proposent d'ordonner les documents par rapport à leurs probabilités de pertinence vis-à-vis d'une requête. L'estimation de ces probabilités nécessite, en effet, de caractériser la distribution statistique des termes de la requête dans les documents, et leur distribution dans toute la collection.

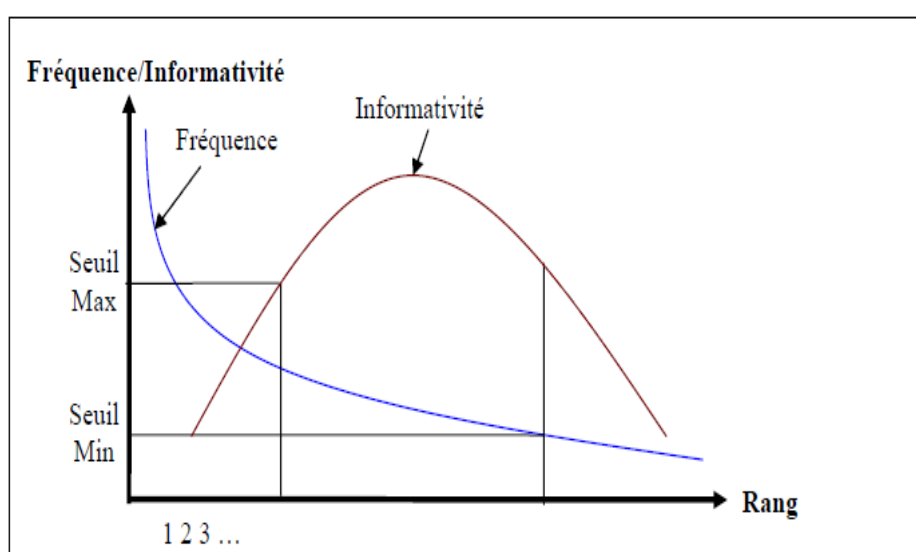


Figure I.3 : La correspondance entre l'informativité et la fréquence.

- ***Distribution binomiale***

La distribution binomiale peut être utilisée pour modéliser les occurrences des termes dans les textes d'une collection de documents. Les termes dans une collection peuvent être vus comme une séquence de n épreuves, où p représente la probabilité qu'un terme soit présent dans la collection, et $(1 - p)$ sinon. Une propriété intéressante de la distribution binomiale est que, pour des valeurs de n telle que $np(1 - p) > 5$, on peut utiliser une approximation par la loi normale. Cependant, dans l'analyse des textes, les hypothèses exigées pour une approximation d'une loi normale, ne sont pas souvent vérifiées en raison du problème des données rares. On sait, d'après la loi de Zipf, que beaucoup de termes apparaissent

rarement dans le texte de la collection. Une alternative d'utiliser une approximation normale est de se baser sur une estimation par une loi multinomiale [Dunning, 1993].

- ***Distribution multinomiale***

La distribution multinomiale est une extension de la distribution binomiale. Au lieu de supposer qu'une épreuve d'une réalisation soit représentée par deux événements possibles, elle est représentée par m événements possibles.

- ***Distribution de poisson***

La distribution de poisson est l'une des distributions probabilistes standard qui est utilisée pour modéliser le nombre d'occurrences d'un certain nombre aléatoire d'événements dans un échantillon de taille fixe. La distribution poissonnienne est décrite par :

$$p(k; \lambda_i) = e^{-\lambda_i} \lambda_i^k / k!$$

Où, $p(k; \lambda_i)$ est la probabilité qu'un événement i se réalise k fois dans l'échantillon. La distribution de poisson est caractérisée par les deux propriétés de l'espérance et de variance qui sont égales à λ_i . La distribution de poisson est une limite de la loi binomiale lorsque le nombre d'événements tend vers l'infini et la probabilité p tend vers zéro.

La distribution de poisson a été utilisée dans le domaine de la RI pour modéliser la distribution des termes dans les documents [Kraaij, 2004]. Par exemple, calculer la probabilité d'apparition d'un certain terme t_i de fréquence f_k dans un nombre aléatoire de documents : $p_i(f_k) = p(f_k; \lambda_i)$. Le paramètre λ_i est la fréquence moyenne du terme i dans une collection. Cette fréquence est égale à la fréquence globale du terme g_{tf_i} (nombre d'occurrences du terme i dans la collection) divisé par le nombre de documents. Ce facteur a été utilisé dans les travaux de Manning et Schutze [Manning and Schutze, 1999].

Une étude intéressante sur l'utilisation de ces statistiques dans le texte peut être trouvée dans Kraaij [Kraaij, 2004].

De manière générale, la majorité des formules de pondération des termes est construite par combinaison de deux facteurs. Un facteur de pondération locale quantifiant la représentativité locale d'un terme dans le document, et un second facteur de pondération

globale mesurant la représentativité globale du terme vis-à-vis de la collection des documents.

Il existe plusieurs techniques de pondération des termes elle se base généralement sur les facteurs suivants:

D.1. Pondération locale

La pondération locale permet de mesurer la représentativité locale d'un terme. Elle prend en compte les informations locales du terme par rapport à un document donné. Elle indique l'importance du terme dans ce document. Les fonctions de pondération locales les plus utilisées sont les suivantes :

Fonction brut de tf_{ij} (term frequency) : correspond au nombre d'occurrences du terme t_i dans le document D_j :

Fonction binaire : elle vaut 1 si la fréquence d'occurrence du terme dans le document est supérieure ou égale à 1, et 0 sinon.

Fonction logarithmique : combine tf_{ij} avec un logarithme, donné par la formule suivante: $\alpha + \log (tf_{ij})$, où α est une constante. Cette fonction permet d'atténuer les effets de larges différences entre les fréquences d'occurrence des termes dans le document.

Fonction normalisée : permet de réduire les différences entre les valeurs associées aux termes du document, et elle est donnée comme suit :

$$0.5 + 0.5 \times \frac{tf_{ij}}{\max_{t_i \in D_j} tf_{ij}}$$

Où $\max_{t_i \in D_j} tf_{ij}$ est la plus grande valeur de tf_{ij} des termes du document D_j .

D.2. Pondération globale

La pondération globale prend en compte des informations concernant un terme par rapport à la collection de documents. Elle indique la représentativité globale du terme dans l'ensemble des documents de la collection. Un poids plus important doit être donné aux termes qui apparaissent moins fréquemment dans la collection : les termes qui sont utilisés dans de nombreux documents sont moins utiles pour la discrimination que ceux qui apparaissent dans peu de documents. Le facteur de pondération globale qui dépend de la

fréquence inverse dans le document a été introduit, comme le facteur IDF (pour Inverted Document Frequency) donné par plusieurs formules :

$$\mathbf{IDF} = \log\left(\frac{N}{n_i}\right)$$

Où

$$\mathbf{IDF} = \log\left(\frac{N \cdot n_i}{N}\right)$$

Où, n_i est le nombre de documents où le terme t_i apparaît dans une collection de documents de taille N .

De ce fait, par cette double pondération locale et globale, les fonctions de pondérations sont souvent référencées sous le nom de TF-IDF.

Les pondérations locales et globales ne tiennent pas compte d'un aspect important du document : *sa longueur*. En général, les documents les plus longs auront tendance à utiliser les mêmes termes de façon répétée, ou à utiliser plus de termes pour décrire un sujet. Par conséquent, les fréquences des termes dans les documents seront plus élevées, et les similarités avec la requête seront également plus grandes. En effet, certaines mesures normalisent la formulation de la fonction de pondération en intégrant la taille des documents, ce qu'on appelle facteur de normalisation. Robertson et Spark-Jones

[Robertson and Sparck-Jones, 1997] proposent de normaliser la fonction de pondération de la façon suivante :

$$\mathbf{wd}_{ij} = \frac{tf_{ij} \cdot (K_1) \cdot \log \frac{N}{n_i}}{K_1 \cdot \left((1-b) + b \cdot \frac{dl_j}{\Delta l} \right) + tf_{ji}}$$

Où, \mathbf{wd}_{ij} est le poids du terme t_i dans le document \mathbf{D}_j ; \mathbf{K}_1 contrôle l'influence de la fréquence du terme t_i , sa valeur optimale dépend de la longueur et de l'hétérogénéité des documents dans la collection de documents (dans TREC, $K_1 = 1,2$) ; \mathbf{b} est une constante appartenant à l'intervalle $[0, 1]$ et contrôle l'effet de la longueur du document (dans TREC, elle est fixée à 0.75) ; \mathbf{dl}_j est la longueur du document \mathbf{D}_j , et $\Delta \mathbf{l}$ est la longueur moyenne des documents dans la collection entière.

E. Construction de l'index [Benaouicha, 09]

Au terme de toutes les différentes étapes expliquées ci-avant (analyse lexicale, élimination des mots vides, radicalisation et choix de la technique de pondération d'un terme), il devient nécessaire d'organiser et de stocker les informations sélectionnées. Le stockage de données peut se faire dans des fichiers structurés ou non structurés. Le fichier de base dans lequel sont stockées les données est appelé fichier direct. L'opération peut durer quelques secondes sur un fichier direct de quelques centaines d'enregistrements, cependant, elle peut se révéler très lente si la base atteint des milliers de documents.

Les fichiers inverses sont créés autour du fichier direct. Ces fichiers comme leur nom l'indiquent, sont le résultat de l'inversion du fichier direct. Plus exactement, au lieu de donner pour chaque document les mots et les fréquences qui le constituent, on donne pour chaque mot les documents qui le contiennent et sa fréquence dans chacun. Le tableau I.1 illustre un exemple de fichier inverse construit à partir de la collection illustrée par le tableau I.2.

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

Tableau I.1: Exemple de fichier direct.

terme	d_1	d_2	d_3
recherche	4	15	3
information	16	27, 56	5
gère	28		
textes	37		
système		4	
restituer		44	
pertinente		68	21
utilisateur		83	52
satisfait			40

Tableau I.2: Exemple de fichier inverse.

I.3.2. Le processus de recherche

La comparaison entre le document et la requête revient à calculer un score, représentant la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction ou d'une probabilité de similarité notée RSV (Q,d) (Retrieval Status Value), où Q est une requête et d un document.

Cette mesure tient compte du poids des termes dans les documents, détermine en fonction d'analyses statistiques et probabiliste. La fonction d'appariement est très étroitement liée aux opérations d'indexation et de pondération des termes de la requête et des documents du corpus. D'une façon générale, l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'information.

La fonction de similarité permet ensuite d'ordonner les documents renvoyés à l'utilisateur. La qualité de cet ordonnancement est primordiale. En effet, l'utilisateur se contente généralement d'examiner les premiers documents renvoyés (les 10 ou 20 premiers). Si les documents recherchés ne sont pas présents dans cette tranche, l'utilisateur considérera le SRI comme mauvais vis-à-vis de sa requête.

Le but de tout SRI est donc évidemment de rapprocher la pertinence système de la pertinence utilisateur (qui comme nous l'avons vu précédemment, est fortement subjective).

I.3.3. Le processus de reformulation de requête

La reformulation de requête consiste, à partir d'une requête initiale formulée par l'utilisateur, à construire une requête qui répond mieux à son besoin informationnel. Les techniques de reformulation de requête se classifient en méthodes locales et méthodes globales [Boubekeur, 08]

Les méthodes locales ajustent une requête relativement aux documents qui sont retournés comme documents pertinents pour la requête initiale. Elles se basent sur la technique dite de *réinjection de pertinence (relevance feedback)*.

Les méthodes globales se basent sur l'expansion de requête en s'appuyant sur des ressources linguistiques (thésaurus ou ontologies), ou sur des techniques d'associations de

termes telles que les relations de cooccurrences. Ces deux dernières méthodes seront détaillées dans le deuxième chapitre.

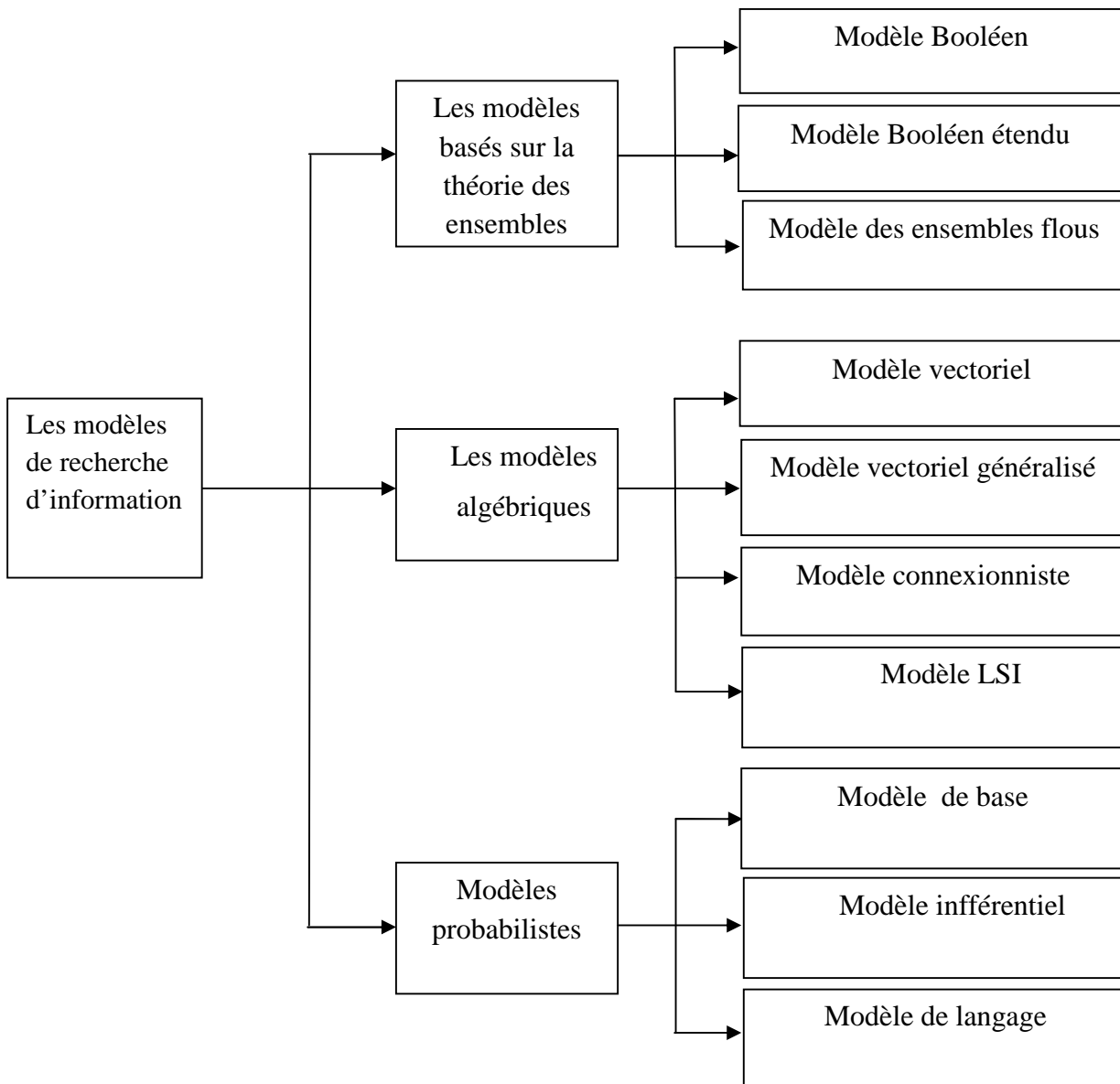
I.4. Modèles de recherche d'information

Un modèle de recherche d'information est une abstraction du processus de RI. Il fournit un cadre théorique permettant ainsi une interprétation formelle de la pertinence. La première fonction d'un système de recherche d'information est de mesurer la pertinence d'un document vis-à-vis d'une requête. Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information. Il doit accomplir deux rôles:

Créer une représentation interne pour un document ou une requête basée sur les termes de l'indexation.

Définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de similarité (mesure de pertinence).

La figure 1.4 présente une classification des principaux modèles utilisés en recherche d'information.



La figure I.4 : Taxonomie des Modèles de Recherche d'Information [Ribeiro, 99]

Les trois grandes familles ou modèles de recherche d'information sont :

I.4.1. Les modèles basés sur la théorie des ensembles

Cet ensemble de modèles se basent sur la théorie des ensembles, ainsi une requête est représentée par un ensemble de termes séparés par des opérateurs logiques (OR, AND, NOT). Le document est, quant à lui, représenté par une liste de mot-clé. Ces modèles permettent d'effectuer des opérations d'union, d'intersection et de différence lors de

l'interrogation. Le modèle le plus connu et le plus simple de cette catégorie est le modèle booléen. On y retrouve également le modèle booléen étendu et le modèle flou.

A. modèle booléen

Le modèle booléen est le plus simple des modèles de *RI*. C'est aussi le premier qui s'est imposé dans le monde de la recherche d'information. Il est basé sur la théorie des ensembles (modèle ensembliste comme le modèle booléen, sont basées sur la théorie des ensembles ce sont les plus simple et les premiers à avoir été mis en place) et *l'algèbre de Boole*. Le modèle booléen considère que les termes de l'index sont présents ou absents d'un document. En conséquence, les poids des termes dans l'index sont binaires, c'est à dire soit 0 soit 1. Une requête q est composée de termes liés par les trois connecteurs logiques ET, OU, NON. La mesure de similarité est la suivante:

$$R_{sv}(q,d) = \begin{cases} 1 & \text{si } d \text{ appartient à l'ensemble décrit par la requête} \\ 0 & \text{sinon} \end{cases}$$

Ainsi, le modèle booléen affirme que chaque document est soit pertinent soit non-pertinent. Il n'y a pas de notion de réponse partielle aux conditions de la requête. Par exemple, considérons un document contenant les trois termes *recherche*, *information* et *traditionnelle*. Ce document ne sera pas pertinent pour la requête *recherche ET information ET traditionnelle ET modèle*.

Le modèle booléen est le pionnier des systèmes de recherche d'information commerciaux. Son principal avantage est sa transparence. En effet, pour l'utilisateur, la raison pour laquelle un document a été sélectionné par le système est claire : il répond exactement à la requête qui a été formulée.

Cependant, il est parfois difficile pour l'utilisateur d'exprimer son besoin en information avec des expressions booléennes, et les expressions booléennes formulées sont généralement très simples, ce qui ne permet pas d'utiliser au mieux les caractéristiques du modèle. De plus, le fait que la pertinence soit basée sur un critère binaire sans notion d'échelle de gradualité empêche le modèle d'avoir de bonnes performances (on ne peut pas dire qu'un document est plus pertinent qu'un autre). Enfin, les résultats de la fonction de similarité (1 ou 0) ne permettent pas de fournir à l'utilisateur une liste ordonnée de résultats.

B. Le modèle booléen étendu

Ce modèle a été introduit par *Salton* cité dans, en 1983. Son apparition a été motivée par l'insuffisance majeure du modèle booléen classique, à savoir, l'inexistence du classement. L'idée générale est d'interpréter la conjonction et la disjonction en termes de distance euclidiennes. L'extension du modèle booléen classiques au modèle étendu se fait en relaxant les conditions d'adhésion aux ensembles et ceci en introduisant la notion d'appariement partiel et de pondération des termes. Ce modèle est une combinaison des caractéristiques du modèle vectoriel (à voir dans les prochaines sections) et des propriétés de l'algèbre booléenne.

Les scores de similarité document - requête sont ainsi calculés :

$$\mathbf{RSV}\left(\vec{d}, q_{ou}\right) = \sqrt{\frac{x^2+y^2}{2}} \text{ pour la requête } x \text{ ou } y$$

$$\mathbf{RSV}\left(\vec{d}, q_{et}\right) = 1 - \sqrt{\frac{(1-x)^2+(1-y)^2}{2}} \text{ pour la requête } x \text{ et } y$$

Où :

X : pour le terme x.

Y : pour le terme y.

Le modèle booléen étendu étend l'algèbre de Boole avec des distances algébriques. Il s'agit ainsi d'un modèle hybride qui inclut les propriétés des modèles ensembliste et algébrique. Le modèle booléen étendu n'a pas été beaucoup utilisé par la suite, mais il donne un cadre nouveau à la recherche d'information, cadre qui pourrait s'avérer utile dans le futur.

C. Le modèle flou

La logique floue a été proposée par Lotfi Zadeh [Zadeh, 1965] dans le milieu des années 1960. A l'inverse de la logique booléenne, la logique floue permet à une condition d'être dans un autre état que vrai ou faux. Elle peut prendre une infinité de valeurs de vérité dans un intervalle [0, 1].

En recherche d'information, une extension du modèle booléen basée sur les ensembles flous a été proposée par Salton [Salton, 1989]. Cette extension vise également à tenir compte de la pondération des termes dans les documents. Un poids d'un terme exprime le degré d'appartenance de ce terme à un ensemble. Ainsi, un document peut être représenté par un ensemble de termes pondérés comme suit : $D_j = \{(td_{1j}, a_{1j}), \dots, (td_{nj}, a_{nj})\}$

où td_{ij} est le i ème terme du document D_j , et a_{ij} est le degré d'appartenance (une valeur comprise entre 0 et 1) du i ème terme au document D_j .

L'évaluation d'une requête Q_k par rapport à un document D_j , peut prendre plusieurs formes. L'une d'elles est la suivante:

$$RSV(D_j, t_i) = a_{ij}$$

$$RSV(D_j, t_1 \wedge t_2) = \min(RSV(D_j, t_1), RSV(D_j, t_2))$$

$$RSV(D_j, t_1 \vee t_2) = \max(RSV(D_j, t_1), RSV(D_j, t_2))$$

$$RSV(D_j, \neg t_i) = 1 - a_{ij}$$

Cette évaluation par min et max est classique et a été proposée par L. Zadeh [Zadeh, 1965] dans le cadre des ensembles flous.

L'un des avantages de ce modèle est que l'on peut mesurer le degré de correspondance entre un document et une requête dans un intervalle $[0,1]$, on peut ordonner ainsi les documents dans l'ordre décroissant de leur correspondance avec la requête.

L'inconvénient principal de ce modèle est que le calcul de la valeur de similarité est toujours dominé par les petits poids des termes dans le cas des conjonctions et les grands poids des termes dans le cas des disjonctions.

I.4.2. Les modèles algébriques

Ces modèles regroupent tous les modèles de RI qui utilisent une représentation vectorielle des documents et des requêtes dans lesquels, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. La similarité est calculée algébriquement en se basant sur la représentation du document et de la requête. Le représentant le plus connu de cette catégorie est le modèle vectoriel. Plusieurs modèles

s'inspirant du modèle vectoriel ont été proposés dans le domaine de la RI : le modèle vectoriel généralisé, le modèle connexionniste et le modèle LSI (Latent Semantic Indexing).

A. Le modèle vectoriel

Le modèle vectoriel fait partie des modèles statistiques. L'utilisation des statistiques a pour but d'une part de caractériser d'un point de vue quantitatif les termes et les documents et d'autre part de mesurer le degré de pertinence d'un document vis à vis d'une requête. Le but final est d'arriver à retourner une liste ordonnée de documents selon ce degré. Un autre avantage réside dans l'expression des besoins de l'utilisateur : contrairement au modèle booléen où les termes de la requête doivent être reliés par des connecteurs logiques, l'utilisateur peut ici aussi exprimer son besoin en information en langage naturel ou sous forme d'une liste de mots clés.

Luhn a été le premier qui a proposé une approche statistique de recherche d'information à la fin des années 1950. Il suggère que l'utilisateur fournisse un document qui ressemble à son besoin en information. La mesure de similarité entre le document fourni et la représentation des documents de la collection est utilisée pour ordonner ces documents. Le critère de similarité est ainsi défini :

Plus deux représentations contiennent les mêmes éléments, plus la probabilité qu'elles représentent la même information est élevée.

Une telle définition revient en fait à compter le nombre d'éléments que partagent la requête et la représentation du document. Pour ce faire, considérons la représentation d'un document comme un vecteur $\vec{d_j} = \{\omega_{1,j}; \omega_{2,j}; \dots; \omega_{t,j}\}$

Où $\omega_{i,j}$ est le poids (0 ou 1) des termes dans le documents, t étant le nombre total de termes de l'index, et considérons la représentation de la requête comme un vecteur

$\vec{q_j} = \{\omega_{1,q}; \omega_{2,q}; \dots; \omega_{t,q}\}$ avec les mêmes notations. La mesure de similarité la plus simple est alors le produit scalaire :

$$RSV(\vec{d_j}, \vec{q}) = \sum_{i=1}^t \omega_{i,j} * \omega_{i,q}$$

Comme les poids des termes sont binaires, la mesure de similarité mesure le nombre de termes partagés entre le document et la requête.

Salton a proposé un modèle basé sur cette mesure de similarité dans son projet SMART (Salton's Magical Automatic Retriever of Text). Le document (vecteur \vec{d}) et la requête (vecteur \vec{q}) sont représentés là encore dans un espace Euclidien de dimension élevée engendré par tous les termes de l'index. La similarité est alors le cosinus de l'angle formé par les deux vecteurs:

$$RSV(\vec{d}, \vec{q}) = \frac{\vec{d} \cdot \vec{q}}{|\vec{d}| * |\vec{q}|} = \frac{\sum_{i=1}^t \omega_{i,j} * \omega_{i,q}}{\sqrt{\sum_{i=1}^t \omega_{i,j}^2} * \sqrt{\sum_{i=1}^t \omega_{i,q}^2}}$$

D'autres fonctions de similarité ont été proposées dans la littérature, parmi lesquelles on peut citer les mesures de Jaccard et Dice.

Le tableau suivant présente les principales mesures de similarité utilisées sont :

	La fonction de similarité
Produit scalaire	$RSV(Q, D_j) = \sum_{i=1}^N q_i \cdot d_{ij}$
Mesure de Jaccard	$RSV(Q, D_j) = \frac{\sum_{i=1}^N q_i \cdot d_{ij}}{\sum_{i=1}^N q_i^2 + \sum_{i=1}^N q_{ij}^2 - \sum_{i=1}^N q_i \cdot d_{ij}}$
La mesure cosinus	$RSV(Q, D_j) = \frac{\sum_{i=1}^N q_i \cdot d_{ij}}{[\sum_{i=1}^N q_i^2]^{1/2} [\sum_{i=1}^N d_{ij}^2]^{1/2}}$

Tableau I.3 : les principales mesures de similarité utilisées.

Les documents sont ainsi classés en fonction de la mesure de l'angle qu'ils forment avec le vecteur requête. Si un terme est présent à la fois dans la requête et le document, il contribue au score. S'il est présent uniquement dans l'un des deux, il diminue le score parce que la requête et le document se correspondent moins.

Les avantages d'un tel modèle sont nombreux : la pondération des termes augmente les performances du système, le modèle permet de renvoyer des documents qui répondent

approximativement à la requête, et la fonction d'appariement permet de trier les documents selon leur degré de similarité avec la requête.

De nombreuses méthodes d'ordonnement des résultats ont été comparées au modèle vectoriel, et celui-ci, malgré sa simplicité, est supérieur ou au moins aussi bon que les autres alternatives.

B. Le modèle vectoriel généralisé

En 1985, Wong, Ziarko et Wong [Wong et al., 1985] proposent une interprétation dans laquelle les vecteurs des termes de l'index sont linéairement indépendants mais non orthogonaux deux à deux. Cette interprétation est appelée le modèle vectoriel généralisé. D'une manière générale, la contribution principale du modèle est l'établissement d'un cadre formel dans lequel les dépendances entre les termes de l'index peuvent être facilement représentées.

Cependant, il est loin d'être prouvé que l'introduction de dépendances entre termes dans un modèle permette d'augmenter son efficacité. De plus, le modèle vectoriel généralisé est plus compliqué et plus lent que le modèle vectoriel classique.

c. Le modèle connexionniste

Ce type de modèle se base sur les fondements des réseaux de neurones biologiques, tant pour la modélisation des documents et des informations descriptives associées (termes, auteurs, mots clés, etc.), que pour la mise en oeuvre du processus de recherche d'information.

Ce modèle peut être vu comme un modèle vectoriel récurrent non linéaire, les neurones formels représentent des objets de la recherche d'information. Un réseau de neurone formel est construit à partir des représentations initiales des documents et de la requête. Le mécanisme de recherche d'information est fondé sur le principe de propagation de valeurs depuis les neurones descriptifs de la requête vers ceux des documents, à travers les connexions du réseau. Les résultats sont présentés à l'utilisateur selon le niveau d'activation des neurones documents. Le modèle connexionniste est connu pour sa capacité d'apprentissage, ce qui permet aux SRI de devenir adaptatifs.

Le fonctionnement du réseau se fait par propagation de signaux de la couche d'entrée vers la couche de sortie. Chaque neurone de la couche d'entrée reçoit une valeur d'activation, calcule une valeur de sortie et la transmet vers les neurones qui lui sont reliés dans la couche suivante. Ce processus se reproduit jusqu'à arriver à la couche de sortie, les valeurs de sortie dans la couche de sortie servant de critères de décision.

Concrètement, trois types de nœuds sont utilisés pour représenter les requêtes, les termes et les documents. Des liens requête-terme et termes-documents sont définis.

Le lien requête-terme indique l'apparition du terme dans la requête. Un poids est associé à ce lien et qui traduit la formule $tf*idf$ de ce terme.

De même pour les liens document-terme et qui définit l'existence d'un terme dans un document.

Un nœud est activé dès que sa valeur en sortie dépasse un certain seuil. Pour commencer, un nœud requête est activé. Grâce aux liens requête-terme, les nœuds termes sont retrouvés et le poids est calculé. Ensuite, les nœuds terme sont activés et les nœuds document sont identifiés par l'intermédiaire des liens document-terme. La valeur de chaque nœud document concerné par la requête est ainsi calculée : $D_j = \sum_i w_{ij}$

Avec w_{ij} est le poids du terme t_i dans le document D_j

Chaque nœud document retrouvé a ainsi une valeur poids qui mesure la pertinence du document dans une requête donnée.

Actuellement, plusieurs modèles basés sur le principe des réseaux de neurones sont utilisés en recherche d'information. Les modèles de RI basés sur les réseaux de neurones sont une solution pour combler les lacunes des modèles vectoriels. La notion de réseau est convenable pour représenter les relations et associations qui existent entre les termes (exemples: synonymie, voisinage, etc.), entre les documents (exemples: similitude, référence, etc.), et enfin entre les termes et les documents (exemples: fréquence, poids, etc.). Cependant, il n'existe pas de représentation unique d'un réseau de neurones pour la recherche d'information, c'est au constructeur du modèle de le définir, et ce en identifiant les éléments suivants :

Les différentes couches du réseau (couche d'entrée, de sortie, intermédiaires, etc.)

Les neurones de chaque couche,

La fonction d'entrée de chaque neurone,

La fonction de sortie de chaque neurone,

Les liens entre les neurones et leurs poids associés.

D. Le modèle LSI (Latent Semantic Indexing)

Dans ce modèle, les documents sont représentés dans un espace de dimension réduite issu de l'espace initial des termes d'indexation. Le but de ce modèle est d'aboutir à une représentation conceptuelle des documents où les effets dus à la variation d'usage des termes dans la collection sont nettement atténués. Ainsi, les documents qui partagent des termes co-occurents ont des représentations proches dans l'espace défini par le modèle. Par conséquent, le système permet de sélectionner des documents même s'ils ne contiennent aucun terme de la requête. Ce modèle se base essentiellement sur la décomposition en valeurs singulières, désignée par SVD (Singular Value Decomposition) de la matrice représentant, en lignes les termes et en colonnes les documents. Un élément de la matrice représente le poids d'un terme dans un document. La SVD permet d'une part de réduire l'espace des termes d'indexation, et d'autre part, de représenter les documents et les requêtes dans un espace qui ne dépend pas des termes d'indexation mais des concepts contenus dans les documents.

Formellement, la transformation par LSI est présentée comme suit :

- initialement, les documents sont représentés par des vecteurs de termes,
- l'ensemble des vecteurs des documents de la collection sont représentés sous forme de matrice X de dimension $d \times t$, où t est le nombre de termes distincts de la collection et d le nombre de documents dans la collection,
- la SVD permet de décomposer toute matrice rectangulaire en un produit de trois matrices. Ainsi, la matrice X est transformée par SVD comme suit :

$$X = T_0 \cdot S_0 \cdot D_0^t$$

avec :

T_0 : une matrice orthogonale de dimension $t \times m$;

t_0D : une matrice orthogonale de dimension $m \times d$;

S_0 : une matrice diagonale de dimension $m \times m$, les valeurs sur la diagonale sont les valeurs propres de X et sont par convention toutes positives et ordonnées par ordre décroissant sur la diagonale ;

m : est le rang de la matrice X ($\leq \min(t, d)$).

L'intérêt de la SVD est qu'elle permet d'utiliser une stratégie plus simple pour l'optimisation de la matrice X , en se basant sur des matrices réduites. Ainsi, les k plus grandes valeurs propres sont supposées suffisantes pour représenter presque toute l'information de la matrice X . Concrètement, toutes les valeurs propres i ($i > k$) sont supposées nulles, et l'équation est réécrite en utilisant la matrice S de dimension $k \times k$, approximation de S_0 réduite aux k premières dimensions. Le résultat de la nouvelle transformation est donné par le modèle réduit suivant :

$$X \approx X' = T.S.D^t$$

avec :

T : une matrice orthogonale réduite de dimension $t \times k$ ($k \leq m$)

D_t : une matrice orthogonale réduite de dimension $k \times d$

S : une matrice diagonale réduite de dimension $k \times k$

m : est le rang de la matrice X ($\leq \min(t, d)$).

- une requête X_q est, comme tout document, un ensemble de termes. Elle peut être représentée dans le nouvel espace des documents comme suit :

$$Q = X_q.T.S^{-1}$$

Où Q est le vecteur des mots de la requête pondéré par les termes appropriés, S^{-1} est la matrice inverse de S .

- Une valeur de similarité est ensuite calculée entre le vecteur requête Q et chaque document de la collection, tous deux représentés dans le nouvel espace vectoriel.

Plusieurs applications de ce modèle ont été proposées en recherche d'information, mais également pour le filtrage d'information et la recherche documentaire multilingue.

I.4.3. Les modèles probabilistes [Tamine, 00]

Ces modèles se basent sur la théorie des probabilités. La pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête. On distingue le modèle BIR (Binary Independence Retrieval), le modèle inférentiel bayésien et le modèle de langage.

A. Le modèle de base

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

$P(w_{ji} / Pert)$: Probabilité que le terme t_i occure dans le document D_j sachant que ce dernier est pertinent pour la requête

$P(w_{ji} / NonPert)$: Probabilité que le terme t_i de poids d_{ji} occure dans le document D_j sachant que ce dernier n'est pas pertinent pour la requête

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

En posant les hypothèses que :

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

$$P(Pert/D_j) = \log \frac{P(W_{ji}/Pert)}{P(W_{ji}/NonPert)}$$

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

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

Croft & Harper [Croft & Harper, 1979] intègrent au modèle les mesures de fréquence plutôt, que de considérer seulement la présence ou l'absence des termes. La similitude requête document est calculée comme suit :

$$RSV(Q_k, D_j) = C \sum_{t=1}^T q_{kt} d_{jt} + \sum_{t=1}^T f_{jt} q_{jt} d_{jt} \log \frac{N-n_t}{n_t}$$

Où :

$$F_{jt} = \frac{tf_{jt}}{\max_t f_j}$$

C : Constante.

B. Le modèle de réseau inférentiel bayésien

Un réseau bayésien est un graphe direct acyclique où les nœuds représentent des variables aléatoires et les arcs des relations causales entre nœuds. Ces derniers sont pondérés par des valeurs de probabilités conditionnelles.

Le travail original en recherche d'information, et basé sur le modèle des réseaux bayésiens, est développé par Turtle [Turtle & Croft, 1991].

Dans l'espace défini par les termes d'indexation, on définit :

T variables aléatoires binaires t_1, \dots, t_T associés aux termes d'indexation

D_j : Variable aléatoire associée à un document

Q_k : Variable aléatoire associée à une requête

On calcule alors la mesure de pertinence de Q_k relativement à D_j en traitant les probabilités conditionnelles de Bayes selon la formule :

$$RSV (Q_k, D_j) = 1 - P(Q_k, D_j)$$

Où

$$P(Q_k, D_j) = \sum_{t=1}^T P(Q_k/t_i) * (\prod_{t_k \in D_j} P(t_k/D_j) * \prod_{t_k \notin D_j} P(\bar{t}_k/D_j)) * P(D_j)$$

Avec :

$P(Q_k/t_i)$: Probabilité que le terme t_i appartienne à un document pertinent de Q_k

$P(t_i/D_j)$: Probabilité que le terme t_i appartienne au document D_j sachant qu'il est pertinent

$$P(\bar{t}_i/D_j) = 1 - P(t_i/D_j)$$

$P(D_j)$: Probabilité d'observer D_j

Les probabilités conditionnelles de chaque noeud sont calculées par propagation des liens de corrélation entre eux.

Le modèle présente l'intérêt de considérer la dépendance entre termes mais engendre une complexité de calcul importante.

D. Le modèle de langage

Dans les modèles de recherche probabilistes "classiques", on cherche à estimer la probabilité que le document réponde à la requête [pont et croft, 98]. L'hypothèse de base dans ces modèles est qu'un document n'est pertinent que s'il ressemble à la requête. Les modèles de langage sont basés sur une hypothèse différente : un utilisateur en interaction avec un système de recherche fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver. La requête est alors inférée par l'utilisateur à partir de ces documents. Un document n'est pertinent que si la requête utilisateur ressemble à celle inférée par le document. On cherche alors à estimer la probabilité que la requête soit inférée par le document. Les modèles de langages calculent cette probabilité et l'utilisent pour ordonner les documents. Etant donné une requête T_1, T_2, \dots, T_n , les documents sont ordonnés selon la mesure suivante :

$$P(T_1, T_2, \dots, T_n | D) = \prod_{i=1}^n ((1 - \lambda_i) P(T_i) + \lambda_i P(T_i | D))$$

Cette mesure est une combinaison linéaire du modèle de document et du modèle de contexte du document (la collection), où : λ_i est la probabilité que le terme à la position i soit important, $1 - \lambda_i$ est la probabilité que le terme ne soit pas important, $P(T_i|D)$ est la probabilité d'un terme important et $P(T_i)$ est la probabilité d'un terme sans importance. Les probabilités sont définies de la manière suivante :

$$P(T_i|D) = \frac{tf(T_i|D)}{\sum_T tf(T,D)}, \text{ terme important}$$

$$P(T_i) = \frac{df(T_i)}{\sum_T df(T)}, \text{ terme sans importance}$$

Où $tf(T_i|D)$ est la fréquence du terme T_i dans le document D et $df(T)$ est le nombre de documents dans lesquels le terme T apparaît. Ces deux probabilités sont estimées en utilisant une estimation de vraisemblance et λ est appelé paramètre de lissage. Le calcul des probabilités peut être réduit à la formule de calcul de scores suivante :

$$S(D, T_1, T_2, \dots, T_n) = \beta \cdot \log(\sum_T tf(T, D)) + \sum_{i=1}^n \log\left(1 + \frac{\lambda \cdot tf(T_i, D) \cdot (\sum_T df(T))}{(1-\lambda) \cdot df(T_i) \cdot (\sum_T tf(T, D))}\right)$$

Le paramètre β sert à estimer des probabilités a priori et est utilisé pour introduire la longueur des documents dans la formule de calcul des scores, c'est à dire pour normaliser ces scores. Une question se pose cependant : comment estimer la valeur de λ_i ? Pour une première recherche, on a :

$\lambda_i = \text{constante}$, c'est à dire que tous les termes sont considérés comme ayant la même importance. λ_i est ensuite réévalué pour chaque terme dans un cycle de réinjection de la pertinence.

I.5. Evaluation des Systèmes de RI

L'évaluation des systèmes de recherche d'information constitue une étape importante dans l'élaboration d'un modèle de recherche d'information. En effet, elle permet de caractériser le modèle et de fournir des éléments de comparaison entre modèles.

Les suggestions de mesures et les techniques d'évaluation des systèmes de recherche d'information, se sont multipliées depuis une vingtaine d'années, dans la lignée des projets de la **DARPA** (*Defense Advanced Research Projects Agency*), dont le plus connu est sans

doute le programme **TREC** (*Text REtrieval Conference*). Il offre une plate forme d'évaluation à grande échelle pour les techniques de RI. Il offre aussi une très large collection de documents de sources très variées.

D'une façon générale, un système de recherche d'information idéal a deux objectifs :

- Retrouver tous les documents pertinents.
- Rejeter tous les documents non pertinents.

La collection TREC [Linda tamine,00]

Les collections de test ont été traditionnellement utilisées en recherche d'information pour évaluer les stratégies de recherche. Cependant, en vue d'être une base de travail fiable, une collection de test doit constituer une référence sûre de comparaison entre stratégies, en accord avec leur efficacité. Plus particulièrement, une collection de référence doit traduire la subjectivité de pertinence des utilisateurs d'une part, et contenir, d'autre part, une masse d'informations assez importante et variée pour constituer un environnement standard d'interrogation.

Dans ce cadre, une série de conférences annuelles TREC a été lancée en 1990 dans le but de conjuguer les efforts de la communauté en recherche d'information et uniformiser les outils d'évaluation.

TREC offre une très large collection de documents organisée en sous collections, qui évoluent d'année en année. Un document TREC est généralement présenté sous le format SGML, identifié par un numéro et décrit par un auteur, une date de production et un contenu textuel.

Une requête TREC est également identifiée par un numéro et décrite par un sujet générique, une description brève et une description étendue sur les caractéristiques des documents pertinents associés à la requête.

Les campagnes d'évaluation [Baziz, 05]

Des campagnes d'évaluation destinées à stimuler et favoriser l'émergence de nouveaux SRI ont été menées, très régulièrement depuis plusieurs années.

La première campagne TREC (TREC1) voit le jour en 1992 avec 25 participants issus du monde académique et industriel. Un certain nombre de tâches existent dans TREC et sont dédiées à certains aspects de la recherche tels que la génomique, le web, le filtrage, les blogs, les questions réponses, etc. La 17^{ème} édition de TREC est TREC 2008 et elle comprend 5 tâches dont la tâche blog track qui s'intéresse à l'étude des informations contenues dans la "blogosphère", la tâche entreprise track qui s'intéresse à l'information manipulée dans les entreprises, etc.

On citera quelques compagnies parmi les plus importantes le programme américain TREC5 (Text Retrieval Evaluation Conference) qui en est à sa 14^{ème} campagne (entamé en 1992) et le programme CLEF (Cross Language Evaluation Forum) qui en est à sa sixième (depuis 2000).

CLEF6 était à l'origine, la tâche multilingue de TREC avant de devenir depuis 2000, un programme à part entière supportant la majorité des langues européennes. De façon analogue, le programme NTCIR7 est conçu pour les langues asiatiques. Amaryllis quand à lui, était un programme Français qui a commencé en 1995, et qui a fusionné depuis 2002 avec le programme CLEF.

Ces programmes fournissent l'infrastructure nécessaire pour monter un banc d'essai "benchmark" permettant une procédure d'évaluation uniforme des différentes techniques et stratégies de recherche que les différents laboratoires et équipes de recherche expérimentent. TREC par exemple fournit une plate-forme comportant des collections de tests, des tâches spécifiques, des questions-type (topics) et les jugements de pertinence (réponses idéales) correspondants à chaque tâche. Comme exemple de tâche, on peut citer la tâche "**Robust**" (pour évaluer les requêtes difficiles) et la tâche "**Genomic**" pour le domaine spécifique de la génomique.

Les principaux programmes mentionnés sont tous inspirés d'un même protocole qui est celui de TREC.

L'objectif de TREC est d'encourager les travaux de recherche en informatique documentaire permettant l'accès à des bases volumineuses en leur fournissant [Tebri, 06]:

- une collection de test importante, constituée d'un ensemble de documents et de requêtes
- la liste de documents pertinents pour chaque requête ;
- des procédures d'évaluation des résultats de recherche.

Dans les collections de TREC, un ensemble de jugements de pertinence est associé aux requêtes TREC. Les jugements de pertinence correspondent pour chaque topic à l'ensemble des documents qui sont jugés pertinents par des juges humains. Les jugements de pertinence permettent de comparer les résultats réels des systèmes aux résultats théoriques établis par les juges.

Différentes tâches sont proposées chaque année dans le cadre du programme d'évaluation de TREC : tâche multilingue (Cross language track), tâche de filtrage (adaptatif, différé et routage), tâche nouveauté (Novelty track), etc. Ces tâches évoluent d'une année à une autre, et elles tiennent compte des résultats obtenus, de l'évolution des attentes des utilisateurs réels et des collections de documents disponibles [Nongdo, 08].

a) La tâche détection de la nouveauté

La tâche détection de la nouveauté a été introduite en 2002 lors de la campagne

TREC11. Étant donné une requête et une liste ordonnée de documents pertinents, l'objectif de cette tâche est de retrouver les passages (phrases) pertinents et nouveaux répondant à la requête. Nous avons utilisé dans nos travaux les documents issus de la première sous-tâche qui consiste à détecter les passages pertinents dans les documents.

La collection de départ est constituée de 50 requêtes provenant des campagnes TREC6,

TREC7, et TREC8. Ces 50 requêtes ont été sélectionnées parmi celles pour lesquelles les jugements de pertinence comprenaient entre 10 et 70 documents pertinents. En 2002,

TREC a choisi de sélectionner 50 requêtes issues des besoins d'information identifiés par les numéros 300 à 450. Le NIST (National Institute of Standards and Technology) a sélectionné les documents effectivement pertinents pour chacun de ces besoins

d'information (jugements de pertinence), avec un maximum de 25 documents par besoin, et les a fournis aux participants.

b) La tâche adhoc

De TREC1 à TREC6, les recherches sont centrées sur deux tâches principales : la tâche de routage et la tâche adhoc. Dans la première tâche, un flot de documents doit être filtré au fur et à mesure pour répondre à un ensemble de profils fixé. À l'inverse de la tâche de routage, la tâche adhoc suppose que les collections de documents sont fixes. Les collections de documents issues des campagnes adhoc de TREC3, TREC5, TREC6, et TREC7. Les sont très variés et proviennent de différentes sources. Selon [BYRN,99], la taille moyenne des documents (pour la majorité d'entre eux) est d'environ 300 mots. La taille de ces collections est variable d'une année à l'autre (plus de 3Gb dans TREC3 par exemple).

Chaque année de TREC comporte un jeu de 50 requêtes qui est soumis aux participants.

Les requêtes de la tâche détection de la nouveauté correspondent à un sous-ensemble des requêtes adhoc de TREC (T, D, et N)

Les mesures principales de TREC (d'évaluation des SRI) sont : le rappel, la précision, la MAP (Mean average Precision), la F-mesure, et les mesures de haute précision (P@5,

P@10, P@15).

I.5.1. Mesures d'évaluation des systèmes

Une mesure d'évaluation appropriée doit estimer la faculté des systèmes à trouver des documents pertinents, sans pour autant favoriser ceux qui retournent plus de documents (et donc potentiellement plus de documents pertinents). Il est donc nécessaire de considérer aussi les documents non pertinents dans une métrique.

Les mesures combinées de rappel et de précision sont les mesures les plus simples et les plus utilisées qui tiennent compte de ces deux impératifs (retrouver des documents pertinents, ne pas retrouver de documents non pertinents).

Soient, pour une requête donnée et un système donné (voir figure 3) :

- P l'ensemble des documents de la collection qui sont pertinents par rapport à cette requête.
- R l'ensemble des documents retrouvés par le système pour cette requête.
- $PR = P \cap R$, l'ensemble des documents pertinents retrouvés par le système.

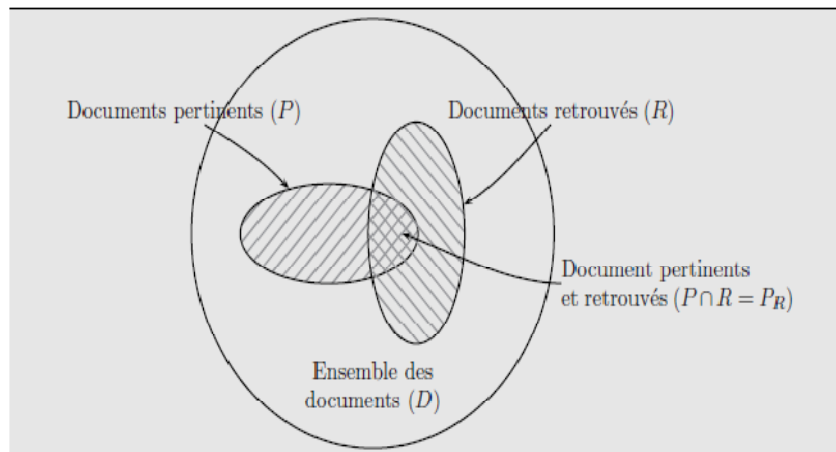


Figure I.5 : Répartition des documents face à une requête.

A. Rappel Précision :

La précision : La précision mesure le taux de documents pertinents parmi tous les documents retrouvés par le système :

$$\textbf{Précision} = \frac{|P_R|}{|R|}$$

Le rappel : Le rappel mesure le taux de documents pertinents retrouvés par le système parmi l'ensemble des documents pertinents de la collection :

$$\textbf{Rappel} = \frac{|P_R|}{|P|}$$

Ces deux valeurs sont indissociables et concurrentes, puisque souvent l'amélioration de l'une d'elle se fait au détriment de l'autre. Une caractéristique intéressante est d'ailleurs l'évaluation de la précision en fonction du rappel. Ces informations sont regroupées dans la courbe de rappel-précision.

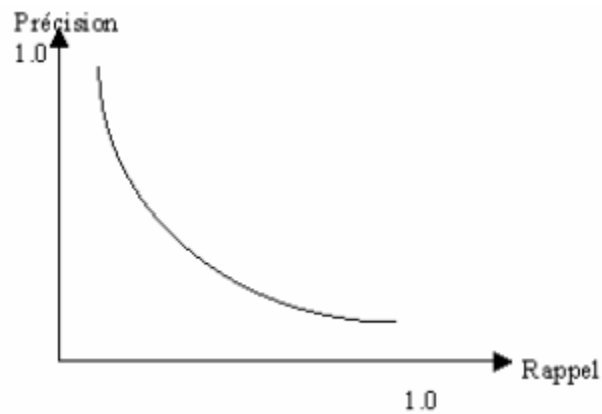


Figure I.6 : Courbe de Rappel/Précision [Fellag,06]

Universellement utilisés, précision et rappel posent cependant quelques problèmes.

Parmi eux :

- La mesure du rappel nécessite de connaître en théorie l'ensemble des documents pertinents par rapport à une requête dans toute la collection.
- Le rappel n'est pas une mesure qui reflète la satisfaction de l'utilisateur moyen.

Celui-ci est rarement intéressé par la connaissance de l'ensemble des documents pertinents, mais souhaite plutôt trouver “vite” (donc bien classés) quelques documents pertinents.

Pour pallier ces inconvénients, de nombreuses métriques héritées de la précision et du rappel ont été proposées.

B. La moyenne des précisions moyennes (MAP) [Nongdo,08]

Cette mesure calcule la moyenne des valeurs de précision moyenne non interpolées sur l'ensemble des documents pertinents. Le calcul de la précision moyenne (pour une requête Q donnée) se fait en déterminant à chaque document pertinent retrouvé la précision correspondante. Lorsqu'un document non pertinent est retrouvé, sa précision est égale à 0. La précision moyenne établit donc la moyenne de toutes ces mesures afin de déterminer la performance du système pour la requête Q . La formule suivante donne la méthode de calcul de la MAP :

$$\text{MAP} = \frac{1}{n} \sum_i q_i \frac{1}{|R_i|} \sum_{j \in R_i} \frac{j}{r_{ij}}$$

Où r_{ij} correspond au rang du $j^{\text{ième}}$ document pertinent pour la requête Q_i . $|R_i|$ indique le nombre de documents pertinents pour la requête Q_i , et

n correspond au nombre de requêtes de test

C. Mesures de haute précision : P@X

Les mesures de haute précision permettent de ne pas évaluer l'ensemble des documents contenus dans la liste restituée par un SRI. En effet, ces mesures permettent d'examiner les listes à différents niveaux de coupe. On considère alors les X premiers documents, et la précision est calculée en fonction de la valeur de X. L'idée est qu'un système qui retourne en tête de liste un grand nombre de documents pertinents obtient une P@X supérieure à un autre système pour lequel les documents pertinents sont dispersés dans la liste restituée. Les valeurs de X peuvent être fixées à 5, 10, 15, 30, ou 100 documents par exemple. Nous parlons ici de mesures de haute précision car on évalue la capacité du système à retrouver un maximum de documents pertinents pour un faible nombre de documents retournés. Cette situation est très courante dans le cas où les utilisateurs recherchent dans les premiers documents l'information pertinente.

Si la valeur de X est plus grande que le nombre total de documents retrouvés, les documents manquants sont considérés non pertinents. La valeur de haute précision correspondante est alors diminuée. Par exemple, un système qui restitue 2 documents tous pertinents aura une valeur de P@5 égale à 2/5, même si seulement 2 documents sont retrouvés. Dans nos travaux, nous nous intéressons à la haute précision lorsque 5, 10, ou 15 documents sont retournés.

D. La R-Précision

Elle mesure la précision (ou le rappel) après que R documents aient été restitués pour la requête. R représente le nombre total de documents pertinents pour la requête.

Cette mesure compense les limites des mesures de haute précision quand la précision est calculée pour x documents et que le nombre total de documents |P| est inférieur à

x. Si la valeur de R est plus grande que le nombre total de documents retrouvés, tous les documents non retrouvés sont alors considérés non pertinents.

E. La F-mesure

D'autres mesures combinant le rappel et la précision sont utilisées en RI pour évaluer les performances des systèmes. Nous présentons ici la F-mesure. Définit la mesure harmonique qui combine les mesures de rappel et de précision selon la formule suivante

$$\mathbf{F\text{-}mesure} = \frac{2}{\frac{1}{Rappel} + \frac{1}{Précision}}$$

$$\mathbf{F\text{-}mesure} = \frac{2 * Rappel * Précision}{Rappel + Précision}$$

Maximiser la F-mesure revient à trouver le meilleur compromis entre le rappel et la précision. La valeur maximale de la F-mesure est 1 et est obtenue quand tous les documents classés sont pertinents, et quand tous les documents pertinents ont été classés.

Les campagnes d'évaluation en RI permettent d'évaluer sur des collections différentes plusieurs SRI, afin de valider les différents modèles mis en œuvre, et comparer les systèmes.

Conclusion

Ce chapitre a porté essentiellement sur l'étude des systèmes de recherche d'information de manière générale et les modèles de recherche et de représentation d'information de manière particulière. Chacun de ces modèles ou stratégies contribue à la résolution des problèmes inhérents à la recherche d'information.

Alors nous avons étudié les différents aspects liés à la RI en général. Nous avons passé en revue les méthodes et les modèles fondamentaux utilisés en RI classique. Les modèles décrits dans ce chapitre fonctionnent sur tous les formats de documents textuels (balisés/non balisés, structurés/non structurés).

Dans le chapitre suivant, nous présentons le principe de la reformulation de requête et ses différentes techniques existantes ainsi que ses paramètres de performances.

Introduction

Souvent l'utilisateur est incapable de formuler son besoin en information d'une manière exacte. Par conséquent, parmi les documents qui lui sont retournés par le SRI, certains l'intéressent moins que d'autres. Compte tenu des volumes croissants des bases d'information, retrouver les informations pertinentes en utilisant seulement la requête initiale de l'utilisateur est une tâche difficile. De nombreux travaux visent à concevoir des SRI capables de répondre aux besoins de l'utilisateur, et pour y arriver une étape supplémentaire est utilisée à savoir la reformulation de la requête.

La reformulation de requêtes, est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur.

Dans ce chapitre on va définir la reformulation de la requête, puis on présente les différentes techniques de la reformulation de requête ainsi que ses paramètres de performances. Parmi ces paramètres la qualité des documents utilisés pour la sélection de termes. Notre travail s'inscrit dans ce contexte.

II.1. Reformulation de requêtes [Bazziz, 05]

La reformulation de la requête consiste à modifier la requête de l'utilisateur par ajout de termes significatifs et/ou réestimation de leurs poids. Si les termes rajoutés proviennent des documents de la collection, on parle de réinjection de pertinence (relevance feedback), si le *processus est supervisé*, et de pseudo réinjection de pertinence si le *processus est automatique*. La modification de la requête peut aussi être basée sur le vocabulaire issu de ressources externes telles que les Ontologies ou les Thesaurus. On parle alors de reformulation de requêtes basée sur les concepts (Concept-based Query Reformulation).

La figure II.1 représente le processus général de la reformulation de requête.

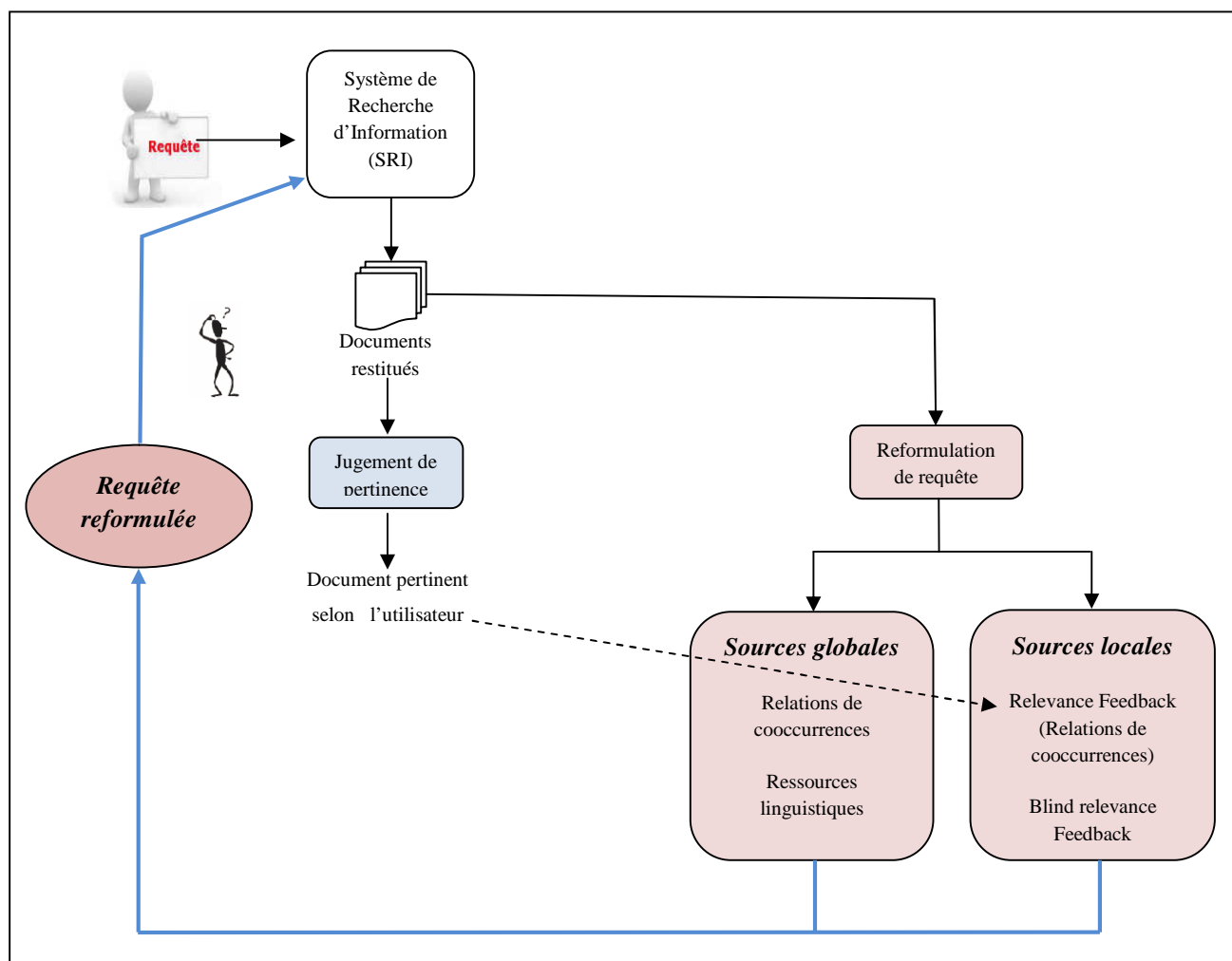


Figure II.1 : La reformulation de requête dans SRI.

Ce schéma illustre la procédure de reformulation de requête qui peut se faire par l'approche locale ou par l'approche globale.

II.2. Les techniques de reformulations de requête

Dans les SRI, la requête initiale seule est souvent insuffisante pour permettre la sélection de documents répondant au besoin de l'utilisateur. De ce fait, plusieurs techniques ont été proposées pour améliorer les performances des SRI. Ces méthodes apportent des solutions aux deux principales questions :

1. Comment peut-on retrouver plus de documents pertinents vis-à-vis d'une requête donnée?
2. Comment peut-on mieux exprimer la requête de l'utilisateur de manière à mieux répondre à son besoin?

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

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

❖ **Approches basées sur le contexte local/global [Tebri, 04]**

Les approches basées sur le contexte local/global consistent à construire automatiquement les liens entre les termes en se basant sur la co-occurrence de ces termes. Ces liens peuvent être construits à partir des documents sélectionnés par le système (contexte local) ou à partir de la collection entière de documents (contexte global). Dans ce qui suit, nous présentons ces deux contextes de reformulations de requête

II.2.1. Analyse par le contexte global

A. Utilisation de ressources linguistiques

L'expansion directe de la requête consiste à rajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Il existe une grande variété de ressources externes. Certains types de ressources sont populaires, nous reprenons leur définition issue du grand dictionnaire dans le domaine des technologies de l'information :

- Vocabulaire contrôlé : « Dans un domaine préalablement défini (d'ordre scientifique, technique, professionnel ou autre, et en général pour une langue donnée), le choix de termes sélectionnés, classés et indexés en vue de faciliter l'indexation, le stockage et la recherche des publications traitant des concepts apparentés à ces termes. »
- Taxonomie : « Construction d'un plan de classification de concepts utilisant des classes disjointes de concepts agrégés. »
- Ontologie : « Ensemble d'informations dans lequel sont définis les concepts utilisés dans un langage donné et qui décrit les relations logiques qu'ils entretiennent entre eux ».

– Un thésaurus : « Vocabulaire contrôlé et dynamique de termes ayant entre eux des relations sémantiques et génériques, et qui s'applique à un domaine particulier de la connaissance ». Ainsi, il permet d'améliorer le système au niveau de l'indexation et de l'interrogation, en précisant le contexte de la recherche, ce qui peut lever un certain degré d'ambiguïté, et en étendant la requête avec des termes considérés comme similaires. Les types de relations traditionnellement définies dans les thésaurus sont :

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

Ces relations seront détaillées dans ce qui suit.

On peut citer WordNet comme exemple de thésaurus hiérarchique. C'est une base de données lexicale pour l'anglais, basée sur des principes linguistiques. Dans cette hiérarchie, les éléments sont en fait des cliques de termes synonymes appelés *synset*.

WordNet est un réseau sémantique organisé autour de la notion de synset. Un synset regroupe des termes (simple ou composés) ayant un même sens dans un contexte donné. Les Synsets sont liés par des relations telles que spécifique-générique ou hyponyme-hyperonyme (is-a), et la relation de composition meronymie-holonymie. [Voorhees, 93]

✓ Relation **Synonymie** : désigne les termes ayant un sens équivalent ou proche et est notée RT pour Related Term. Les synonymes étant associés à la classe **Concept**.

✓ Relation **Hyperonymie**: désigne les termes ayant un sens plus large. Elle est parfois notée BT pour Broader Term, c'est le terme utilisé pour désigner une classe englobant des instances de classes plus spécifiques. Y est un *hyperonyme* de X si X est un type de (kind of) Y.

✓ Relation **Hyponymie**: désigne les termes ayant un sens plus spécifique utilisé pour désigner un membre d'une classe, et est donc la relation symétrique de la précédente. C'est la relation is a. Elle est notée NT pour Narrower Term. X est un *hyponyme* de Y si X est un type de (kind of) Y.

✓ Relation **Holonymie**: Le nom de la classe globale dont les noms *meronymes* font partie. Y est un *holonyme* de X si X est une partie de (is a part of) Y.

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

B. Extension de requêtes par relation de co-occurrences (statistique) [Moreau &Claveau, 06]

Contrairement aux techniques locales comme le LCA (Local Context Analysis) qui ne considère que des cooccurrences locales au document considéré, cette approche permet de prendre en compte les cooccurrences de mots dans le corpus de document tout entier. La création des groupements de mots est basée sur l'hypothèse que *deux mots cooccurrents dans le même contexte sont sémantiquement similaires*. Cette hypothèse s'interprète de la manière suivante : Les mots qui sont souvent utilisés ensemble dans un contexte (i.e. paragraphe) ont une forte probabilité de synonymie : pour décrire un phénomène on utilise souvent, dans un contexte local des synonymes relatifs au phénomène.

II.2.2. Analyse par le contexte local

La méthode d'analyse du contexte local (Local Context Analysis (LCA)) a été développée par Croft et Xu [Croft and Xu, 1995] et utilisée dans leur système de recherche Inquiry. A la différence avec les autres techniques de reformulation, elle utilise la règle de passage.

Elle consiste à modifier la requête initiale de l'utilisateur à partir des proportions des contenus des meilleurs documents retrouvés.

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

- ✓ Sélectionner n passages des n meilleurs documents retrouvés (un passage est une classe de termes de taille fixe, par exemple 300 termes).
- ✓ Extraire des concepts (groupes de mots) à partir des ces passages. Ces concepts sont ensuite ordonnés selon l'équation ci-dessous.
- ✓ Les 70 meilleurs termes des concepts ordonnés sont utilisés dans l'expansion de la requête initiale.

$$bel(Q, c) = \prod_{t_i \in Q} (\delta + \log(af_{c, t_i} idf_c / \log(n))^{idf_i})$$

Où :

$$af_{c, t_i} = \sum_{j=1}^{j=n} ft_{ij} fc_j$$

$$idf_i = \max(1.0, \log 10(N/N_i)/5.0)$$

$$idf_c = \max(1.0, \log 10(N/N_c)/5.0)$$

c : est un concept

ft_{ij} : est le nombre d'occurrences du terme t_i dans le passage p_j ,

fc_j : est le nombre d'occurrences du concept c dans le passage p_j ,

N : est le nombre de passages dans la collection,

N_i : est le nombre de passages contenant le terme t_i,

N_c : est le nombre de passages contenant le concept c,

δ : est une constante (égale à 0.1) qui permet d'éviter des valeurs nulles.

A. La technique de réinjection de pertinence

La *réinjection de pertinence* (*relevance feedback*) « *RF* » est l'une des techniques de modification de requêtes les plus utiles dans le domaine de la recherche d'information.

Cette méthode est mise en pratique quand l'utilisateur doit améliorer la requête qu'il a formulée au système de recherche d'information parce que les documents trouvés à l'étape initiale de la recherche ne répondent pas de manière pertinente aux besoins en information de l'utilisateur.

La technique fonctionne comme suit :

- L'utilisateur soumet une requête au SRI, qui produit une liste ordonnée de documents selon leurs degrés correspondants de pertinence à la requête.
- L'utilisateur examine cette liste triée et détermine quels sont les documents pertinents et non pertinent.
- Composition automatique d'une nouvelle requête à partir des données issues de la requête initiale et des informations fournies par les documents pertinentes et non pertinentes sélectionnées.

Traitement avec la nouvelle requête reformulée automatiquement et transmise au moteur de recherche,

- Génération des documents détectés avec la nouvelle requête reformulée.
- Ce processus est répété jusqu'à ce que l'utilisateur soit complètement satisfait de l'ensemble des documents trouvés.

L'emploi de RF procure des avantages réels [Salton, 90]

La RF libère l'utilisateur des contraintes liées à la reformulation de requête dans un environnement plus ou moins connu, Permet de mener des opérations de recherche par étapes successives, favorisant ainsi une approche graduelle et rationnelle du champ d'intérêt.

- Procure un outil approprié de recherche permettant de pondérer les termes suivant leur importance relative et d'adapter les recherches aux caractéristiques de la collection sur laquelle elle s'applique.

L'application de la technique du RF part du principe que tous les documents pertinents dépistés par un moteur de recherche suite à une requête ont des caractéristiques communes.

Dans un modèle de recherche à base d'espace vectoriel, cela signifie que les vecteurs représentant les documents extraits et le vecteur de requête ont une similarité notable. Cela implique, par voie de conséquence, que les vecteurs de termes des documents non pertinents par rapport à la requête et les vecteurs de termes des documents pertinents sont dissemblables. Ces considérations ont conduit à la conception d'approches de reformulation automatique de requête. En effet, le premier prototype implémentant la technique du RF était basée sur le modèle de recherche vectoriel et consistait en un ensemble de termes, éventuellement pondérés. Ainsi, les requêtes étaient exprimées sous forme de vecteurs :

$$Q_0 = (r_0, r_1, r_2 \dots, r_t)$$

Où Q_0 est la requête initiale; r_i représente la pondération du terme i pouvant prendre les valeurs 0 ou 1 suivant l'absence ou la présence de ce terme dans la requête. Les termes de la requête initiale peuvent être des mots choisis dans un thesaurus, un dictionnaire de termes contrôlés, ou choisis librement parmi les mots du domaine.

À partir des documents pertinents produits par la requête Q_0 , le processus du RF génère automatiquement une nouvelle requête :

$$Q' = (r'_0, r'_1, r'_2 \dots, r'_t)$$

Où r'_i représente la pondération du terme i de la première requête Q_0 modifiée dans la requête Q' .

B. Les techniques de réinjection de pertinence sans jugement de l'utilisateur (automatique) [Mataoui, 07]

La technique de réinjection de pertinence, décrite précédemment dépend des jugements de l'utilisateur sur la liste des documents trouvés. Une autre approche alternative, appelée **pseudo** ou **blind RF** emploie la technique de réinjection de pertinence automatiquement sans utiliser l'information issue des jugements de l'utilisateur.

Dans cette technique le système génère une liste triée de documents pour la première requête initiale. Ensuite, il sélectionne un certain nombre de documents (petit ensemble) à partir des tops documents du classement. Une nouvelle requête générée par le processus de reformulation de requêtes est utilisée pour produire une nouvelle liste triée de documents.

Le principe de base du pseudo RF est qu'une itération feedback basée sur les premiers documents sélectionnés pendant l'exécution de la requête initiale, va engendrer un meilleur classement pour les documents pertinents.

Selon Croft & Harper [Croft and Harper, 1979], cette technique souffre d'un problème majeur qui est appelé «query drift». Ce problème apparaît l'ensemble des documents pertinents (ou ne contient pas du tout). Donc, cette technique ne fonctionne bien que dans le cas où la requête initiale permet d'extraire de bon résultats (beaucoup de documents pertinents)

Le schéma ci-dessous explique la procédure de réinjection automatique.

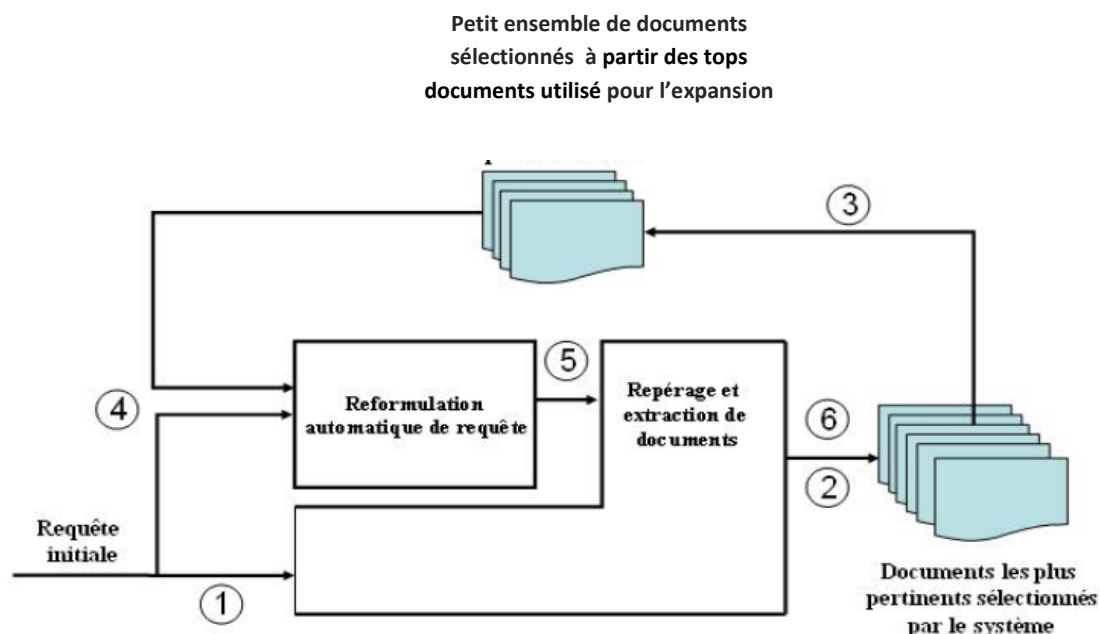


Figure II.2 : Processus de fonctionnement du blind RF.

II.3. La réinjection de pertinence dans les modèles de recherche d'information [Linda Tamine, 00]

La technique du RF (Relevance feedback) peut être appliquée aux principaux modèles de recherche que sont: le modèle vectoriel, le modèle booléen et le modèle probabiliste. Nous étudions ici le modèle vectoriel, et modèle probabiliste.

II.3.1. Réinjection de pertinence dans le modèle vectoriel (Approche de Rocchio)

Les stratégies de reformulation développées dans le modèle vectoriel induisent une repondération de requête avec expansion. La reformulation consiste alors à orienter le vecteur requête vers les vecteurs documents pertinents et de l'éloigner des vecteurs documents non pertinents.

Rocchio [Rocchio, 1971] décrit une stratégie permettant de dériver itérativement le vecteur requête optimal à partir d'opérations sur les vecteurs documents pertinents et vecteurs documents non pertinents. La formule proposée est la suivante :

$$Q_{i+1} = \alpha Q_i + \frac{\beta}{P} \sum_{dp \in Dp} dp - \frac{\delta}{Np} \sum_{dnp \in Dnp} dnp \quad (1)$$

Où :

Q_{i+1} : Requête construite à la $i+1$ ème itération de feedback

Q_i : Requête construite à la i ème itération de feedback

Dp : Ensemble des documents jugés pertinents

Dnp : Ensemble des documents jugés non pertinents

P : Nombre de documents jugés pertinents

Np : Nombre de documents jugés non pertinents

α, β, δ : Constantes.

Le jugement de l'utilisateur est ainsi exploité pour :

- ajouter des termes issus des documents pertinents : leurs coefficients deviennent non nuls dans le nouveau vecteur requête,
- Repondérer les termes de la requête : les poids des termes de la requête sont réévalués sur la base de leur fréquence d'occurrence dans les documents pertinents et documents non pertinents.

Salton et Buckley [Salton & Buckley, 1990] ont comparé l'effet de formule de Rocchio, Ide-Regular et Ide-DecHi, sur différentes collections.

$$Q_{i+1} = \alpha Q_i + \beta \sum_{dp \in Dp} dp - \delta \sum_{dnp \in Dnp} dnp \quad Ide - Regular \quad (2)$$

$$Q_{i+1} = \alpha Q_i + \beta \sum_{dnp \in Dp} dp - \delta dnp \quad Ide - Dec - Hi \quad (3)$$

Où :

dnp : premier document jugé non pertinent

Dec-Hi: **D**ecrease using **H**ighest ranking non relevant documents.

Les auteurs ont mené une série d'expérimentations pour évaluer la reformulation par injection de pertinence, en comparant l'impact de l'utilisation des trois formules sur les résultats de recherche d'information, effectuées dans les collections CRANFIELD, CISI et MED. Les résultats présentés montrent que la formule (3) de Ide-Dec-Hi donne les meilleurs résultats avec les paramètres $\alpha = 1$, $\beta = 0.75$, $\delta = 0.25$.

Buckley & al [Buckley & al, 1994] se sont intéressés à l'application de la technique de relevance feedback dans la base TREC. La nouvelle requête est obtenue selon la formule Ide-Regular avec les paramètres $\alpha = 8$, $\beta = 16$, $\delta = 4$.

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

Par ailleurs, les auteurs définissent des mini-documents par groupage de 200 mots interconnectés dans le but de restituer les parties de documents pertinents (passage retrieval). Chaque terme du mini-document est localement pondéré. La combinaison de la similitude locale et similitude globale dans le calcul de similitude requête/documents, a permis d'améliorer d'avantage la reformulation de requête avec un taux de 16%.

II.3.2. Réinjection de pertinence dans le modèle probabiliste

Sur la base du modèle probabiliste, Harman [Harman, 1992], Haines [Haines & Croft, 1993] et Robertson [Robertson & al, 1995] ont développé des formules de pondération de requête en utilisant le jugement de l'utilisateur sur la pertinence des documents restitués par le système.

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

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

Où : q_{ki} : est le poids d'indexation du terme i dans la requête q_k .

P_i : Probabilité ($d_{ji} = 1$ / D_j est Pertinent)

U_i : Probabilité ($d_{ji} = 1$ / D_j est Non Pertinent)

C : Constante

d_{ji} : Le poids d'indexation de terme t_i dans le document d_j .

Avec :

$d_{ji} = 1$ si t_i occure dans D_j , 0 sinon

$P_{init} = 0.5$

$U_{init} = n_i / N$

N : Nombre total de documents dans la collection

n_i : Nombre de documents de la collection contenant le terme t_i

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

$$VS (t_i) = \log \frac{P_i * (1 - q_i)}{q_i * R(1 - p_i)}$$

Où : q_i : est le poids de terme i dans la requête q .

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

R : Nombre de documents jugés pertinents

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

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

Où :

NR : Nombre de documents non pertinents retrouvés

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

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

Il en résulte ainsi une repondération de la requête avec expansion. Haines & Croft [Haines & Croft, 1991] étendèrent le modèle d'inférence introduit par Turtle & Croft [Turtle & Croft, 1991] en incluant des techniques de reformulation de requête. Un nouveau type de nœud et deux couches associées y ont été intégrés dans le réseau afin de traduire le jugement de pertinence de l'utilisateur, relativement à la présence des concepts modélisés dans les documents restitués. Ces travaux ont montré la faisabilité de la relevance feedback dans le modèle d'inférence bayésien. Cependant, la complexité engendrée dans la structure du réseau et dans le calcul de probabilités conditionnelles dégradent les performances globales de la stratégie [Ponte, 1998]. Dans le cas du modèle probabiliste, la stratégie d'injection de pertinence présente l'intérêt d'être directement reliée à la dérivation de poids des termes de la requête. Cependant, elle pose le problème de la complexité de calcul des probabilités conditionnelles.

II.4. Analyse des approches de reformulations de requêtes

Partons du fait que la reformulation de requête est un mécanisme pouvant s'appliquer aux différents modèles de recherche et de représentation de l'information, nous avons jugé opportun de le caractériser par des critères spécifiques.

Ces critères nous servent de comparaison pour deux techniques de reformulation de requête : reformulation directe et reformulation par injection de pertinence. Ces critères sont les suivants :

- *Source de modification* : précise l'origine des informations exploitées pour effectuer la reformulation de requête.
- *Interaction avec l'utilisateur* : précise si la technique considérée impose une interaction avec l'utilisateur.
- *Hypothèse de base* : spécifie l'hypothèse sur laquelle est fondée l'approche

- *Traitement des dépendances* : précise si la technique considérée traite les termes les uns indépendamment des autres ou par combinaison.
- *Paramètre d'efficacité* : spécifie les facteurs influents sur les performances de la technique considérée.

Le tableau ci-dessous présente les caractéristiques des stratégies fondamentales de reformulation de requête. Ce mécanisme est proposé, pour soutenir les modèles de représentation et de recherche d'information en exploitant des informations internes et/ou externes permettant d'enrichir la description initiale de l'utilisateur.

	Source de modification	Hypothèse de base	Interaction avec L'utilisateur	Paramètre d'efficacité
Reformulation Directe	Les liens sémantiques entre terme	Les liaisons sémantiques préétablies procurent une information fondée	Non	Degré d'exhaustivité du mécanisme d'indexe
Reformulation Par injection de pertinence	Lien sémantiques entre jugement de pertinence de l'utilisateur	Les documents à une même requête sont ressemblants	Oui	Degré de ressemblance requête/document Qualité de la recherche initiale

Tableau II.1 : Comparaison des techniques de reformulation de requête [Linda Tamine, 00].

II.5. Paramètres de performance d'un SRI [Linda Tamine, 00]

Un nombre considérable d'expérimentation ont été effectuées sur les collections de documents pour l'évaluation de l'impact induit par la reformulation de requête sur le processus de recherche d'information. Les auteurs révèlent que son apport en performance est en fonction d'un nombre considérable de paramètres.

Ces paramètres sont très dépendants de :

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

Nous évoquons dans ce qui suit les principaux paramètres :

1. Nombre de termes ajoutés à la requête

L'ajout de termes à la requête accroît la performance du SRI. Buckley & al [Buckley & al, 94] ont expérimenté le retour de pertinence dans l'environnement multi-fond documentaire TREC ; ils ont montré que le taux de performance est d'avantage corrélé avec le nombre de termes ajoutés qu'avec le nombre de documents initialement retrouvés.

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

$$RP(N) = A \log(N) + B \log(X) + C$$

Où

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

N : Nombre de documents restitués

X : Nombre de termes ajoutés à la requête

A, B, C : sont des constantes

Ils ont conclu que le seuil critique du nombre de termes à ajouter à la requête dépend des caractéristiques de la collection.

En outre, la pondération différenciée des termes ajoutés à la requête accroît la performance du système. On attribue un poids :

- Moins important aux termes ajoutés [Haines & Croft, 93]
- plus important aux termes issus des documents pertinents que ceux issus des documents non pertinents [Salton & Buckley, 90].

2. La méthode de sélection des termes

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

Salton et Buckley [Salton et Buckley, 90] ont expérimenté séparément, l'ajout de tous les nouveaux termes, tous les termes issus des documents pertinents et les termes les plus fréquents dans les documents restitués à la requête initiale. L'expansion de la requête avec tous les nouveaux termes offre de meilleurs résultats que les autres méthodes ; toutefois l'écart de performance n'est pas très considérable relativement aux exigences de temps et d'espace mémoire.

Robertson [Robertson & al, 95] et Haines [Haines & Croft, 93] adoptent une méthode de sélection de nouveaux termes sur la base d'une fonction qui consiste à attribuer pour chaque terme un nombre traduisant sa valeur de pertinence. Les termes sont alors triés puis sélectionnés sur la base d'un seuil.

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

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

Où :

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

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

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

RSV(Retrieval Statue Value)

Harman [Harman, 1992] propose quant à elle, les principales fonctions suivantes :

- **IDF** : Inverse Document Frequency

$$IDF = - \log P_t$$

Où P_t est la probabilité qu'un document contient le terme t .

La sélection de termes ayant une valeur IDF importante, revient en fait, à sélectionner les termes ayant une faible probabilité d'occurrence dans un document de façon aléatoire.

- **RTF – IDF : Relevant Term Frequency – IDF**

$$RTF - IDF_t = RTF * IDF_t$$

Où RTF est la fréquence d'apparition d'un terme dans les documents pertinents.

La sélection de termes ayant une valeur RTF – IDF importante revient à sélectionner les termes caractéristiques des documents pertinents (faible probabilité d'apparition aléatoire dans les documents pertinents). C'est la méthode qui a donné les meilleurs résultats.

Boughanem [Boughanem, 95] a expérimenté la reformulation de requêtes sur un SRI basé sur l'approche connexionniste. Les termes ajoutés à la requête sont sélectionnés sur l'approche d'un seuil de cooccurrence avec les termes initiaux de la requête. Il conclut que la valeur idéal du seuil varie de façon inversement proportionnelle à la taille de la base et taille des documents.

En outre, interviennent de façon moins considérable [Salton & Buckley, 90] .

3. Longueur moyenne de requête

L'accroissement des performances de la reformulation de requête est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite [Buckley & al, 1994]. Dans ce sens, des expérimentations intéressantes ont été réalisées sur la base TREC7 et présentées dans [Cormack & al, 1999]. Les auteurs montrent en effet que la dérivation automatique de courtes requêtes à partir de documents jugés ou supposés pertinents à la suite d'une recherche initiale, permettent d'atteindre des résultats très performants pour différentes tâches : recherche, filtrage et routing.

4. Méthode de sélection de document

Il existe peu de travaux réalisés pour la sélection des documents comparativement à la sélection des termes, la méthode présentée dans [Fiana & Oren, 00] considère le document pertinent comme un bon représentatif d'un ensemble de D_{rel} (les documents pertinents dans le corpus) dans la mesure où il peut «aider efficacement» pour trouver les documents pertinents à partir de D_{rel} par l'intermédiaire d'une recherche effectuée sur le corpus.

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

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

- *La première* estime la représentativité d'un document indépendamment des autres documents dans D_{rel} , en sélectionnant les k - top documents classés, Cette méthode assigne

au document d ($\in D_{rel}$) un score, $Score(d)$, qui reflète la représentativité de d dans D_{rel} . Les k -tops documents dans D_{rel} sont ensuite utilisés comme entrée à la méthode de sélection des termes d'expansion.

Parmi les caractéristiques utilisées pour la sélection des documents :

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

$score_{(Random)}(d) \stackrel{def}{=} \frac{1}{n}$; Puis, choisir les k documents à partir de D_{rel} au hasard.

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

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

$$Score_{QuerySim}(d) \stackrel{def}{=} sim(q, d)$$

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

$$Score_{Length}(d) \stackrel{def}{=} -|d|$$

$|d|$ est la taille de document d .

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

✓ *La méthode centroïde* : En termes de modèle de langage, la *centroïde* est une probabilité de distribution de tout le vocabulaire.

$$p(w|Cent(D_{rel})) \stackrel{def}{=} \frac{1}{n} \sum_{d' \in D_{rel}} p(w|d')$$

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

$$Score_{centroid}(d) \stackrel{def}{=} -D(p(\cdot|cent(D_{rel})) || p(\cdot|d))$$

✓ *Les méthodes basées sur le Graphe* : Certains travaux sur le ré-classement de la première liste de documents trouvés en réponse à une requête ont montré que les documents qui sont très semblables aux autres documents dans la liste ont une forte probabilité de la pertinence. L'idée est que ces documents représentent la liste entière, et par la vertu de la façon dont la liste a été créé - c'est à dire, en réponse à la requête, ils pourraient être pertinents au besoin fondamental de l'information. Toutefois, la liste est composée de plusieurs documents à la fois pertinents et non pertinents.

Nous explorons davantage dans notre approche l'idée de ré-classement de la première liste de documents trouvés en réponse à une requête en signalant explicitement la contrainte de similarité entre ces documents retournés et la contrainte de la longueur de document.

Contrairement aux approches décrites auparavant notre approche que nous allons détailler et expérimenté dans le chapitre III tient compte du score initial des documents retourné avec la première requête en les combinant avec le score initial de ces documents.

II.6. Avantages et risques d'expansion de requête [Léon Bouraoui, 10]

Avantages:

Augmentation du rappel

Augmentation de précision

Suggestion à l'utilisateur de termes qu'il ne connaissait pas

Risques:

"Query drift" : déviation par rapport à l'intention initiale de l'utilisateur

Exemple: java (programmation) \Rightarrow java (île) = Indonésie, javanais, etc.

Cette exemple montre bien cette déviation, lorsque l'utilisateur cherche le mot java qui est un langage de programmation, et en faisant l'expansion de requête, le résultat peut être java qui est une île qui se trouve à l'Indonésie parce que le document qui parle de cette île peut contenir le mot java (traitement indépendant des termes). C'est ce qui fait que le moteur de recherche confond entre java (programmation) et java (île).

Exemple de reformulation de requête dans les moteurs de recherche

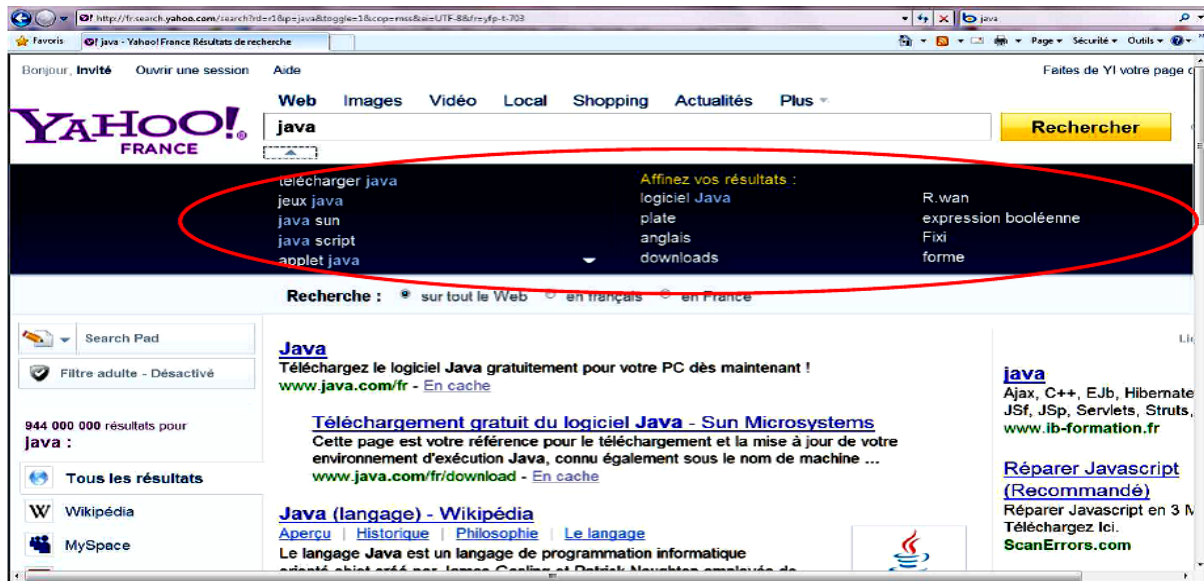


Figure II.2 : expansion de requête dans des moteurs de recherche: Yahoo [Léon Bouraoui, 10]

L'exemple ci-dessus montre la recherche d'information avec le moteur de recherche (yahoo) navigateurs web, par exemple ici avec la requête java le moteur de recherche fait une expansion automatique et affiche les différentes informations sur java pour choisir exactement celle recherché.

Conclusion

Ce chapitre a porté essentiellement sur l'étude de la reformulation de requêtes dans les SRI de manière générale, dans la première partie on a entamé les deux principales contextes de reformulation de requête (contexte locale/globale), dans la seconde partie on abordé la reformulation de requête dans le cadre des différents modèles de recherches d'informations.

La reformulation de requêtes est une phase importante du processus de recherche d'information. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin. Cette technique peut être appliquée automatiquement ou d'une façon interactive, c'est à dire avec l'intervention de l'utilisateur.

Nous decrivons dans le chapitre suivant notre approche de sélection de documents pour l'expansion de requête ainsi que les résultats obtenus de l'évaluation.

Introduction

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

Nous avons présenté dans le chapitre précédent les différentes approches de reformulation de requête, dans ce chapitre nous allons présenter une nouvelle approche dont l'objectif est de réordonner les documents restitués dans la première recherche en se basant sur la similarité et la taille de document, car pour choisir les bons termes à ajouter à la requête il faut au préalable choisir les bons documents afin de répondre efficacement aux besoins des utilisateurs.

Dans ce qui suit nous allons donner les fondements théoriques (architecture) de notre approche qui est implémentée en utilisant la plateforme Terrier, puis un exemple illustratif ainsi que les outils de développement et en fin quelques résultats expérimentaux obtenus sur deux collections de tests TREC AP88, WSJ9092.

I. Architecture générale de notre approche

La figure suivante illustre l'architecture de notre approche :

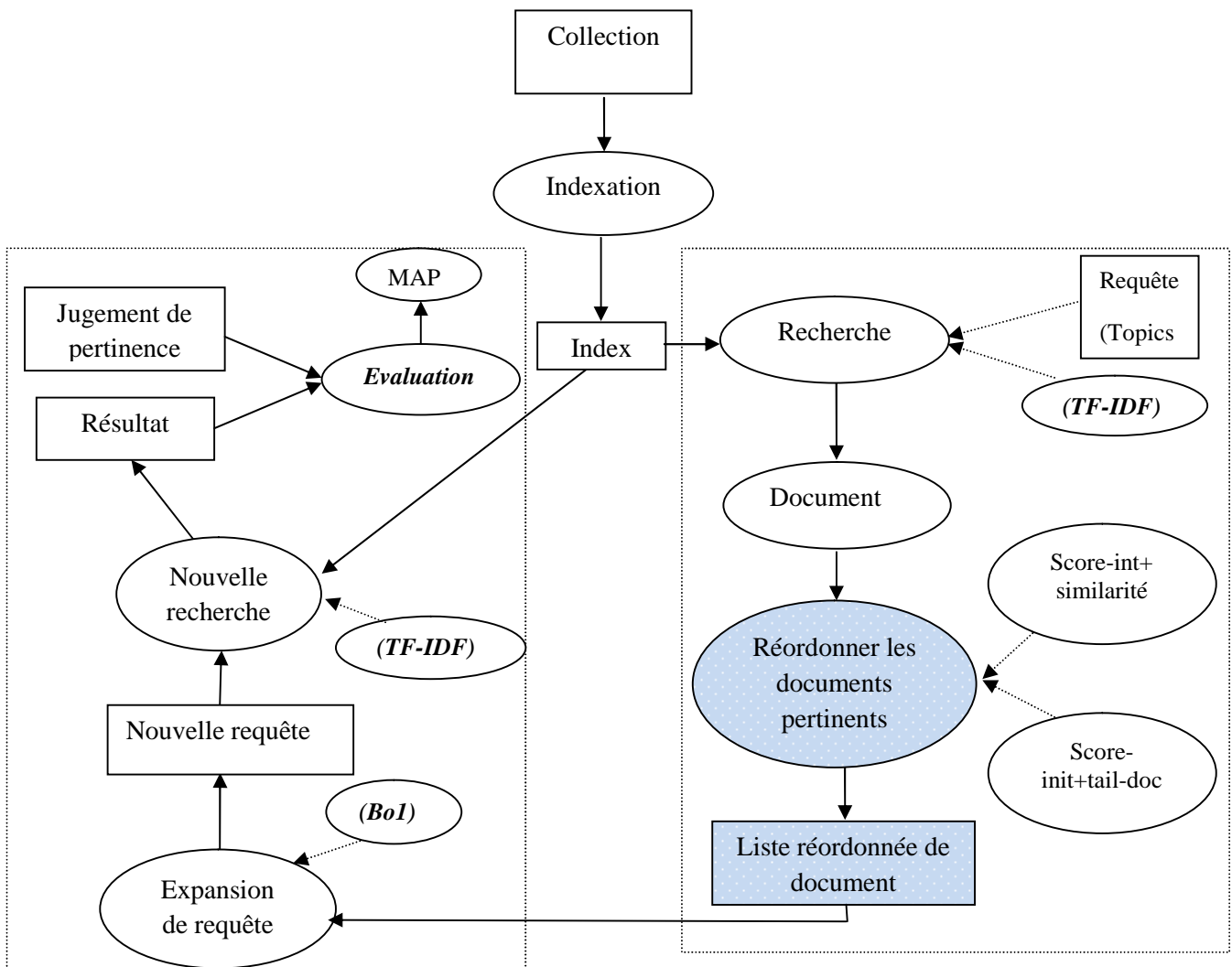


Figure III.1: Architecture générale de notre approche.

I.1. Présentation de notre approche de sélection des documents utilisés pour l'expansion de la requête

La reformulation de requête est proposée comme une méthode élaborée pour la RI, s'inscrivant dans la voie de conception des SRI adaptatif aux besoins des utilisateurs. C'est un processus permettant de générer une requête plus adéquate à la RI dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est de modifier la requête de l'utilisateur par ajout de termes significatifs.

Dans cette section nous allons présenter notre travail qui consiste à l'implémentation d'une nouvelle approche de sélection de documents d'expansion de la requête dans la plate forme de RI Terrier en utilisant la similarité et la taille de document dans le but de réordonner les documents restitués dans la première recherche.

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

Etape 1 : La première recherche

- En se basant sur la structure de donnée (Index) produite par le processus d'indexation, le module de recherche assigne un score pour chaque terme de la requête dans le document puis assigne un score (score initial) aux documents pour les retourner dans une liste ordonnée (résultat de la première recherche) en se basant sur un modèle de pondération TF-IDF qui comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer.

Etape 2 : Réordonnement des documents de la première recherche

- C'est dans cette étape que se situe notre travail qui consiste à calculer la similarité entre les documents retournés dans la première recherche selon la mesure de cosinus suivante :

$$\cos(D_k, D_j) = \frac{\sum_{i=1}^N d_{ik} \cdot d_{ij}}{[\sum d_{ik}^2]^{1/2} [\sum d_{ij}^2]^{1/2}}$$

$$Sim(D_k) = \sum_{j \in D_{rel}} \cos(D_k, D_j)$$

Tel que:

D_{rel} : Les documents retournés par la première recherche ().

D_k et $D_j \in D_{rel}$.

- Une fois le score de similarité est calculé nous appliquons des nouvelles formules de classement des documents pertinents que nous avons proposés pour réordonner la liste de documents retournés dans la première recherche.

✓ La première formule que nous avons proposée est dans le but de réordonner la liste de documents retournés selon leur degré de ressemblance en calculant la similarité entre les documents

✓ retournés par la première recherche. La formule est la suivante :

$$score-final(D) = \lambda * score-init(D) + (1 - \lambda) * \log(1 + sim-doc(D)) \quad (1)$$

✓ La seconde formule proposée fait intervenir la similarité maximale d'un document pour diminuer l'écart entre les valeurs de similarités. la formule est la suivante :

$$Score-final(D) = \lambda * score-init(D) + (1-\lambda)(\log(1+(sim-doc(D) \setminus sim-max(D)))) \quad (2)$$

✓ La troisième formule proposée fait intervenir la moyenne de similarité d'un document pour diminuer l'écart entre les valeurs de similarités. la formule est la suivante :

$$Score-final(D) = \lambda * score-init(D) + (1-\lambda)(\log(1+(sim-doc(D) \setminus moy-sim(D)))) \quad (3)$$

Notons que l'utilisation de **log** à pour but de diminuer l'écart entre les valeurs de similarités (lissage).

✓ La quatrième formule proposée fait intervenir la taille de document du fait que les expérimentations réalisées par plusieurs auteurs ont montré que les documents longs ont une plus grande probabilité de pertinence parce qu'ils contiennent plus de termes d'appariements avec la requête. la formule est la suivante :

$$Score-final(D) = \lambda * score-init(D) + (1-\lambda)(\log(tail-doc(D))) \quad (4)$$

✓ La cinquième formule proposée fait intervenir la taille maximale d'un document pour diminuer l'écart entre les valeurs de la taille de document. la formule est la suivante :

$$Score-final(D) = \lambda * score-init(D) + (1-\lambda)(\log(1+(tail-doc(D) / tail-max(D)))) \quad (5)$$

✓ La sixième formule proposée fait intervenir la taille moyenne d'un document pour diminuer l'écart entre les valeurs de la taille de document. la formule est la suivante :

$$Score-final(D) = \lambda * score-init(D) + (1-\lambda)(\log(1+(tail-doc(D) / tail-moy(D)))) \quad (6)$$

Tel que :

$\lambda \in [0,1]$: est le coefficient qui estime l'importance entre la similarité (ou la taille de document) et le score initial.

Scor-final : est le score des documents retournés après l'ajout de l'une des évidences cités (similarité, taille de document).

Scor-init : est le score initial des documents.

Sim-doc : est la similarité entre les documents pertinents retournés par la première recherche.

sim-max : est la similarité maximale entre les documents pertinents retournés par la première recherche.

moy-sim : est la similarité moyenne entre les documents pertinents retournés par la première recherche.

tail-doc : est la taille d'un document.

tail-max : est la taille maximale des documents pertinents retournés par la première recherche.

tail-moy : est la taille moyenne des documents pertinents retournés par la première recherche.

Nous avons proposé aussi d'autres formules en faisant intervenir les deux facteurs, la similarité et la taille de document par la combinaison de ces derniers dans le but d'optimiser la précision moyenne.

Les formules de combinaisons que nous avons proposées sont les suivantes:

$$\checkmark \quad \text{Score-final}(D) = \lambda * \text{score-init}(D) + (1-\lambda) * (\text{Log}(\text{sim-doc}(D) + \text{taille-doc}(D))) \quad (7)$$

$$\checkmark \quad \text{Score-final}(D) = \lambda * \text{score-init}(D) + (1-\lambda) * (\log(\text{sim-doc}(D) / \text{taille-doc}(D))) \quad (8)$$

$$\checkmark \quad \text{Score-final}(D) = \lambda * \text{score-init}(D) + (1-\lambda) * (\text{Log}(A * (\text{sim-doc}(D)) + (1-A) * (\text{taille-doc}(D)))) \quad (9)$$

Sachant que :

$A \in [0,1]$: c'est le coefficient qui estime l'importance entre la similarité et la taille d'un document.

A la fin de cette étape on aura une nouvelle liste de documents réordonnée selon le nouveau score (score avant l'expansion).

- **L'algorithme de similarité**

Nous avons calculé la similarité entre documents selon l'algorithme suivant :

Debut

docIDS : Tableau [] ; // identifiant des documents retournés dans la 1ère recherche

score : Tableau [] ; // score des documents retournés dans la 1ère recherche (score initial)

// pour chaque document retourné

faire // calculer la norme f

f ← 0 ;

i ← 0 ;

```

f ← racine (somt_doc) ;

hm : une table ;

hm_doc : une table,

Sim_hm : une table ;

hm(doc_id, f); // mettre la norme de chaque document dans la table hm;

hm(doc_id) ; // retourne la norme pour chaque document j tel que (j < 1000) ;

fait ;

// Calcule du facteur Xi * Yi

hm_doc ← term_doc[] ; // mettre les termes des documents dans la table hm_doc

mul ← 0 ;

mul ← mul + (term_doc[] * hm_doc(term_doc[]));

// Calcul de score de la similarité

cosinus ← 0 ;

cosinus ← mul / (hm(doc_id) * hm(doc_id_comp));

// mettre la similarité de (doc_id et doc_id_comp) dans la table sim_hm ;

Sim_hm(doc_id, cosinus);

Sim_hm(doc_id_comp, cosinus);

// Calcul du nouveau score (score final)

Const  $\lambda$  = constante; //  $0 < \lambda < 1$ 

// Pour tous les documents retournés de la 1ere recherche

// Faire la fonction de pondération

Pour  $0 \leq i \leq \text{doc\_id}$  faire

Score[i] ←  $\lambda \text{ score}[i] + (1 - \lambda)(\log(1 + \text{sim\_hm}[i]))$  ;

fait ;

Fin.
```

Etape3 :L'expansion de requête

- Terrier inclut la pseudo-relevance feedback automatique, sous forme de *Query Expansion*. Cette méthode fonctionne en faisant l'extraction des tops termes informatives à partir des tops documents classés (un nombre de document spécifié) par attribution du score pour chaque terme en utilisant le modèle d'expansion Bo1 par défaut et les ajouter à la requête , La nouvelle requête est re-pondérée (avec TF-IDF) et fournit un ensemble de documents qui seront stockés dans *le fichier.res (pertinence système)*.

I.2. Terminologie et notation**❖ Le modèle d'expansion Bo1 [Ounis,2007]**

Terrier fournit plusieurs modèles de pondération des termes dans le cadre de **DFR** (Divergence From Randomness) qui sont utiles pour identifier les termes informatives des tops documents classés.

La divergence de caractère aléatoire (DFR) emploie un mécanisme d'expansion de requête qui est une généralisation de la méthode de Rocchio (Rocchio, 1971). Le mécanisme de DFR d'expansion de requête comporte deux étapes.

-D'abord, il applique un modèle de pondération de terme DFR pour mesurer l'informativité des termes dans les tops documents ordonné. L'idée du modèle de pondération de terme DFR est d'en déduire l'informativité d'un terme par la divergence de sa distribution dans les tops documents classés à partir d'une distribution aléatoire.

Le modèle de pondération de terme de DFR le plus efficace est le modèle Bo1 qui utilise les statistiques de Bose-Einstein. En utilisant ce modèle, le poids w d'un terme t dans les tops documents classés est donné par:

$$w(t) = tf_x \cdot \log_2 \frac{1 + P_n}{P_n} + \log_2(1 + P_n)$$

où tf_x est la fréquence du terme de requête dans les tops documents classés, P_n est donné par la FN , F la fréquence de le terme dans la collection, et N est le nombre de documents dans la collection, le réglage par défaut de Bo1 permet d'extraire 10 termes les plus informatives à partir des 3 tops documents retournés.

Les termes de la requête d'origine peuvent également apparaître dans les 10 termes extraits.

Dans la deuxième étape, le mécanisme **DFR** d'expansion de requête étend la requête par la fusion des termes extraits avec les termes de la requête originale. Le poids d'un terme de la

requête qtw est donné par une formule d'expansion de requête de paramètre-libre (parameter-free) :

$$qtw = \frac{qtf}{qtf_{max}} + \frac{w(t)}{\lim_{F \rightarrow t_{fx}} w(t)} = F_{max} \log_2 \frac{1 + P_{n,max}}{P_{n,max}} + \log_2(1 + P_{n,max})$$

où $\lim_{F \rightarrow t_{fx}} w(t)$ est la limite supérieure de $w(t)$, $P_{n,max}$ est donnée par F_{max}/N , et F_{max} est la fréquence F de la terme avec le $w(t)$ maximum dans les tops documents classés. Si un terme de requête originale n'apparaît pas dans les termes les plus informatifs extraits des tops documents classés, son poids de terme de requête reste égal à l'original.

❖ TF-IDF [Benaouicha, 2009]

La mesure $tf \times idf$ en RI désigne un ensemble de schémas de pondération de termes, c'est la fonction de pondération la plus répandue. tf désigne une mesure en rapport avec l'importance d'un terme pour un document. En général, cette valeur est déterminée par la fréquence du terme dans le document. Par idf , on mesure si le terme est discriminant (un terme est discriminant s'il apparaît peu dans d'autres documents). Cette mesure est utilisée en RI pour calculer la pertinence d'un document par rapport à une requête, ceci traduit le fait qu'un document est pertinent à une requête s'ils partagent assez de termes importants.

La mesure $tf \times idf$ est communément utilisée en RI, vu que cette mesure donne une bonne approximation de l'importance du terme dans le document, particulièrement dans des corpus de documents de tailles intermédiaires.

Les fonctions TF et IDF sont les suivantes :

$$TF = (k_1 * tf_{ij}) / (tf_{ij} + (k_1 * (1 - b + b * \frac{|D|}{\Delta_{lg}})))$$

$$IDF = \log \left(\frac{N}{D_i} \right)$$

Tel que : $K=1,2$ contrôle l'influence de la fréquence du terme t_i

B : est une constante appartenant à l'intervalle $[0, 1]$ et contrôle l'effet de la longueur du document (dans TREC, elle est fixée à 0.75).

N : est la longueur d'un document.

D_i : est le nombre de documents où le terme t_i apparaît dans une collection de documents de taille N .

tf_{ij} : correspond au nombre d'occurrences du terme t_i dans le document D_j

$|D|$: est la longueur d'un document.

Δ_{lg} : est la taille moyenne d'un document.

I .3. Exemple illustratif

Nous expliquons comment que les documents de la première recherche seront réordonnés avec notre approche à travers l'exemple montré dans le tableau III.1 et le tableau III.2 en prenant les quatre premiers documents:

Sachant que

- La fonction de pondération utilisée est :

$$\text{score-final} = \lambda * \text{score-initial} + (1 - \lambda) * \log(1 + \text{similarité})$$

- avec λ égal à 0.1 et le nombre de document utilisé pour l'expansion est égal à 14.

Score-initial	Doc-id	La similarité
7,78	AP881006-0283	134.05
7,71	AP880519-0286	188.33
7,62	AP880607-0250	256.24
7,6	AP880621-0182	166.40

Tableau III.1 : le classement des documents dans la première recherche.

Score avant expansion	Doc-id	La similarité
5,75	AP880607-0250	256.24
5,73	AP880517-0295	298.31
5,71	AP880518-0324	285.13
5,7	AP880414-275	263.69

Tableau III.2 : la liste des documents réordonnés après l'implémentation de notre approche.

On peut remarquer que le rang du document (AP880607-0250) à passé de 3 à 1 en utilisant le facteur de similarité et que les autres documents n'apparaient pas(ils ont changé de rang).

Avec le nouveau classement, nous choisissons k-tops documents (14 document) pour extraire les bons termes qu'on utilisera pour l'expansion de requête, ensuite nous obtiendrons le classement final des documents pertinents comme mentionner dans le tableau ci-dessous.

Score-final	Doc-id
13,31	AP880607-0250
12,68	AP880607-0043
11,66	AP880301-267
11,08	AP880525-323

Tableau III.3 : Le réordonnancement des documents après l'expansion.

Dans le tableau ci-dessus on remarque que le rang du document (AP880607-0250) reste toujours 1 après l'expansion de requête et que les autres documents n'apparaient pas(ils ont changé de rang).

III. L'Environnement Technique

Notre travail consiste à implémenter notre approche dans 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.1. Présentation de la plate forme Terrier

TERRIER, **TER**abyte **Retr**IEve**R** : est un moteur de recherche robuste et efficace, développé par le département informatique de l'université Glasgow de Scotland. Utilisé avec succès dans:

- La recherche Ad-hoc
- La recherche sur le web
- La recherche multilingue

Terrier offre une plate forme idéale destinée à l'indexation de volumes importants de documents: jusqu'à 25 millions de documents. C'est un logiciel Open Source écrit en Java.

Comme tous moteurs de recherche, terrier permet :

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

III.1.1. Architecture de Terrier

La figure III.3 montre l'architecture générale de Terrier.

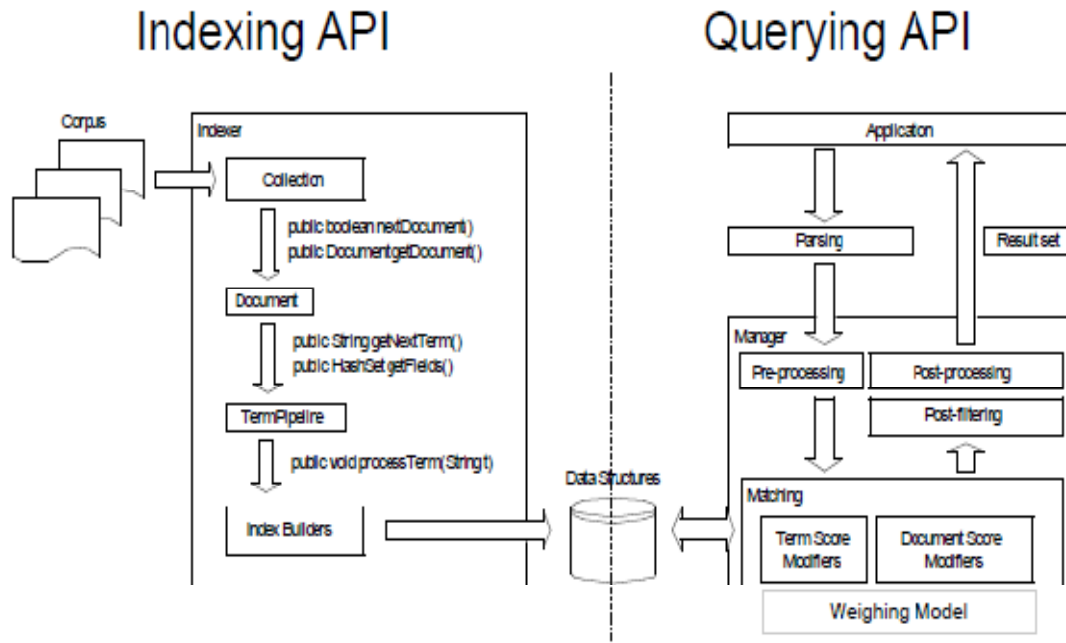


Figure III.2 : Vue d'ensemble de l'architecture Terrier.

Indexing et Querying APIs (Application Programming Interface) est facile à étendre, adapte de nouvelle applications, dispose d'une architecture modulaire, facile pour commencer à travailler avec, elle offre beaucoup d'option de configuration.

A. API d'indexation

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

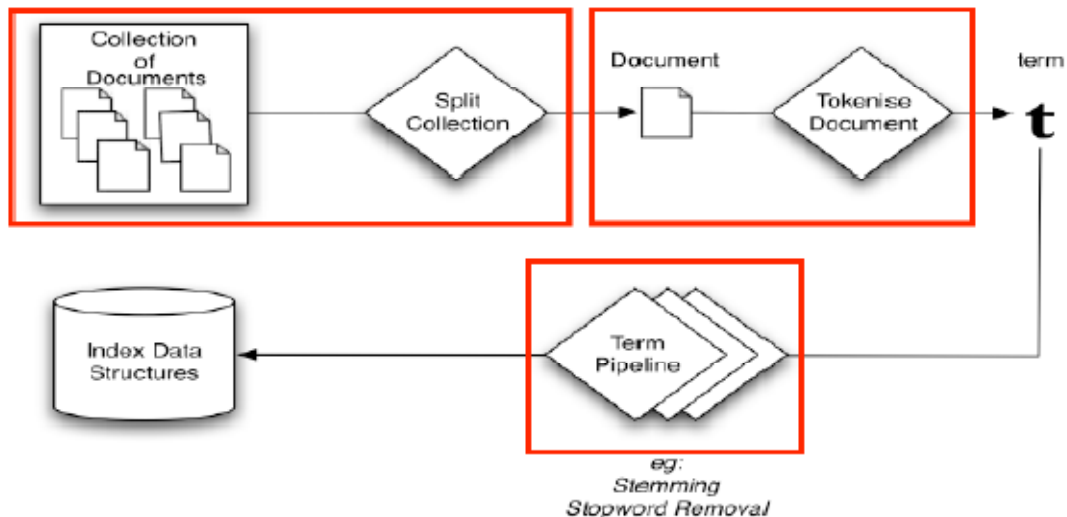


Figure III.3: le processus d'Indexation dans Terrier.

- **Collection** : Interface dans le package ...\\src\\uk\\ac\\gla\\terrier\\indexing, permet de splitter une collection (ou corpus) en documents.
- **Document** : Interface dans le package ...\\src\\uk\\ac\\gla\\terrier\\indexing, permet de parcourir les documents et extraire les Termes (en utilisant Tokeniser).
- **Term Pipeline** (Interface) : effectue les traitements des termes extraits :
 - Eliminer les mots vides (Stopwords)
 - Lemmatisation des termes selon la langue.

B. Les structures de l'indexe

Après l'indexation, les termes sont stockés en structures de données suivantes:

- **lexicon**: contient les Informations sur chaque terme de la collection (Terme, Id terme, Nombre docs qui contiennent le terme, Fréquence de terme dans la collection, Offset dans le fichier inverse).
- **Inverted Index** : Fichier inverse (Id Terme, Id document, Fréquence terme dans le document, #Filds).
- **Direct index**: Index (Id Terme, Id document, Fréquence terme dans le document, #Filds).
- **Document Index** :(Id Terme, Fréquence Terme, #Filds).

C. API de recherche

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

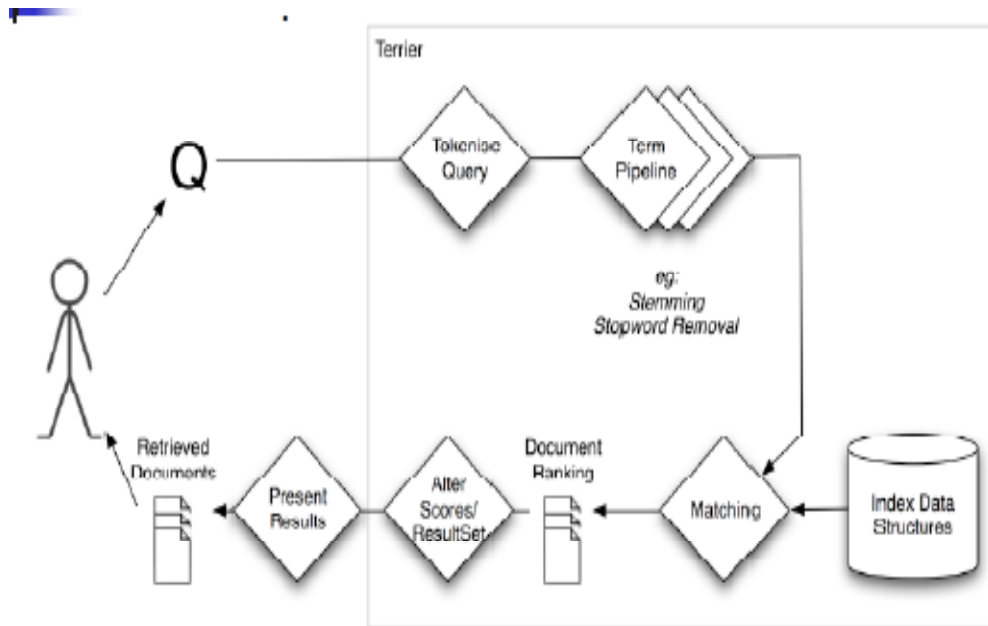


Figure III.4: le processus de recherche dans Terrier.

L'API de recherche contient les classes suivantes :

- **Query:** Classe abstraite qui représente la requête.

Terrier supporte Trois modèles de requête:

SingleTermQuery: Désigne la requête qui contient un seul terme.

MultiTermQuery : Désigne la requête qui contient plusieurs termes.

FieldQuery : Terme qualifié par un champ (Exemple: dans le titre du document).

- **Manager:** Chargé de la gestion de la recherche, il contient les étapes suivantes :

Pre-processing: Appliquer l'élimination des mots vides et troncature.

Matching : Déterminer les documents qui répondent à la requête en initialisant:

- *WeightingModels* :

Assigner un score pour chaque terme de la requête dans le document (Pondération), Plusieurs Modèles de pondération sont implémentés : TF_IDF, BM25, ...

- *DocumentScoreModifiers* :

Il permet de Modifier le score d'un document en fonction du langage de la requête.

Post-filtrng: Filtrer les documents pertinents selon un critère bien défini (exemple : type de document).

Post-processing: Reclasser les documents pertinents s'il y a une expansion de la recherche

- **Set-Results:** contient la liste des documents retournés classés selon leur degré de pertinence.

Une description détaillée de Terrier est donné annexe.

III.2. Le langage java [Groussard, 2000]

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

- des applications, sous forme de fenêtre ou de console ;
- des applets, qui sont des programmes Java incorporés à des pages web ;
- des applications pour appareils mobiles, avec J2ME ;
- et bien d'autres ! J2EE, JMF, J3D pour la 3D...

Java se voit attribuer plusieurs qualités dont voici quelques unes :

- **Java est orienté objet :** Java se veut un pur langage de P.O.O, c'est-à-dire qu'un programme s'y trouvera formé d'une classe ou de la réunion de plusieurs classes et il instanciera des objets.
- **Java est simple :** La syntaxe de Java est, en grande partie, tirée de celle de C++, avec l'avantage de l'élimination des mécanismes complexes, comme la gestion des pointeurs et de la mémoire, rendant ainsi les programmes java plus faciles à écrire et à compiler avec le moins d'erreurs possibles.
- **Java est distribué :** Java implémente les protocoles réseau standards, ce qui permet de développer des applications client/serveur en architecture distribuée, afin d'invoquer des traitements et/ou de récupérer des données sur des machines distantes.
- **Java est interprétée :** Un programme Java n'est pas exécuté, il est interprété par la machine virtuelle ou JVM (*Java Virtual Machine*), ce qui le rend un peu plus lent. Mais cela apporte des avantages, notamment celui de ne pas être obligé de recompiler un programme Java d'un système à un autre car il suffit, pour chacun des systèmes, de posséder sa propre machine virtuelle Java.
- **Java est robuste :** Java est un langage fortement typé et très strict. Par exemple la déclaration des variables doit obligatoirement être explicite en Java. Le code est vérifié (syntaxe, types) à la compilation et également au moment de l'exécution, ce qui permet de réduire les bugs et les problèmes d'incompatibilité de versions.

- **Java est sécurisé :** Au moment de l'exécution d'un programme Java, le JRE utilise un processus nommé le *ClassLoader* qui s'occupe du chargement du *byte code* (ou langage binaire intermédiaire) contenu dans les classes Java. Le byte code est ensuite analysé afin de contrôler qu'il n'a pas fait de création ou de manipulation de pointeurs en mémoire et également qu'il n'y a pas de violation d'accès.
- **Java est portable :** cette caractéristique est l'une de celles qui ont contribué à sa grande réputation parmi les communautés d'internet et ceci grâce à son indépendance de toute plate forme d'exécution car un programme Java peut tourner sur n'importe quelle machine possédant une JVM (Java Virtual Machine).

III.3. Présentation de NetBeans

NetBeans est à l'origine un EDI (Environnement de Développement Intégré) Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelque part en 2002, Sun a décidé de rendre NetBeans open-source. Mais NetBeans n'est pas uniquement un EDI Java, c'est également une plateforme. Il vous est possible de créer votre propre application Awt ou Swing, basée sur la plateforme NetBeans. Pour celles et ceux d'entre vous qui viennent du monde Eclipse, cela correspond à Eclipse RCP.

Sa conception est complètement modulaire : Tout est module, même la plateforme. Ce qui fait de NetBeans une boîte à outils facilement améliorable ou modifiable. La license de NetBeans permet de l'utiliser gratuitement à des fins commerciales ou non. Les modules que vous pourriez écrire peuvent être open-source comme ils peuvent être closed-source, Ils peuvent être gratuits, comme ils peuvent être payants.

Il présente une interface conviviale **GUI** (Graphical User Interface) qui nous permet d'éditer, compiler et exécuter un programme écrit en langage Java.

Netbeans est téléchargeable sur le site officiel www.netbeans.org et plus précisément dans la section téléchargement : NetBeans Downloads, son installation est facile.

IV. Résultats et Expérimentation

Dans la section IV.II nous allons présenter les expérimentations réalisées ainsi que les résultats obtenus avec notre approche.

IV.1. Collection de test

Pour évaluer les différentes formules proposées dans notre approche, nous nous appuyons sur deux collections de tests TREC. Ces deux collections contiennent des documents plats, la première AP88 (Associated Press 1988) et la deuxième c'est WSJ9091 (Wall Street Journal 90.92) qui sont de tailles moyennes. Nous avons utilisé 50 requêtes simples dans un fichier nommé Topics251-300.

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

Collection	Taille	Documents	Topics
AP88	237 Mo	79 919	251-300
WSJ90-92	241 Mo	74 520	251-300

Tableau III.4. Statistiques sur les collections de tests et les Topics utilisés.

IV.2. Evaluation et résultats des formules de classement des documents pertinent

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 par la suite comparés à ceux obtenus après la reformulation avec notre approche en appliquant les mêmes paramètres de recherche (le nombre de documents d'expansion, le modèle de pondération, le modèle d'expansion).

La mesure d'évaluation : pour évaluer les différents modèles nous avons utilisé la MAP (Précision moyenne).

Nous résumons dans les tableaux ci-dessous les résultats obtenus en appliquant les différentes formules :

IV.2.1. Les résultats obtenus sur la collection AP88

Le tableau III.5 présente les résultats obtenus avec le modèle de base sur la collection AP88 (en faisant varier le nombre de document) :

Nbr_documents	MAP	Nbr_documents	MAP	Nbr_documents	MAP
1	0.1867	19	0.2058	37	0.1996
2	0.1923	20	0.2049	38	0.1964
3	0.1930	21	0.2068	39	0.1948
4	0.1952	22	0.2064	40	0.1942
5	0.1936	23	0.2028	41	0.1951
6	0.1952	24	0.2042	42	0.1931
7	0.1953	25	0.2055	43	0.1919
8	0.1987	26	0.2064	44	0.1912
9	0.1992	27	0.2052	45	0.1912
10	0.1994	28	0.2063	46	0.1885
11	0.2076	29	0.2055	47	0.1872
12	0.2069	30	0.2089	50	0.1851
13	0.2075	31	0.2100	75	0.1883
14	0.2128	32	0.2101	100	0.1850
15	0.2060	33	0.2087	200	0.1857
16	0.2071	34	0.2059	300	0.1768
17	0.2079	35	0.2009	400	0.1667
18	0.2018	36	0.1997		

Tableau III.5 : la précision moyenne en faisant varier le nombre de document utilisés pour l'expansion dans le modèle de base.

Nous remarquons dans le tableau ci-dessus que le meilleur résultat est obtenu avec un nombre de document d'expansion égal à 14, et une précision moyenne égale à 0.2128.

Pour cela nous avons alors testé notre approche en considérant le même paramètre (14 documents) en variant le coefficient λ des formules (1), (2), (3), (4), (5) et (6). Le tableau ci-dessus montre les résultats obtenus avec les formules (1), (2) et (3) comparativement avec le modèle de base et cela sur la collection AP88.

λ	La formule (1)		La formule (2)		La formule (3)	
	MAP	Taux	MAP	Taux	MAP	Taux
0.08	-----	-----	0,2103	-	-----	-----
0.15	-----	-----	0,2166	1,78%	-----	-----
0.1	0,2037	-	0,2173	2,11%	0,21	-
0.2	0,2151	1,08%	0,2169	1,92%	0,216	1,50%
0.22	-----	-----	0,2169	1,92%	-----	-----
0.23	-----	-----	0,2173	2,11%	0,2178	2,34%
0.24	-----	-----	-----	-----	0,2195	3,14%
0.25	-----	-----	-----	-----	0,2203	3,52%
0.26	-----	-----	-----	-----	0,2202	3,47%
0.27	-----	-----	-----	-----	0,2195	3,14%
0.3	0,2198	3,28%	0,2149	0,98%	0,2161	1,55%
0.37	0,2207	3,71%	-----	-----	-----	-----
0.4	0,2195	3,14%	0,2109	-	0,2124	-
0.5	0,2199	3,33%	0,2089	-	0,2099	-
0.6	0,211	-	0,2101	-	0,2087	-
0.7	0,2089	-	0,2144	0,75%	0,2102	-
0.8	0,2105	-	0,2125	-	0,2135	0,32%
0.9	0,2135	0,32%	0,213	0,09%	0,2126	-

Le tableau III.6 : la précision moyenne en faisant varier λ dans les formules (1), (2), (3).

On note que : - le signe (-) : signifie que la valeur de λ testée n'a pas donné une amélioration.

- le signe (-----) : signifie que la valeur de λ n'a pas été testée.

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

- La formule (1) a apporté une amélioration légère dans la plus part des valeurs qu'on a attribué à λ . Notons que la précision moyenne optimale est égale à 0.2207 avec λ égal à 0.37 et le taux d'amélioration est de 3.71%.
- La formule (2) a apporté une petite amélioration pour la plus part des valeurs de λ . Mais le taux d'amélioration est inférieur à celui obtenu avec la formule (1).

La meilleure précision obtenue avec cette formule (2) est de 0.2173 pour un λ égal à 0.1 et 0.23.

- Avec la formule (3) on a obtenu une amélioration très proche de celle obtenue avec la formule (1).

La plus grande précision obtenue avec la formule (3) est égale à 0.2203 avec λ égal à 0.25.

D'après les résultats obtenus avec les formules (1), (2), (3) nous remarquons que :

- l'amélioration est importante avec la formule (1)
- l'amélioration diminue avec la formule (3) par rapport à (1).
- l'amélioration diminue avec la formule (2) par rapport à (1) et (3).

On peut déduire donc que la division de la similarité sur la moyenne des similarités diminue sa valeur et sa division sur la similarité maximale la diminue plus, ce qui la rend moins importante par rapport au score initial.

Le tableau suivant montre les résultats obtenus avec les formules (5), (6) et (7) qui font intervenir la taille des documents.

	La formule (4)		La formule(5)		La formule(6)	
λ	MAP	Taux	MAP	Taux	MAP	Taux
0.1	0,1771	-	0,1985	-	0,1953	-
0.2	0,1985	-	0,2033	-	0,1995	-
0.3	0,1994	-	0,2119	-	0,2014	-
0.4	0,2001	-	0,2151	1,08%	0,2134	0,28%
0.42	-----	-----	-----	-----	0,2144	0,75%
0.43	-----	-----	0,2152	1,12%	0,2164	1,69%
0.44	-----	-----	0,2153	1,17%	0,2146	0,84%
0.45	-----	-----	0,2153	1,17%	0,2143	0,70%
0.46	-----	-----	0,2143	0,70%	-----	-----
0.48	-----	-----	0,2143	0,70%	0,2143	0,70%
0.5	0,211	-	0,2144	0,75%	0,2142	0,65%
0.55	-----	-----	0,2131	0,14%	0,2131	0,14%
0.57	-----	-----	0,2129	0,04%	0,2129	0,04%
0.58	0,2153	1,17%	0,2153	1,17%	0,2153	1,17%
0.59	0,2161	1,55%	-----	-----	-----	-----
0.6	0,2145	0,79%	0,212	-	0,2153	1,17%
0.61	0,2119	-	-----	-----	0,2131	0,14%
0.7	0,2124	-	0,2125	-		
0.78	0,2152	1,12%	-----	-----	-----	-----
0.79	0,2153	1,17%	-----	-----	-----	-----
0.8	0,2139	0,51%	0,2126	-	0,2125	-
0.81	0,2139	0,51%	-----	-----	-----	-----
0.9	0,2125	-	0,213	0.09%	0,2127	-

Le tableau III.7 : la précision moyenne en faisant varier λ dans les formules (4), (5), (6).

Le tableau ci-dessus nous montre que :

- Avec la formule (4) introduisant la taille de document a influencé très légèrement sur la moyenne de précision par rapport aux trois premières formules qui tiennent compte de la similarité. Nous constatons ici que la meilleure valeur de MAP est égale à 0.2161 avec λ égal à 0.59 et le taux d'amélioration est égal à 1.55%.
- Avec la formule (5) le degré d'influence sur la moyenne de précision est moins important par rapport aux trois premières formules qui tiennent compte de la similarité. Nous constatons ici que la meilleure valeur du MAP est égale à 0.2153 avec λ égal à 0.58, 0.44 et 0.45 et le taux d'amélioration est égal à 1.17%.
- Dans la formule (6) la taille de document a amélioré la MAP et cette amélioration reste inférieure par rapport aux améliorations constatées avec les formules qui tiennent compte de la similarité. Nous constatons ici que la meilleur valeur de MAP=0.2164 avec $\lambda=0.43$ et le taux d'amélioration est égale à 1.69%.

D'après les résultats obtenus illustré dans les tableaux III.6 et III.7 nous prévoyons d'avoir des résultats plus performants en variant le nombre de document d'expansion et λ . Sachant qu'on a choisit seulement trois valeurs pour λ (la mauvaise, la moyenne et la meilleure) car le temps nécessaire pour faire un seul test dépasse les 40 minutes.

Le tableau suivant montre la précision moyenne en faisant varier le nombre de document et λ avec les formules (1), (2), (3).

nbr_doc	terrier	Formule (1)			Formule (2)			Formule (3)		
		λ	MAP	Taux	λ	MAP	Taux	λ	MAP	Taux
5	0.1936	0.1	0,2134	10,22%	0.1	0,2088	7,85%	0.25	0,2038	5,26%
		0.37	0,1958	1,13%	0.5	0,2008	3,71%	0.6	0,2008	3,71%
		0.9	0,1942	0,30%	0.9	0,1934	-	0.9	0,1943	0,36%
10	0.1994	0.1	0,2083	4,46%	0.1	0,2124	6,51%	0.25	0,2077	4,16%
		0.37	0,2103	5,46%	0.5	0,2099	5,26%	0.6	0,2099	5,26%
		0.9	0,1984	-	0.9	0,1973	-	0.9	0,1976	-
14	0.2128	0.1	0,2037	-	0.1	0,1985	-	0.25	0,2203	3,52%
		0.37	0,2207	3,71%	0.5	0,2144	0,75%	0.6	0,2087	-
		0.9	0,2135	0,32%	0.9	0,213	0,09%	0.9	0,2126	-
20	0.2049	0.1	0,2003	-	0.1	0,2083	1,65%	0.25	0,2112	3,07%
		0.37	0,21	2,48%	0.5	0,2054	0,24%	0.6	0,2054	0,24%
		0.9	0,205	0,04%	0.9	0,2052	0,14%	0.9	0,2053	0,19%
25	0.2055	0.1	0,205	-	0.1	0,2103	2,33%	0.25	0,2099	2,14%
		0.37	0,2119	3,11%	0.5	0,2098	2,09%	0.6	0,2098	2,09%
		0.9	0,2067	0,58%	0.9	0,2055	0%	0.9	0,2054	-
30	0.2089	0.1	0,1969	-	0.1	0,2049	-	0.25	0,2119	1,43%
		0.37	0,2119	1,43%	0.5	0,2119	3,11%	0.6	0,2117	1,34%
		0.9	0,2092	0,14%	0.9	0,2095	1,94%	0.9	0,2053	-

Tableau III.8 : la précision moyenne en faisant varier le nombre de documents et λ dans les formules (1), (2), (3).

Nous remarquons dans le tableau précédent que :

- la formule (1) a présenté une amélioration avec tous les nombres de documents qu'on a attribués. Par exemple, avec le nombre de document égal à 5 présente un résultat très satisfaisant (MAP=0.2134) avec un taux d'amélioration de 10.22%. Ceci tend à prouver que, la prise en compte de nombre de document d'expansion améliore les performances de la formule(1).

Le meilleur résultat présenté est : une MAP égale à 0.2207, avec un λ égal à 0.37 présentant un taux d'amélioration égal à 3.71%.

- La formule (2) à eu une amélioration légère avec toutes les valeurs attribuées au nombre de document d'expansion mais les résultats restent inférieurs à ceux obtenus avec la formule (1).

- La formule (3) donne des résultats semblables à ceux des formules (1) et (2).

Le tableau suivant montre la précision moyenne en faisant varier le nombre de document et λ avec les formules (4), (5) et (6).

nbr_doc	terrier	Formule (4)			Formule (5)			Formule (6)		
		λ	MAP	Taux	λ	MAP	Taux	λ	MAP	Taux
5	0.1936	0.1	0,2045	5,63%	0.1	0,2041	5,42%	0.1	0,2011	3,87%
		0.59	0,1934	-	0.44	0,1932	-	0.43	0,1933	-
		0.9	0,1934	-	0.9	0,1934	-	0.9	0,1934	-
10	0.1994	0.1	0,1717	-	0.1	0,2059	3,25%	0.1	0,2039	2,25%
		0.59	0,2028	1,70%	0.44	0,2017	1,15%	0.43	0,2005	0,55%
		0.9	0,1986	-	0.9	0,1986	-	0.9	0,199	-
14	0.2128	0.1	0,1771	-	0.1	0,1985	-	0.1	0,1953	-
		0.59	0,2161	1,55%	0.44	0,2153	1,17%	0.43	0,2164	1,69%
		0.9	0,2125	-	0.9	0,213	0,09%	0.9	0,2127	-
20	0.2049	0.1	0,1844	-	0.1	0,1986	-	0.1	0,1851	-
		0.59	0,2008	-	0.44	0,2021	-	0.43	0,1996	-
		0.9	0,2025	-	0.9	0,2036	-	0.9	0,2017	-
25	0.2055	0.1	0,1786	-	0.1	0,2008	-	0.1	0,1883	-
		0.59	0,2004	-	0.44	0,2051	-	0.43	0,2018	-
		0.9	0,2063	0,38%	0.9	0,2055	0	0.9	0,2056	0,04%
30	0.2089	0.1	0,1791	-	0.1	0,1942	-	0.1	0,1843	-
		0.59	0,2032	-	0.44	0,21	0,52%	0.43	0,2037	-
		0.9	0,2093	0,19%	0.9	0,2099	0,47%	0.9	0,2097	0,38%

Tableau III.9 : la précision moyenne en faisant varier le nombre de document et λ dans les formules (4), (5), (6).

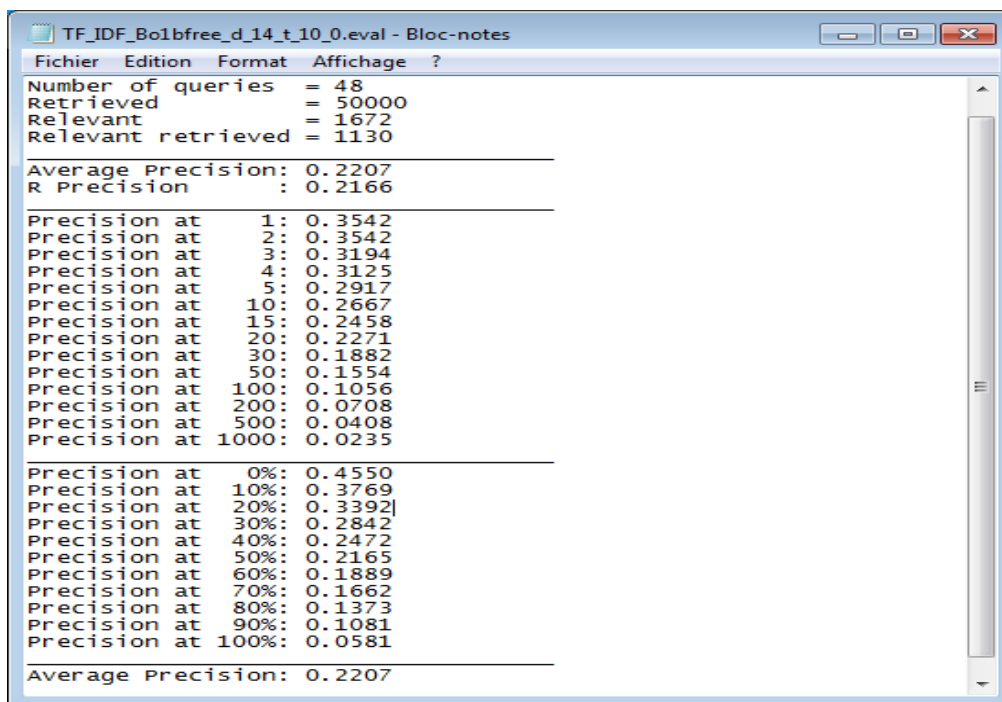
D'après le tableau III.9 nous constatons que les formules qui tiennent compte de la taille d'un document donnent une amélioration avec toutes les valeurs de nombre de document sauf le cas où le nombre de document égal à 20. cependant il est possible d'avoir des résultats plus importants que ceux obtenus en donnant d'autres valeurs à λ (exigence de temps).

❖ **Synthèse sur les résultats obtenus sur la collection AP88**

Sur la base de nos expérimentations sur la collection AP88, on constate que :

- Le nombre de document d'expansion optimal est égal à 14 pour toutes les formules proposées.
- Toutes les formules ont présenté des améliorations.
- La formule optimale pour cette collection est : la formule (1) avec un λ égal à 0.37 et le nombre de document égal à 14. Une amélioration d'ordre 3,7% est constatée.

Nous présentons dans le tableau ci-dessous l'application des paramètres qui ont donné le meilleur résultat avec la collection AP88.



TF_IDF_Bo1bfree_d_14_t_10_0.eval - Bloc-notes		
Fichier	Edition	Format Affichage ?
Number of queries	=	48
Retrieved	=	50000
Relevant	=	1672
Relevant retrieved	=	1130
<hr/>		
Average Precision:		0.2207
R Precision	:	0.2166
<hr/>		
Precision at	1:	0.3542
Precision at	2:	0.3542
Precision at	3:	0.3194
Precision at	4:	0.3125
Precision at	5:	0.2917
Precision at	10:	0.2667
Precision at	15:	0.2458
Precision at	20:	0.2271
Precision at	30:	0.1882
Precision at	50:	0.1554
Precision at	100:	0.1056
Precision at	200:	0.0708
Precision at	500:	0.0408
Precision at	1000:	0.0235
<hr/>		
Precision at	0%:	0.4550
Precision at	10%:	0.3769
Precision at	20%:	0.3392
Precision at	30%:	0.2842
Precision at	40%:	0.2472
Precision at	50%:	0.2165
Precision at	60%:	0.1889
Precision at	70%:	0.1662
Precision at	80%:	0.1373
Precision at	90%:	0.1081
Precision at	100%:	0.0581
<hr/>		
Average Precision:		0.2207

Figure III.5 : Meilleur résultat de l'évaluation de notre approche sur la collection AP88.

IV.2.2. Les résultats de la collection WSJ9092

Afin de trouver le modèle de base avec lequel on comparera notre approche, nous avons varié le nombre de document à utiliser pour l'expansion de la requête sur la collection WSJ9092. Le tableau suivant illustre les résultats obtenus.

Nbr_documents	MAP	Nbr_documents	MAP	Nbr_documents	MAP
1	0.1487	19	0.1673	37	0.1730
2	0.1578	20	0.1670	38	0.1724
3	0.1560	21	0.1700	39	0.1712
4	0.1590	22	0.1722	40	0.1728
5	0.1615	23	0.1718	41	0.1724
6	0.1641	24	0.1732	42	0.1723
7	0.1645	25	0.1743	43	0.1715
8	0.1639	26	0.1710	44	0.1708
9	0.1611	27	0.1688	45	0.1724
10	0.1639	28	0.1693	46	0.1712
11	0.1724	29	0.1699	47	0.1678
12	0.1696	30	0.1695	50	0.1683
13	0.1674	31	0.1683	75	0.1684
14	0.1661	32	0.1748	100	0.1701
15	0.1685	33	0.1737	200	0.1523
16	0.1678	34	0.1727	300	0.1421
17	0.1682	35	0.1720	400	0.1382
18	0.1672	36	0.1710		

Tableau III.10 : la précision moyenne en faisant varier le nombre de document utilisé pour l'expansion dans terrier.

Nous remarquons dans le tableau ci-dessus que le meilleur résultat présenté est donné avec le nombre de document d'expansion égal à 32 qui donne une MAP égale à 0.1748.

Pour cela nous avons utilisé dans notre approche 32 documents pour l'expansion en variant le coefficient λ puis nous comparons les résultats obtenus au résultat de modèle de base (MAP égale à 0.1748).

Le tableau ci-dessus montre les résultats obtenus avec les formules (1), (2) et (3) qui font intervenir le facteur de similarité comparativement avec le modèle de base et cela avec la collection WSJ9092.

λ	La formule(1)		La formule (2)		La formule (3)	
	MAP	Taux	MAP	Taux	MAP	Taux
0.1	0,16	-	0,1664	-	0.1	0,1649
0.2	0,1681	-	0,1692	-	0.2	0,1692
0.3	0,168	-	0,1682	-	0.3	0,1686
0.4	0,169	-	0,1728	-	0.4	0,1701
0.5	0,1691	-	0,1702	-	0.5	0,1695
0.6	0,1695	-	0,1702	-	0.6	0,1701
0.7	0,1693	-	0,1722	-	0.7	0,1696
0.8	0,17	-	0,173	-	0.8	0,1726
0.9	0,1737	-	0,1746	-	0.9	0,1738
0.95	0,1738	-	0,1748	0%	0.95	0,1748
0.97	0,1746	-	0,1748	0%	-----	-----
0.99	0,1748	0%	0,1748	0%	0.99	0,1748
0.995	0,1748	0%	-----	-----	-----	-----

Tableau III.11 : la précision moyenne en faisant varier λ dans les formules (1), (2) et (3).

D'après le tableau III.11 nous constatons que les formules (1), (2) et (3) n'ont pas apporté une amélioration avec toutes les valeurs qu'on a attribuées à λ . Notant que la précision optimale est égale à celle obtenue avec le modèle de base (0.1748).

Le tableau ci-dessus montre les résultats obtenus avec les formules (4), (5) et (6) qui font intervenir le facteur taille de document comparativement avec le modèle de base et cela avec la collection WSJ9092.

λ	La formule (4)		La formule (5)		La formule(6)	
	MAP	Taux	MAP	Taux	MAP	Taux
0.1	0,13	-	0,1564	-	0,1366	-
0.2	0,1431	-	0,1658	-	0,1534	-
0.3	0,1557	-	0,1726	-	0,1614	-
0.4	0,1662	-	0,1715	-	0,1672	-
0.5	0,1631	-	0,1706	-	0,169	-
0.6	0,1682	-	0,1718	-	0,1717	-
0.7	0,1699	-	0,1735	-	0,1712	-
0.8	0,1713	-	0,1743	-	0,1711	-
0.81	-----	-----	0,1743	-	-----	-----
0.82	-----	-----	0,1743	-	-----	-----
0.83	-----	-----	0,1739	-	-----	-----
0.89	-----	-----	0,1741	-	-----	-----
0.9	0,1728	-	0,1741	-	0,1738	-
0.94	0,1739	-	-----	-----	0,1739	-
0.95	0,1745	-	-----	-----	0,1741	-
0.96	0,1741	-	-----	-----	0,1741	-
0.98	-----	-----	-----	-----	0,1738	-

Tableau III.12 : la précision moyenne en faisant varier le λ dans les formules (4), (5) et (6).

Le tableau III.12 nous montre que les formules qui tiennent compte de la taille de document n'ont pas apporté aucune amélioration avec toutes les valeurs qu'on a attribuées à λ par rapport à la précision optimale obtenue avec le modèle de base.

D'après les résultats obtenus précédemment nous déduisant que le nombre de document égal à 32 n'apporte pas d'amélioration pour cela nous avons opté à identifier le nombre de document qui répond à notre objectif en faisant des tests présentés dans les tableaux ci-dessous.

nbr_doc	terrier	Formule (1)			Formule(2)			Formule(3)		
		λ	MAP	Taux	λ	MAP	Taux	λ	MAP	Taux
5	0.1615	0.1	0,1725	6,81%	0.1	0,1766	9,34%	0.1	0,1768	9,47%
		0.5	0,1608	-	0.4	0,1657	2,60%	0.6	0,1632	1,05%
		0.99	0,1615	0	0.95	0,1615	0	0.95	0,1615	0
10	0.1639	0.1	0,1732	5,67%	0.1	0,1768	7,87%	0.1	0,1782	8,72%
		0.5	0,1719	4,88%	0.4	0,1682	2,62%	0.6	0,1674	2,13%
		0.99	0,1639	0	0.95	0,1639	0	0.95	0,1639	0%
15	0.1685	0.1	0,1696	0,65%	0.1	0,1789	6,17%	0.1	0,1789	6,17%
		0.5	0,1733	2,84%	0.4	0,1706	1,24%	0.6	0,167	-
		0.99	0,1685	0	0.95	0,1683	-	0.95	0,1683	-
20	0.1670	0.1	0,1616	-	0.1	0,1741	4,25%	0.1	0,1720	2,99%
		0.5	0,177	5,98%	0.4	0,1754	5,02%	0.6	0,1714	2,63%
		0.99	0,167	0	0.95	0,1671	0,05%	0.95	0,1673	0,17%
25	0.1743	0.1	0,1613	-	0.1	0,1701	-	0.1	0,1683	-
		0.5	0,1724	-	0.4	0,1689	-	0.6	0,1685	-
		0.99	0,1743	0	0.95	0,1715	-	0.95	0,1694	-
32	0.1748	0.1	0,16	-	0.1	0,1664	-	0.1	0,1649	-
		0.5	0,1691	-	0.4	0,1728	-	0.6	0,1701	-
		0.99	0,1748	0	0.95	0,1748	0	0.95	0,1748	0%

Tableau III.13 : la précision moyenne en faisant varier le nombre de document et le λ dans les formules (1), (2) et (3).

D'après le tableau III.13 nous constatons que les formules (1), (2) et (3) donnent une amélioration avec toutes les valeurs de nombre de document sauf avec le nombre de document égal à 25. Cependant il est toujours possible d'avoir des résultats plus importants que ceux obtenus en donnant d'autres valeurs à λ .

Le meilleur résultat présenté est obtenu avec la formule (2) et formule (3) avec un nombre de document égal à 15. (MAP=0.1789) présentant un taux d'amélioration égal à 6.17%.

Le tableau suivant montre la précision moyenne en faisant varier le nombre de document et le λ avec les formules (4), (5) et (6).

nbr_doc	terrier	Formule (4)			Formule(5)			Formule(6)		
		λ	MAP	Taux	λ	MAP	Taux	λ	MAP	Taux
5	0.1615	0.1	0,1375	-	0.1	0,1723	6,68%	0.1	0,1559	-
		0.5	0,1792	10,95	0.5	0,1644	1,79%	0.5	0,1709	5,82%
		0.95	0,161	-	0.8	0,161	-	0.95	0,1617	0,12%
10	0.1639	0.1	0,1378	-	0.1	0,1656	2,53%	0.1	0,143	-
		0.5	0,1669	1,83	0.5	0,1631	-	0.5	0,1655	0,97%
		0.95	0,1668	1,76	0.8	0,1627	-	0.95	0,1624	-
15	0.1685	0.1	0,1248	-	0.1	0,1576	-	0.1	0,1379	-
		0.5	0,1655	-	0.5	0,1696	0,65%	0.5	0,1656	-
		0.95	0,1688	0,17	0.8	0,1688	0,17%	0.95	0,1688	0,17%
20	0.1670	0.1	0,1285	-	0.1	0,1562	-	0.1	0,137	-
		0.5	0,1641	-	0.5	0,1708	2,27%	0.5	0,1695	1,49%
		0.95	0,1692	1,31	0.8	0,1673	0,17%	0.95	0,1667	-
25	0.1743	0.1	0,1378	-	0.1	0,1588	-	0.1	0,132	-
		0.5	0,1725	-	0.5	0,1712	-	0.5	0,1699	-
		0.95	0,1687	-	0.8	0,1702	-	0.95	0,1698	-
32	0.1748	0.1	0,13	-	0.1	0,1564	-	0.1	0,1366	-
		0.5	0,1631	-	0.5	0,1706	-	0.5	0,169	-
		0.95	0,1745	-	0.8	0,1743	-	0.95	0,1741	-

Tableau III.14 : la précision moyenne en faisant varier le nombre de document et le λ (4), (5) et (6).

D'après le tableau III.14 nous constatons que les formules qui tiennent compte de la taille d'un document donnent une amélioration avec toutes les valeurs de nombre de document sauf dans le nombre de document égal à 25 et 32.

La valeur la plus optimale avec ces trois formules est obtenue avec un nombre de document égal à 5. (MAP=0.1792) présentant un taux d'amélioration égal à 10.95%.

❖ Synthèse sur les résultats obtenus sur la collection WSJ9092

Sur la base de nos expérimentations sur la collection WSJ9092, on constate que :

- Le nombre de document d'expansion optimal est égal à 5 pour les formules qui tiennent compte de la taille d'un document et de la similarité entre documents.
- Toutes les formules ont présenté des améliorations par rapport au modèle de base.
- La formule optimale pour cette collection est la formule numéro (4): avec un λ égal à 0.5 et le nombre de document égal à 5.

Nous présentons dans le tableau ci-dessous l'application des paramètres qui ont donné le meilleur résultat avec la collection WSJ9092.

```

TF_IDF_Bo1bfree_d_5_t_10_0.eval - Bloc-notes
Fichier  Edition  Format  Affichage  ?
Number of queries = 45
Retrieved = 50000
Relevant = 1064
Relevant retrieved = 611
Average Precision: 0.1792
R Precision : 0.1793
Precision at 1: 0.3111
Precision at 10: 0.2022
Precision at 100: 0.0640
Precision at 1000: 0.0136
Precision at 0%: 0.4443
Precision at 10%: 0.3710
Precision at 20%: 0.3016
Precision at 30%: 0.2244
Precision at 40%: 0.2019
Precision at 50%: 0.1659
Precision at 60%: 0.1376
Precision at 70%: 0.1129
Precision at 80%: 0.0711
Precision at 90%: 0.0547
Precision at 100%: 0.0422
Average Precision: 0.1792
    
```

Figure III.6 : Meilleur résultat d'évaluation de notre approche pour la collection WSJ9092.

Dans le tableau ci-dessous nous appliquons les paramètres qui ont donnés les meilleurs résultats avec la collection AP88 sur la collection WSJ9092, et vice versa.

La collection AP88						
nbr-doc	λ	Modèle de base	La formule(1)	La formule(2)	La formule(3)	La formule(4)
14	0.37	0.2128	0.2207 (3,71%)			
14	0.25	0.2128			0.2203 (3,52%)	
La collection WSJ9092						
14	0.37	0.1661	0.1765 (6,26%)			
14	0.25	0.1661			0.1764 (6,20%)	
La collection WSJ9092						
15	0.1	0.1685		0.1789 (6,17%)	0.1789 (6,17%)	
5	0.5	0.1615				0.1792 (10,95%)
La collection AP88						
15	0.1	0.2060		0.2151 (4,41%)	0.2073 (0,63%)	
5	0.5	0.1936				0.2025 (4,59%)

Tableau III.15 : la précision moyenne avec les meilleurs paramètres dans les deux collections.

D'après les résultats obtenus dans le tableau ci-dessus nous remarquons que l'application des paramètres qui ont donnés de meilleurs résultats avec la collection AP88 sur la collection WSJ9092 nous retourne de bons résultats, et l'application des paramètres qui ont donnés de meilleurs résultats avec la collection WSJ9092 sur AP88 nous retourne aussi de bons résultats.

Nous pouvons déduire que les meilleurs paramètres obtenus dans les deux collections de tests sont efficace (donnent une amélioration) en les appliquant sur n'importe qu'elle collection.

IV.2.3. Les résultats de formules de combinaison sur la collection AP88

Dans cette section, nous présentons les résultats obtenus par la combinaison de la similarité et la taille d'un document, en s'appuyant sur les valeurs de λ et le nombre de document d'expansion qui ont donné les meilleurs résultats dans les deux collections.

La collection AP88				
nbr-doc	λ	Modèle de base	La formule (7)	La formule (8)
14	0.37	0.2128	0.2061	0.2059
14	0.25	0.2128	0.2053	0.2091
15	0.1	0.2060	0.1908	0.2032
5	0.5	0.1936	0.2029 (4,8%)	0.1878

Tableau III.16 : la précision moyenne en appliquant les formules de combinaison numéro (7) et (8).

Le tableau ci-dessus montre que la formule de combinaison n'apporte pas d'amélioration qu'avec le nombre de document égal à 5 et un λ égal à 0.5 présentant un taux d'amélioration égal à 4.8%.

Pour cela nous avons opté à faire des tests (fixer le nombre de document à 5 et λ à 0.5) en appliquant une nouvelle formule numéro (9) de combinaison en faisant varier le coefficient A qui estime l'importance entre la similarité et la taille d'un document.

La collection AP88				
nbr-doc	λ	A	La formule (9)	Taux d'amélioration
5	0.5	0.08	0,2024	4,54%
		0.1	0,2024	4,54%
		0.3	0,2025	4,59%
		0.5	0,2029	4,80%
		0.7	0,202	4,33%
		0.9	0,2035	5,11%
		0.95	0,2042	5,47%

Tableau III.17 : la précision moyenne en variant le coefficient A dans la formule (9).

Le tableau précédent montre que cette formule a apporté une amélioration par rapport au modèle de base (MAP=0.1936) avec toutes les valeurs attribuées à A, ainsi que par rapport au meilleur résultat obtenu dans le tableau III.16 (MAP=0.2029). Une amélioration d'ordre 5,47% est constatée.

IV.2.4. Les résultats des formules de combinaison sur la collection WSJ9092

Le tableau ci-dessous montre les résultats obtenus avec les formules de combinaison.

La collection WSJ9092				
nbr-doc	λ	terrier	La formule (7)	La formule (8)
14	0.37	0.1661	0.1689 (1,68%)	0.1693 (1,92%)
14	0.25	0.1661	0.1730 (4,15%)	0.1686 (1,5%)
15	0.1	0.1685	0.1317	0.1425
5	0.5	0.1615	0.1783 (10,4%)	0.1659 (2,72%)

Tableau III.18 : la précision moyenne en appliquant les formules de combinaison.

Les résultats du tableau ci-dessus montrent que la combinaison a donné de bons résultats dans tous les tests effectués sauf avec le nombre de document égal à 15. Nous remarquons que la meilleure valeur reste toujours avec le nombre de document égal à 5 et λ égal à 0.5.

Pour cela nous avons procédé comme précédemment pour effectuer les tests suivants en utilisant la formule (9) :

La collection WSJ9092				
nbr-doc	λ	A	La formule (9)	Taux d'amélioration
5	0.5	0.08	0,1792	10,95 %
		0.1	0,1792	10,95%
		0.3	0,1785	10,526%
		0.5	0,1783	10,40%
		0.7	0,1739	7,67 %
		0.9	0,1653	2,35%
		0.95	0,1608	-

Tableau III.19 : la précision moyenne en variant le coefficient A.

Le tableau précédent montre que la formule (9) a apporté une amélioration par rapport au modèle de base (MAP=0.1615) avec toutes les valeurs attribuées à A, ainsi que par rapport au meilleur résultat obtenu dans le tableau III.18 (MAP=0.1783).

❖ Synthèse sur les résultats obtenus avec les formules de combinaison sur les deux collections

Sur la base de nos expérimentations sur les deux collections de tests avec les formules de combinaison on constate que :

- Les deux premières formules de combinaison n'ont pas amélioré les résultats dans la plus part des cas sur la collection AP88, par contre elles ont donné une amélioration légère sur la collection WSJ9092.
- La formule de combinaison numéro (9) a présenté une amélioration par rapport au modèle de base, ainsi que par rapport aux deux premières formules de combinaison.

Malgré l'amélioration donnée avec la formule de similarité et celle de la taille de document, la combinaison de ces deux facteurs (la similarité et la taille d'un document) n'a pas donné de bons résultats.

V. Conclusion

Nous avons proposé dans ce chapitre plusieurs formules de classement de document pertinent, qui ont été testées et expérimentées sur deux collections de tests en utilisant la plateforme Terrier. Les résultats des tests, ont révélé que la prise en compte de la taille d'un document et la similarité entre documents donne de meilleures performances. Et puisque l'expérimentation a été réalisée sur deux collections de tests de grandeur nature, on peut déduire que le facteur de similarité est une bonne évidence pour réordonner les documents feedback.

Introduction générale

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 système de recherche d'information 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.

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

Notre travail se situe dans le contexte de la reformulation de requête plus particulièrement dans le cadre de réordonnancement des documents retournés avec la première requête formulée par l'utilisateur, en utilisant deux facteurs (la similarité et la taille du document) on considérant toujours que le score des documents de la première recherche comme une évidence. Ces documents ainsi réordonnés vont être utilisés pour choisir les termes d'expansion.

L'idée de l'approche proposée et implémentée est de combiner les résultats de la première recherche avec le facteur de similarité d'une part et avec la taille du document d'une autre part. Pour la réalisation et l'évaluation de nos expérimentations nous utilisons la plateforme de RI Terrier (détaillé en annexe).

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

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

Introduction Générale

Le second chapitre traite la reformulation (expansion) de requête et ses différentes techniques ainsi que les stratégies qui sont développés dans les différents modèles.

Le troisième chapitre présente notre approche, les outils utilisés pour son implémentation, ainsi que les résultats d'évaluation effectué sur deux collections TREC (AP88, WSJ 90-92).

Liste des tableaux

I.1. Exemple de fichier direct	14
I.2. Exemple de fichier inverse	14
I.3. Les principales mesures de similarité utilisées	22
II.1. Comparaison des techniques de reformulation de requête.....	52
III.1. Le classement des documents dans la première recherche.....	67
III.2. la liste des documents réordonnés après l'implémentation de notre approche.....	67
III.3. Le réordonnement des documents après l'expansion.....	67
III.4. Statistiques sur les collections de test et les Topics utilisé.....	74
III.5. la précision moyenne en faisant varier le nombre de document utilisé pour l'expansion dans terrier.....	75
III.6. la précision moyenne en faisant varier λ dans les formules (1), (2), (3).....	76
III.7. la précision moyenne en faisant varier λ dans les formules (4), (5), (6).....	77
III.8. la précision moyenne en faisant varier le nombre de document et λ dans les formules (1), (2), (3).....	79
III.9. la précision moyenne en faisant varier le nombre de document et λ dans les formules (4), (5), (6).....	80
III.10. la précision moyenne en faisant varier le nombre de document utilisé pour l'expansion dans terrier.....	82
III.11. la précision moyenne en faisant varier λ dans les formules (1), (2) et (3).....	83
III.12. la précision moyenne en faisant varier λ dans les formules (4), (5) et (6).....	84
III.13. la précision moyenne en faisant varier le nombre de document et λ dans les formules (1), (2) et (3).....	85

III.14. la précision moyenne en faisant varier le nombre de document et λ (4), (5) et (6).....	86
III.15. la précision moyenne avec les meilleurs paramètres dans les deux collections.....	87
III.16. la précision moyenne en appliquant les formules de combinaison.....	88
III.17. la précision moyenne en variant le coefficient A.....	89
III.18. la précision moyenne en appliquant les formules de combinaison.....	89
III.19. la précision moyenne en variant le coefficient A.....	90

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 : Conduite de Projet Informatique*

Thème

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

Proposé et dirigé par :

M^{er} A. Hammache

Réalisé par :

M^{elle} Ouiza Ould Youcef.

M^{elle} Sabrina Taleb.

Promotion 2011/2012



Remerciement

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 tenons à exprimer notre gratitude aux membres du jury pour l'honneur qu'ils nous ont fait en acceptant de juger notre travail.

Nous tenons également à remercier M^{elle} Ait Adda.S pour son aide et sa gentillesse.

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 Dédicaces

Je dédie ce modeste travail

*À ceux qui m'ont tout donné sans rien attendre en retour mis
à part ma réussite, à ceux qui m'ont appris à aller au bout de mes
ambitions, à ceux qui ont toujours cru en moi : à mes très chers
parents.*

À mes sœurs Malika, Fatima, Dalila, Chabha, Louzou.

À mes frères Malik, Mahdi.

À mon neveu Iliane

À mes beaux frères.

À tout mes proches.

À tout mes amis(es).

À ma chère collègue Ouiza Ould Youcef et à toute sa famille.





Dédicaces Dédicaces

Je dédie ce modeste travail:

*À ceux qui m'ont donné la vie et fait de moi ce que je suis mes très chers
parents.*

*À mes joyaux précieux qui illuminent ma vie, mes adorables frères Boussad
et le petit charmant Riad,*

*À ma sœur Sabrina qui me conseille et m'encourage toujours sans oublier ma
petite sœur et adorable Houa.*

À ma chère collègue Sabrina Taleb et à toute sa famille.

*Je ne peux enfin clôturer ces dédicaces sans le dédier du fond de mon cœur à
tous mes très chères amis(es) à celui qui sera le père de mes enfants et à tous mes
proches.*



Sommaire :

Introduction Générale.....	1
-----------------------------------	----------

Chapitre I:La Recherche d'Information

Introduction	3
I.1. La recherche d'information (RI)	3
I.1.1. Définition	3
I.1.2. notions de base	3
I.2. Le système de recherche d'information	5
I.2.1. Définition	5
I.2.2. Fonctionnement	5
I.3. Processus de Recherche d'Information	5
I.3.1. Le processus d'indexation	6
I.3.1.1. Les étapes du processus d'indexation.....	7
I.3.2. Le processus de recherche.....	15
I.3.3. le processus de reformulation des requêtes	15
I.4. Modèles de recherche d'information	16
I.4.1. Les modèles basés sur la théorie des ensembles	17
I.4.2. Les modèles algébriques	20
I.4.3. Les modèles probabilistes	27
I.5. Evaluation des Systèmes de RI	30
I.5.1. Mesures d'évaluation des systèmes	34
Conclusion.....	38

Chapitre II: La Reformulation de la Requête

Introduction.....	39
II.1. Reformulation de requêtes.....	39
II.2. Les techniques de reformulations de requête.....	40
II.2.1. Analyse par le contexte global.....	41
II.2.2. Analyse par le contexte local.....	43
II.3. La réinjection de pertinence dans les modèles de recherche d'information.....	47
II.3.1. Réinjection de pertinence dans le modèle vectoriel.....	48
II.3.2. Réinjection de pertinence dans le modèle probabiliste.....	50
II.4. Analyse des approches de reformulations de requêtes.....	51

II.5. Paramètres de performance d'un SRI.....	53
II.6. Avantages et risques d'expansion de requête.....	57
Conclusion.....	58

Chapitre III: Evaluation et Expérimentation

Introduction.....	59
I. Architecture générale de notre approche.....	59
I.1. Présentation de notre approche de sélection des documents utilisés pour l'expansion de la requête.....	60
I.2. Terminologie et notation.....	65
I.3. Exemple illustratif.....	67
III. L'Environnement Technique.....	68
III.1. Présentation de la plate forme Terrier.....	68
III.1.1. Architecture de Terrier.....	68
III.2. Le langage java.....	72
III.3. Présentation de NetBeans.....	73
IV. Résultats et Expérimentation.....	74
IV.1. Collection de test.....	74
IV.2. Evaluation et résultats des formules de classement des documents pertinent.....	74
IV.2.1. Les résultats obtenus sur la collection AP88.....	74
IV.2.2. Les résultats de la collection WSJ9092.....	81
IV.2.3. Les résultats de formules de combinaison sur la collection AP88.....	88
IV.2.4. Les résultats des formules de combinaison sur la collection WSJ9092.....	89
V. Conclusion.....	90
Conclusion & Perspectives.....	91

Bibliographie

Bibliographie

Annexes

Annexe: La plate forme terrier.

Liste des figures

I.1. Processus de Recherche d'Information.....	6
I.2. Étapes du Processus d'indexation automatique	7
I.3. La correspondance entre l'informativité et la fréquence	10
I.4. Taxonomie des Modèles de Recherche d'Information	17
I.5. Répartition des documents face à une requête	35
I.6. Courbe de Rappel/Précision	36
II.1. La reformulation de requête dans SRI	40
II.2. Processus de fonctionnement du RF	47
II.3. Expansion de requête dans des moteurs de recherche	58
III.1. Présentation de notre approche implémentée dans terrier	60
III.2. Vue d'ensemble de l'architecture Terrier.....	69
III. 3. Architecture d'Indexation dans Terrier	69
III. 4. Recherche dans Terrier.....	71
III. 5. Meilleur résultat de l'évaluation de notre approche pour la collection AP88.....	81
III. 6. Meilleur résultat d'évaluation de notre approche pour la collection WSJ9092	87

Conclusion générale

Conclusion général

Conclusion & Perspective

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

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

Pour optimiser les résultats retournés on a proposé dans notre approche d'améliorer le processus de sélection de documents utilisés pour l'expansion en se basant sur deux facteurs : le premier porte sur la combinaison de la similarité des documents avec le score initial (le score des documents obtenus dans la première recherche) et le second porte sur la combinaison de la taille des documents avec le score initial.

Dans notre approche nous avons utilisé le facteur de similarité calculé avec la mesure de cosinus proposé dans le modèle vectoriel, par ailleurs plusieurs formules sont dérivées par la combinaison de similarité, taille document et le score initial.

Pour valider les formules proposées, nous les avons évalué en utilisant deux collections de test TREC (AP88, WSJ9092), sous de la plateforme de RI "*Terrier*" qui est un système de recherche robuste et efficace.

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

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

Les perspectives envisageables à notre travail portent principalement sur une étude plus approfondie des méthodes de sélection de termes à ajouter à la requête. Pour cela nous

Conclusion général

proposons d'appliquer la caractéristique de similarité pour la sélection de termes à ajouter à la requête puisque elle a donné de bons résultats pour la sélection de document.

Bibliographie

Bibliographie

[Baziz, 05]: Mustapha Baziz, "Indexation conceptuelle guidée par ontologie pour la recherche d'information" thèse de Doctorat à l'Université de Paul Sabatier, 2005.

[Benaouicha, 09] : Mohamed Benaouicha, "Une approche algébrique pour la recherche d'information structurée" thèse de Doctorat à l'Université de Paul Sabatier, Toulouse, 2009.

[Boubekeur, 08]: Fatiha Boubekeur, "Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets", thèse de Doctorat à l'Université de Paul Sabatier, 2008.

[Boughanem, 95] : M.Boughanem. "Un système de Recherche d'Informations Orienté Objet Basé sur l'Approche Connexionniste", Rapport de Recherche, Toulouse (France), décembre 1995.

[Boughnem, 97]: M.Boughanem, C. SOULE-DUPUY. *Query modification based on Relevance Back-propagation*, 5Pème Conférence internationale RIAO Recherche d'Information Assistée par Ordinateur. 1997.

[Boughanem et al., 99]: Mohand Boughanem, Claude Chrisment, C. Soulé-Dupuy: Query Modification Based on Relevance Back-Propagation in an Ad hoc Environment. Inf. Process. Manage. 35(2): 121-139 (1999).

[Buckley & al, 94] : C. Buckley, G. Salton & J. Allan : *The Effect of Adding*

Information in a Relevance Feedback Environment, Conference on Research and

Development in Information Retrieval (SIGIR), 1994

[BYRN, 99]: R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN :0-201-39829-X.

[Croft & Xu, 95]: J. Xu & W.B. Croft : *Query Expansion Using Local and Global Document Analysis*. In Proc. ACM SIGIR Annual Conference on Research and Development, Zurich, 1995

[Cormack & al, 99]: G. Cormack, C.R. Palme, M.V Biesbrouk & C.L.A. Clarck, "Deriving Very Short Queries for High Precision and Recall", In Proceedings of the 7th Text Retrieval Conference TREC7, July 1999.

[Croft & Harper, 79]: Croft, W.B., Harper D.J. "Using probabilistic models of document retrieval without relevance information". *Journal of Documentation* 35 4 (1979), p. 285–295.

[Dunning, 93]: Dunning, T. (1993). "Accurate methods for the statistics of surprise and coincidence". *Computational Linguistics*, pages 61–74.

[FAQ NetBeans, 09]: <http://www.developpez.com>FAQ, " NetBeans", Date de publication 14/10/2006, Dernière mise à jour 17/05/2009.

[Fellag, 06] : Samia Fellag née Berchiche, "Recherche d'information dans les documents semi-structurés XML", thèse de magistère à l'université Mouloud Mammeri, Tizi Ouzou, 2006.

[Fiana&Oren, 00]: Fiana Raiber, &Oren Kurland, "On Identifying Representative Relevant Documents" Faculty of Industrial Engineering and Management

[Groussard, 00] : Thierry Groussard, " java 6 les fondamentaux du langage java", ENI editions, 2009.

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

[Harman, 92]: Donna Harman, "Relevance Feedback and Other Query Modification Techniques", *Information Retrieval: Data Structures & Algorithms* 1992: 241-263.

[Hlaoua, 07] : Lobna Hlaoua, "Reformulation de Requêtes par Réinjection de Pertinence dans les Documents Semi-Structurés", thèse de doctorat à l'Université Paul Sabatier, Toulouse, 2007.

[Kraaij, 2004]: Kraaij, W. (2004). "Variations on Language Modeling for Information Retrieval". Phd thesis, University of Twente.

[Léon Bouraoui, 10] : Jean-Léon Bouraoui, Emilie Guimier de Neef, Benoît Gaillard, Malek Boualem, Olivier Collin, "Expansion sémantique de requêtes", Mercredi 31 mars 2010.

[Lynda Tamine, 00]: Lynda Tamine, "Optimisation de requêtes dans un système de recherche d'information " thèse de Doctorat à l'Université de Paul Sabatier, 2000.

[Manning and Schütze, 99]: Manning, C. and Schütze, H. (1999). "*Foundations of Statistical Natural Language Processing*", chapter 16: Text Categorization, pages 575–608. The MIT Press, Cambridge, US.

[Mataoui, 07] : M'Hamed Mataoui, " thème : reformulation de requête dans les systèmes de recherche d'information dans les documents XML ", thèse de magister soutenu le 29-04-2007, université M'hmed BOUGARA de Boumerdes.

[Moreau & Claveau, 06] : "Extension de requêtes par relations morphologiques acquises automatiquement"* IRISA – CNRS Campus universitaire de Beaulieu Avenue du Général Leclerc 35042 Rennes cedex, France, {Fabienne.Moreau,Vincent.Claveau@irisa.fr}, 2006.

[Nie, 04] : Jian-Yun Nie, "Cours Hiver 2004, Module recherche d'information département d'informatique et de recherche opérationnelle", Université de Montréal, Hiver 2004.

[Nongdo, 08] : Nongdo Désiré Yawbom Kompaoré, "Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes : vers un processus de RI adaptatif" thèse de Doctorat à l'Université de Paul Sabatier, 2008.

[Ounis, 07]: "Combining fields for query expansion and adaptive query expansion", Ben He - Iadh Ounis, Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, United Kingdom, Available online 26 January 2007.

[Ponte, 98]: J.M. Ponte, "*A Language Modelling Approach to Information Retrieval*", Doctor of philosophy tgesis, University of Massachusetts, September 1998

[Ponte et Croft, 98] : Ponte, J. et Croft, W. (1998). "A language modelling approach to information retrieval". In Proc. of the 21st Int. ACM SIGIR, Conference on research and development in information retrieval, pages 40{48. ACM Press.

[Ribeiro, 99]: R. A. Ribeiro-Neto. *Modern Information Retrieval*. New York : ACM Press ; Harlow England : Addison-Wesley, cop., 1999.

[Robertson & al, 95]: S.E Robertson, S.E. Walker & M.M Hnacock-Beaulieu: *Large Test Collection Experiments on an Operational Interactive System : Okapi at TREC*, in IP&M, pp 260-345, 1995

[Rocchio, 71]: J.J. Rocchio. "*Relevance feddback in Information Retrieval*", in *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall Series in Automatic Computation, 1971.

[Robertson and Sparck-Jones, 97]: Robertson, S. and Sparck-Jones, K. (1997). Simple, proven approaches to text retrieval. Tech. rep. tr356, Computer Laboratory University of Cambridge.

[Salton, 89]: Salton, G., "Automatic Text Processing", Addison Wesley, 1989.

[Salton & Buckley, 90]: G.Salton & C.Buckley : *Improving Retrieval Performance By Relevance Feedback*, Journal of The American Society for Information Science, Vol. 41, N°4, pp 288-297, 1990.

[Salton, 90]: G. Salton & C. Buckley. *Improving Retrieval Performance By Relevance Feedback*, Journal of Thé American Society for Information Science. 1990. 41(4): 288-297.

[Tebri, 04]: Hamid Tebri, "Formalisation et spécification d'un système de filtrage incrémental d'information", thèse de Doctorat à l'Université de Paul Sabatier, 2004.

[Urfist, 04] :Alexandre SERRES, ''Cours de problématique générale de la recherche d'information '' , URFIST Bretagne pays de Loire, 2002.

[Turtle & Croft, 91]: Turtle H.R, & Croft W.B, "Evaluation of an Inference Network-Based Retrieval Model", ACM Transactions On Information Systems, 9(3):187{222, 1991.

[Voorhees, 93] : E. Voorhees, "Using WordNet to Disambiguate Word Senses for Text Retrieval", Proceedings of the 16th Annual Conference on Research and Development in Information Retrieval, SIGIR'93, Pittsburgh, PA, 1993.

[Wong et al., 85]: Wong, S., Ziarko, W., and Wong, P. (1985). Generalized vector space model information retrieval. In Proceedings of the 8th annual international ACM SIGIR Conference on Research and development in Information Retrieval, pages 18–25. ACM.

[Zadeh, 65]: Zadeh, L. (1965). Fuzzy sets. Information and control 8, pages 338–353.

[Zipf, 49]: Zipf, G. (1949). Human Behaviour and the Principle of Least Effort. 1st edition, Addison Wesley.

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 effectives méthodes de recherche pour les collections de documents, nouvelle combinaison et idées de découpage-bord (cutting-edge) de théorie probabiliste, analyse statistique, et techniques de compression de données.

Ceci a mené au développement de diverses approches de recherche en utilisant un nouveau et un fort cadre probabiliste pour RI, dans le but pratique de la combinaison efficace et effective de diverses sources pour augmenter la performance de recherche.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requêtes 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 indexed unit).

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

2. Installation de Terrier :

Pour pouvoir utiliser terrier il est nécessaire d'installer une JRE (1.5.0 ou plus). La JRE ou la JDK peuvent être téléchargé sur le site de Java. Puis aller sur le site de Terrier pour le télécharger <http://WWW.terrier.org>

Ensuite dézipper la copie téléchargé dans le répertoire que vous souhaitez installer Terrier

3. La structure des répertoires de terrier :

Les répertoires de Terrier sont structurés comme suit :

- bin\ : contient les scripts nécessaires pour démarrer Terrier.
- doc\ : contient la documentation relative à Terrier.
- etc\ : contient les fichiers de configuration de Terrier.
- lib\ : contient les classes compilées de Terrier et les différentes librairies externes utilisées par Terrier.
- share\ : contient la liste des mots vides (stop word list).
- src\ : contient le code source de Terrier.

Terrier 2.1

- var/index : contient les structures de données : fichier inverse, fichier lexicon, index direct, document index.
- var/result : contient les résultats de la recherche.
- Licenses/ : contient les informations sur la licence des différents composants inclus dans Terrier.

4. Les applications de Terrier :

Terrier offre trois applications :

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.

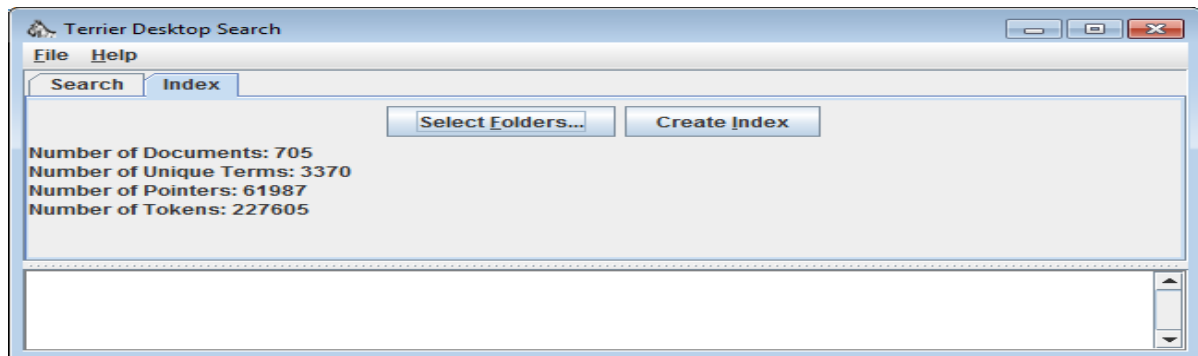


Figure1 : Présentation de l'interface Desktop Terrier.

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'utilisation de la classe *Indexer*.

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

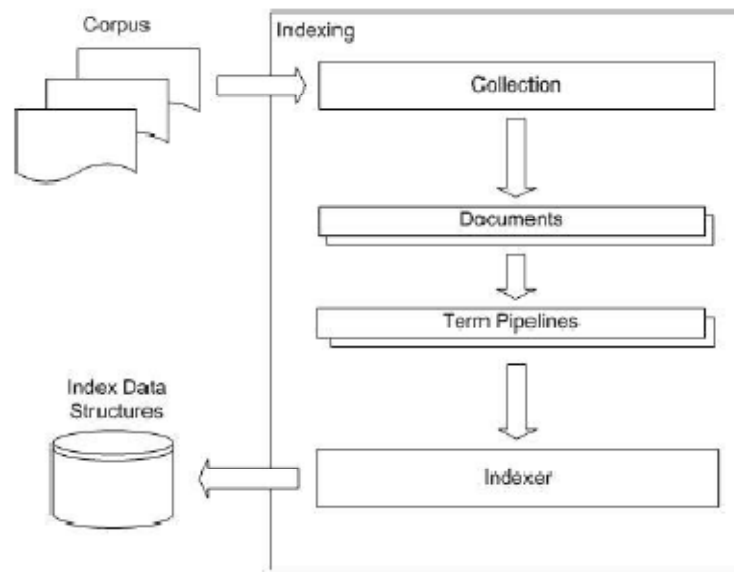


Figure 2 : Présentation du processus d'indexation dans Terrier.

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 sources de données (nouveau corpus) et cela en ajoutant une nouvelle classe java qui implémente cette interface, dans notre cas nous avons utilisé la classe `TRECCollection` qui permet de traiter les collections de type TREC (à savoir AP88 et WSJ9092). Elle permet de parcourir un ensemble de document et de renvoyer un objet document en faisant appel à sa méthode `nextDocument()`.

Terrier offre plusieurs classes qui implémentent cette interface telle que `SimpleFileCollection`, `TRECUTFCollection`, `SimpleXMLCollection`, `TRECCollection`.

5.2 Document :

C'est une interface qui se trouve dans le package `uk.ac.gla.terrier.indexing`. Cette composante englobe le concept de Document. Les classes qui implémentent cette interface s'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 TermPipeline :

C'est une interface qui se trouve dans le package `uk.ac.gla.terrier.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 données appropriée.

Terrier offre deux types d'indexeur : `BasickIndexer`, `BlockIndexer`.

5.5 Les structures d'Index :

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. Term Frequency : fréquence globale dans la collection. Byte offset in inverted file : pour avoir la position dans le fichier inverse. Bite offset in inverted file : pour avoir la position dans le fichier inverse.
Inverted Index	Document id : identificateur du document d'appariement.

Terrier 2.1

	<p>Term Frequency : la fréquence du terme dans le document.</p> <p>Fields (# of fields bits) : Les champs dans lesquels le terme apparaît sont codés en utilisant un bit Set.</p> <p>Block Frequency : fréquence dans une fenêtre.</p> <p>[Block id] : l'identificateur du bloc dans le document ou le terme apparaît.</p>
Document Index	<p>Document id : identificateur de document.</p> <p>Document Length : la longueur de document.</p> <p>Document Number : numéro de document.</p> <p>Byte offset in direct file : pour avoir la position dans le fichier direct.</p> <p>Bit offset in direct file : pour avoir la position dans le fichier direct.</p>
Direct Index	<p>Term id : identificateur du terme.</p> <p>Term frequency : la fréquence du terme.</p> <p>Fields (# of fields bits)</p> <p>Block frequency : fréquence dans une fenêtre.</p> <p>[Block id] : identification du bloc.</p>

Tableau 1 : présentation des structures d'Index.

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

6. L'API de recherche de Terrier :

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

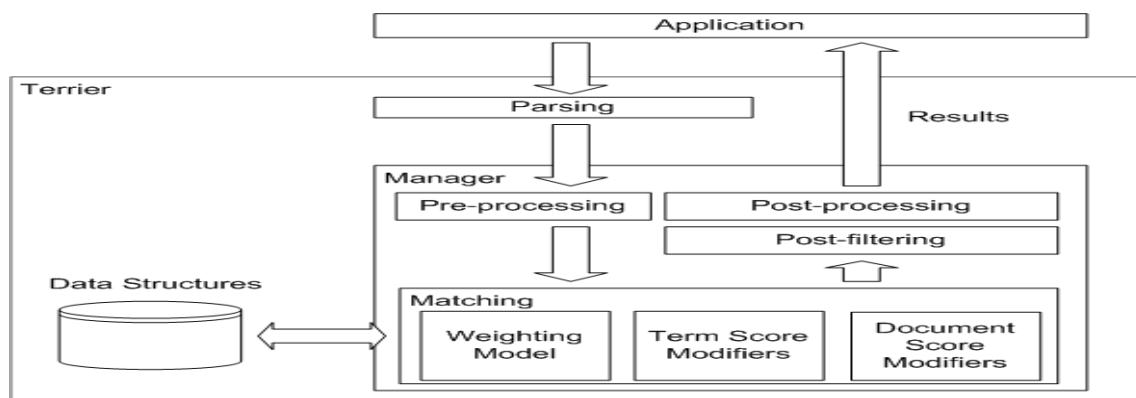


Figure 3 : Présentation du processus de recherche de Terrier.

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 attribut un score au document qui respecte la requête.

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

6.2 Manager :

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

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

Post-processing

Après l'application de *TermScoreModifiers* ou *DocumentScoreModifiers*, l'ensemble de documents recherchés (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, où une série de filtres peut enlever les documents déjà recherchés, qui ne satisfont pas une condition donnée. Par exemple, dans le contexte d'un moteur de recherche Web, un *Post filtre* peut être utilisé pour réduire le nombre de documents recherchés depuis le même site Web, afin d'augmenter la diversité dans les résultats.

7. Modification de Terrier :

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

8. Utilisation de Batch (TREC) Terrier :

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

8.1 Indexation :

1. Aller dans le répertoire où Terrier est installé en 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 en utilisant l'option suivante :

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

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

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

Remarque :

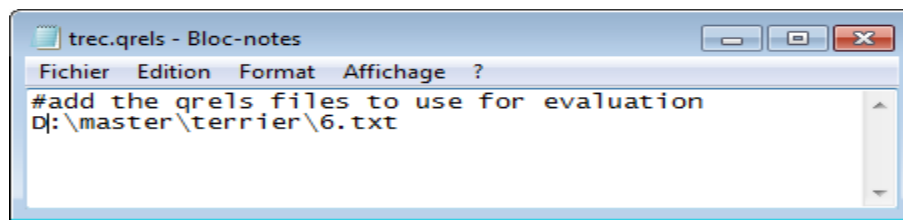
Pour indexer une collection autre qu'une Collection TREC, il est nécessaire d'apporter quelques modifications au fichier *terrier.properties*.

8.2 Recherche :

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

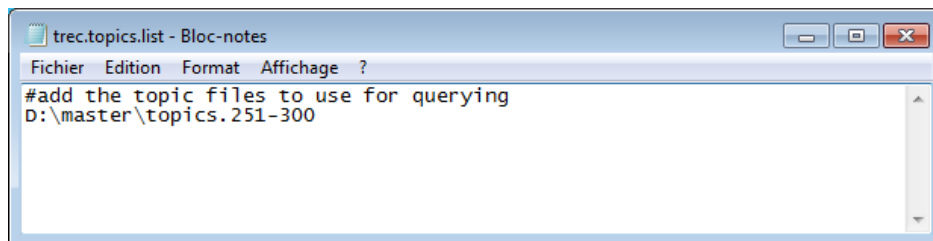
1. Spécification du fichier de jugement de pertinence (dans notre cas *6.txt*) dans le fichier `etc\trec.qrls`.

Terrier 2.1



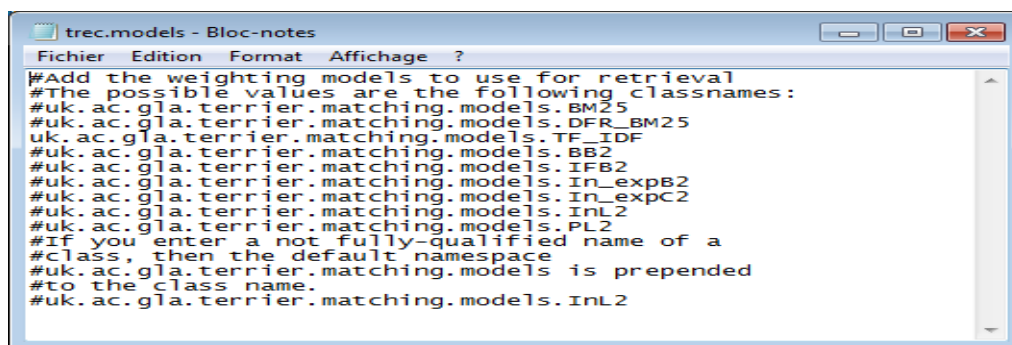
Tel que *6.txt* est le fichier de jugement de pertinence.

2. Spécification du fichier contenant les requêtes (topics.251-300) dans le fichier etc\terrec.topics.list.



3. Spécification du modèle de pondération à utiliser dans le fichier etc\terrec.models.

Pour cela il suffit de décocher le modèle choisit 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\terrec_terrier -r`

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

`.\bin\terrec_terrier -e`