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

MEMOIRE

En vue de l'obtention du Diplôme de Master2 en informatique

Thème

Expansion de requêtes de l'utilisateur à base des termes composés

Dirigé et proposé par :

Mr HAMMACHE Arezki

Réalisé par :

Melle ARHAB Karima
Melle GUEZOUI Faroudja

Promotion 2011/2012

Remerciements

Nous tenons à exprimer notre vive reconnaissance et notre très grande considération à M. Arezki HAMMACHA pour avoir accepté de diriger notre travail et pour sa disponibilité tout au long de l'année, sa patience et sa compréhension. Nous avons toujours travaillé avec lui dans un cadre professionnel avec une bonne humeur. C'était vraiment un plaisir de travailler avec lui.

Nous remercions aussi les membres du jury pour l'attention qu'ils ont porté à la lecture de ce mémoire

Introduction générale

Chapitre I : état de l'art sur la recherche d'information

Chapitre II : reformulation de requête de l'utilisateur et modèle de langage

Chapitre III: Approche d'expansion de la requête basée sur les termes composés

Chapitre IV: Expérimentations et évaluations

Conclusion générale

Table des figures

Liste des tableaux

Introduction générale

L’apparition et la popularisation des ordinateurs, des documents électroniques, de différents types de support de stockages pour un coût qui ne cesse pas de diminuer, ont profondément bouleversé le monde de l’informatique documentaire en favorisant la multiplicité des ressources documentaires grand public.

La recherche d'information (RI) est un vaste domaine d'étude apparu dans les années 60. L'avènement d'Internet et plus particulièrement du Web a conduit à révéler la RI au grand jour, notamment par le biais des moteurs de recherche. La profusion de données numériques disponibles a rendu indispensables des moyens de recherche performants et automatiques, permettant à tout un chacun de trouver une information précise. La RI a alors évolué vers des tâches de plus en plus nombreuses et diversifiées.

Les systèmes de RI, quels que soient leurs objectifs, la nature ou la provenance de l'information manipulée, tendent en fait vers le même but : établir une correspondance entre l'information disponible et celle recherchée par l'utilisateur. L’un des problèmes-clés des systèmes de recherche d’informations (SRI) est la définition d’une fonction de correspondance entre la représentation du contenu sémantique des documents et la requête de l’utilisateur. Cette fonction doit modéliser la *pertinence* d’un document pour l’utilisateur. De fait, il existe deux formes de pertinence : la *pertinence système* et la *pertinence utilisateur*. Améliorer la qualité d’un système de recherche d’information (SRI) consiste donc à réduire la distance existant entre la *pertinence utilisateur* et la *pertinence système.* La stratégie traditionnelle consiste à représenter la requête et le document par un vecteur selon les termes observés, et à mesurer la similarité entre eux.

La mise en œuvre de processus de recherche d’information passe par une spécification d’un modèle de RI qui sont basés sur un processus de mise en correspondance entre requêtes utilisateurs et documents de la collection. De nombreux modèles sont développés dans la littérature dont l’objectif commun est de satisfaire au mieux les besoins de l’utilisateur. Chacun de ces modèles se différencie par sa manière de représenter les documents et la requête, et de les mettre en correspondance. L’inconvénient majeur des modèles classiques, réside dans le fait qu’ils induisent une manipulation de concepts de manière indépendante. Contrairement aux autres modèles de RI, les modèles de langue se sont distingués par leur efficacité et leur fondement mathématique solide, (Ponteet al.,1998), (Song et al.,1998). Les modèles de langue ne modélisent pas directement la notion de pertinence d’un document face

à une requête. La pertinence d'un document vis-à-vis d'une requête est vue comme la probabilité que la requête soit générée par le modèle de langue du document appelée aussi *score de pertinence*. Dans notre travail nous utilisons ce modèle.

La quantité d'information en ligne croît très rapidement, ainsi que le nombre de langues dans lesquelles ces contenus sont disponibles. Des traitements spécifiques s'avèrent donc nécessaires pour préciser le sens de certaines requêtes, ou au contraire pour en élargir la portée, *l'Expansion de Requêtes* (ER), par exemple, permet, en ajoutant des mots aux requêtes afin d'optimiser leur mise en relation avec les documents. La première approche suivie pour améliorer les performances des systèmes a été la reformulation de la requête qu'est une phase importante dans les systèmes de recherche d'information. Elle se base sur le principe que l'utilisateur n'est souvent pas capable de formuler ses besoins en informations et consiste à récrire la requête de l'utilisateur selon les informations retrouvées par la requête initiale. De manière générale, ceci consiste, dans le cas notamment de la réinjection de la pertinence, d'extraire à partir des premiers documents retournés par le système, les mots clés importants puis les rajouter à la requête initiale.

L'objectif de ce projet est de proposer une approche d'expansion de requête d'utilisateur à base des termes composés qui se fonde sur l'hypothèse qu'un mot composé est moins ambigu qu'un mot isolé. Dans ce travail nous avons tenté d'apporter des réponses aux différentes questions proposées, à savoir : comment effectuer une expansion de requête à base des termes composés ? Comment extraire les meilleurs termes composés à partir des N premiers documents pertinents retournés par la première recherche? Quels poids doit-on assigner à ces différents termes ajoutés?.

Nous présentons dans le premier chapitre, un état de l'art des systèmes de recherche d'information en montrant les différents mécanismes sous-jacents à la conception de ces systèmes (appariement, indexation, filtrage, reformulation, etc…), nous détaillons les principes de base de ces modèles et pressentons leurs limites.

Nous décrivons dans le second chapitre l'approche d'expansion de requête en présentant les différentes sources utilisées (les documents résultants de la première recherche, des ressources externes telles que les ontologies, les thésaurus et la relation de cooccurrence entre termes dans la collection), ensuite nous détaillons les approches d'expansion de requêtes utilisées dans les modèles de langage.

Dans le troisième chapitre, nous présentons la partie contribution de ce mémoire en décrivant une nouvelle approche d'expansion de requête basée sur les termes composés en

utilisant la relation de cooccurrence entre termes. Cette technique est modélisée dans le cadre de modèle de langage.

Le quatrième chapitre présente globalement l'expérimentation de notre approche dans la plateforme de recherche d'information Terrier et à tester sur la collection AP88, et les résultats obtenus dans le but de valider notre approche d'expansion de requête à base des termes composés. Cette évaluation a pour but de mesurer l'efficacité de cette approche proposé et estimer son l'impact sur les résultats de la recherche.

Enfin, nous concluons en proposant une perspective de recherche permettant d'améliorer notre approche, une annexe pour décrire le système de recherche d'information Terrier, utilisé pour la réalisation de nos expérimentations.

Chapitre I :

La recherche d'information

1. Introduction

Le domaine de recherche d'information remonte peu après l'invention des ordinateurs. Comme plusieurs autres domaines informatiques, les pionniers de l'époque étaient enthousiastes à utiliser l'ordinateur pour automatiser la recherche des informations, qui dépassaient la capacité humaine. On peut aujourd'hui dire que la recherche d'information est un champ transdisciplinaire, qui peut être étudié par plusieurs disciplines, approche qui devrait permettre de trouver des solutions pour améliorer son efficacité.

Ce chapitre commence par présenter les concepts de base de la Recherche d'Information (RI), avant de décrire l'architecture générale d'un système de recherche d'information. Par la suite, une description des principaux modèles de RI qui servent à formaliser le processus de recherche sont présentés, en fin les mesures et les collections de test utilisés pour évaluer les systèmes de recherche d'information sont exposés.

2. La recherche d'information

La Recherche d'Information (RI) [Van Rijsbergen., 77] [Grossman et al., 98] [Salton, 71] est traditionnellement définie comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur.

Les systèmes de recherche d'information (SRI) [Baziz M., 05], servent d'interface entre une source (collection) contenant des quantités considérables de documents et des utilisateurs cherchant, via des requêtes, des informations susceptibles de se trouver dans cette collection. Les SRI intègrent un ensemble de techniques permettant de sélectionner ces informations. Elles peuvent être résumées en quatre fonctions, qui sont le stockage de l'information, l'organisation de ces informations, la recherche d'informations en réponse à des requêtes utilisateurs et la restitution des informations pertinentes pour ces requêtes. La dernière fonction est celle qui est visible pour l'utilisateur.

3. Bref historique de la RI :

La RI n'est pas un domaine récent. Il date des années 1940, dès la naissance des ordinateurs. Au début, la RI se concentrait sur les applications dans des bibliothèques, d'où aussi le nom "automatisation de bibliothèques". Depuis le début de ces études, la notion de

pertinence a toujours été un sujet de recherche. Dans les années 1950, on commençait de petites expérimentations en utilisant des petites collections de documents. Dans les années 1960 et 1970, des expérimentations plus larges ont été menées, et on a développé une méthodologie d'évaluation du système qui est aussi utilisée maintenant dans d'autres domaines.

Les années 1980 ont été influencées par le développement de l'intelligence artificielle. Ainsi, on tentait d'intégrer des techniques de l'IA en RI, par exemple, système expert pour la RI, etc.

Les années 1990 (surtout à partir de 1995) sont des années de l'Internet. Cela a pour effet de propulser la RI en avant scène de beaucoup d'applications. La problématique est élargie. Par exemple, on traite maintenant plus souvent des documents multimédia qu'avant. Cependant, les techniques de base utilisées dans les moteurs de recherche sur le web restent identiques.

4. Quelques concepts clés de la recherche d'information :

A partir de la définition de la recherche d'information on peut distinguer des concepts suivants :

- la collection de documents,
- les documents,
- le besoin en information (requête),
- l'appariement requête-document
- La pertinence

4.1. La collection de documents

La collection de documents constitue l'ensemble des informations exploitables et accessibles par l'utilisateur. Elle est constituée d'un ensemble de documents. Dans le cas général et pour des raisons d'optimalité, la collection constitue des représentations très simplifiées mais suffisantes de ces documents. Ces représentations sont étudiées de telle sorte que la gestion (ajout, suppression d'un document) et l'interrogation (recherche) de la collection se font dans les meilleures conditions de cout.

4.2. Le document

Le document constitue l'information élémentaire d'une collection documentaire. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document.

4.3. Le besoin en information (requête)

Un besoin en information d'un utilisateur est exprimé par une requête. La littérature propose divers types de langages d'interrogation pour formuler cette requête. Nous citons les plus répandus :

1. Interrogation en langage booléen : l'utilisateur exprime sa requête sous forme d'un ensemble de termes relies entre eux par des operateurs booléens (ET, OU, NON). Beaucoup de moteurs de recherche, se basent sur ce mode d'interrogation, citons les plus connus.

2. Interrogation en langage naturel ou quasi naturel : l'utilisateur exprime sa requête en langage libre (langage naturel) sous forme de mots clés. Le système se charge de traduire (analyser) ces mots clés en une requête de langage de base de données ou une autre forme interne utilisable par le système. Les systèmes SMART [Salton 1989], SPIRIT [Fluhr et al 1985], OKAPI [Robertson et al 1976] sont interrogeables en langage naturel. Exemple : trouver toutes les usines de fabrication de voitures et leurs adresses.

3. Interrogation en langage graphique : une interface d'aide à la formulation de la requête est proposée à l'utilisateur. En effet, une vue d'ensemble de la base d'information et en particulier une vue de termes représentant le contenu sémantique des documents, est donnée à l'utilisateur pour l'assister à formuler sa requête. Dans PROTEUS [Signore et al 1992], l'interface d'aide à la formulation de requête propose un gestionnaire de thesaurus. Ce dernier est représenté par un graphe, les nœuds étant les termes du thesaurus et les liens étant les relations sémantiques entre ces termes. L'utilisateur peut identifier le type de relation qu'il souhaite utiliser et sélectionner un terme.

Le projet NEURODOC [Lelu et al 1992] est plus adapte a l'utilisation d'un thesaurus volumineux. NEURODOC offre à l'utilisateur un tableau de bord ou chaque nœud possède un nom et résume le sous-ensemble de mots et de documents fortement lies.

4.4. L'appariement requête-document

La comparaison entre le document et la requête permet de calculer une mesure appelée pertinence système, supposée représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction 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. 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. L'ordre dans lequel les documents susceptibles de répondre à la requête sont retournés est important. 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.

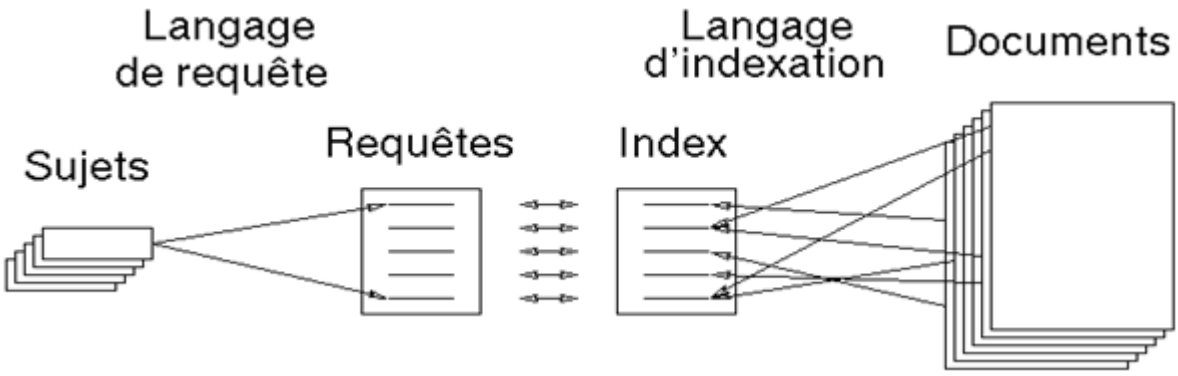


Figure 1.1 : processus d'appariement [G. salton, M. McGill, 84**].**

4.5. Pertinence

Pertinence est la notion centrale dans la RI car toutes les évaluations s'articulent autour de cette notion. Mais c'est aussi la notion la plus mal connue, malgré de nombreuses études portant sur cette notion. Voyons quelques définitions de la pertinence pour avoir une idée de la divergence.

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;
- un degré de la surprise qu'apporte un document, qui a un rapport avec le besoin de l'utilisateur;
- une mesure d'utilité du document pour l'utilisateur;

Même dans ces définitions, les notions utilisées (informativité, relativité, surprise, …) restent très vagues. Pourquoi on arrive à cette situation? C'est parce que les utilisateurs d'un système de RI ont des besoins très variés. Ils ont aussi des critères très différents pour juger si un document est pertinent. Donc, la notion de pertinence est utilisée pour recouvrir un très vaste éventail des critères et des relations. Par exemple, un utilisateur qui a formulé la requête sur "système expert" peut être satisfait par un document décrivant toutes les techniques utilisées dans "MYCIN" qui est un exemple typique de système expert. Cependant, un deuxième utilisateur peut juger ce même document non-pertinent car il cherche plutôt une description non-technique. Dans les deux situations, on appelle la relation entre le document et la requête "pertinence".

Beaucoup de travaux ont été menés sur cette notion. On s'est vite aperçu que la pertinence n'est pas une relation isolée entre un document et une requête. Elle fait appel aussi au contexte de jugement.

La question qu'on peut se poser est: à quoi sert d'étudier la notion de pertinence si on sait qu'elle est très variable?

Une des raisons est de tenter de trouver certains comportements communs entre les utilisateurs, et essayer de les formaliser. Si on arrive à cerner une partie de pertinence commune, on pourra l'implanter dans les systèmes pour répondre au moins à une partie commune des besoins. On connaît maintenant certains facteurs communs. Par exemple, le sujet (ou en anglais topic) est le facteur le plus important dans la pertinence

D'après les définitions précédentes, on peut déduire qu'il y a deux types de pertinence (pertinence système et pertinence utilisateur) qui sont illustrés par la figure suivante :

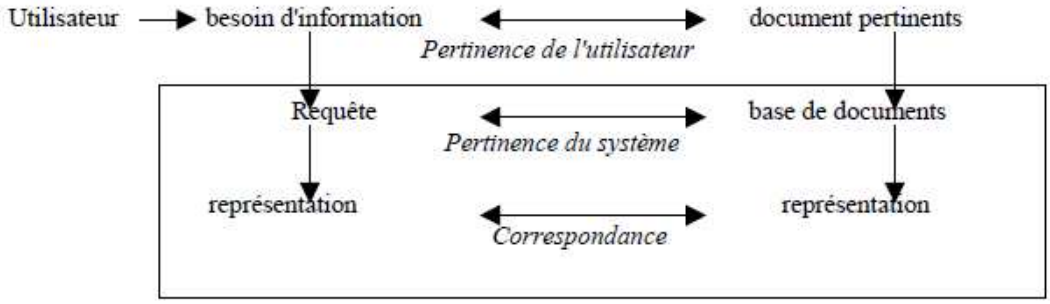


Figure1.2. pertinence système et pertinence utilisateur, [1].

On remarque qu'il y a trois niveaux différents:

1. **Le niveau utilisateur**:

A ce niveau, l'utilisateur a un besoin d'information dans sa tête, et il espère obtenir les documents pertinents pour répondre à ce besoin.

La relation entre le besoin d'information et les documents attendus est la relation de pertinence.

2. **Le niveau système**:

A ce niveau, le système répond à la requête formulée par l'utilisateur par un ensemble de documents trouvés dans la base de documents qu'il possède.

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

3. **Le niveau interne du système**:

La requête formulée par l'utilisateur (souvent en langue naturelle) ne peut pas se comparer directement avec des documents en langue naturelle eux aussi. Il faut donc créer des représentations internes pour la requête et pour les documents. Ces

représentations doivent être manipulables par l'ordinateur. Le processus de création de ces représentation est appelé *l'indexation* (à développer dans ce qui suit). Il est aussi à noter que les représentations crées ne reflètent qu'une partie des contenus de la requête et des documents. La technologie de nos jours ne nous permet pas encore de créer une représentation complète.

Pour déterminer si la représentation d'un document correspond à celle de la requête, on doit développer un processus d'évaluation. Différentes méthodes d'évaluation ont été développées, en relation avec la représentation de documents et de requête. C'est cet ensemble de représentation et la méthode d'évaluation qu'on appelle un *modèle* de RI (qui serra expliqué dans ce qui suit).

On remarque qu'il y a des différences entre deux niveaux différents. En ce qui concerne le besoin d'information, il est transformé en une requête, puis en une représentation de cette dernière aux niveaux inférieurs. Du côté document, il y a des changements similaires. Les relations qu'on peut déterminer à chaque niveau ne sont pas pareilles non plus. Ce qu'on espère est qu'un bon système de RI puisse donner une évaluation de *correspondance* qui reflète bien la *pertinence du système*, qui à son tour, correspond bien au jugement de *pertinence de l'utilisateur*. [1][2]

5. Approches possibles pour réaliser un système de recherche d'information

On peut imaginer quelques approches possibles pour réaliser un système de RI. [L.TAMINE, 97]

1. Une première approche très naïve consiste à considérer une requête comme une chaîne de caractères, et un document pertinent comme celui qui contient cette chaîne de caractères. À partir de cette vision simpliste, on peut imaginer l'approche qui consiste à balayer les documents séquentiellement, en les comparants avec la chaîne de caractères qui est la requête. Si on trouve la même chaîne de caractère dans un document, alors il est sélectionné comme réponse.

Cette approche est évidemment très simple à réaliser. Cependant, elle a plusieurs lacunes:

- *Vitesse:* L'opération de recherche est très lente. Pour chaque requête, on doit parcourir tous les documents dans la base. En général, il y en a des centaines de milliers, voire des millions. Il n'est donc envisageable d'utiliser cette approche que sur des collections très petites jusqu'à quelques centaines de documents.
- *Pouvoir d'expression d'une requête: Une* requête étant une simple chaîne de caractères, il est difficile d'exprimer des besoins complexes comme "Trouver des documents concernant les bases de données et l'intelligence artificielle utilisées dans l'industrie".

2. La deuxième approche que la plupart de systèmes existants l'utilisent est basée sur une indexation (sera détaillée dans ce qui suit).

Par rapport à l'approche précédente (première approche), la deuxième approche a les avantages suivants:

- Elle est plus rapide. En effet, on n'a plus besoin du parcours séquentiel. Avec la structure d'index, on peut directement savoir quels documents contiennent tel ou tel mot.
- L'expression des requêtes peut être très complexe, exprimant des besoins d'information complexes.

Le prix à payer pour ces avantages est le besoin de l'espace de stockage supplémentaire pour la structure d'index (qui sera détaillé dans ce qui suit). Mais ce besoin d'espace pose de moins en moins de problème maintenant avec l'arrivée d'un nouveau système de stockage plus rapide : le disque dur.

On utilisant la deuxième approche (indexation), on peut donner l'architecture générale de système de recherche d'information.

6. Architecture générale d'un Système de Recherche d'Information

Pour répondre aux besoins en information de l'utilisateur, un SRI met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans

un fond documentaire d'une part, et des besoins en information des utilisateurs d'autre part. Ces processus supposent que la collection de documents est unique. Pour des raisons d'optimisation du coût de recherche notamment en temps de réponse, plusieurs travaux se sont intéressés à la recherche parallèle sur plusieurs collections ayant des caractéristiques plus au moins proches [MacFarlane et al. 00] [Beigbeder et al, 05].

Un système de recherche d'information intègre trois fonctions principales représentées schématiquement par le processus en U de recherche d'information [Belkin et al. 92]. La Figure 1-3 illustre l'architecture générale d'un système de recherche d'information. D'un coté, on a l'information accessible dans le système. Elle est en général le résultat de collection de documents ou de sous collections de documents traitant d'un même domaine ou de domaines proches. D'un autre coté, on a le besoin en information exprimé par l'utilisateur, en général sous forme de requête, une fois stabilisé. Ensuite, l'information aussi bien que le besoin en information passe par des étapes de traitement pour être exploitables. Ces processus s'appuient sur un certain nombre de modèles permettant de sélectionner des informations pertinentes en réponses à une requête utilisateur. Il s'agit principalement du processus de représentation et du processus de recherche :

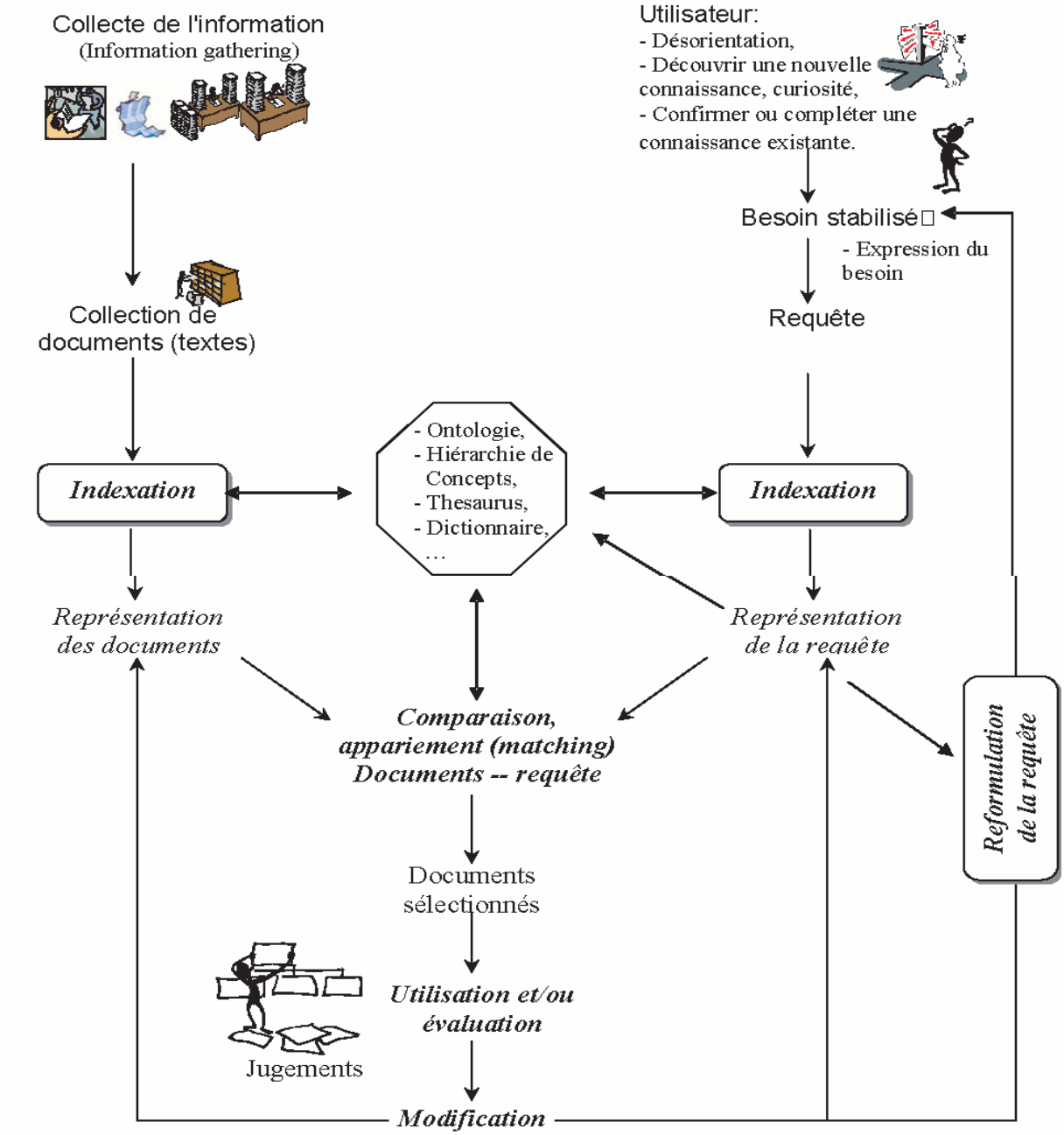


Figure 1-3. Processus général de recherche d'information.

6.1. Processus de représentation :

Un processus de représentation a pour rôle d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique (*indexation).* Le résultat de l'indexation constitue le *descripteur* du document ou de la requête, qui est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés

généralement des poids pour différencier leur degré de représentativité. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée *dictionnaire* constituant le *langage d'indexation.*

6.1.1. Indexation

Pour que le coût de la recherche soit acceptable, il convient d'exécuter une étape primordiale sur la collection de documents. Cette étape consiste à analyser les documents à fin de créer un ensemble de mots-clés : on parle de l'étape d'indexation [Deerwester et al. 90][Soule Dupuy, 90]. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet de créer une vue logique du document. On entend par vue logique la représentation des documents dans le système [M. Baziz, 05]. L'indexation peut être :

- **Manuelle** : chaque document est analysé par un spécialiste du domaine ou par un documentaliste
- **Semi-automatique :** l'indexeur intervient souvent pour choisir d'autres termes significatifs (synonymes, etc.) à partir de thesaurus ou d'une ontologie.
- **Automatique :** le processus d'indexation est entièrement informatisé

De manière générale, l'indexation automatique est la plus importante, elle est réalisée selon les étapes suivantes :

A. Analyse lexicale :

L'analyse lexicale (tokenization en anglais) est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est un groupe de caractères constituant un mot significatif. L'analyse lexicale permet de reconnaitre les espaces de séparation des mots, des chiffres, les ponctuations, etc.

B.L'élimination des mots vides :

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

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

C. Lemmatisation :

Un mot donné peut avoir différentes formes dans un texte. On peut par exemple citer *économie*, *économiquement*, *économétrie*, *économétrique*, etc. Il n'est pas forcement nécessaire d'indexer tous ces mots et un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème :

1. Une première façon consiste à examiner seulement la forme de mot, et selon la forme, on essaie de déduire ce qui est la racine [Porter 80].

2. On peut aussi utiliser un dictionnaire dans la lemmatisation. Pour savoir si une séquence de lettre à la fin correspond à une terminaison, il suffit de faire une élimination ou une transformation tentative, et de voir si la forme obtenue existe dans le dictionnaire. Si non, ce n'est pas une terminaison correcte, et d'autres possibilités sont ensuite envisagées. Par exemple, on peut accepter la règle qui remplace -ation par -er. Par exemple, transform-ation, élimin-ation, etc. Cependant, pour "vocation", si on applique cette règle, on obtiendra "vocer". Ce n'est pas une transformation correcte. Pour éviter cela, on peut vérifier dans le dictionnaire si le mot "vocer" existe. Sinon, on ne le transforme pas. L'utilisation d'un dictionnaire ajoute certains avantages, mais elle est au prix de se disposer d'un dictionnaire. La plupart de système de RI en dispose pas, et un tel dictionnaire électronique est encore peu accessible.

3. Une lemmatisation correcte requière souvent une reconnaissance correcte de catégorie grammaticale. Ainsi, on peut penser à utiliser un taggeur (ou un analyseur de catégorie) automatique dans un processus de lemmatisation. Une des approches possibles est de déterminer la catégorie d'un mot de façon probabiliste. Pour cela, il faut d'abord qu'on entraîne un modèle probabiliste en utilisant un ensemble de textes catégorisés

manuellement (le corpus d'entraînement). Ce modèle détermine la probabilité d'un mot d'être dans une catégorie selon sa forme, et selon les mots qui l'entourent.

D. pondération des termes

La pondération des termes d'indexation consiste à associer un poids d'importance (ou valeur de représentativité) $\boldsymbol{Wij}$ à chaque terme $\boldsymbol{tj}$ d'un document $\boldsymbol{di}$. De manière générale, les formules de pondération utilisées sont basées sur la combinaison d'un facteur de pondération local quantifiant la représentativité locale du terme dans le document, et d'un facteur de pondération global quantifiant la représentativité globale du terme vis-à-vis de la collection de documents. Plusieurs formules existent, [Buckley et al., 95] dont :

$$\boldsymbol{w_{ij} = \frac{tf_{ij}}{df_j} = tf_{ij} \times \frac{1}{df_j} = tf_{ij} \times idf_j}$$

Où :

- ✓ ***tfij*** est la fréquence d'occurrences du terme ***tj*** dans le document ***di***.
- ✓ ***dfj*** est la fréquence documentaire du terme ***tj*** (i.e. la proportion de documents de la collection qui contiennent ***tj***)
- ✓ ***idfj*** sa fréquence documentaire inverse.

La mesure **tf* idf** est une bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles intermédiaires. Pour des documents plus longs des normalisations ont été proposées, dont :

- ➢ La normalisation pivotée de Singhal [Singhal et al., 97] :

$$\boldsymbol{w_{ij} = \frac{tf_{ij} * idf_j}{1 + \frac{slope}{(1 - slope) * pivot} * \sqrt{\sum_j (tf_{ij} * idf_j)^2}}}$$

Où :

- ✓ **tfij** est le nombre d'occurrences du terme tj dans l'unité documentaire **di**
- ✓ **idfj** est la fréquence documentaire inverse définie classiquement par :

log(n/Nj) tel que **n** est le nombre de documents de la collection et **Nj** le nombre de documents indexés par le terme **tj**.

- ✓ **pivot** est une constante qui représente l'écart nul entre la probabilité de pertinence et la probabilité de sélection des documents.
- ✓ **slope** est un facteur de normalisation fixé empiriquement, de sorte à minimiser l'écart entre la pertinence et la sélection.

➢ La formule de Robertson [Robertson et al., 97]

$$w_{ij} = \frac{tf_{ij} * (k_1 + 1)}{k_1 \left((1 - b) + b * \frac{dl_i}{l} \right) + tf_{ij}}$$

Où :

- ✓ ***wij*** est le poids du terme tj dans le document **di**.
- ✓ ***K1*** constante qui permet de contrôler l'influence de la fréquence du terme **tj** dans le document **di**. Sa valeur dépend de la longueur des documents dans la collection. Le plus souvent, sa valeur est fixée à 1,2.
- ✓ ***b*** constante qui permet de contrôler l'effet de la longueur du document. Sa valeur la plus souvent utilisée est : 0,75.
- ✓ ***dli*** est la longueur du document **di**.
- ✓ Δ**l** est la longueur moyenne des documents dans la collection entière.

6.1.2. Structure des fichiers index :

Ce qui fait toujours le centre des débats dans le domaine de la recherche d'information est la manière dont l'index doit être organisé (stocké) pour répondre efficacement et rapidement aux besoins d'information des utilisateurs. Les structures d'index déterminent la méthode d'indexation et agissent directement sur les performances du système de recherche d'information. [Rijsbergen, 79]

Plusieurs structures ont été proposées, parmi elles on peut citer :

A. Les fichiers Séquentiels (Sequential files) :

Un fichier séquentiel est le moyen le plus simple de stocker un fichier de données puisque l'on stocke les enregistrements les uns à la suite des autres dans leur ordre d'insertion. Jusqu'au milieu des années 70, tous les systèmes textuels utilisaient les fichiers séquentiels du fait de l'utilisation de bandes magnétiques. Une requête sur un document consistait alors à parcourir toute la bande jusqu'à ce que l'on trouve le bon document, d'où une lenteur certaine du système malgré l'optimisation des algorithmes de recherche.

L'amélioration du processus était bloqué par le matériel (bandes magnétiques) ce qui changea radicalement avec l'arrivée d'un nouveau système de stockage plus rapide : le disque dur.

B. Les fichiers de signature (Signature files) :

Une signature de texte est une chaîne de bits (chaîne binaire) dans laquelle les bits sont positionnés pour décrire le document. La chaîne de bit est créée en appliquant une opération de hachage sur tous les mots clés décrivant un document. Une fois qu'un fichier signature est créé, la réponse à une requête consiste à calculer une chaîne pour la requête et à la comparer à celle calculée pour chaque document. Les manipulations de bits étant codées dans le processeur, cette technique est beaucoup plus rapide que la manipulation de chaînes de caractères. Cette méthode est donc très rapide, bien que le fichier signature ait besoin d'être parcouru entièrement à chaque requête.

Le principal désavantage de cette méthode est que le fait qu'un bit signale la présence d'un mot dans un fichier ne donne ni son emplacement, ni le nombre de fois qu'il apparaît dans ce fichier. Néanmoins, les résultats obtenus par cette méthode sont relativement bons.

C. Les fichiers inversés (Inverted files)

La structure de fichier inversé est à la base de tous les systèmes de recherche d'information. Un système à base de fichiers inversés contient trois composants principaux :

- Un dictionnaire : Le fichier dictionnaire contient tous les mots ou groupes nominaux spécifiques pouvant servir de mots-clés pour l'indexation et la recherche dans

l'ensemble des fichiers à traiter. A chaque entrée du dictionnaire est associé le nombre de fois où l'entrée apparaît dans l'ensemble documentaire.

- Un fichier de hachage : Ce fichier contient pour chaque entrée du dictionnaire une liste décrivant dans quel fichier apparaît cette entrée.
- Les fichiers de données : Qui représentent les documents du corpus à indexer.

La figure suivante est une représentation d'un système à base de fichiers inversés.

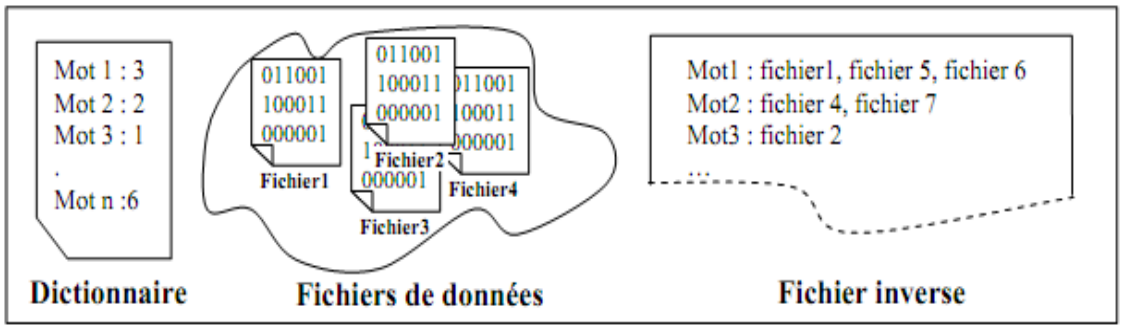


Figure1.4 : Système à base de fichiers inversés [Rijsbergen, 79].

Ce système de fichiers, s'il est rapide pour trouver un résultat, consomme énormément de place de stockage, les fichiers index étant parfois aussi gros que les fichiers de données, surtout dans le cas où les positions où apparaissent les mots clés dans les fichiers sont stockées. Les mises à jour sont aussi coûteuses puisqu'il faut refaire l'index à chaque ajout.

6.2. Processus de recherche :

Il représente le processus du noyau d'un SRI. Il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer. Les modèles de recherche représentent ce qui diffère le plus entre les SRI. Ils sont inspirés de concepts mathématiques afin de pouvoir évaluer certaines relations, notamment la relation d'appariement entre la requête et les documents. La problématique majeure des SRI est de retrouver les quelques dizaines ou milliers de documents pertinents parmi des millions de documents. Cet écart de cardinalité rend cette tâche encore plus difficile.

En plus des étapes de représentation et de recherche, quelques systèmes peuvent supporter une étape supplémentaire qui est :

6.3. La re-formulation de requête

L'utilisateur ne sait pas toujours choisir les bons termes qui expriment le mieux ses besoins d'information. En introduisant la reformulation de requête, la RI est alors envisagée comme une suite de formulations et de re-formulations de requêtes jusqu'à la satisfaction du besoin d'information de l'utilisateur, la requête initiale permettant rarement d'aboutir à un résultat qui satisfait ce dernier. Il s'agit en particulier d'ajouter des termes à la requête initiale de l'utilisateur et on parle alors d'expansion de la requête de l'utilisateur. On distingue trois niveaux permettant de différencier entre les techniques d'expansion de requêtes [Ihadjaden, 1994]:

- *La source des termes utilisés dans la reformulation* et qui peuvent provenir des résultats de recherches précédentes ou d'une base de connaissance (réseau sémantique, thesaurus).
- *Le choix de la méthode* ou de l'algorithme qui permet de sélectionner les termes à ajouter à la requête initiale.
- *Le rôle de l'usager* dans le processus de sélection des termes et qui peut être actif ou passif.

6.1. Reformulation manuelle

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

6.2. Reformulation interactive

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

6.3. Reformulation automatique de requêtes :

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

Cette reformulation intervient dans un processus plus général d'optimisation de la fonction de pertinence. Celle-ci a pour but de rapprocher la pertinence système de la pertinence utilisateur. Elle se présente comme une opération primordiale dans un SRI. En effet, compte tenu des volumes croissants des bases d'information, retrouver des informations pertinentes en utilisant seulement la requête initiale est une tache quasi-impossible [Voorhees 99].

La dimension de l'espace de recherche est élevée, c'est ainsi que la difficulté fondamentale de la reformulation de requêtes est la définition de l'approche à adopter en vue de réduire l'espace de recherche. Cette approche passe par la détermination de [Efthimiadis 96] :

- ✓ critères de choix des termes d'extension,
- ✓ règles de calcul des poids des nouveaux termes,
- ✓ hypothèse de base quant aux liens entre termes et documents.

Ces trois principales fonctions peuvent utiliser une ressource externe qui peut être une ontologie, une hiérarchie de concept ou le vocabulaire contrôlé d'un thesaurus ou des ressources internes qui seront détaillées dans le chapitre II.

7. Modèles de recherche d'information (RI) :

Si c'est l'indexation qui choisit les termes pour représenter le contenu d'un document ou d'une requête, c'est au modèle de leur donner une interprétation. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit les deux rôles suivants:

- ➢ créer une représentation interne pour un document ou pour une requête basée sur ces termes;
- ➢ définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité).

Le modèle joue un rôle central dans la RI. C'est le modèle qui détermine le comportement clé d'un système de RI. Dans ce chapitre, on décrit quelques modèles souvent utilisés dans la RI.

7.1. Le modèle booléen :

C'est le premier modèle utilisé en RI [Salton 71] ; il est basé sur la théorie des ensembles et l'algèbre de Boole [Gessler 93]. Le modèle booléen propose la représentation d'une requête sous forme d'une équation logique. Les termes d'indexation sont reliés par des connecteurs logiques ET, OU et NON.

L'approche booléenne consiste à trouver les documents qui ont exactement les mêmes termes qu'une requête construite par mots clefs. Les requêtes peuvent être affinées grâce aux opérateurs OR ou AND ou encore au moyen d'opérateurs comme NEAR. Ce type de recherche est à la base des moteurs de recherche comme Altavista1 ou Google2.

Cette approche est très efficace pour des requêtes utilisant des termes très spécifiques ou portants sur des domaines techniques particuliers avec leur vocabulaire propre mais son intérêt reste néanmoins limité.

L'inconvénient majeur du modèle booléen réside dans sa caractéristique de fournir une réponse binaire soit 1, soit 0 (les documents contiennent les termes demandés ou ne les contiennent pas). Ceci induit un volume de réponses important sans ordre spécifique des documents résultants.

-Tous les termes dans un document ou dans une requête étant pondérés de la même façon simple (0 ou 1), il est difficile d'exprimer qu'un terme est plus important qu'un autre dans leur représentation.

-Les formules de requêtes sont complexes, non accessibles à un large public. Elles nécessitent une maîtrise parfaite des opérateurs booléens, car leur signification est différente de celle qu'ils ont dans la langue naturelle.

Les deux modèles présentés ci-dessous largement utilisés en pratique, permettent de remédier à ces inconvénients.

7.2. Le modèle vectoriel :

Le modèle vectoriel introduit par Salton [Salton 71], repose sur les bases mathématiques des espaces vectoriels. Dans ce modèle, les documents et les requêtes sont représentés dans un espace vectoriel engendré par l'ensemble des termes d'indexation $t_1, t_2, t_3 \ldots, t_T$ où T est le nombre total de termes issus de l'indexation de la collection des documents.

Chaque document est représenté par un vecteur : $D_J = (d_{1j}, d_{2j}, \ldots d_{ij} \ldots, d_{Tj})$

Chaque requête est représentée par un vecteur $Q = (q_1, q_2 \ldots q_i \ldots, q_T)$

Avec : d_{ij} Poids du terme t_i dans le document D_j

q_i Poids du terme t_i dans la requête Q

Les termes de poids nul représentent les termes absents dans un document alors que les poids positifs représentent les termes assignés.

La fonction de calcul du coefficient de similarité entre chaque documentD_i , représenté par le vecteur$(d_1, d_2, \ldots d_{Tj})$ et la requête Q, représentée par le vecteur$(q_1, q_2 \ldots q_i \ldots, q_T)$ est appelée Retrieval Status Value ou RSV.

Ce coefficient de similarité est calculé sur la base d'une fonction qui mesure la colinéarité des vecteurs document et requête. On peut citer notamment les fonctions suivantes :

- **Produit scalaire :**

$$\boldsymbol{RSV(Q, D_j) = \sum_{i=1}^{T} q_i * d_{ij}}$$

- **Mesure de Jaccard** :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\sum_{i=1}^{T} {q_i}^2 \ + \sum_{i=1}^{T} {d_{ij}}^2 - \sum_{i=1}^{T} q_i * d_{ij}}$$

- **Mesure de cosinus** :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\left(\sum_{i=1}^{T} {q_i}^2\right)^{1/2} * \left(\sum_{i=1}^{T} {d_{ij}}^2\right)^{1/2}}$$

La similarité entre deux textes (requêtes ou documents) dépend ainsi des poids des termes coïncidant dans les deux textes. Il est donc possible de classer les documents par ordre de pertinence décroissante.

L'avantage du modèle vectoriel par rapport au modèle booléen réside particulièrement dans l'ordonnancement des documents sélectionnés selon leurs pertinences. Cependant, l'inconvénient majeur de l'approche vectorielle réside dans le fait que l'association entres les termes d'indexation n'est pas considérée. Il est impossible de représenter des phrases ou des mots multi termes. On considère effectivement que les termes sont indépendants [Yates 99].

7.3. Le modèle probabiliste :

Le modèle probabiliste a été proposé par Robertson et Sparck Jones [Robertson 76], il utilise un modèle mathématique fondé sur la théorie de la probabilité conditionnelle (appelé aussi modèle de la théorie de pertinence). Lors du processus d'indexation deux probabilités conditionnelles sont utilisées :

- *P(t/pert) :* La probabilité pour que le terme t apparaisse dans un document donné sachant que ce document est pertinent pour la requête
- *P(t/NonPert)* : La probabilité pour que le terme t apparaisse dans un document donné sachant que ce document est non pertinent pour la requête.

En supposant que la distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents, et que les variables "documents pertinents" et "document non pertinent" sont indépendantes, la fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document P (Pert/D).

$$P(Pert/D) = \frac{P(D/Pert) * P(Pert)}{P(D)},$$

$$P(NonPert/D) = \frac{P(D/NonPert) * P(NonPert)}{P(D)}$$

Avec :

$$P(D) = P(D/Pert) * P(Pert) + P(D/NonPert) * P(NonPert)$$

Où :

- *P (D/Pert*) (respectivement *P(D/NonPert)*) : Probabilité d'observer D sachant qu'il est pertinent (respectivement non pertinent).

-P(Pert) (respectivement P(NonPert)) : Probabilité à priori qu'un document soit pertinent (respectivement non pertinent).

Le coefficient de similarité requête document (RSV) peut être calculé par différentes formules.

Robertson et Spark-Jones [Robertson 96] proposent la formule suivante :

$$RSV = \sum log \left(\frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \right)$$

Où

- ✓ N: nombre total de documents de la base,
- ✓ n: nombre de documents contenant le terme,
- ✓ R: nombre de documents connus comme étant pertinents,
- ✓ r: nombre de documents connus comme étant pertinents et contenants le terme.

L'ajout de 0.5 à tous les membres s'explique par la nécessité d'écarter tous les cas limites qui entraîneraient des valeurs nulles de ces membres.

Ce modèle a donné lieu à de nombreuses extensions. Il est à l'origine du système OKAPI qui est l'un des systèmes les plus performants selon les compagnes d'évaluation TREC3 [Walker 97]. L'inconvénient majeur de ce modèle est que les calculs des probabilités sont complexes et que l'indépendance des variables n'est pas toujours vérifiée voir pas prise en compte.

7.4. Les modèles de langage

Le modèle de langage est emprunté de la linguistique informatique. Son objectif est de capter les régularités linguistiques d'un langage, en observant la distribution des mots, succession de mots dans le langage donné. Il désigne une fonction de probabilité qui assigne à chaque séquence de mots une probabilité de sa génération.

Dans le domaine de la recherche d'information, Ponte et Croft [Pont and Croft, 1998] l'ont introduit en premier. Il permet de déterminer la probabilité qu'une requête utilisateur Q puisse être générée par un modèle de langage d'un document spécifique. L'estimation de cette probabilité est plus en moins complexe, à cet effet l'indépendance des termes de la requête Q est généralement supposée; d'où la formule générale suivante :

$$P(Q/M_d) = \prod_{t \in Q} P(t \setminus M_d)$$

Après la recherche les documents seront rangés par l'ordre décroissant de cette probabilité.

Dans notre travail, nous utilisons le modèle de langage comme cadre pour implémenter notre approche. Le chapitre III décrit d'une manière détaillée ce modèle

.

8. Evaluation des performances des systèmes de recherche d'information

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. D'une façon générale, tout système de recherche d'information présente deux objectifs:

- ✓ retrouver tous les documents pertinents,
- ✓ rejeter tous les documents non pertinents.

Ces deux objectifs sont évalués par les mesures de précision et de rappel définis ci-dessous. Nous allons définir également les mesures à *x* documents et d'autres mesures de performance.

8.1. Les mesures de Précision/Rappel

Les mesures de précision/rappel sont obtenues en partitionnant l'ensemble des documents restitués par le SRI en deux catégories : les documents pertinents et les documents non pertinents. Ces deux catégories se définissent comme suit:

- **Taux de précision :** La précision mesure la capacité du système de rejeter tous les documents non pertinents à une requête. Il est donné par le rapport entre l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés.
- **Taux de rappel :** Le rappel mesure la capacité du système à retrouver tous les documents pertinents répondants à une requête. Il est donné par le rapport entre les documents retrouvés pertinents et l'ensemble des documents pertinents de la base.

Les taux de précision et de rappel sont donnés par les formulations suivantes :

$$\textbf{Précision} = \frac{R+}{M}$$

$$\textbf{Rappel} = \frac{R+}{R}$$

Où:

***R*:** le nombre total de documents pertinents dans la collection,

- ✓ ***M* :** le nombre de documents sélectionnés,
- ✓ ***R+* :** le nombre de documents pertinents sélectionnés.

La figure 1.5 illustre la précision et le rappel d'une requête d'une façon générale

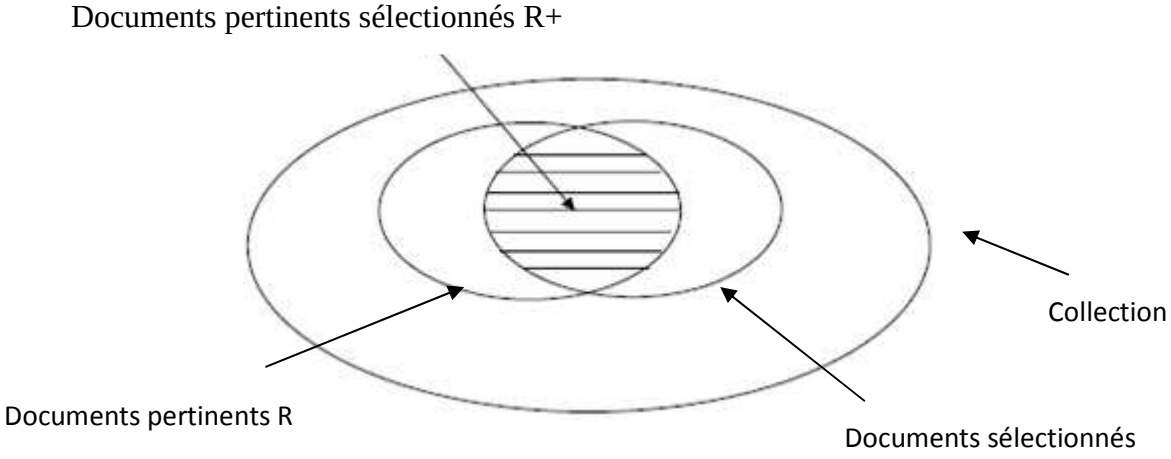


Figure. 1.5 – Exemple de rappel et de précision pour une requête

Le taux de rappel et le taux de précision évaluent respectivement les notions de bruit et de silence documentaire qui constituent les deux premiers indicateurs de performance d'un SRI.

La notion de silence et bruit présentent respectivement la taux de documents pertinents non sélectionnés et le taux de documents non pertinents sélectionnés.

8.2. Autres mesures de performance

- **Le temps de réponse acceptables :** un SRI doit pouvoir fournir à l'utilisateur les documents correspondants à sa demande dans des temps très courts.
- **La présentation des résultats claire et facilité d'utilisation :** capacité du système à comprendre les besoins de l'utilisateur et a mettre en valeur les documents correspondants a ceux-ci. Ceci est lie à l'interface avec l'utilisateur.
- **Le nombre total de documents pertinents retournés, ou le rappel à 1000 documents :** ces mesures permettant d'évaluer la performance globale du système au final, en fonction ou non du nombre de documents pertinents total.
- **La longueur de recherche :** elle est égale au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre *n* de documents pertinents.

8.3. Compagnes d'évaluation :

De nombreux projets basés sur des corpus d'évaluation se multiplient depuis les années 70.

On peut par exemple citer la compagne CLEF (Cross Language Evaluation Forum) ou la compagne d'évaluation TREC, Co-organisée par le NIST4 et la DARPA, a commencé en 1992. Elle a pour but d'encourager la recherche documentaire basée sur de grandes collections de test, tout en fournissant l'infrastructure nécessaire pour l'évaluation des méthodologies de recherche et de filtrage d'information.

9. Conclusion

Dans ce premier chapitre, nous nous sommes essentiellement intéressés à donner une présentation générale de recherche d'information qui désigne le processus qui permet, à partir d'une expression des besoins d'information d'un utilisateur, de retrouver l'ensemble des documents contenant l'information recherchée. Comme on a déjà illustré précédemment, les documents retournés par le SRI peuvent ne pas répondre au besoin de l'utilisateur. Pour prendre en compte cette difficulté, des techniques de reformulation (expansion) de la requête qui fait l'objet de deuxième chapitre sont utilisées, afin d'obtenir des requêtes optimales.

1. Introduction

Le domaine de recherche d'information remonte peu après l'invention des ordinateurs. Comme plusieurs autres domaines informatiques, les pionniers de l'époque étaient enthousiastes à utiliser l'ordinateur pour automatiser la recherche des informations, qui dépassaient la capacité humaine. On peut aujourd'hui dire que la recherche d'information est un champ transdisciplinaire, qui peut être étudié par plusieurs disciplines, approche qui devrait permettre de trouver des solutions pour améliorer son efficacité.

Ce chapitre commence par présenter les concepts de base de la Recherche d'Information (RI), avant de décrire l'architecture générale d'un système de recherche d'information. Par la suite, une description des principaux modèles de RI qui servent à formaliser le processus de recherche sont présentés, en fin les mesures et les collections de test utilisés pour évaluer les systèmes de recherche d'information sont exposés.

2. La recherche d'information

La Recherche d'Information (RI) [Van Rijsbergen., 77] [Grossman et al., 98] [Salton, 71] est traditionnellement définie comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur.

Les systèmes de recherche d'information (SRI) [Baziz M., 05], servent d'interface entre une source (collection) contenant des quantités considérables de documents et des utilisateurs cherchant, via des requêtes, des informations susceptibles de se trouver dans cette collection. Les SRI intègrent un ensemble de techniques permettant de sélectionner ces informations. Elles peuvent être résumées en quatre fonctions, qui sont le stockage de l'information, l'organisation de ces informations, la recherche d'informations en réponse à des requêtes utilisateurs et la restitution des informations pertinentes pour ces requêtes. La dernière fonction est celle qui est visible pour l'utilisateur.

3. Bref historique de la RI :

La RI n'est pas un domaine récent. Il date des années 1940, dès la naissance des ordinateurs. Au début, la RI se concentrait sur les applications dans des bibliothèques, d'où aussi le nom "automatisation de bibliothèques". Depuis le début de ces études, la notion de

pertinence a toujours été un sujet de recherche. Dans les années 1950, on commençait de petites expérimentations en utilisant des petites collections de documents. Dans les années 1960 et 1970, des expérimentations plus larges ont été menées, et on a développé une méthodologie d'évaluation du système qui est aussi utilisée maintenant dans d'autres domaines.

Les années 1980 ont été influencées par le développement de l'intelligence artificielle. Ainsi, on tentait d'intégrer des techniques de l'IA en RI, par exemple, système expert pour la RI, etc.

Les années 1990 (surtout à partir de 1995) sont des années de l'Internet. Cela a pour effet de propulser la RI en avant scène de beaucoup d'applications. La problématique est élargie. Par exemple, on traite maintenant plus souvent des documents multimédia qu'avant. Cependant, les techniques de base utilisées dans les moteurs de recherche sur le web restent identiques.

4. Quelques concepts clés de la recherche d'information :

A partir de la définition de la recherche d'information on peut distinguer des concepts suivants :

- la collection de documents,
- les documents,
- le besoin en information (requête),
- l'appariement requête-document
- La pertinence

4.1. La collection de documents

La collection de documents constitue l'ensemble des informations exploitables et accessibles par l'utilisateur. Elle est constituée d'un ensemble de documents. Dans le cas général et pour des raisons d'optimalité, la collection constitue des représentations très simplifiées mais suffisantes de ces documents. Ces représentations sont étudiées de telle sorte que la gestion (ajout, suppression d'un document) et l'interrogation (recherche) de la collection se font dans les meilleures conditions de cout.

4.2. Le document

Le document constitue l'information élémentaire d'une collection documentaire. L'information élémentaire, appelée aussi granule de document, peut représenter tout ou une partie d'un document.

4.3. Le besoin en information (requête)

Un besoin en information d'un utilisateur est exprimé par une requête. La littérature propose divers types de langages d'interrogation pour formuler cette requête. Nous citons les plus répandus :

1. Interrogation en langage booléen : l'utilisateur exprime sa requête sous forme d'un ensemble de termes relies entre eux par des operateurs booléens (ET, OU, NON). Beaucoup de moteurs de recherche, se basent sur ce mode d'interrogation, citons les plus connus.

2. Interrogation en langage naturel ou quasi naturel : l'utilisateur exprime sa requête en langage libre (langage naturel) sous forme de mots clés. Le système se charge de traduire (analyser) ces mots clés en une requête de langage de base de données ou une autre forme interne utilisable par le système. Les systèmes SMART [Salton 1989], SPIRIT [Fluhr et al 1985], OKAPI [Robertson et al 1976] sont interrogeables en langage naturel. Exemple : trouver toutes les usines de fabrication de voitures et leurs adresses.

3. Interrogation en langage graphique : une interface d'aide à la formulation de la requête est proposée à l'utilisateur. En effet, une vue d'ensemble de la base d'information et en particulier une vue de termes représentant le contenu sémantique des documents, est donnée à l'utilisateur pour l'assister à formuler sa requête. Dans PROTEUS [Signore et al 1992], l'interface d'aide à la formulation de requête propose un gestionnaire de thesaurus. Ce dernier est représenté par un graphe, les nœuds étant les termes du thesaurus et les liens étant les relations sémantiques entre ces termes. L'utilisateur peut identifier le type de relation qu'il souhaite utiliser et sélectionner un terme.

Le projet NEURODOC [Lelu et al 1992] est plus adapte a l'utilisation d'un thesaurus volumineux. NEURODOC offre à l'utilisateur un tableau de bord ou chaque nœud possède un nom et résume le sous-ensemble de mots et de documents fortement lies.

4.4. L'appariement requête-document

La comparaison entre le document et la requête permet de calculer une mesure appelée pertinence système, supposée représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction 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. 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. L'ordre dans lequel les documents susceptibles de répondre à la requête sont retournés est important. 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.

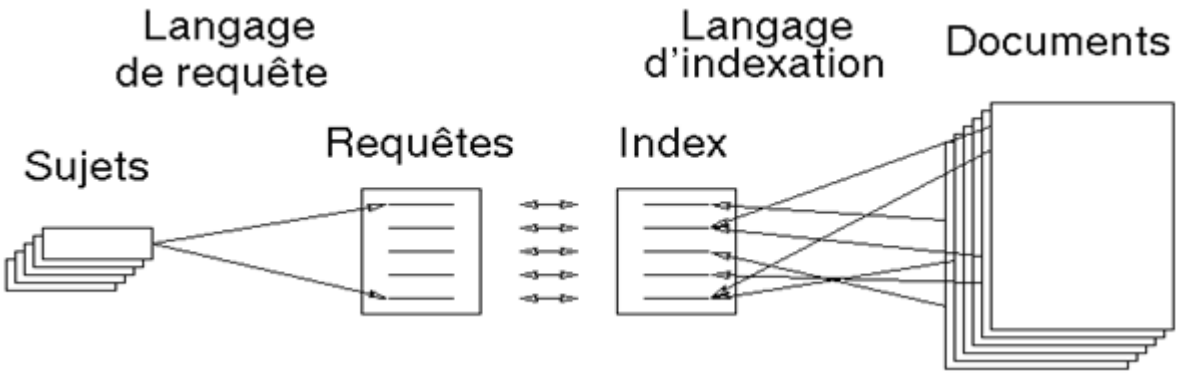


Figure 1.1 : processus d'appariement [G. salton, M. McGill, 84**].**

4.5. Pertinence

Pertinence est la notion centrale dans la RI car toutes les évaluations s'articulent autour de cette notion. Mais c'est aussi la notion la plus mal connue, malgré de nombreuses études portant sur cette notion. Voyons quelques définitions de la pertinence pour avoir une idée de la divergence.

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;
- un degré de la surprise qu'apporte un document, qui a un rapport avec le besoin de l'utilisateur;
- une mesure d'utilité du document pour l'utilisateur;

Même dans ces définitions, les notions utilisées (informativité, relativité, surprise, …) restent très vagues. Pourquoi on arrive à cette situation? C'est parce que les utilisateurs d'un système de RI ont des besoins très variés. Ils ont aussi des critères très différents pour juger si un document est pertinent. Donc, la notion de pertinence est utilisée pour recouvrir un très vaste éventail des critères et des relations. Par exemple, un utilisateur qui a formulé la requête sur "système expert" peut être satisfait par un document décrivant toutes les techniques utilisées dans "MYCIN" qui est un exemple typique de système expert. Cependant, un deuxième utilisateur peut juger ce même document non-pertinent car il cherche plutôt une description non-technique. Dans les deux situations, on appelle la relation entre le document et la requête "pertinence".

Beaucoup de travaux ont été menés sur cette notion. On s'est vite aperçu que la pertinence n'est pas une relation isolée entre un document et une requête. Elle fait appel aussi au contexte de jugement.

La question qu'on peut se poser est: à quoi sert d'étudier la notion de pertinence si on sait qu'elle est très variable?

Une des raisons est de tenter de trouver certains comportements communs entre les utilisateurs, et essayer de les formaliser. Si on arrive à cerner une partie de pertinence commune, on pourra l'implanter dans les systèmes pour répondre au moins à une partie commune des besoins. On connaît maintenant certains facteurs communs. Par exemple, le sujet (ou en anglais topic) est le facteur le plus important dans la pertinence

D'après les définitions précédentes, on peut déduire qu'il y a deux types de pertinence (pertinence système et pertinence utilisateur) qui sont illustrés par la figure suivante :

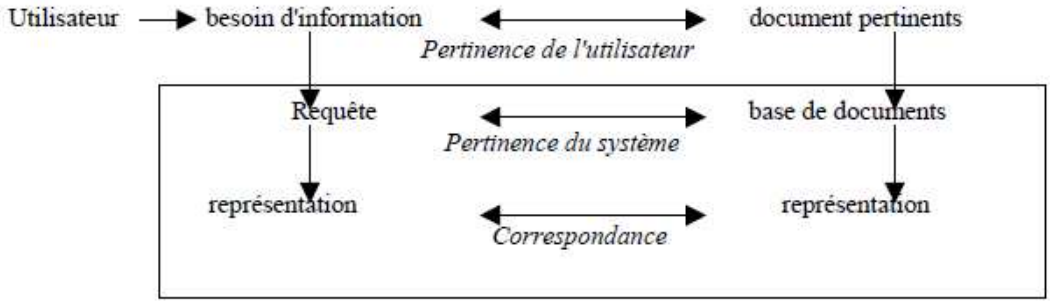


Figure1.2. pertinence système et pertinence utilisateur, [1].

On remarque qu'il y a trois niveaux différents:

1. **Le niveau utilisateur**:

A ce niveau, l'utilisateur a un besoin d'information dans sa tête, et il espère obtenir les documents pertinents pour répondre à ce besoin.

La relation entre le besoin d'information et les documents attendus est la relation de pertinence.

2. **Le niveau système**:

A ce niveau, le système répond à la requête formulée par l'utilisateur par un ensemble de documents trouvés dans la base de documents qu'il possède.

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

3. **Le niveau interne du système**:

La requête formulée par l'utilisateur (souvent en langue naturelle) ne peut pas se comparer directement avec des documents en langue naturelle eux aussi. Il faut donc créer des représentations internes pour la requête et pour les documents. Ces

représentations doivent être manipulables par l'ordinateur. Le processus de création de ces représentation est appelé *l'indexation* (à développer dans ce qui suit). Il est aussi à noter que les représentations crées ne reflètent qu'une partie des contenus de la requête et des documents. La technologie de nos jours ne nous permet pas encore de créer une représentation complète.

Pour déterminer si la représentation d'un document correspond à celle de la requête, on doit développer un processus d'évaluation. Différentes méthodes d'évaluation ont été développées, en relation avec la représentation de documents et de requête. C'est cet ensemble de représentation et la méthode d'évaluation qu'on appelle un *modèle* de RI (qui serra expliqué dans ce qui suit).

On remarque qu'il y a des différences entre deux niveaux différents. En ce qui concerne le besoin d'information, il est transformé en une requête, puis en une représentation de cette dernière aux niveaux inférieurs. Du côté document, il y a des changements similaires. Les relations qu'on peut déterminer à chaque niveau ne sont pas pareilles non plus. Ce qu'on espère est qu'un bon système de RI puisse donner une évaluation de *correspondance* qui reflète bien la *pertinence du système*, qui à son tour, correspond bien au jugement de *pertinence de l'utilisateur*. [1][2]

5. Approches possibles pour réaliser un système de recherche d'information

On peut imaginer quelques approches possibles pour réaliser un système de RI. [L.TAMINE, 97]

1. Une première approche très naïve consiste à considérer une requête comme une chaîne de caractères, et un document pertinent comme celui qui contient cette chaîne de caractères. À partir de cette vision simpliste, on peut imaginer l'approche qui consiste à balayer les documents séquentiellement, en les comparants avec la chaîne de caractères qui est la requête. Si on trouve la même chaîne de caractère dans un document, alors il est sélectionné comme réponse.

Cette approche est évidemment très simple à réaliser. Cependant, elle a plusieurs lacunes:

- *Vitesse:* L'opération de recherche est très lente. Pour chaque requête, on doit parcourir tous les documents dans la base. En général, il y en a des centaines de milliers, voire des millions. Il n'est donc envisageable d'utiliser cette approche que sur des collections très petites jusqu'à quelques centaines de documents.
- *Pouvoir d'expression d'une requête: Une* requête étant une simple chaîne de caractères, il est difficile d'exprimer des besoins complexes comme "Trouver des documents concernant les bases de données et l'intelligence artificielle utilisées dans l'industrie".

2. La deuxième approche que la plupart de systèmes existants l'utilisent est basée sur une indexation (sera détaillée dans ce qui suit).

Par rapport à l'approche précédente (première approche), la deuxième approche a les avantages suivants:

- Elle est plus rapide. En effet, on n'a plus besoin du parcours séquentiel. Avec la structure d'index, on peut directement savoir quels documents contiennent tel ou tel mot.
- L'expression des requêtes peut être très complexe, exprimant des besoins d'information complexes.

Le prix à payer pour ces avantages est le besoin de l'espace de stockage supplémentaire pour la structure d'index (qui sera détaillé dans ce qui suit). Mais ce besoin d'espace pose de moins en moins de problème maintenant avec l'arrivée d'un nouveau système de stockage plus rapide : le disque dur.

On utilisant la deuxième approche (indexation), on peut donner l'architecture générale de système de recherche d'information.

6. Architecture générale d'un Système de Recherche d'Information

Pour répondre aux besoins en information de l'utilisateur, un SRI met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans

un fond documentaire d'une part, et des besoins en information des utilisateurs d'autre part. Ces processus supposent que la collection de documents est unique. Pour des raisons d'optimisation du coût de recherche notamment en temps de réponse, plusieurs travaux se sont intéressés à la recherche parallèle sur plusieurs collections ayant des caractéristiques plus au moins proches [MacFarlane et al. 00] [Beigbeder et al, 05].

Un système de recherche d'information intègre trois fonctions principales représentées schématiquement par le processus en U de recherche d'information [Belkin et al. 92]. La Figure 1-3 illustre l'architecture générale d'un système de recherche d'information. D'un coté, on a l'information accessible dans le système. Elle est en général le résultat de collection de documents ou de sous collections de documents traitant d'un même domaine ou de domaines proches. D'un autre coté, on a le besoin en information exprimé par l'utilisateur, en général sous forme de requête, une fois stabilisé. Ensuite, l'information aussi bien que le besoin en information passe par des étapes de traitement pour être exploitables. Ces processus s'appuient sur un certain nombre de modèles permettant de sélectionner des informations pertinentes en réponses à une requête utilisateur. Il s'agit principalement du processus de représentation et du processus de recherche :

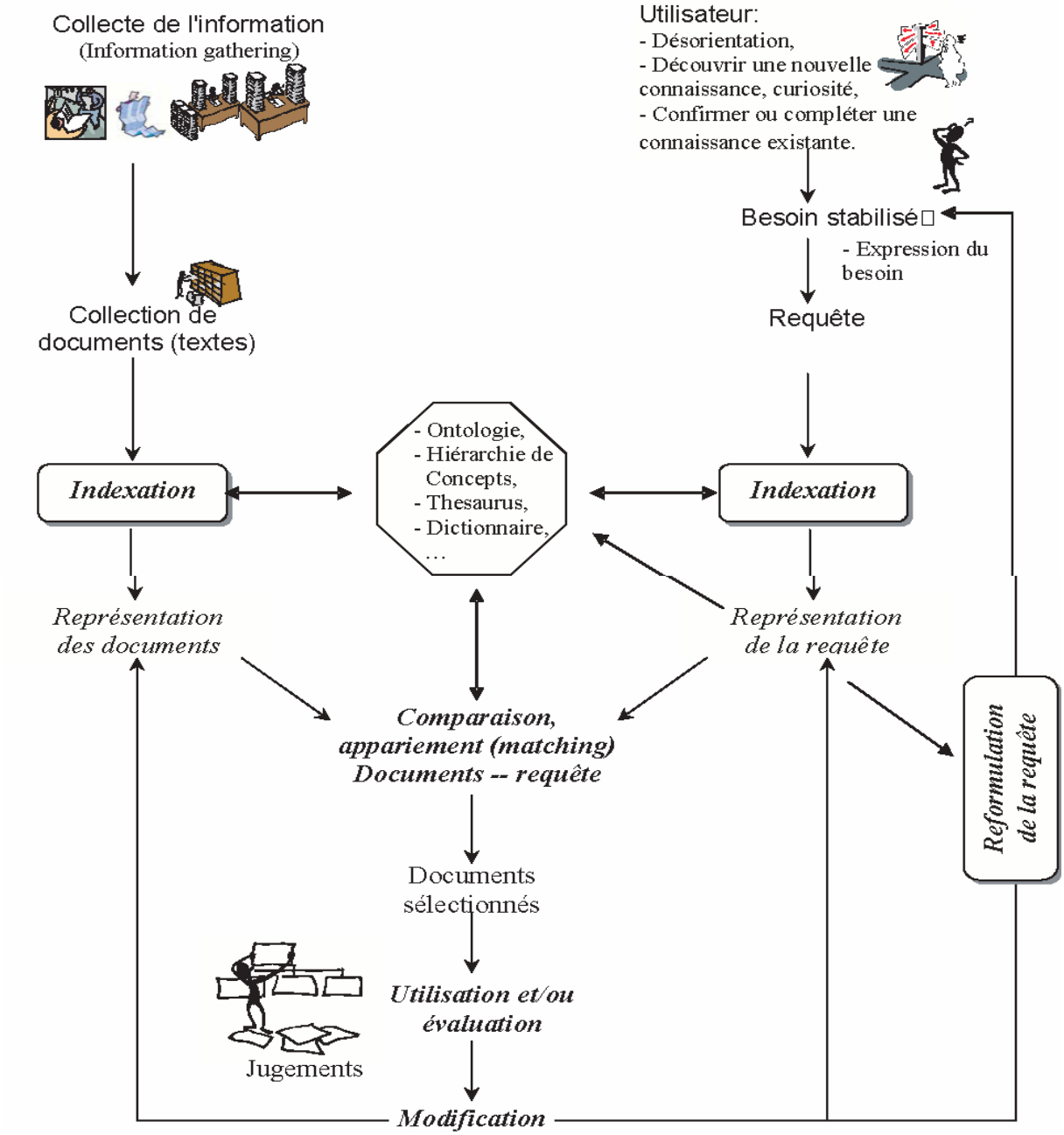


Figure 1-3. Processus général de recherche d'information.

6.1. Processus de représentation :

Un processus de représentation a pour rôle d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique (*indexation).* Le résultat de l'indexation constitue le *descripteur* du document ou de la requête, qui est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés

généralement des poids pour différencier leur degré de représentativité. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée *dictionnaire* constituant le *langage d'indexation.*

6.1.1. Indexation

Pour que le coût de la recherche soit acceptable, il convient d'exécuter une étape primordiale sur la collection de documents. Cette étape consiste à analyser les documents à fin de créer un ensemble de mots-clés : on parle de l'étape d'indexation [Deerwester et al. 90][Soule Dupuy, 90]. Ces mots-clés seront plus facilement exploitables par le système lors du processus ultérieur de recherche. L'indexation permet de créer une vue logique du document. On entend par vue logique la représentation des documents dans le système [M. Baziz, 05]. L'indexation peut être :

- **Manuelle** : chaque document est analysé par un spécialiste du domaine ou par un documentaliste
- **Semi-automatique :** l'indexeur intervient souvent pour choisir d'autres termes significatifs (synonymes, etc.) à partir de thesaurus ou d'une ontologie.
- **Automatique :** le processus d'indexation est entièrement informatisé

De manière générale, l'indexation automatique est la plus importante, elle est réalisée selon les étapes suivantes :

A. Analyse lexicale :

L'analyse lexicale (tokenization en anglais) est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est un groupe de caractères constituant un mot significatif. L'analyse lexicale permet de reconnaitre les espaces de séparation des mots, des chiffres, les ponctuations, etc.

B.L'élimination des mots vides :

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

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

C. Lemmatisation :

Un mot donné peut avoir différentes formes dans un texte. On peut par exemple citer *économie*, *économiquement*, *économétrie*, *économétrique*, etc. Il n'est pas forcement nécessaire d'indexer tous ces mots et un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème :

1. Une première façon consiste à examiner seulement la forme de mot, et selon la forme, on essaie de déduire ce qui est la racine [Porter 80].

2. On peut aussi utiliser un dictionnaire dans la lemmatisation. Pour savoir si une séquence de lettre à la fin correspond à une terminaison, il suffit de faire une élimination ou une transformation tentative, et de voir si la forme obtenue existe dans le dictionnaire. Si non, ce n'est pas une terminaison correcte, et d'autres possibilités sont ensuite envisagées. Par exemple, on peut accepter la règle qui remplace -ation par -er. Par exemple, transform-ation, élimin-ation, etc. Cependant, pour "vocation", si on applique cette règle, on obtiendra "vocer". Ce n'est pas une transformation correcte. Pour éviter cela, on peut vérifier dans le dictionnaire si le mot "vocer" existe. Sinon, on ne le transforme pas. L'utilisation d'un dictionnaire ajoute certains avantages, mais elle est au prix de se disposer d'un dictionnaire. La plupart de système de RI en dispose pas, et un tel dictionnaire électronique est encore peu accessible.

3. Une lemmatisation correcte requière souvent une reconnaissance correcte de catégorie grammaticale. Ainsi, on peut penser à utiliser un taggeur (ou un analyseur de catégorie) automatique dans un processus de lemmatisation. Une des approches possibles est de déterminer la catégorie d'un mot de façon probabiliste. Pour cela, il faut d'abord qu'on entraîne un modèle probabiliste en utilisant un ensemble de textes catégorisés

manuellement (le corpus d'entraînement). Ce modèle détermine la probabilité d'un mot d'être dans une catégorie selon sa forme, et selon les mots qui l'entourent.

D. pondération des termes

La pondération des termes d'indexation consiste à associer un poids d'importance (ou valeur de représentativité) $\boldsymbol{Wij}$ à chaque terme $\boldsymbol{tj}$ d'un document $\boldsymbol{di}$. De manière générale, les formules de pondération utilisées sont basées sur la combinaison d'un facteur de pondération local quantifiant la représentativité locale du terme dans le document, et d'un facteur de pondération global quantifiant la représentativité globale du terme vis-à-vis de la collection de documents. Plusieurs formules existent, [Buckley et al., 95] dont :

$$\boldsymbol{w_{ij} = \frac{tf_{ij}}{df_j} = tf_{ij} \times \frac{1}{df_j} = tf_{ij} \times idf_j}$$

Où :

- ✓ ***tfij*** est la fréquence d'occurrences du terme ***tj*** dans le document ***di***.
- ✓ ***dfj*** est la fréquence documentaire du terme ***tj*** (i.e. la proportion de documents de la collection qui contiennent ***tj***)
- ✓ ***idfj*** sa fréquence documentaire inverse.

La mesure **tf* idf** est une bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles intermédiaires. Pour des documents plus longs des normalisations ont été proposées, dont :

- ➢ La normalisation pivotée de Singhal [Singhal et al., 97] :

$$\boldsymbol{w_{ij} = \frac{tf_{ij} * idf_j}{1 + \frac{slope}{(1 - slope) * pivot} * \sqrt{\sum_j (tf_{ij} * idf_j)^2}}}$$

Où :

- ✓ **tfij** est le nombre d'occurrences du terme tj dans l'unité documentaire **di**
- ✓ **idfj** est la fréquence documentaire inverse définie classiquement par :

log(n/Nj) tel que **n** est le nombre de documents de la collection et **Nj** le nombre de documents indexés par le terme **tj**.

- ✓ **pivot** est une constante qui représente l'écart nul entre la probabilité de pertinence et la probabilité de sélection des documents.
- ✓ **slope** est un facteur de normalisation fixé empiriquement, de sorte à minimiser l'écart entre la pertinence et la sélection.

➢ La formule de Robertson [Robertson et al., 97]

$$w_{ij} = \frac{tf_{ij} * (k_1 + 1)}{k_1 \left((1 - b) + b * \frac{dl_i}{l} \right) + tf_{ij}}$$

Où :

- ✓ ***wij*** est le poids du terme tj dans le document **di**.
- ✓ ***K1*** constante qui permet de contrôler l'influence de la fréquence du terme **tj** dans le document **di**. Sa valeur dépend de la longueur des documents dans la collection. Le plus souvent, sa valeur est fixée à 1,2.
- ✓ ***b*** constante qui permet de contrôler l'effet de la longueur du document. Sa valeur la plus souvent utilisée est : 0,75.
- ✓ ***dli*** est la longueur du document **di**.
- ✓ Δ**l** est la longueur moyenne des documents dans la collection entière.

6.1.2. Structure des fichiers index :

Ce qui fait toujours le centre des débats dans le domaine de la recherche d'information est la manière dont l'index doit être organisé (stocké) pour répondre efficacement et rapidement aux besoins d'information des utilisateurs. Les structures d'index déterminent la méthode d'indexation et agissent directement sur les performances du système de recherche d'information. [Rijsbergen, 79]

Plusieurs structures ont été proposées, parmi elles on peut citer :

A. Les fichiers Séquentiels (Sequential files) :

Un fichier séquentiel est le moyen le plus simple de stocker un fichier de données puisque l'on stocke les enregistrements les uns à la suite des autres dans leur ordre d'insertion. Jusqu'au milieu des années 70, tous les systèmes textuels utilisaient les fichiers séquentiels du fait de l'utilisation de bandes magnétiques. Une requête sur un document consistait alors à parcourir toute la bande jusqu'à ce que l'on trouve le bon document, d'où une lenteur certaine du système malgré l'optimisation des algorithmes de recherche.

L'amélioration du processus était bloqué par le matériel (bandes magnétiques) ce qui changea radicalement avec l'arrivée d'un nouveau système de stockage plus rapide : le disque dur.

B. Les fichiers de signature (Signature files) :

Une signature de texte est une chaîne de bits (chaîne binaire) dans laquelle les bits sont positionnés pour décrire le document. La chaîne de bit est créée en appliquant une opération de hachage sur tous les mots clés décrivant un document. Une fois qu'un fichier signature est créé, la réponse à une requête consiste à calculer une chaîne pour la requête et à la comparer à celle calculée pour chaque document. Les manipulations de bits étant codées dans le processeur, cette technique est beaucoup plus rapide que la manipulation de chaînes de caractères. Cette méthode est donc très rapide, bien que le fichier signature ait besoin d'être parcouru entièrement à chaque requête.

Le principal désavantage de cette méthode est que le fait qu'un bit signale la présence d'un mot dans un fichier ne donne ni son emplacement, ni le nombre de fois qu'il apparaît dans ce fichier. Néanmoins, les résultats obtenus par cette méthode sont relativement bons.

C. Les fichiers inversés (Inverted files)

La structure de fichier inversé est à la base de tous les systèmes de recherche d'information. Un système à base de fichiers inversés contient trois composants principaux :

- Un dictionnaire : Le fichier dictionnaire contient tous les mots ou groupes nominaux spécifiques pouvant servir de mots-clés pour l'indexation et la recherche dans

l'ensemble des fichiers à traiter. A chaque entrée du dictionnaire est associé le nombre de fois où l'entrée apparaît dans l'ensemble documentaire.

- Un fichier de hachage : Ce fichier contient pour chaque entrée du dictionnaire une liste décrivant dans quel fichier apparaît cette entrée.
- Les fichiers de données : Qui représentent les documents du corpus à indexer.

La figure suivante est une représentation d'un système à base de fichiers inversés.

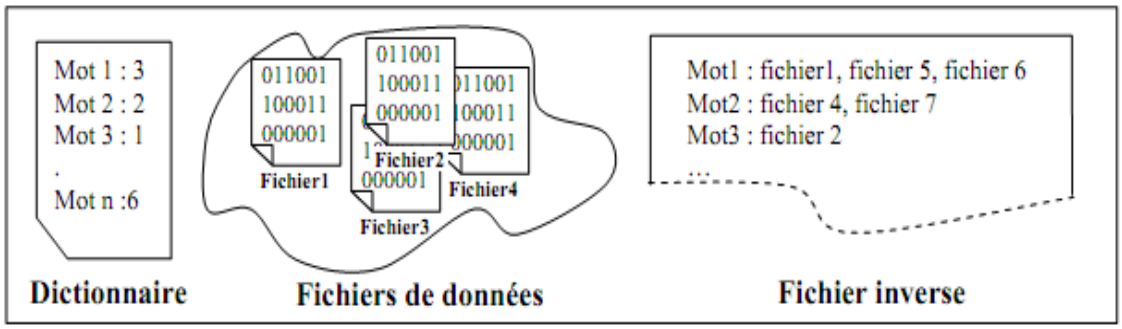


Figure1.4 : Système à base de fichiers inversés [Rijsbergen, 79].

Ce système de fichiers, s'il est rapide pour trouver un résultat, consomme énormément de place de stockage, les fichiers index étant parfois aussi gros que les fichiers de données, surtout dans le cas où les positions où apparaissent les mots clés dans les fichiers sont stockées. Les mises à jour sont aussi coûteuses puisqu'il faut refaire l'index à chaque ajout.

6.2. Processus de recherche :

Il représente le processus du noyau d'un SRI. Il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer. Les modèles de recherche représentent ce qui diffère le plus entre les SRI. Ils sont inspirés de concepts mathématiques afin de pouvoir évaluer certaines relations, notamment la relation d'appariement entre la requête et les documents. La problématique majeure des SRI est de retrouver les quelques dizaines ou milliers de documents pertinents parmi des millions de documents. Cet écart de cardinalité rend cette tâche encore plus difficile.

En plus des étapes de représentation et de recherche, quelques systèmes peuvent supporter une étape supplémentaire qui est :

6.3. La re-formulation de requête

L'utilisateur ne sait pas toujours choisir les bons termes qui expriment le mieux ses besoins d'information. En introduisant la reformulation de requête, la RI est alors envisagée comme une suite de formulations et de re-formulations de requêtes jusqu'à la satisfaction du besoin d'information de l'utilisateur, la requête initiale permettant rarement d'aboutir à un résultat qui satisfait ce dernier. Il s'agit en particulier d'ajouter des termes à la requête initiale de l'utilisateur et on parle alors d'expansion de la requête de l'utilisateur. On distingue trois niveaux permettant de différencier entre les techniques d'expansion de requêtes [Ihadjaden, 1994]:

- *La source des termes utilisés dans la reformulation* et qui peuvent provenir des résultats de recherches précédentes ou d'une base de connaissance (réseau sémantique, thesaurus).
- *Le choix de la méthode* ou de l'algorithme qui permet de sélectionner les termes à ajouter à la requête initiale.
- *Le rôle de l'usager* dans le processus de sélection des termes et qui peut être actif ou passif.

6.1. Reformulation manuelle

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

6.2. Reformulation interactive

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

6.3. Reformulation automatique de requêtes :

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

Cette reformulation intervient dans un processus plus général d'optimisation de la fonction de pertinence. Celle-ci a pour but de rapprocher la pertinence système de la pertinence utilisateur. Elle se présente comme une opération primordiale dans un SRI. En effet, compte tenu des volumes croissants des bases d'information, retrouver des informations pertinentes en utilisant seulement la requête initiale est une tache quasi-impossible [Voorhees 99].

La dimension de l'espace de recherche est élevée, c'est ainsi que la difficulté fondamentale de la reformulation de requêtes est la définition de l'approche à adopter en vue de réduire l'espace de recherche. Cette approche passe par la détermination de [Efthimiadis 96] :

- ✓ critères de choix des termes d'extension,
- ✓ règles de calcul des poids des nouveaux termes,
- ✓ hypothèse de base quant aux liens entre termes et documents.

Ces trois principales fonctions peuvent utiliser une ressource externe qui peut être une ontologie, une hiérarchie de concept ou le vocabulaire contrôlé d'un thesaurus ou des ressources internes qui seront détaillées dans le chapitre II.

7. Modèles de recherche d'information (RI) :

Si c'est l'indexation qui choisit les termes pour représenter le contenu d'un document ou d'une requête, c'est au modèle de leur donner une interprétation. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit les deux rôles suivants:

- ➢ créer une représentation interne pour un document ou pour une requête basée sur ces termes;
- ➢ définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité).

Le modèle joue un rôle central dans la RI. C'est le modèle qui détermine le comportement clé d'un système de RI. Dans ce chapitre, on décrit quelques modèles souvent utilisés dans la RI.

7.1. Le modèle booléen :

C'est le premier modèle utilisé en RI [Salton 71] ; il est basé sur la théorie des ensembles et l'algèbre de Boole [Gessler 93]. Le modèle booléen propose la représentation d'une requête sous forme d'une équation logique. Les termes d'indexation sont reliés par des connecteurs logiques ET, OU et NON.

L'approche booléenne consiste à trouver les documents qui ont exactement les mêmes termes qu'une requête construite par mots clefs. Les requêtes peuvent être affinées grâce aux opérateurs OR ou AND ou encore au moyen d'opérateurs comme NEAR. Ce type de recherche est à la base des moteurs de recherche comme Altavista1 ou Google2.

Cette approche est très efficace pour des requêtes utilisant des termes très spécifiques ou portants sur des domaines techniques particuliers avec leur vocabulaire propre mais son intérêt reste néanmoins limité.

L'inconvénient majeur du modèle booléen réside dans sa caractéristique de fournir une réponse binaire soit 1, soit 0 (les documents contiennent les termes demandés ou ne les contiennent pas). Ceci induit un volume de réponses important sans ordre spécifique des documents résultants.

-Tous les termes dans un document ou dans une requête étant pondérés de la même façon simple (0 ou 1), il est difficile d'exprimer qu'un terme est plus important qu'un autre dans leur représentation.

-Les formules de requêtes sont complexes, non accessibles à un large public. Elles nécessitent une maîtrise parfaite des opérateurs booléens, car leur signification est différente de celle qu'ils ont dans la langue naturelle.

Les deux modèles présentés ci-dessous largement utilisés en pratique, permettent de remédier à ces inconvénients.

7.2. Le modèle vectoriel :

Le modèle vectoriel introduit par Salton [Salton 71], repose sur les bases mathématiques des espaces vectoriels. Dans ce modèle, les documents et les requêtes sont représentés dans un espace vectoriel engendré par l'ensemble des termes d'indexation $t_1, t_2, t_3 \ldots, t_T$ où T est le nombre total de termes issus de l'indexation de la collection des documents.

Chaque document est représenté par un vecteur : $D_J = (d_{1j}, d_{2j}, \ldots d_{ij} \ldots, d_{Tj})$

Chaque requête est représentée par un vecteur $Q = (q_1, q_2 \ldots q_i \ldots, q_T)$

Avec : d_{ij} Poids du terme t_i dans le document D_j

q_i Poids du terme t_i dans la requête Q

Les termes de poids nul représentent les termes absents dans un document alors que les poids positifs représentent les termes assignés.

La fonction de calcul du coefficient de similarité entre chaque documentD_i , représenté par le vecteur$(d_1, d_2, \ldots d_{Tj})$ et la requête Q, représentée par le vecteur$(q_1, q_2 \ldots q_i \ldots, q_T)$ est appelée Retrieval Status Value ou RSV.

Ce coefficient de similarité est calculé sur la base d'une fonction qui mesure la colinéarité des vecteurs document et requête. On peut citer notamment les fonctions suivantes :

- **Produit scalaire :**

$$\boldsymbol{RSV(Q, D_j) = \sum_{i=1}^{T} q_i * d_{ij}}$$

- **Mesure de Jaccard** :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\sum_{i=1}^{T} {q_i}^2 \ + \sum_{i=1}^{T} {d_{ij}}^2 - \sum_{i=1}^{T} q_i * d_{ij}}$$

- **Mesure de cosinus** :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\left(\sum_{i=1}^{T} {q_i}^2\right)^{1/2} * \left(\sum_{i=1}^{T} {d_{ij}}^2\right)^{1/2}}$$

La similarité entre deux textes (requêtes ou documents) dépend ainsi des poids des termes coïncidant dans les deux textes. Il est donc possible de classer les documents par ordre de pertinence décroissante.

L'avantage du modèle vectoriel par rapport au modèle booléen réside particulièrement dans l'ordonnancement des documents sélectionnés selon leurs pertinences. Cependant, l'inconvénient majeur de l'approche vectorielle réside dans le fait que l'association entres les termes d'indexation n'est pas considérée. Il est impossible de représenter des phrases ou des mots multi termes. On considère effectivement que les termes sont indépendants [Yates 99].

7.3. Le modèle probabiliste :

Le modèle probabiliste a été proposé par Robertson et Sparck Jones [Robertson 76], il utilise un modèle mathématique fondé sur la théorie de la probabilité conditionnelle (appelé aussi modèle de la théorie de pertinence). Lors du processus d'indexation deux probabilités conditionnelles sont utilisées :

- *P(t/pert) :* La probabilité pour que le terme t apparaisse dans un document donné sachant que ce document est pertinent pour la requête
- *P(t/NonPert)* : La probabilité pour que le terme t apparaisse dans un document donné sachant que ce document est non pertinent pour la requête.

En supposant que la distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents, et que les variables "documents pertinents" et "document non pertinent" sont indépendantes, la fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document P (Pert/D).

$$P(Pert/D) = \frac{P(D/Pert) * P(Pert)}{P(D)},$$

$$P(NonPert/D) = \frac{P(D/NonPert) * P(NonPert)}{P(D)}$$

Avec :

$$P(D) = P(D/Pert) * P(Pert) + P(D/NonPert) * P(NonPert)$$

Où :

- *P (D/Pert*) (respectivement *P(D/NonPert)*) : Probabilité d'observer D sachant qu'il est pertinent (respectivement non pertinent).

-P(Pert) (respectivement P(NonPert)) : Probabilité à priori qu'un document soit pertinent (respectivement non pertinent).

Le coefficient de similarité requête document (RSV) peut être calculé par différentes formules.

Robertson et Spark-Jones [Robertson 96] proposent la formule suivante :

$$RSV = \sum log \left(\frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \right)$$

Où

- ✓ N: nombre total de documents de la base,
- ✓ n: nombre de documents contenant le terme,
- ✓ R: nombre de documents connus comme étant pertinents,
- ✓ r: nombre de documents connus comme étant pertinents et contenants le terme.

L'ajout de 0.5 à tous les membres s'explique par la nécessité d'écarter tous les cas limites qui entraîneraient des valeurs nulles de ces membres.

Ce modèle a donné lieu à de nombreuses extensions. Il est à l'origine du système OKAPI qui est l'un des systèmes les plus performants selon les compagnes d'évaluation TREC3 [Walker 97]. L'inconvénient majeur de ce modèle est que les calculs des probabilités sont complexes et que l'indépendance des variables n'est pas toujours vérifiée voir pas prise en compte.

7.4. Les modèles de langage

Le modèle de langage est emprunté de la linguistique informatique. Son objectif est de capter les régularités linguistiques d'un langage, en observant la distribution des mots, succession de mots dans le langage donné. Il désigne une fonction de probabilité qui assigne à chaque séquence de mots une probabilité de sa génération.

Dans le domaine de la recherche d'information, Ponte et Croft [Pont and Croft, 1998] l'ont introduit en premier. Il permet de déterminer la probabilité qu'une requête utilisateur Q puisse être générée par un modèle de langage d'un document spécifique. L'estimation de cette probabilité est plus en moins complexe, à cet effet l'indépendance des termes de la requête Q est généralement supposée; d'où la formule générale suivante :

$$P(Q/M_d) = \prod_{t \in Q} P(t \setminus M_d)$$

Après la recherche les documents seront rangés par l'ordre décroissant de cette probabilité.

Dans notre travail, nous utilisons le modèle de langage comme cadre pour implémenter notre approche. Le chapitre III décrit d'une manière détaillée ce modèle

.

8. Evaluation des performances des systèmes de recherche d'information

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. D'une façon générale, tout système de recherche d'information présente deux objectifs:

- ✓ retrouver tous les documents pertinents,
- ✓ rejeter tous les documents non pertinents.

Ces deux objectifs sont évalués par les mesures de précision et de rappel définis ci-dessous. Nous allons définir également les mesures à *x* documents et d'autres mesures de performance.

8.1. Les mesures de Précision/Rappel

Les mesures de précision/rappel sont obtenues en partitionnant l'ensemble des documents restitués par le SRI en deux catégories : les documents pertinents et les documents non pertinents. Ces deux catégories se définissent comme suit:

- **Taux de précision :** La précision mesure la capacité du système de rejeter tous les documents non pertinents à une requête. Il est donné par le rapport entre l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés.
- **Taux de rappel :** Le rappel mesure la capacité du système à retrouver tous les documents pertinents répondants à une requête. Il est donné par le rapport entre les documents retrouvés pertinents et l'ensemble des documents pertinents de la base.

Les taux de précision et de rappel sont donnés par les formulations suivantes :

$$\textbf{Précision} = \frac{R+}{M}$$

$$\textbf{Rappel} = \frac{R+}{R}$$

Où:

***R*:** le nombre total de documents pertinents dans la collection,

- ✓ ***M* :** le nombre de documents sélectionnés,
- ✓ ***R+* :** le nombre de documents pertinents sélectionnés.

La figure 1.5 illustre la précision et le rappel d'une requête d'une façon générale

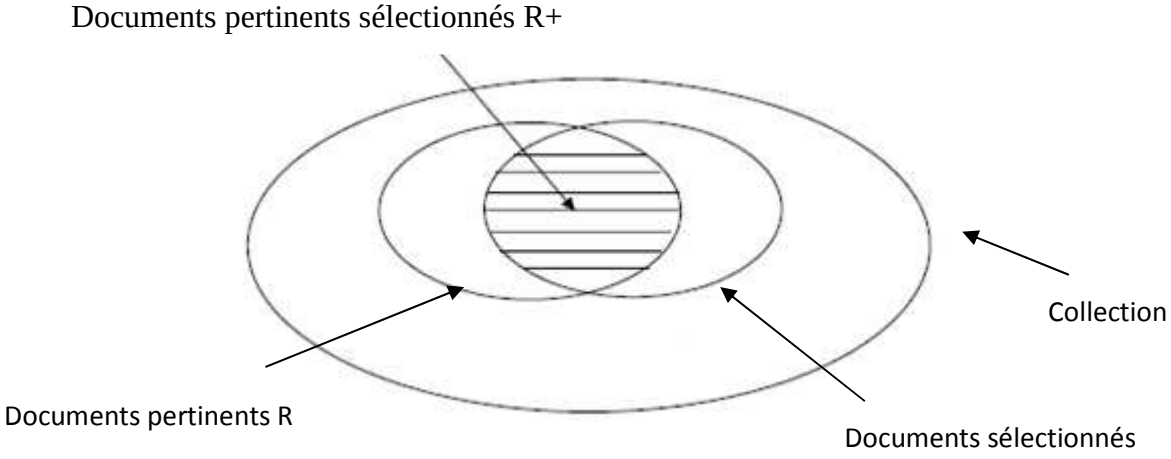


Figure. 1.5 – Exemple de rappel et de précision pour une requête

Le taux de rappel et le taux de précision évaluent respectivement les notions de bruit et de silence documentaire qui constituent les deux premiers indicateurs de performance d'un SRI.

La notion de silence et bruit présentent respectivement la taux de documents pertinents non sélectionnés et le taux de documents non pertinents sélectionnés.

8.2. Autres mesures de performance

- **Le temps de réponse acceptables :** un SRI doit pouvoir fournir à l'utilisateur les documents correspondants à sa demande dans des temps très courts.
- **La présentation des résultats claire et facilité d'utilisation :** capacité du système à comprendre les besoins de l'utilisateur et a mettre en valeur les documents correspondants a ceux-ci. Ceci est lie à l'interface avec l'utilisateur.
- **Le nombre total de documents pertinents retournés, ou le rappel à 1000 documents :** ces mesures permettant d'évaluer la performance globale du système au final, en fonction ou non du nombre de documents pertinents total.
- **La longueur de recherche :** elle est égale au nombre de documents non pertinents que doit lire l'utilisateur pour avoir un certain nombre *n* de documents pertinents.

8.3. Compagnes d'évaluation :

De nombreux projets basés sur des corpus d'évaluation se multiplient depuis les années 70.

On peut par exemple citer la compagne CLEF (Cross Language Evaluation Forum) ou la compagne d'évaluation TREC, Co-organisée par le NIST4 et la DARPA, a commencé en 1992. Elle a pour but d'encourager la recherche documentaire basée sur de grandes collections de test, tout en fournissant l'infrastructure nécessaire pour l'évaluation des méthodologies de recherche et de filtrage d'information.

9. Conclusion

Dans ce premier chapitre, nous nous sommes essentiellement intéressés à donner une présentation générale de recherche d'information qui désigne le processus qui permet, à partir d'une expression des besoins d'information d'un utilisateur, de retrouver l'ensemble des documents contenant l'information recherchée. Comme on a déjà illustré précédemment, les documents retournés par le SRI peuvent ne pas répondre au besoin de l'utilisateur. Pour prendre en compte cette difficulté, des techniques de reformulation (expansion) de la requête qui fait l'objet de deuxième chapitre sont utilisées, afin d'obtenir des requêtes optimales.

Chapitre II :

Reformulation de requête de l'utilisateur

1. Introduction :

L'un des problèmes-clés des systèmes de recherche d'informations (SRI) est la définition d'une fonction de correspondance entre la représentation du contenu sémantique des documents et la requête de l'utilisateur. Cette fonction doit modéliser la *pertinence* d'un document pour l'utilisateur. De fait, il existe deux formes de pertinence : la *pertinence système* et la *pertinence utilisateur*. La première correspond à la pertinence que le système a calculée, la seconde à la façon dont l'utilisateur juge de la pertinence des documents trouvés par le système pour son besoin d'information. Améliorer la qualité d'un système de recherche d'information (SRI) consiste donc à réduire la distance existant entre la *pertinence utilisateur* et la *pertinence système.* Les études pour réduire cette distance portent sur la difficulté de l'utilisateur pour définir ses objectifs de recherche. En effet, indépendamment du problème lié à la capacité qu'a un utilisateur d'exprimer ses besoins de façon précise, il existe un autre problème que G. Salton avait déjà souligné dans [salton, 83], dû au fait que les auteurs de documents et les utilisateurs de SRI utilisent une grande variété de mots pour exprimer le même concept.

De nombreux travaux visent à concevoir des systèmes de recherche d'information (SRI) capables de remédier aux problèmes suscités. La reformulation de la requête est sans doute la piste la plus investie dans ce contexte.

Dans ce chapitre, nous allons présenter les différentes techniques de reformulation de requête, nous citons quelques paramètres de performance inhérents à chacune des techniques de reformulation, ensuite nous détaillons les approches de reformulation de requêtes utilisées dans chacun des modèles classiques (vectoriel et probabiliste) et dans les modèles de langage, comme notre travail s'insère dans le cadre de ce dernier modèle, nous présentons en détaile les différents approches développées dans ce cadre .

2. La reformulation de requêtes

2.1. Définition:

L'utilisateur formule son besoin en information par une requête composée de ses propres mots clés et le choix de chaque terme a une influence directe sur l'ensemble des documents restitués par le système. La requête initiale seule est souvent insuffisante pour permettre la sélection de document répondant au besoin de l'utilisateur. Pour cela, une étape de

reformulation de la requête est souvent utilisée dans le but de retrouver plus de documents pertinents. Donc la reformulation de requête est un processus permettant de générer une requête plus adéquate à la recherche d'information dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est d'enrichir la requête initiale de l'utilisateur par ajout de termes significatifs et/ou la suppression des termes inutiles d'une part, et d'autre part la repondération des termes de la requête. Cette reformulation permet de coordonner le langage de recherche, utilisé par l'utilisateur dans sa requête et le langage d'indexation. Par conséquent, elle limite le bruit (les documents non pertinents retrouvés par le système de recherche) et le silence (défini par les documents pertinents non retrouvés) dus à un mauvais choix des termes d'indexation dans l'expression de la requête d'une part, et les lacunes du processus d'indexation d'autre part.

2.2. Les différentes techniques de reformulation de la requête

Dans la reformulation de la requête diverses sources sont utilisées pour choisir les termes à ajouter. Elle peut être basée sur le vocabulaire issu de ressources externes telles que les ontologies ou les Thesaurus (approche globale) ou par le calcule de degré de cooccurrence entre termes, ou bien les termes rajoutés proviennent des documents de la collection (retournés dans une première recherche), dans ce cas on parle de réinjection de pertinence (approche locale) (Xu et *croft.*, 96) :

2.2.1. Approche globale

L'idée de base des méthodes globales est d'ajouter à la requête initiale des termes issus de ressources externes existantes (ressources linguistiques) ou bien de ressources internes construites à partir de la relation de cooccurrence:

2.2.1.1. Utilisation de ressources externes (Méthodes linguistiques)

Dans le contexte de la recherche d'information (RI), déterminer la pertinence d'un texte vis-à-vis d'une requête d'utilisateur est une tâche qui intéresse l'analyse linguistique à divers titres. Les méthodes linguistiques sont en particulier sollicitées pour améliorer les capacités des systèmes à repérer des textes qui répondent au besoin informationnel de l'utilisateur mais l'expriment d'une autre manière. Le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. Il s'agit principalement de chercher des associations inter-termes extraites à partir des ontologies linguistiques (tel que WordNet [G. Miller.95]), ou à partir de thésaurus

[G. Brajnik, S. Mizzaro, C. Tasso. 96]. On parle alors de reformulation de requêtes basée sur les concepts (Concept-based Query Reformulation) [Khan, 00].

A. Expansion de requête avec des ontologies

Dans la littérature, de nombreux travaux récents portent sur l'utilisation d'ontologies pour l'aide à la recherche d'information L'une des définitions d'une ontologie les plus connues est comme suit: «*les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire* » [SAID L'HADJ L, 09]. Les ontologies s'appuient sur des ressources généralistes comme WordNet (qui couvre la grande majorité des noms, verbes, adjectifs et adverbes de la langue anglaise. les termes (simple ou composés) dans WordNet sont organisés en ensembles de synonymes appelés « Synsets ».), elles s'appuient également sur des ressources plus spécialisées comme MESH.

Buscaldi et al [D. Buscaldi, P. Rosso, 05] proposent d'étendre les requêtes des utilisateurs en s'appuyant sur la ressource WordNet et plus spécifiquement sur les relations de synonymie et de méronymie. La démarche d'expansion de requête suivie par Baziz [M Baziz, 05] consiste d'abord à détecter les termes de la requête qui renvoient à des concepts d'une ressource externe (WordNet, puis, de les étendre par des termes représentant d'autres concepts proches de ceux de la requête. Ces termes sont identifiés grâce aux liens sémantiques entre concepts qu'offre l'ontologie. Baziz ajoute un processus de désambiguïsation afin que l'expansion n'amène pas trop de bruit, en prenant en compte le contexte de la requête et en cherchant à identifier les concepts correspondant aux plus longs termes que l'on peut former à partir des mots de la requête. Ces méthodes se basent sur le fait que l'expansion à partir de connaissances explicites entre les termes permet d'étendre la requête de façon plus sémantique qu'une approche basée sur la co-occurrence des mots dans les textes. Ce même type d'hypothèse est à l'origine de l'indexation sémantique. Dans ce type d'indexation, les termes retenus pour représenter un document ne sont plus les termes issus des documents, mais les concepts associés à ces termes.

Voorhees [Voorhees, 94] ayant observé que la polysémie et synonymie des mots influent négativement sur la précision et le rappel, a proposé une procédure d'indexation automatique en utilisant WordNet. Elle est partie du principe qu'un groupe de mots utilisés dans un certain contexte a un sens plus précis (non ambigu) même si les mots le constituant sont isolément ambigus. Ainsi, elle a proposé une technique d'expansion des requêtes avec les sens. Pour ce faire, elle a utilisé les synsets correspondant aux noms dans WordNet et les relations

hiérarchiques. Pour désambiguïser une occurrence d'un mot ambigu, les synsets de ce mot sont classés selon une valeur de cooccurrence calculée entre le contexte de ce mot et un voisinage contenant les mots du synset dans la hiérarchie de WordNet.

Voorhees définit le voisinage du sens d'un mot contenu dans un synset "s" comme suit :

« *En considérant l'ensemble des synsets et les relations d'Hyperonymie et d'Hyponymie dans WordNet comme les sommets et les arcs dirigés d'un graphe, le voisinage d'un synset donné s, serait alors le plus large sous-graphe connexe qui contient s, qui contient seulement les descendants d'un ancêtre de s et ne contient aucun synset ayant un descendant qui inclut une autre instance d'un membre (un mot) de s* ».

Par exemple, à partir du fragment de la structure de WordNet donnée par la *figure 10,*

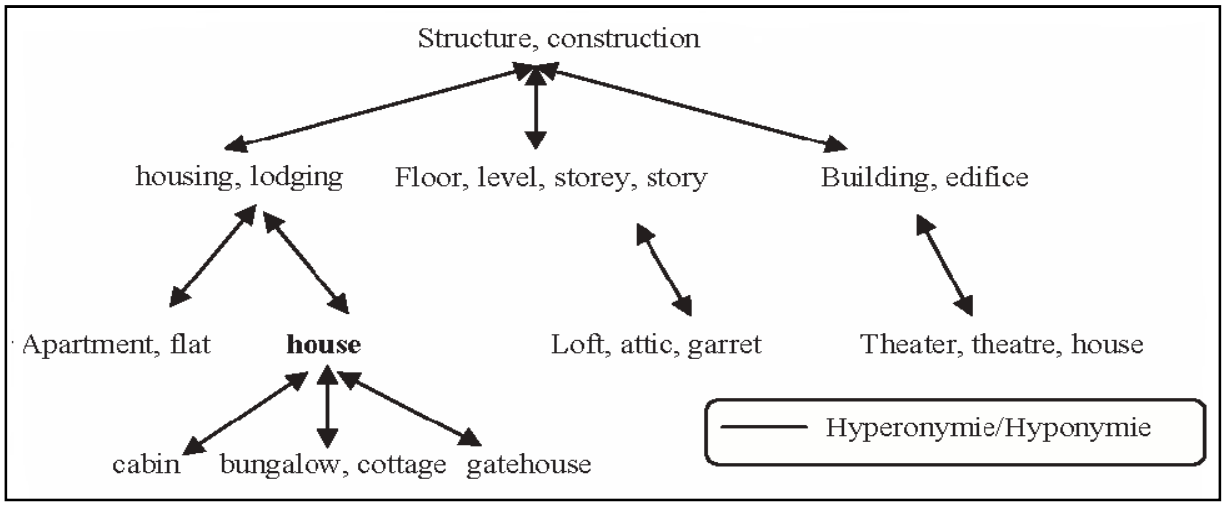


Figure 2.1: Exemple de voisinage de mot selon Voorhees

Le voisinage du premier sens de "house" inclurait les termes: housing, lodging, apartment, flat, cabin, gatehouse, bungalow, cottage. Les termes "structure" et "construction" (situés en haut de la hiérarchie), ne seraient pas inclus puisque un des descendants de leur synset contient un autre sens du terme "house".

B. Expansion de requête avec les thésaurus :

Le thésaurus est un outil classique en recherche d'informations : il consiste en la définition d'un certain nombre de termes du domaine, généralement appelés *concepts* et la représentation de *relations sémantiques.* Un thésaurus joue un rôle important dans un SRI. L'utilisation la plus courante est celle de l'expansion de requêtes, cela consiste à étendre la requête de l'utilisateur en ajoutant ou en remplaçant des termes par ceux du thésaurus pour

permettre de faire correspondre les thèmes exprimés dans une requête et l'indexation des documents qui a été réalisée avec d'autres termes. En effet, le thésaurus définit les relations entre les différents termes de l'index et permet de sélectionner de nouveaux termes à ajouter à la requête initiale. Plus les termes ajoutés sont pertinents, plus la performance du SRI est élevée.

L'évaluation d'un thésaurus est donc liée avec l'évaluation de la performance d'un SRI via l'extension de requête. Autrement dit, la performance d'un thésaurus en RI est évaluée par l'augmentation de la performance de recherche via l'expansion de requête. Le thesaurus regroupe plusieurs informations de type linguistique (équivalence, association, hiérarchie) et statistique (pondération des termes).

2.2.1.2. Utilisation de ressources internes (la relation de cooccurrence) :

La notion de cooccurrence fait référence au phénomène général par lequel des mots sont susceptibles d'être utilisés dans un même contexte. Autrement dit, on considère qu'il y a cooccurrence lorsque la présence d'un mot dans un texte donne une indication sur la présence d'un autre mot, exemple simple comme *«avion»* et *«aéroport»*, deux mots qui, selon toute vraisemblance, sont utilisés la plupart du temps dans un contexte commun, celui de l'aviation. Ces deux mots ne sont ni synonymes ni antonymes. On choisit donc de dire qu'ils sont cooccurrents.

Les méthodes basées sur la relation de cooccurrence ont été utilisées depuis les années 70. En alternative à l'utilisation de ressources généralistes, certaines études dérivent directement de la collection de documents les connaissances sémantiques ensuite exploitées dans le processus de recherche. L'utilisation de cooccurrences a notamment fait l'objet très tôt de plusieurs travaux [Salton et al., 68], [G, Cao, Nie, 05], [van Rijsbergen, 77].

Van Rijsbergen [Rijsbergen, 77] a proposé l'idée d'utiliser les statistiques de cooccurrence pour détecter la similarité sémantique entre les termes d'indexation et ceux qui ont été indiquées dans une requête d'utilisateur et l'exploiter ensuite pour étendre les requêtes des utilisateurs, Ex : Les termes *«recherche d'information »* et *«pertinence»* sont similaires à cause de la cooccurrence que l'on peut trouver dans les textes. On peut donc ajouter *«pertinence »* aux requêtes contenant *«recherche d'information »*.

La cooccurrence entre deux mots est vue comme étant la tendance de deux mots à apparaître à l'intérieur d'une même phrase, d'un même paragraphe, d'un même texte ou d'une même collection de textes, selon le cas. Il s'agit de chercher des associations de termes afin

d'ajouter des termes voisins à la requête. Le degré de cooccurrence est calculé à partir des méthodes essentiellement statistiques qui seront présentées, dans ce qui suit.

A. Mesures d'association utilisées :

Dans le cadre de l'approche statistique du traitement de la langue naturelle, plusieurs façons d'attribuer un score d'association à une paire de mots ont été élaborées. Parmi ces mesures, certaines se basent sur de solides fondements théoriques, alors que d'autres relèvent plutôt du domaine de l'heuristique. Certaines sont directement tirées de la discipline des statistiques alors que d'autres sont nées dans le domaine de la recherche d'information. Nous avons choisi d'étudier plus en profondeur deux de ces mesures. Le choix s'est arrêté sur celles-ci en fonction de leur popularité et de leur bonne performance dans certaines applications.

Pour chacune des mesures qui seront présentées ici, le calcul se base sur les données de la table de contingence (tableau 2.1), qui contient les fréquences d'apparition des paires de mots. C'est à partir des quatre valeurs présentes dans cette table que l'on va calculer le degré d'association entre deux mots : (*a*) le nombre de fois où les deux mots apparaissent ensemble, (*b*) le nombre de fois où le premier mot est présent sans que le deuxième ne le soit, (*c*) le nombre de fois où le deuxième mot est présent sans que le premier ne le soit, et (*d*) le nombre de fois où aucun des deux mots n'apparaît.

	Mot 2 présent	Mot 2 absent
Mot 1 présent	*a*	*b*
Mot 1 absent	*c*	*d*

Tableau 2.1 - Table de contingence pour mesurer le degré d'association entre deux mots

A. Test du χ^2

Cette mesure a été imaginée dans le cadre du problème classique en statistique qu'est la validation d'hypothèses. L'idée générale de la mesure du χ^2 est de comparer les fréquences observées dans un corpus avec les fréquences attendues s'il y avait indépendance. Si la différence entre les deux est grande, on peut rejeter l'hypothèse d'indépendance. Basée sur le tableau 4.1, l'équation qui traduit cette notion est la suivante :

$$x^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

où

- ***i*** couvre les rangées du tableau 4.1
- ***j*** couvre les colonnes du tableau 4.1
- O_{ij} est la valeur observée pour la cellule **(*i* , *j*)**
- E_{ij} est la valeur attendue pour la cellule **(*i* , *j*)**

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

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

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

$$\mathbf{X^2} = \frac{N(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)}$$

Quand la taille de l'échantillon est suffisamment grande, le test du χ^2 suit une distribution χ^2. On consulte une table de cette distribution pour connaître le degré de confiance associé à la valeur obtenue par le calcul. Par exemple, en haut de la valeur critique $\chi^2 = 3.841$, on peut rejeter l'hypothèse d'indépendance avec une confiance de 95%.

B. Information mutuelle

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

$$\boldsymbol{I(mot_1, mot_2) = \log_2 \frac{P(mot_1 \& mot_2)}{P(mot_1)P(mot_2)}}$$

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

$$\boldsymbol{I(mot_1, mot_2) = \log_2 \frac{Na}{(a+c)(a+b)}}$$

L'information mutuelle est considérée comme étant une bonne mesure de l'indépendance entre deux mots. Sa valeur devient nulle si les deux mots sont parfaitement indépendants. Mais pour des paires de mots parfaitement dépendants (qui apparaissent toujours ensemble), la valeur de l'information mutuelle augmente plus.

Une remarque importante à faire est que, l'efficacité de ces approches d'expansion de requêtes par cooccurrence est variable selon les travaux , mais aucune amélioration franche des résultats ne semble se dégager quelle que soit la collection de documents. H. Peat & P. [H. Peat & P. Wilett, 91] expliquent ce phénomène par le fait que les méthodes utilisées pour l'extraction des cooccurrences favorisent l'acquisition de termes approximativement de même fréquence. Or si les termes de la requête sont très fréquents, les termes ajoutés sont eux aussi trop fréquents pour être discriminants.

Une autre source d'information utilisée pour l'extension de la requête : est le retour de pertinence, dite approche locale qui sera expliqué dans ce qui suit.

2.2.2. Approche locale :

On parle de méthodes locales si les termes ajoutés proviennent des documents résultants de la première recherche, ou les jugements explicites ou implicites de l'utilisateur après une première recherche.

2.2.2.1. La reformulation par réinjection de pertinence :

Les chercheurs ont conclu que bien que les utilisateurs aient souvent la difficulté d'exprimer leurs besoins informationnels avec précision, ils pourraient identifier l'information utile quand elle leurs sera présentée. C'est-à-dire, qu'une fois que le système présente un premier ensemble de documents, ils peuvent facilement différencier entre les documents qui contiennent de l'information utile et ceux qui ne la contiennent pas. C'est ce qu'on appelle communément la réinjection de pertinence.

La reformulation de requête par réinjection de pertinence est plus connue sous le nom de *Relevance Feedback* [J.J. Rocchio, 71]. Cette méthode permet une modification de la requête initiale, sur la base des jugements de pertinence de l'utilisateur sur les documents restitués par le système. La relevance feedback est une forme de recherche évolutive et interactive. Son principe fondamental est d'utiliser la requête initiale pour amorcer la recherche d'information puis exploiter itérativement les jugements de pertinence de l'utilisateur afin d' « ajuster » la requête par expansion ou repondération. La nouvelle requête obtenue à chaque itération de feedback, permet de « corriger » la direction de recherche dans le fond documentaire, et ce, dans le sens des documents pertinents.

Trois méthodes essentielles existent et cela selon le type de jugement utilisé:

A. Réinjection automatique de pertinence :

Réinjection automatique de pertinence est connue sous le nom de *pseudo-réinjection* ou *blind Relevance Feedback* Dans ce cas, on applique le même principe de la réinjection de pertinence mais en considérant les *n* premiers documents renvoyés par le système comme pertinents [B. Croft, D. Harper, 79], [M. Mitra, A. Singhal, 98] (sans utiliser l'information issue des jugements de l'utilisateur). Plus précisément, le système de recherche restitue un ensemble de documents répondant à la requête initiale. Ainsi au lieu de juger explicitement

les documents, on suppose que les *k* premiers documents comme étant pertinents (documents pseudo-pertinents). On peut également considérer les documents qui sont restitués en fin de liste comme non pertinents. L'idée de base derrière la pseudo réinjection de pertinence est qu'une itération de réinjection basée sur les documents les plus similaires à la requête initiale de l'utilisateur pourrait donner une meilleure restitution des documents.

Cette technique a été développée la première fois par Croft & Harper [B. Croft and D. Harper. 79.], en tant qu'un moyen d'estimation des probabilités dans le modèle probabiliste pour une première recherche. Depuis, cette technique a été largement étudiée pour améliorer les classements des documents en particulier dans le cadre de TREC [S. Walker, S.E. Robertson, 97]. Croft et Harper ont également indiqué que cette méthode présente un inconvénient majeur : si une grande partie des documents les mieux classés sont non pertinents, alors les termes ajoutés à la requête sont susceptibles d'être sans rapport avec le contexte de la requête, et par conséquent, la qualité des documents retournés avec la nouvelle requête est susceptible d'être plus mauvaise.

La réinjection automatique peut être bénéfique si les requêtes initiales permettent de retrouver des documents pertinents, dans le cas contraire elle provoque une dégradation des performances.

Des chercheurs comme Mitra et *al.* [M. Mitra, A. Singhal, C. Buckley,98.] et [C. Buckley, all, 95.], ont essayé avec un certain succès de surmonter ce problème en améliorant le taux de précision dans les *k* meilleurs documents, c'est ce qu'on nomme habituellement la "haute précision".

Il est prouvé dans la majorité des travaux que la réinjection automatique présente une solution pratique pour l'amélioration des performances de la recherche en ligne sous un certain nombre de conditions. En particulier, c'est une technique très utile pour améliorer la recherche quand il s'agit de requêtes courtes ou de requêtes qui ne permettent pas de restituer assez de documents pertinents.

Pour répondre aux limites de cette technique, il est nécessaire de faire intervenir l'utilisateur dans le processus de réinjection de pertinence. Une approche qui permet la modification de la requête utilisateur d'une manière interactive sera détaillée dans ce qui suit.

- **Expansion interactive de requêtes**

Dans le cas des méthodes d'expansion automatique des requêtes décrites précédemment, les termes sont extraits à partir des documents et ajoutés en totalité à la requête. Une alternative est de permettre aux utilisateurs de choisir les termes pouvant être ajoutés : on

parle d'*Expansion Interactive des Requêtes* (EIR) [D. Harman, 88.]. L'utilisateur qui est le mieux placé pour déterminer la pertinence, a alors plus de contrôler sur les termes qui seront ajoutés à la requête. Cette technique est défendue par le fait que l'utilisateur peut mieux sélectionner les termes pertinents que le système.

B. Reformulation par les jugements explicites ou implicites de l'utilisateur :

D'après Ruthven et Lalmas [Ruthven, M. Lalmas. 03.], la majorité des techniques proposées en *RF* est basée sur la différence entre le contenu des documents pertinents et celui des documents non pertinents. Ces derniers se rapportent à deux groupes de documents :

1. ceux qui ont été jugés non pertinents explicitement par l'utilisateur ;
2. ceux qui n'ont pas été jugés par l'utilisateur. Ces documents sont soit non sélectionnés, l'utilisateur ne les a pas jugés, soit l'utilisateur les a rejetés implicitement sans fournir une évaluation de pertinence.

La *RF* utilisant le groupe des documents jugés explicitement non pertinents est appelée *RF* négative. D'après [Ruthven, M. Lalmas. 03], la *RF* négative présente une difficulté au niveau du traitement des informations négatives par le système. Une pratique courante en RI est de supprimer les termes ayant un poids négatif. Ces termes permettent plutôt la recherche de documents non pertinents que de documents pertinents.

La Réinjection de pertinence négative peut être utilisée pour indiquer les termes devant avoir un poids négatif.

Dans ce contexte, Belkin et Cool, dans une étude de la participation de l'utilisateur dans la réinjection de pertinence [N. J. Belkin, C. Cool, 96], proposent un modèle alternatif. Leur hypothèse est qu'un terme appartenant à un document pertinent ou à un document non pertinent peut être intéressant puisqu'il permet d'augmenter le nombre de documents pertinents (s'il appartient à un document pertinent) ou de diminuer le nombre de documents non pertinents (dans le cas contraire).

Le but de la réinjection de pertinence négative abordé par Sumner et al [R. G. Sumner, K. Yang, 1998.], était la suppression des documents non pertinents précédemment vus par l'utilisateur mais pouvant réapparaître dans la liste des résultats s'ils répondent à la nouvelle requête. Les expérimentations dans [N. J. Belkin, C. Cool, 97] ont montré que bien que les utilisateurs puissent utiliser la réinjection de pertinence négative, l'amélioration des performances n'est pas significative.

Le jugement de non pertinence est une tâche plus délicate que le jugement de pertinence [Ruthven, M. Lalmas. 03]. En général, un utilisateur qui juge un document pertinent donne

souvent des raisons détaillées, mais les raisons de la non-pertinence sont susceptibles d'être basées sur ce qui manque dans le document, plutôt que sur ce qui est présent.

2.2.2.2. Processus général de *RF*

Le processus de réinjection de pertinence, comme schématisé sur la figure 2.2, comporte principalement trois étapes : l'échantillonnage, l'extraction des évidences et la réécriture de la requête.

- ✓ *L'échantillonnage*: cette étape permet de construire un échantillon de documents à partir des éléments jugés par l'utilisateur. Cet échantillon est caractérisé par le nombre d'éléments jugés et le nombre d'éléments jugés pertinents.

- ✓ *L'extraction des évidences :* est l'étape la plus importante, elle consiste en général à extraire les termes pertinents qui serviront à l'enrichissement de la requête initiale. Plusieurs approches ont été développées, la plus reconnue est celle de Rocchio [J.J. Rocchio, 71] adaptée au modèle vectoriel (sera détaillé dans ce qui suit).

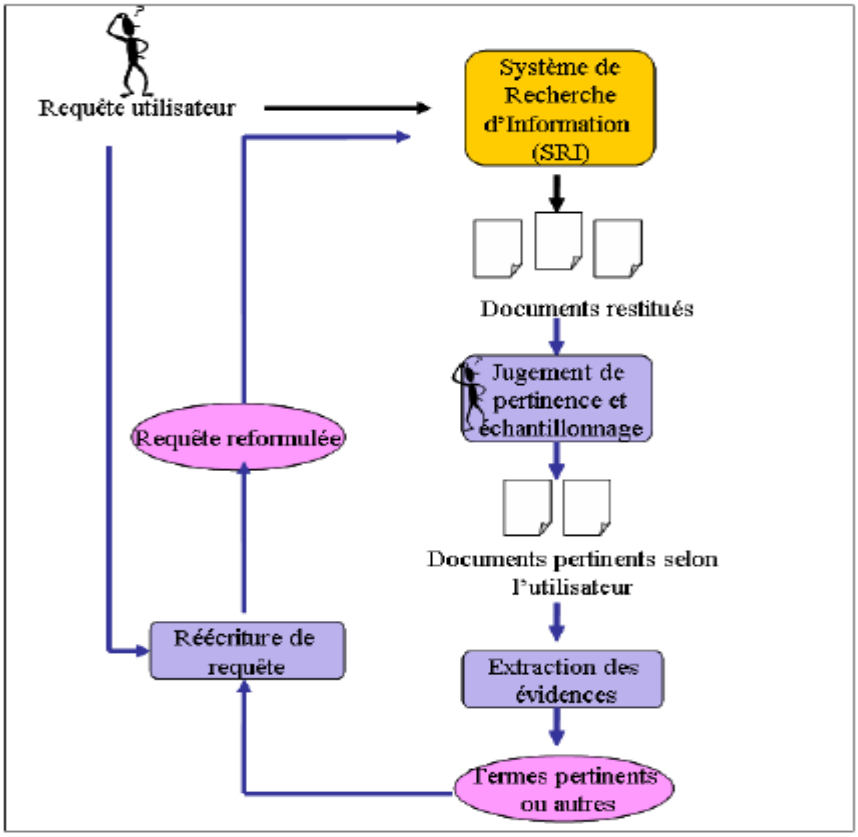


Figure 2.2. Le Processus général de la réinjection de pertinence [L.HLAOUA, 07]

- ✓ *La réécriture de la requête :* consiste à construire une nouvelle requête en Combinant la requête initiale avec les informations extraites dans l'étape précédente.

Le processus général de la réinjection de pertinence peut être renouvelé plusieurs fois pour une même séance de recherche : on parle alors de la réinjection de pertinence à itérations multiples.

2.3. Paramètres de performance [L. Tamine et al, 00, 00]

Un nombre considérable d'expérimentations 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. Une étude synthétique de ces différents travaux, permet de déduire que l'ordre des performances imputées à l'intégration de cette stratégie est variable, dépendant de diverses conditions d'exploitation :

- ✓ Modèle de recherche,
- ✓ Hypothèse de base quant à la distribution des termes dans les documents, sémantique d'un terme, concept, phrase et relation sémantique entre eux,
- ✓ Caractéristiques des collections de documents : taille, nombre, source etc…

En faisant abstraction des paramètres caractéristiques inhérents à chacune des techniques de reformulation de requête présentées, en dégageant dans ce qui suit, les paramètres de performance intrinsèques.

2.3.1. Nombre de termes ajoutés à la requête

Buckley & al [Buckley & al, 94] ont expérimenté la relevance feed-back dans l'environnement 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 :

$$\boldsymbol{RP(N) = A.\log(N_s) + B.\log(X) + C}$$

Où :

- ✓ $\boldsymbol{RP(N)}$: Performance du système pour N documents restitués
- ✓ ***Ns***: Nombre de documents restitués
- ✓ ***X*** : Nombre de termes ajoutés à la requête
- ✓ ***A, B, C*** : Constantes

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

Harman [Harman, 92] a par ailleurs montré, que la meilleure méthode de sélection des termes issues des documents pertinents devient inefficace après l'ajout de *20* à *40* termes à la requête initiale, sur des bases de tailles moyenne (CACM, Cranfield ...).

2.3.2. 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, 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.

- Harman [Harman, 92] propose les fonctions suivantes :

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

Où :

- ✓ $\boldsymbol{RT_j}$: Nombre total de documents retrouvés par la requête
- ✓ $\boldsymbol{df_i}$: Fréquence d'occurrence du terme **ti** dans la collection

- ✓ $\boldsymbol{N}$: Nombre total de documents dans la collection

$$\textbf{2. } \boldsymbol{SV(i) = \frac{r_i}{R} - \frac{df_i}{N}}$$

Où :

- ✓ $\boldsymbol{r_i}$**:** Nombre de documents pertinents contenant **ti**
- ✓ $\boldsymbol{R}$: Nombre de documents pertinents

$$\textbf{3. } \boldsymbol{SV(i) = \log_2 \frac{p_i(1-q_i)}{(1-p_i)}}$$

Avec :

- ✓ $\boldsymbol{p_i}$: Probabilité que ti appartienne aux documents pertinents
- ✓ $\boldsymbol{q_i}$: Probabilité que ti appartienne aux documents non pertinents

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

2.3.3. Longueur moyenne de requête

L'accroissement des performances est plus important lorsque les collections sont interrogées par des requêtes de longueur relativement petite [Buckley & al, 94].

Dans ce sens, des expérimentations intéressantes ont été réalisées sur la base TREC7 et présentées dans [Cormack, al, 99]. 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.

3. Reformulation de requête dans les modèles de recherche d'information classiques

Nous allons dans cette section détailler les approches de reformulation de requêtes utilisées dans chacun des modèles de la RI classiques (vectoriel et probabiliste).

3.1. Modèle vectoriel

Contrairement au modèle booléen qui permet d'extraire les documents qui satisfont exactement la requête, le modèle vectoriel permet d'extraire tout les documents qui

contiennent au moins un terme de la requête. Le classement de ces documents se fait par le biais des fonctions de mesure de similarité.

La première formalisation de la technique de réinjection dans le modèle vectoriel est celle de Rocchio [Rocchio, 71]. Ce dernier a défini le problème par la détermination de la requête optimale. Cette détermination peut être faite par l'ajout de nouveaux termes ou/et la repondération des termes de la requête initiale. L'équation originale de Rocchio est :

$$\boldsymbol{Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{n1} R_i - \frac{1}{n_2}\sum_{i=1}^{n2} S_i}$$

Où :

- ✓ $\boldsymbol{Q_0}$: Représente le vecteur de la requête initiale ;
- ✓ $\boldsymbol{Q_1}$: Représente le vecteur de la nouvelle requête ;
- ✓ $\boldsymbol{n_1}$: Représente le nombre de documents pertinents ;
- ✓ $\boldsymbol{n_2}$: Représente le nombre de documents non pertinents ;
- ✓ $\boldsymbol{R_i}$: Représente le vecteur du $\boldsymbol{i^{éme}}$ document pertinent ;
- ✓ $\boldsymbol{S_i}$: Représente le vecteur du $\boldsymbol{i^{éme}}$document non pertinent ;

La nouvelle requête contienne de nouveaux termes (à partir des documents jugés pertinents), et de nouveaux poids pour les termes de la requête initiale. Une variante de cette formule a été implémentée sous le système SMART [Rocchio, 71]. Cette formule a obtenue de bons résultats.

Ide [Ide, 71] a testé plusieurs aspects de la technique de réinjection de pertinence sur le système SMART [Salton, Buckley, 90]. Parmi ces aspects : l'utilisation des documents pertinents uniquement dans le processus de feedback et la variation du nombre de documents utilisés dans le feedback. Une variation de la formule de Rocchio est celle appelée : « IDE-DEC-HI ». Cette formule n'utilise que le premier document non pertinent trouvé mais aucune performance n'a été mise en évidence.

$$\boldsymbol{Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{nr} R_i S_i}$$

Où :

$\boldsymbol{n_r}$ **:** Représente le nombre de documents pertinents

Une nouvelle variante de formule de Rocchio a été proposée. Elle permet de donner un degré d'importance (α, β et δ) à chacun des composants utilisés dans le processus de reformulation de requêtes.

$$\boldsymbol{Q_1 = \alpha Q_0 + \frac{\beta}{n_1}\sum_{i=1}^{n1} R_i - \frac{\delta}{n_2}\sum_{i=1}^{n2} S_i}$$

3.2. Modèle probabiliste

Dans son principe, le modèle probabiliste est dédié à la reformulation de requêtes. Au départ de la recherche, aucune information de pertinence n'est disponible pour estimer qu'un document est pertinent pour un des termes de la requête. La solution de ce problème est d'utiliser une des fonctions de pondération (exemple : tf.idf) pour la recherche initiale. Ensuite, après avoir obtenu la première liste ordonnée de documents, la fonction suivante peut être utilisée.

$$\boldsymbol{W_i = \log \frac{r_i/(R-r_i)}{(n_i-r_i)/(N-n_i-R+r_i)} \cdot \left(\frac{r_i}{R} - \frac{n_i-r_i}{N-R}\right)}$$

Oû :

- ✓ $\boldsymbol{r_i}$ **:** Le nombre de document pertinent cotenans le terme **i**
- ✓ $\boldsymbol{n_i}$: Le nombre de documents contenants le terme **i**
- ✓ $\boldsymbol{R}$: Le nombre de documents pertinents à la requête
- ✓ $\boldsymbol{N}$: Le nombre de documents dans la collection

4. La reformulation de requêtes dans le cadre de modèles de langage :

Durant ces dernières années, un autre type de modèles est apparu : les modèles de langue. Ils se distinguent des autres modèles classiques par leur fondement mathématique solide : *la théorie des probabilités*. Ils traduisent la pertinence d'un document vis-à-vis d'une requête en mesurant la probabilité qu'une requête puisse être générée par le modèle de langue du document.

4.1. Idée de base sur les modèles de langage :

Dans les modèles classiques de recherche, on cherche à mesurer la similarité entre un document $\boldsymbol{d_i}$ et une requête $\boldsymbol{q}$ ou à estimer la probabilité que le document réponde à la requête ($\boldsymbol{P(Q_j/D_k)}$**.** L'hypothèse de base dans ces modèles consiste à dire qu'un document

n'est pertinent que s'il "ressemble" à la requête. Les modèles de langage, comme décrits par leurs initiateurs Ponte et Croft [Ponte et al., 98], sont eux basés sur une hypothèse différente. Cette hypothèse admet qu'un utilisateur en interaction avec un système de recherche d'information, fournit une requête en pensant à un ou plusieurs documents qu'il souhaite retrouver. Par conséquent, un document n'est pertinent que si la requête de l'utilisateur ressemble à celle inférée, par le système pour ce document. On cherche alors à estimer la probabilité que la requête soit inférée par le document ($\boldsymbol{P(Q_j/D_k)}$**.**

On désigne Par « modèle de langue » une fonction de probabilité **P** qui assigne une probabilité **P(S)** à un mot ou à une séquence de mots **S** en une langue [Boughanem & al. , 04]. Une fois cette fonction définie, il est possible d'estimer la probabilité de générer cette séquence de mots à partir du modèle de la langue.

Considérons la séquence **S** composée des mots indépendants suivants : $\boldsymbol{m_1}$, $\boldsymbol{m_2}$,…. $\boldsymbol{m_n}$. Un modèle général d'unigram est formulé comme suit :

$$\boldsymbol{P(S) = \prod_{i=1}^{n} P(m_i)}$$

Il est difficile d'estimer ces probabilités pour une langue dans l'absolu. Cette estimation ne peut se faire que par rapport à un corpus d'apprentissage ou de référence. Ainsi le modèle de langue peut être approximativement celui du corpus.

Ces probabilités sont plus justes quand le corpus de référence et de grande taille. Cependant, aussi grand soit il, le corpus de référence ne peut couvrir la totalité d'une langue.

Pour les mots absents, l'estimation de leurs probabilités sera nulle. Par conséquent, on affectera une probabilité nulle à toute séquence de mots ou phrase contenant ce mot.

Ce problème est appelé « sous représentation des données » [Boughanem & al. , 04].

Pour pallier ce problème, on utilise une technique qu'on l'appelle : le lissage.

4.1.1 Le lissage *:* Dans sa définition la plus simple, le lissage consiste à affecter une probabilité non nulle (enlevée de la masse des probabilités des n-grammes observés) aux mots non rencontrés dans le corpus. Sur la façon d'enlever et de redistribuer une partie de masse de probabilité, il y a une série de méthodes proposées dans la littérature. [Stanly & Goodman, 98] On peut citer quelques techniques de lissage les plus connues dans le tableau suivant *:*

Méthode de Lissage	**Formule**	**Description**
Lissage de Laplace	$P_{ajouter-un}(\alpha/C) = \frac{\lvert\alpha\rvert+1}{\sum_{\alpha_i \in V}(\lvert\alpha_i\rvert+1)}$	V: ensemble du vocabulaire d'indexes $\lvert\alpha\rvert$: nombre d'occurrences du n-grammes α C: correspond au corpus -Pas de performance sur les corpus de petite taille.
Lissage Good-Turing (GT)	$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_i \in C}(\lvert\alpha_i\rvert)}$ $r^* = (r+1)\frac{n_{r+1}}{n_r}$	r: fréquence d'occurrence d'un n-grammes α n_r: nombre de n-grammes apparus r fois -recommandée pour les n-grammes de faibles fréquences, car l'estimation GT est instable les n-grammes de grandes fréquences.
Jelinek et Mercer (JM) ou Lissage par interpolation	$P_{JM}(m_i/m_{i-1}) = \lambda_{m_{i-1}} P_{ML}(m_i/m_{i-1}) + (1-\lambda) P_{JM}(m_i)$ - En RI, cette méthode prend un autre sens : $(1-\alpha)P(w/d) + \alpha P(w/C)$	α: ici est un paramètre déterminé pour maximiser l'espérance des données. w: correspond à un mot observé dans le document ou dans la collection -Interpolation du modèle du document avec celui de la collection

Dirichlet	$\frac{c(w,d)+\mu P(w/C)}{\sum_w c(w,d)+\mu}$	$c(w,d)/P(w/C)$fréquence d'apparition du mot w dans le document d /ou dans la collection C. μ est un paramètre appelé pseudo fréquence. On peut aisément remarquer qu'il s'agit d'une généralisation du lissage de Laplace

Tableau 2.2 : Les différentes techniques de lissage

4.2. Approches des modèles de langage :

Le principe des approches utilisant un modèle de langue est différent. On ne tente pas de modéliser directement la notion de pertinence dans le modèle; mais on considère que la pertinence d'un document face à une requête est en rapport avec la probabilité que la requête puisse être générée par le modèle de langue du document. Ainsi, on considère qu'un document $\boldsymbol{D}$ incarne un sous-langage, pour lequel on tente de construire un modèle de langue $\boldsymbol{M_D}$**.** Le score du document face à une requête $\boldsymbol{Q}$ est déterminé par la probabilité que son modèle génère la requête :

$$\boldsymbol{Score(Q,D) = P(Q,M_D)}$$

Il est aussi possible de construire un modèle de langue pour le document $\boldsymbol{P(.|M_D)}$ et un autre pour la requête $\boldsymbol{P(.|M_Q)}$. Le score d'un document face à la requête peut être déterminé par une comparaison entre les deux modèles. C'est le principe utilisé dans la méthode d'entropie croisée.

4.2.1. Modèle de Hiemstra :

Hiemstra traite la requête comme une séquence de mots indépendants [Lavrenko, 01] et la probabilité de générer une requête **Q** sachant le modèle de document est calculée selon la formule :

$$\boldsymbol{P(Q/M_d) = \prod_{i=1}^{n} P(m_i/M_d)^{tf(m_i,Q)}}$$

Où

$\boldsymbol{tf(m_i, Q)}$ est la fréquence du terme $\boldsymbol{m_i}$ dans la requête **Q**.

L'intégration de l'élément $\boldsymbol{tf(m_i, Q)}$ - la fréquence du terme $\boldsymbol{m_i}$ dans la requête **Q**- dans ce modèle était pour mieux modéliser la requête, dans laquelle certains mots peuvent être non pertinents.

Aussi pour l'estimation de $\boldsymbol{P(m_i/M_d)}$, Hiemstra emploie l'approche par interpolation qui combine le modèle de document avec le modèle de la collection. Il introduit un coefficient $\boldsymbol{\lambda_{m_i}}$ pour chaque terme $\boldsymbol{m_i}$ de la requête afin d'estimer son importance [Tebri, 04].

Le calcule de la probabilité interpolée est alors exprimée par :

$$\boldsymbol{P(m_i/M_d) = \lambda_{m_i} P_{MLE}(m_i/D) + (1 - \lambda_{m_i}) P_{MLE}(m_i/C)}$$

L'interpolation dans ce modèle est appliquée sans effectuer aucun lissage sur le modèle de document, tel que Hiemstra utilise directement la technique d'estimation par le maximum de vraisemblance pour le calcul de :

$$\boldsymbol{P_{MLE}(m_i/D) = \frac{tf(m_i)}{\sum_{t_j \in d} tf(t_j)}} \quad \text{et} \quad \boldsymbol{P_{MLE}(m_i/C) = \frac{tf(m_i)}{|c|}}.$$

4.2.2. Modèle basé sur l'entropie croisée :

Une variante du modèle, basé sur le ratio de vraisemblance, il permet de représenter la recherche d'information (RI) comme une entropie croisée (ou écart d'entropie) .

L'équation de ration de vraisemblance est la suivante :

$$\boldsymbol{LR(D, Q) = \frac{P(D/Q)}{P(D)} = \frac{P(Q/D)}{P(Q)}}$$

Sous l'hypothèse d'indépendance des termes et en considérant une fonction logarithmique. Pour passer de l'équation du ratio vers une forme entropique, des transformations ont été faites, cette équation peut s'écrire de la façon suivante :

$$\boldsymbol{LR(Q|D) = \log \frac{P(Q|M_D)}{P(Q|M_C)}} \quad (1)$$

En supposant une distribution multinomiale de termes dans le document et dans le corpus :

$$\boldsymbol{P(Q|M_D) = \frac{|Q|}{\prod_{t_i \in Q} tf(t_i, Q)} \prod_{t_i \in Q} P(t|D)^{tf(t_i, Q)}}$$

$$\boldsymbol{P(Q|M_C) = \frac{|Q|}{\prod_{t_i \in Q} tf(t_i, Q)} \prod_{t_i \in Q} P(t|C)^{tf(t_i, Q)}} \quad (2)$$

Où $\boldsymbol{|Q|}$ est la taille de la requête (le nombre d'occurrences de mots). Ainsi, nous avons:

$$\boldsymbol{LR(Q|D) = \sum_{i=1}^{n} tf(t_i, Q) * \log \frac{\alpha P(t_i|M_D) + (1-\alpha) P(t_i|M_C)}{P(t_i|M_C)}} \quad (3)$$

Où $\boldsymbol{P(t_i|M_C)}$ est la probabilité générative de la requête étant donné un modèle de langage estimé sur un corpus $\boldsymbol{C}$ et $\boldsymbol{tf(t_i, Q)}$ est la fréquence du terme $\boldsymbol{t_i}$ dans la requête.

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

Les scores dans l'équation dépendent de longueur de la requête, ils peuvent être facilement normalisés en les divisant par la longueur de requête (comme la longueur est constante, elle n'intervient pas dans le score). La forme normalisée de l'équation 3 peut donc s'exprimer comme suit :

$$\boldsymbol{NLR(Q|D) = \log \frac{P(Q|M_D)}{P(Q|M_C)}}$$

$$\boldsymbol{= \sum_{i=1}^{n} \frac{tf(t_i, Q)}{\sum_i tf(t_i, Q)} * \log \frac{\alpha P(t_i|M_D) + (1-\alpha) P(t_i|M_C)}{P(t_i|M_C)}} \quad (4)$$

L'étape suivante est de considérer le rapport $\frac{\boldsymbol{tf(t_i,Q)}}{\sum_i \boldsymbol{tf(t_i,Q)}}$, (comme une estimation du maximum de vraisemblance de la distribution de probabilité représentant la requête $\boldsymbol{P(t_i|M_Q)}$**.** L'équation 4 peut être réinterprétée comme la relation entre les deux distributions de probabilité $\boldsymbol{P(t|M_D)}$ et $\boldsymbol{P(t|M_Q)}$, normalisées par la troisième distribution $\boldsymbol{P(t|M_C)}$:

$$\boldsymbol{NLR(Q|D) = \sum_{i=1}^{n} P(t_i|Q) * \frac{P(t_i|M_D)}{P(t_i|M_c)}} \qquad (5)$$

Le modèle mesure de combien le modèle de document pourrait coder des événements du modèle de requête mieux que le modèle de corpus. En théorie de l'information, ceci est interprété comme une différence entre deux entropies croisées [Lavrenko, 01], exprimée comme suit :

$$\boldsymbol{NLR(Q|D) = H(X|C) - H(X|D)} \qquad (6)$$

Où $\boldsymbol{X}$ est une variable aléatoire avec la distribution de probabilité $\boldsymbol{P(t_i) = P(t_i|M_Q)}$ et $\boldsymbol{C}$ et $\boldsymbol{D}$ sont des fonctions de masse de probabilité représentant respectivement la distribution marginale (du corpus) et le modèle du document.

L'entropie croisée permet donc de mesurer en moyenne l'écart entropique sur le fait que le modèle de document correspond (suit) bien la distribution probabiliste de la requête. Remarquons que dans l'entropie croisée, on voit apparaître implicitement le modèle de langue de la requête $\boldsymbol{M_Q}$.

4.2.3. Le modèle de Berger et Lafferty (approche basée sur traduction) :

La modélisation de langage est une approche très efficace dans le domaine de recherche de l'information. Toutefois, les modèles précédents ne traitent pas toutes les formes et styles de requêtes, ni les problèmes de synonymies et de polysémies de terme : multiples termes partageant des significations semblables et le même terme ayant des significations multiples [Berger et Lafferty, 99].

Pour aborder ces problèmes, la recherche d'information est traitée comme un processus de traduction statistique, qui traduit la requête utilisateur **Q** en un document **D**.

Cette idée peut être exprimée comme suit :

1. L'utilisateur a un besoin d'information **B** ;
2. A partir de ce besoin, l'utilisateur génère un fragment d'un document $\boldsymbol{D_B}$ supposé idéal ;
3. Il choisit un ensemble de mots clés à partir du fragment $\boldsymbol{D_B}$ et génère sa requête.
   La tâche du processus de recherche d'information est alors de trouver les documents similaires au fragment $\boldsymbol{D_B}$ et le calcul de leurs probabilités sachant l'utilisateur et sa requête est donné selon la loi de Bayes :

$$\boldsymbol{P(D/Q\,,U)} = \frac{\boldsymbol{P(Q/D\,,U)P(D/U)}}{\boldsymbol{P(Q/U)}}$$

La probabilité $\boldsymbol{P(Q/U)}$ est une constante fixe pour une requête et un utilisateur donnés, qu'on peut ignorer dans la classification des documents. Donc la probabilité de pertinence d'un document **D**, face à une requête **Q** est donnée par :

$$\boldsymbol{P(D/Q\,) = P(Q/D)P(D)}$$

Pour le calcul de la probabilité $\boldsymbol{(Q/D)}$, Berger et Lafferty se sont inspiré du modèle combinatoire (mixture model) de l'approche de traduction statistique, proposée par les travaux d'IBM [Berger et Lafferty, 99]. Leur modèle est le résultat d'une combinaison d'un modèle de langage de document $\boldsymbol{l(d_j/D\,)}$ avec un modele de traduction $\boldsymbol{d(m_i/d_j\,)}$ qui permet la traduction des termes $\boldsymbol{d_j}$ (j varie de 1 …r) du document **D** en termes $\boldsymbol{m_i}$ (i varie de 1…n), genérant ainsi la requete **Q**, ce qui donne d'une manière simplifiée :

$$\boldsymbol{P(D/Q\,) = \emptyset(n)\prod_{i=1}^{n}\sum_{j=1}^{r} t(m_i/d_j)\, l(d_j/D\,)}$$

Avec : $\boldsymbol{\emptyset(n)}$ la probabilité de générer une requête de longueur **n** ;

r le nombre de termes dans le document

Pour l'appliquer à la recherche d'information, le lissage du modèle de Berger et Lifferty est effectué selon la technique d'interpolation linéaire ; combinant un modèle unigramme $\boldsymbol{P(m_i)}$ avec un modèle basé sur la traduction statistique, donné selon la formule :

$$\boldsymbol{P(D\,, m_1, \dots, m_n) = P(D\,)\prod_{i=1}^{n} \lambda\, P(m_i/D) + (1-\lambda)P(m_i)}$$

Avec :

$$\boldsymbol{P(m_i/D) = \sum_{j=1}^{r} t(m_i/d_j)\ l(d_j/D\ )}$$

$\boldsymbol{P(D\ )}$ la probabilité de pertinence à priori de document.

4.3. Les extensions du modèle uni-gramme(bi-gramme et n-gramme):

Pour aller au delà de l'hypothèse de l'indépendance entre termes des extensions du modèle uni-gram sont proposés, ces modèles se nomment n-gramme leurs principe est le suivant :

Pour la prédiction d'un mot $\boldsymbol{m_i}$, le calcule ne tient compte que des n-1 mots qui le précèdent immédiatement (directement), donc l'historique $\boldsymbol{h_i}$ du mot $\boldsymbol{m_i}$ est:

$$\boldsymbol{h_i = \{m_{(i-(n-1))} \dots \dots m_{i-1}\}}$$

D'où :

$$\mathbf{P}\ \boldsymbol{(m_i/m_1\ , m_2\ , \dots , m_{i-1}) =\ P(m_i/m_{i-n+1}\ , \dots , m_{i-1})}.$$

Si **n**=2 alors ce modèle est dit bigramme : suppose que chaque mot de la séquence dépend de son prédécesseur direct. L'historique d'un mot $\boldsymbol{m_i}$ prend en compte son précédent $\boldsymbol{m_{i-1}}$:

$$bi-grammes: \qquad \boldsymbol{P(S) = \prod_{i=1}^{n} P(m_i/m_{i-1})}$$

Si **n**=3 alors ce modèle est dit trigramme : suppose que chaque mot de la séquence dépend de ses deux prédécesseurs directs.

$$Tri-grammes: \qquad \boldsymbol{P(S) = \prod_{i=1}^{n} P(m_i/m_{i-2}, m_{i-1})}$$

La valeur de n dépasse rarement 3 en pratique [Yannick, 02].

4.4. Expansion de la requête dans le modèle de langage :

Le modèle de langage est un nouveau cadre probabiliste pour la description du processus de la Recherche d'Information (RI) [Ponte et *al.*, 98]. Les résultats obtenus avec ce modèle ont montré des performances équivalentes voire supérieures à celles des modèles classiques (vectoriel, probabiliste).

La formule qui calcule le score d'un document est basée sur sa probabilité de générer la requête **Q**. Généralement cette formule est utilisée pour l'estimation de modèle de document. Afin d'intégrer d'autres informations pour l'estimation de modèle de la requête, Lafferty et al [Lafferty et *al.,* 01] ont proposé une autre implémentation de la fonction de calcul de score basée sur la mesure de divergence de Kullback-Leibler (KL-divergence) entre le modèle de la requête et celui de document.

4.4.1. Le modele Kullback-Leibler (KL-divergence) :

Une autre formulation populaire de modèle de langage dans RI est la Kl-divergence (Kullback-Leibler) [Croft et Lafferty, 02].Dans ce cas, un modèle de document est estimé, est ainsi un modèle de la requête. Le score est alors déterminé par Kl-divergence entre les deux modèles comme suit :

$$\mathbf{Score(Q,D)} = \sum_{\mathbf{t_i \in V}} \mathbf{P(t_i|Q)} \times \mathbf{\log P(\, t_i|D)} \qquad (1)$$

Où $\boldsymbol{V}$ est le vocabulaire de la collection, $\boldsymbol{M_Q}$ et $\boldsymbol{M_D}$ sont respectivement des modèles de langue pour la requête $\boldsymbol{Q}$ et pour le document $\boldsymbol{D}$.

Dans des études précédentes) $\boldsymbol{P(t_i|Q)}$ souvent est directement déterminé par l'Estimation de Maximum Likelihood (ML) c.-à-d.:

$$\boldsymbol{P(t_i|Q)} = \boldsymbol{tf(t_i\,,Q)/\sum_{t_i} tf(t_i\,,Q)}$$

Où $\boldsymbol{tf(t_i\,,Q)}$ est la fréquence de terme $\boldsymbol{t_i}$ dans la requête $\boldsymbol{Q}$, de cette façon, la somme peut être restreinte pour les termes de la requête seulement, c.-à-d :

$$\mathbf{Score(Q,D)} = \sum_{\mathbf{q_i \in Q}} \mathbf{P(q_i|Q)} \times \mathbf{\log P(\, q_i|D)} \qquad (2)$$

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

Nous pouvons observer que les deux formules exigent toujours que les termes de la requête doivent être apparaît dans le document pour que le dernier soit recherché. Mai avec le lissage du modèle de document, on peut augmenter la probabilité $\boldsymbol{P(q_i|D)}$ d'un terme $\boldsymbol{q_i}$ absent dans $\boldsymbol{D}$ de zéro à une petite valeur.

4.4.2. Estimation du modèle de la requête dans le modèle de langage

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

4.4.2.1. Approches basées sur la traduction :

Dans cette approche Bai et al [Bai et al., 05] examinent comment un terme de la requête est important dans un document (c.-à-d. le document est approprié à la requête).

D'une part si le terme de la requête est contenu dans le document, donc ce dernier est approprié à la requête à un certain degré. D'autre part, si un terme de la requête n'apparait pas dans un document ne signifie pas nécessairement que ce dernier n'est pas approprié. Il peut y avoir quelques autres termes dans le document qui sont fortement liées au terme de la requête, par exemple, synonymes. Dans ce deuxième cas, le document peut encore être jugé approprié dans une certaine mesure par les relations entre termes.

En ce qui concerne la formule (1) dans la section 4.4.1, l'expansion de requête consiste à trouver une meilleure manière d'estimer le modèle de la requête $\boldsymbol{P(t_i|Q)}$, de sorte que non seulement les termes exprimés dans la requête aient une probabilité différente de zéro, mais également d'autres termes relatifs. Tandis que cette idée est intuitive, elle n'a pas été entièrement incorporée au cadre de modèle de langage.

Une utilisation naïve de lissage par un modèle de collection a comme conséquence un grand modèle de requête (avec tous les termes ayant la probabilité différente de zéro), de ce fait augmentant le coût de la fonction de calcul entre la requête et les modèles de document.
La solution est de lisser le modèle original $\boldsymbol{P_{ML}(t_i|Q)}$ de requête par une autre fonction $\boldsymbol{P_R(t_i|Q)}$ de probabilité déterminée en utilisant les relations entre termes :

$$\boldsymbol{P(t_i|Q) = \lambda P_{ML}(t_i|Q) + (1-\lambda) P_R(t_i|Q)} \quad (3)$$

Avec : $\boldsymbol{\lambda}$ est un paramètre de lissage.

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

$$\boldsymbol{Score(Q,D)} = \sum_{t_i \in V} \boldsymbol{P(t_i|Q)} \times \log \boldsymbol{P(t_i|D)}$$

$$= \sum_{t_i \in V} [\boldsymbol{\lambda P_{ML}(t_i|Q)} + (\mathbf{1}-\boldsymbol{\lambda})\, \boldsymbol{P_R(t_i|Q)}] \times \log \boldsymbol{P(t_i|D)}$$

$$= \boldsymbol{\lambda} \sum_{q_i \in Q} \boldsymbol{P_{ML}(q_i|Q)} \times \log \boldsymbol{P(q_i|D)} + (\mathbf{1}-\boldsymbol{\lambda}) \sum_{t_i \in V} \boldsymbol{P_R(t_i|Q)} \times \log \boldsymbol{P(t_i|D)} \quad (4)$$

Où : **V** est l'ensemble de vocabulaire de la collection, $\boldsymbol{Q}$ la requête.

Noter que le premier terme est une somme sur des mots de requête (au lieu de tout le vocabulaire).C'est parce que $\boldsymbol{P_{ML}(t_i|Q) = 0}$ pour $\boldsymbol{t_i \notin Q}$. Le deuxième terme correspond au processus classique d'expansion de requête, dans lequel quelques (nouveaux) termes $\boldsymbol{t_i}$ liés à la requête **Q** sont déterminés, et leurs probabilités dans le modèle de document sont employées dans l'évaluation de la requête.[Zhai et al ,01] [Bai et al, 05]

Quoi que les rapports utilisés, il est important de limiter le nombre de termes $\boldsymbol{t_i}$ à considérer comme dans le deuxième terme de la formule (5) afin de rendre l'approche plus informatique et efficace.

4.4.2.2. Approche basé sur le retour de pertinence

Zhai et al [Zhai et *al.*, 01] ont proposé un modèle basé sur l'approche de feedback qui peut être incorporé dans le cadre de KL-divergence présenté dans [Lafferty et al, 01]. Ils travaillent avec les documents de retour de pertinence, et estiment un modèle de requête qui peut être employé pour mettre à jour un modèle existant de la requête.

Dans ce cas, ils ont proposé une approche basée sur la minimisation de la KL-divergence au-dessus des documents de feedback pour mettre à jour le modèle de langage de requête.

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

Une stratégie différente pour estimer un modèle de requête basé sur des documents de feedback est de réduire au minimum la divergence entre le modèle et les documents de feedback.

Soit $\boldsymbol{F = (d_1, d_2, \ldots d_n)}$ un ensemble de documents de feedback.

La définition de KL-divergence entre le modèle de requête $\boldsymbol{\theta}$ et $\boldsymbol{F}$ l'ensemble des documents de feedback est comme suit :

$$\boldsymbol{D_e(\theta; F) = \frac{1}{|F|} \sum_{i=1}^{n} D(\theta||\widehat{\theta}_{d_i})}$$

C'est la divergence moyenne entre la distribution empirique des mots de chaque document ($\widehat{\boldsymbol{\theta}}_{\boldsymbol{d_i}}$).

Le modèle estimé de requête sera près de chaque modèle de document de feedback, cependant, les documents de feedback partagent beaucoup de mots communs dus à la langue et les caractéristiques du domaine. En préférant un modèle qui encourt une plus grande divergence en ce qui concerne le modèle de collection, qui est une approximation du modèle de langue pour le contenu non pertinent.

Incorporant cette condition, la fonction de divergence d'un modèle de requête de feedback est la suivante :

$$\boldsymbol{D_e(\theta; F, C) = \frac{1}{|F|} \sum_{i=1}^{n} D(\theta||\widehat{\theta}_{d_i}) - \lambda D(\theta||P(w|C))}$$

Avec $\boldsymbol{\lambda} \in [0,1]$ contrôle le poids sur le modèle de langue de collection, et $\boldsymbol{P(w|C)}$ est le modèle de langue de collection. Réduire au minimum cette divergence est équivalent à maximiser l'entropie du modèle sous une contrainte de préférence codée dans le deuxième terme.

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

$$\boldsymbol{P\left(w\middle|\widehat{\theta}_F\right)} \quad \boldsymbol{\propto \exp\left(\frac{1}{(1-\lambda)}\frac{1}{|F|}\sum_i \log P(w|\,\widehat{\theta}_{d_i}) - \frac{\lambda}{1-\lambda} log P(w|C)\right)}$$

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

Le paramètre $\boldsymbol{\lambda}$ contrôle le poids sur le modèle de langue de collection, quand $\boldsymbol{\lambda}$ est mise à zéro, l'effet du modèle de langue de collection est ignoré, et nous avons alors un modèle de requête qui réduit au minimum la divergence sur des documents de feedback.

Avant d'exploiter $\hat{\theta}_F$ dans le modèle de recherche de KL-divergence, il faut d'abord l'interpoler avec le modèle original de requête $\widehat{\boldsymbol{\theta}}_{\boldsymbol{Q}}$ pour obtenir un modèle mis à jour $\hat{\theta}_{Q'}$, ensuite sélectionnant un document d par $\boldsymbol{D}(\widehat{\boldsymbol{\theta}}_{\boldsymbol{Q'}}||\widehat{\boldsymbol{\theta}}_{\boldsymbol{d}})$

4.4.2.3. Approches basées sur la dépendance des termes :

Lv et al [Lv et *al.*, 09] présentent une approche qui permet de choisir efficacement parmi des documents de feedback les mots qui sont concentrés sur thème de requête, cette approche est l'exploitation de la position des termes dans des documents de feedback.

Ils ont proposé un modèle de pertinence basé sur la position des termes (PRM).

- **modèle de pertinence (PRM) :**

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

PRM est une distribution polynôme qui essaye de capturer la probabilité que le terme $\boldsymbol{w}$ est vu dans un document approprié. Cependant, PRM estime la probabilité conditionnelle en termes de probabilité commune d'observer $\boldsymbol{w}$ avec la requête.

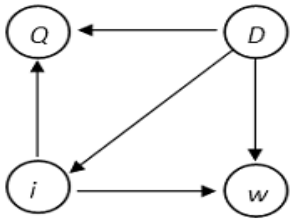


Figure 2.3: réseau de la dépendance, pour estimer le modèle basé sur la position de pertinence.

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

$$\boldsymbol{P(w|Q)} = \frac{\boldsymbol{P(w,Q)}}{\boldsymbol{P(Q)}} \propto \boldsymbol{P(w,Q)} = \sum_{\boldsymbol{D\in F}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{|D|}} \boldsymbol{P(w,Q,D,i)} \quad (1)$$

Avec $\boldsymbol{i}$ indique une position dans le document $\boldsymbol{D}$, et $\boldsymbol{F}$ est l'ensemble de documents de feedback.

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

La génération du mot $\boldsymbol{w}$ et celui de la requête $\boldsymbol{Q}$ sont indépendante, on a alors :

$$\boldsymbol{P(w,Q,D,i)} \propto \frac{\boldsymbol{P(w,Q|D,i)}}{\boldsymbol{|D|}} = \frac{\boldsymbol{P(Q|D,i)P(w|D,i)}}{\boldsymbol{|D|}} \quad (2)$$

Remplaçant l'équation (2) dans l'équation (1), on obtient la formule suivante :

$$\boldsymbol{P(w|Q)} \propto \boldsymbol{P(w,Q)} \propto \sum_{\boldsymbol{D\in F}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{|D|}} \frac{\boldsymbol{P(Q|D,i)P(w|D,i)}}{\boldsymbol{|D|}} \quad (3)$$

$\boldsymbol{P(w|D,i)}$ est la probabilité de prélèvement du mot $\boldsymbol{w}$ à la position $\boldsymbol{i}$ dans le document $\boldsymbol{D}$.
Pour une meilleure efficacité du modèle, ils ont simplifié le calcul de la probabilité $\boldsymbol{P(w|D,i)}$ comme suit :

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

Le terme $P(Q|D,i)$ dans l'équation (3) est la composante clé en estimant le modèle de pertinence. C'est la probabilité de Likelihood de la requête à la position $\boldsymbol{i}$ du document $\boldsymbol{D}$.
$\boldsymbol{P(Q|D,i)}$ peut mesurer les poids relatifs de positions dans un document : une position plus près des mots de la requête génère plus probablement la requête, et en conséquence un terme qui se produit à cette position recevrait un poids plus élevé.

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

$$\boldsymbol{P(w|D,i)} = \frac{\boldsymbol{c'(w,i)}}{\sqrt{\boldsymbol{2\pi\sigma^2}}} \qquad (4)$$

Avec $\boldsymbol{c'(w,i)}$ est la fréquence globale propagée de terme $\boldsymbol{w}$ à la position **i** des occurrences de $\boldsymbol{w}$ dans toutes les autres positions.

Suivant [Yuanhua, 09] , l'estimations $\boldsymbol{c'(w,i)}$ en utilisant la fonction de Gaussian kernel:

$$\boldsymbol{c'(w,i)} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{|D|}} \boldsymbol{c(w,j)} \ \mathbf{exp}\ [\frac{\boldsymbol{-(i-j)^2}}{\boldsymbol{2\sigma^2}}] \qquad (5)$$

Avec $\boldsymbol{i}$ et $\boldsymbol{j}$ sont les positions absolues des termes correspondants dans le document, et $|\boldsymbol{D}|$ est la longueur du document ; $\boldsymbol{c(w,j)}$ est la fréquence observée de $\boldsymbol{w}$ à la position $\boldsymbol{j}$.

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

$$\boldsymbol{P_\lambda(w|D,i)} = \boldsymbol{(1-\lambda)P(w|D,i) + \lambda P(w|C)} \qquad (6)$$

Avec ¸ $\boldsymbol{\lambda} \in [0\,,1\,]$ est un paramètre de lissage et $\boldsymbol{P(w|C)}$ est le modèle de langue de la collection. Enfin pour le calcul de la probabilité $\boldsymbol{P(Q|D,i)}$, on utilise la formule suivante :

$$\boldsymbol{P(Q|D,i)} = \prod_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{P_\lambda(q_j|D,i)} \qquad (7)$$

En remplaçant l'équation (7) dans l'équation (3), on obtient le modèle utilisé pour le calcul de pertinence.

Conclusion :

La reformulation de requête est une phase importante dans les systèmes de recherche d'information. Elle consiste à modifier la requête initiale de l'utilisateur par l'ajout de termes significatifs et/ou la ré-estimation de leur poids.

Dans ce chapitre nous avons présenté les techniques de reformulation de requête, les différentes ressources utilisées pour choisir les termes à ajouter soit externes|internes ou bien à partir des jugements de pertinence de l'utilisateur sur les documents restitués par le système , ensuite nous avons étudié la reformulation de requête dans les modèles traditionnels et dans le modèle de langage et comme notre travail s'insère dans le cadre de ce dernier, une grande partie lui a été consacrée.

Les techniques traditionnelles de la RI, considèrent un document comme un ensemble de termes simples, Cependant, l'une des critiques formulées à l'utilisation ces termes comme unité d'indexation est que le contenu d'un document ne peut pas être capturé précisément par un ensemble de mots clés indépendants, la solution proposée est d'utiliser des unités d'indexation plus complexes (termes composés) en plus de l'utilisation des termes simples pour une représentation plus détaillée du contenu des documents et des requêtes.

Dans ce contexte le chapitre suivant aura comme contribution la reformulation de requête basée sur des termes composés.

Chapitre III :

Approche d'expansion de la requête basée sur les termes composés

1. Introduction :

La plupart des systèmes de recherche d'informations (RI) utilisent des mots simples pour l'expansion de requête de l'utilisateur. Il est reconnu que cette technique est insuffisante à cause de l'ambiguïté des mots isolés et du non prise en compte des relations entre les mots.
Pour remédier à ce problème, il est suggéré que les termes composés, au lieu de mots simples, seraient la solution.

Dans notre approche on considère certaines relations entre termes de la requête. Plus explicitement on considère une requête comme un ensemble de termes composés et un ensemble de termes simple sous l'hypothèse suivante : « les termes qui appariassent prés des termes composés sont plus précis ».
En considère la relation de cooccurrence pour le choix des termes d'expansion.

2. Approche proposée:

Premièrement, on présente la formalisation de notre modèle. On considère une requête $\boldsymbol{Q}$ représentée dans le vocabulaire $\boldsymbol{V = \{T_1, \dots, T_m, t_1, \dots, t_n\}}$ contenant des termes simples $\boldsymbol{t_i}$ et des termes composés $\boldsymbol{T_j}$**.** Les termes composés $\boldsymbol{T_j}$ peuvent être formés par deux ou plusieurs mots non vides adjacents.

Pour le calcul de score d'un document $\boldsymbol{D}$ vis-à-vis d'une requête $\boldsymbol{Q}$, on utilise la mesure de divergence de Kullback-Leibler (KL-divergence) exprimée ainsi :

$$\boldsymbol{Score(Q, D) = \sum_{W \in V} P(w/Q) log P(w/D)}$$

L'estimation de modèle de la requête $\boldsymbol{P(w/Q)}$ et de modèle de document $\boldsymbol{P(w/D)}$ sont décrit dans les sections suivantes.

2.1. Estimation de Modèle de la requête

On formule le modèle de la requête $\boldsymbol{P(w/Q)}$ comme une mixture entre le modèle original de la requête $\boldsymbol{P_{ML}(w/\theta_Q{}^O)}$ et le modèle considérant la relation de cooccurrence

$\boldsymbol{P}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}})$ afin de reformuler la requête initiale avec les termes originaux et les termes avec lesquels ces derniers cooccurent les Il est exprimé ainsi :

$$\boldsymbol{P}(\boldsymbol{w}/\boldsymbol{Q}) = \boldsymbol{\varphi} * \boldsymbol{p_{ML}}(\boldsymbol{w}/\ \boldsymbol{\theta_Q}^{\boldsymbol{o}}) + (\ \mathbf{1} - \boldsymbol{\varphi}) * \boldsymbol{p}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}}) \qquad [1]$$

Où : $\boldsymbol{\varphi} \in [0\ ,1]$ est un paramètre qui contrôle l'apport de modèle de la requête original par rapport au modèle considérant la relation de cooccurrence.

La fonction de score devient alors:

$$\boldsymbol{score}(\boldsymbol{Q},\boldsymbol{D}) = \sum_{\boldsymbol{w}\epsilon\boldsymbol{V}}(\boldsymbol{\varphi} * \boldsymbol{P_{ML}}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{o}}) + (\mathbf{1} - \boldsymbol{\varphi}) * \boldsymbol{p}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}}))\ \mathbf{log}\ \boldsymbol{P}(\boldsymbol{w}/\boldsymbol{\theta_D}) \qquad [2]$$

$$\boldsymbol{score}(\boldsymbol{Q},\boldsymbol{D}) = \boldsymbol{\varphi} * \sum_{\boldsymbol{w}\epsilon\boldsymbol{Q}} \boldsymbol{P_{ML}}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{o}})\boldsymbol{logP}(\boldsymbol{w}/\boldsymbol{\theta_D}) +$$

$$(\mathbf{1} - \boldsymbol{\varphi}) * \sum_{\boldsymbol{w}\epsilon\boldsymbol{V}} \boldsymbol{p}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}})\boldsymbol{logP}(\boldsymbol{w}/\boldsymbol{\theta_D}) \qquad [3]$$

On note que le premier terme de la formule est une sommation à travers les termes de la requête (non pas à travers tous les termes du vocabulaire), car $\boldsymbol{P_{ML}}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{o}}) = \mathbf{0}$ pour tout terme n'appartenant pas à la requête.

Pour des raisons de calcul, on considérera qu'un sous ensemble de $\boldsymbol{V}$. On note cet ensemble $\boldsymbol{G}$; ou $|\boldsymbol{G}| = \boldsymbol{N}$; où $\boldsymbol{N}$ est le nombre de termes à ajouter à la requête originale. Par conséquent la formule [3] devient ainsi :

$$\boldsymbol{score}(\boldsymbol{Q},\boldsymbol{D}) = \boldsymbol{\varphi} * \sum_{\boldsymbol{w}\epsilon\boldsymbol{Q}} \boldsymbol{P_{ML}}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{o}})\boldsymbol{logP}(\boldsymbol{w}/\boldsymbol{\theta_D}) +$$

$$(\mathbf{1} - \boldsymbol{\varphi}) * \sum_{\boldsymbol{w}\epsilon\boldsymbol{G}} \boldsymbol{p}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}})\boldsymbol{logP}(\boldsymbol{w}/\boldsymbol{\theta_D}) \qquad [4]$$

2.1.1. Estimation de modèle de la requête basé sur la cooccurrence $P(w/\theta_Q{}^{coo})$:

En considérant tous les termes de la requête (simples et composés), le modèle de la requête $\boldsymbol{P}(\boldsymbol{w}/\boldsymbol{\theta_Q}^{\boldsymbol{coo}})$ basé sur la relation de cooccurrence est exprimé ainsi:

$$P(\boldsymbol{w}/\theta_Q{}^{coo}) = \sum_{t_i \epsilon Q} P_{coo}(w/t_i) * \boldsymbol{P}(\boldsymbol{t_i}/\boldsymbol{Q}) + \sum_{T_j \epsilon Q} P_{coo}(w/T_j) * \boldsymbol{P}(\boldsymbol{T_j}/\boldsymbol{Q}) \quad [5]$$

Avec :

- ✓ $\boldsymbol{P_{coo}}(\boldsymbol{w}/\boldsymbol{t_i})$: la probabilité de cooccurrence d'un terme $\boldsymbol{w}$ sachant un terme simple $\boldsymbol{t_i}$ appartenant à la requête originale.
- ✓ $\boldsymbol{P_{coo}}(\boldsymbol{w}/\boldsymbol{T_j})$: la probabilité de cooccurrence d'un terme $\boldsymbol{w}$ sachant un terme composé $\boldsymbol{T_j}$ appartenant à la requête originale.
- ✓ $\boldsymbol{P}(\boldsymbol{t_i}/\boldsymbol{Q})$: le poids d'un termes simple dans la requête originale $\boldsymbol{Q}$.
- ✓ $\boldsymbol{P}(\boldsymbol{T_j}/\boldsymbol{Q})$: le poids d'un terme composé dans la requête originale $\boldsymbol{Q}$

A. Estimation des probabilités $P(t_i/Q)$ et $P(T_j/Q)$:

On estime ces deux probabilités ainsi :

$$\boldsymbol{P}(\boldsymbol{t_i}/\boldsymbol{Q}) = \frac{|\boldsymbol{t_i}|}{|\boldsymbol{Q}|} \quad [6]$$

De même :

$$\boldsymbol{P}(\boldsymbol{T_j}/\boldsymbol{Q}) = \frac{|\boldsymbol{T_j}|}{|\boldsymbol{Q}|} \quad [7]$$

Où :

- ✓ $|\boldsymbol{T_j}|$: fréquence de terme dans composé la requête originale × 2 (2= taille de $\boldsymbol{T_j}$)
- ✓ $|\boldsymbol{t_i}|$: fréquence de terme simple dans la requête originale.
- ✓ $|\boldsymbol{Q}|$: taille de la requête originale.

B. Estimation de la probabilité $P_{coo}(w/T_j)$:

L'estimation de la probabilité d'un terme w sachant un terme composé T_j appartenant à la requête originale est exprimée ainsi :

$$P_{coo}(w/T_j) = \alpha P_{coo\_int}(w/T_j) + (1-\alpha) P_{coo}(w/c(T_j)) \quad [8]$$

Où :

$c(T_j)$: est l'ensemble des termes simples qui composent le terme T_j , et α est un facteur d'interpolation pour contrôler l'apport d'un terme composé par rapport à ses termes composants.

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

$$P_{coo}(w/T_j) = \alpha P_{coo\_int}(w/T_j) + (1-\alpha) \sum_{t_k \in T_j} P_{coo\_int}(w/t_k)\, P(t_k/T_j) \quad [9]$$

Oû :

- ✓ $P_{coo\_int}(w/T_j)$: la probabilité initiale de cooccurrence d'un terme w sachant un terme composé T_j appartenant à la requête originale calculée avec la formule [18].
- ✓ $P_{coo\_int}(w/t_k)$: la probabilité initiale de cooccurrence d'un terme w sachant un terme simple qui appartient au terme composé T_j appartenant à la requête originale calculée avec la formule [18].
- ✓ $P(t_k/T_j)$: la probabilité d'un terme simple t_k qui appartient au terme composé T_j calculée avec la formule [11].

❖ **Estimation de la probabilité $P(t_k/T_j)$**

Pour le calcul de cette probabilité on pose l'hypothèse suivante : « les termes composants d'un terme composé ne sont pas de même importance ». L'un des termes

peut être plus important que d'autres. Exemple : le terme *«ordinateur»* est plus important que le terme *«personnel»* dans le terme composé *«ordinateur personnel»*.
On considère intuitivement que la dominance d'un terme est déterminée par sa spécificité, on propose de l'estimer de la manière suivante:

$$\boldsymbol{imp(t_k) = N/df} \qquad [10]$$

Où $\mathbf{t_k}$ est un terme simple. Ainsi on calcule la probabilité $\boldsymbol{P(t_k/T_j)}$ comme suit :

$$\boldsymbol{P\left(t_k/T_j\right) = \frac{imp(t_k)}{\sum_{t_i \in T_j} imp(t_i)}} \qquad [11]$$

Où $\boldsymbol{df}$ est le nombre de documents où le terme ***t*** apparaît, et $\boldsymbol{N}$ le nombre de documents dans la collection **C**.

C. Estimation de la probabilité $P_{coo}(w/t_i)$:

Pour calculer cette probabilité, on s'est basé sur l'intuition suivante: On suppose que l'utilisateur lorsqu'il utilise un terme simple dans sa requête, il fait référence généralement, à un ou plusieurs concepts « terme composé ». Par exemple un utilisateur qui utilise le terme *«énergie»* dans sa requête peut faire référence au terme composé *«énergie électrique»*.

On propose donc de lisser la probabilité $\boldsymbol{P_{coo}(w/t_i)}$ en tenant compte de la probabilité de cooccurrence de terme $\boldsymbol{w}$ dans l'ensemble des termes composés auxquels le terme $\boldsymbol{t_i}$ appartient. La probabilité $\boldsymbol{P_{coo}(w/t_i)}$ est exprimée comme suit :

$$\boldsymbol{P_{coo}(w/t_i) = \beta P_{coo\_init}(w/t_i) + (1-\beta)P_{coo}(w/C(t_i))} \qquad [12]$$

Où :

- ✓ $\boldsymbol{P_{coo\_int}(w/t_i)}$: la probabilité initiale de cooccurrence d'un terme $\boldsymbol{w}$ sachant un terme composé $\boldsymbol{t_i}$ appartenant à la requête originale calculée par la formule [18].

- ✓ $C(t_i)$ est l'ensemble des termes composés auxquels le terme t_i appartient. Et β est un facteur d'interpolation pour contrôler l'apport d'un terme simple par rapport aux termes composés auxquels il appartient.
- ✓ $P_{coo}(w/C(t_i))$: la probabilité de cooccurrence d'un terme w sachant l'ensemble des termes composés auxquels le terme simple t_i appartient calculée avec la formule [13].

De même, que la formule [8], on utilise alors une méthode de translation [Berger et *al.*, 99]. Ainsi la seconde partie de la formule est exprimée comme suit :

$$P_{coo}(w/C(t_i)) = \sum_{t_i \in T_l} P_{coo\_ini}(w/T_l)\, P(T_l/t_i) \qquad [13]$$

Où :

- ✓ T_l: terme composé où le terme simple t_i appartient.

- ❖ **Estimation de la probabilité $P(T_l/t_i)$:** En appliquant le théorème de Bayes on obtient :

$$P(T_l/t_i) = \frac{P(t_i/T_l)P(T_l)}{P(t_i)} = P(t_i/T_l) * P(T_l) \qquad [14]$$

Tel que $P(t_i) = 1$, $P(t_i/T_l)$ est calculée en utilisant la formule [11] et $P(T_l)$ est calculée ainsi :

$$P(T_l) = \frac{1/idf(T_l)}{\sum_{T_m \in C(t_i)} (1/idf(T_m)} \qquad [15]$$

Le calcul de cette probabilité est basé sur l'hypothèse suivante : « on affecte une grande probabilité aux termes composés qui apparaissent plus dans l'ensemble des premiers

documents pertinents retournés de la première recherche, car l'utilisateur utilise généralement les termes les plus fréquents ».

De plus, si on pose l'hypothèse suivante : « lorsque l'utilisateur utilise un terme simple dans sa requête, il fait allusion à un terme composé donné ». On considère que ce terme composé est le plus fréquent dans la l'ensemble des premiers documents pertinents retournés de la première recherche et il est plus probable d'être exprimé par le terme simple seul. On note ce terme composé $\boldsymbol{T_{max}}$, il est sélectionné comme suit:

$$\boldsymbol{T_{max}} = \boldsymbol{argmax}_{\forall \boldsymbol{T_l} \in \boldsymbol{V} \wedge \mathbf{t_i} \in \boldsymbol{T_l}} \boldsymbol{P}(\boldsymbol{T_l}/\mathbf{t_i}) \quad [16]$$

La formule [13] devient alors :

$$\boldsymbol{P_{coo}}(\boldsymbol{w}/\boldsymbol{C}(\boldsymbol{t_i})) = \boldsymbol{P_{coo}}(\boldsymbol{w}/\boldsymbol{T_{max}})\boldsymbol{P}(\boldsymbol{T_{max}}/\boldsymbol{t_i}) \quad [17]$$

❖ Estimation de la probabilité de cooccurrence :

On estime la probabilité de cooccurrence de la manière suivante :

$$\boldsymbol{P_{coo}}(\boldsymbol{w}/\boldsymbol{w_j}) = \frac{\boldsymbol{Count}(\boldsymbol{w},\boldsymbol{w_j})}{\sum_{\boldsymbol{w_i}} \boldsymbol{Count}(\boldsymbol{w},\boldsymbol{w_i})} \quad [18]$$

Où:

$\boldsymbol{w}$, $\boldsymbol{w_i}$, $\boldsymbol{w_j} \in \boldsymbol{V}$ et $\boldsymbol{Count}(\boldsymbol{w},\boldsymbol{w_j})$ est la fréquence de cooccurrence de couple $(\boldsymbol{w},\boldsymbol{w_j})$ dans une fenêtre de taille *F* sur les document retournés de la première recherche .

2.2. Estimation de Modèle de document :

Nous supposons que le modèle de document peut être estimé en utilisant deux modèles : modèle de terme simple $\boldsymbol{(M_{Dt})}$ et modèle de terme composé $\boldsymbol{(M_{DT})}$**.** Etant donné la nouvelle requête $\boldsymbol{Q_{new}}$(obtenue après expansion de requête), exprimé par des termes simples $\mathbf{t_i}$ et termes composés $\mathbf{T_j}$. Les deux modèles de document sont estimés comme suit :

$$\mathbf{P(t_i|D) = P_{Dir}\left(t_i \middle| M_{D_t}\right)} \quad [19]$$

$$\mathbf{P\left(T_j|D\right) = \omega P_{Dir}\left(T_j|M_{D_T}\right) + (1-\omega)\prod_{t_k \in T} P_{Dir}\left(t_k|M_{D_t}\right)} \quad [20]$$

Où : $\boldsymbol{\omega}$ ϵ [0, 1] paramètre de lissage, $\mathbf{P_{Dir}\left(T_j|M_{D_T}\right)}$ et $\mathbf{P_{Dir}\left(T_k|M_{D_t}\right)}$ peut être évalué en utilisant n'importe quel modèle de langue uni-gramme. Dans ce cas le modèle de Dirichlet est utilisé.

Le modèle est représenté comme suit :

$$\mathbf{P_{Dir}\left(t_i \middle| M_{D_t}\right) = \frac{F(t_i, D_t) + \mu P(t_i|C_t)}{|D_t| + \mu}} \quad [21]$$

Où : $\mathbf{F(t_i, D_t)}$ est la fréquence de terme de $\mathbf{t_i}$ dans le document $\mathbf{D_t}$, $\mathbf{P(t_i|C_t)}$ est le modèle de langue de collection de fond (la fréquence globale de terme est utilisé), $\mathbf{|D_t|}$ la taille de document et μ le paramètre de lissage.

De la même manière :

$$\mathbf{P_{Dir}\left(T_j \middle| M_{D_T}\right) = \frac{F\left(T_j, D_T\right) + \mu P(T_j|C_T)}{|D_T| + \mu}} \quad [22]$$

Où :

- ✓ $\mathbf{P\left(T_j|C_T\right)}$ est le modèle de langue de collection de fond.
- ✓ $\mathbf{F\left(T_j, D_T\right)}$ est la fréquence de terme composé, elle est calculée par compte de terme $\mathbf{T_j}$.
- ✓ $\mathbf{|D_T| = \sum_{T \in C_T} F(T, D_T)}$ est la taille du document représenté par des termes composés.

2. Exemple d'illustration:

Soit la requête suivante Q: t_3 ″t_1t_2″ t_3

on cherche dans cet exemple la probabilité de : P(**W**|ϑ_Q^{coo})

Où :

t_3 est un terme simple de la requête

t_1t_2 est un terme composé de la requête

t_5 est un terme à ajouter à la requête.

Cet exemple est représenté sous forme de schéma suivant :

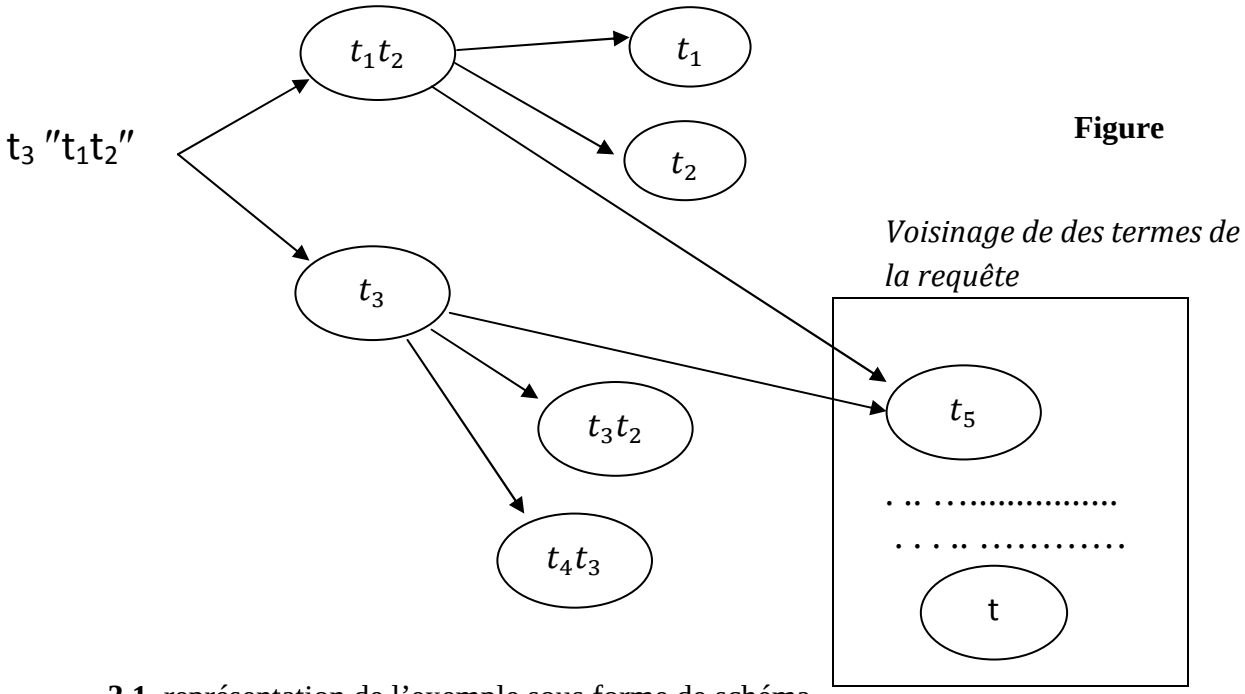


Figure 3.1. représentation de l'exemple sous forme de schéma

Les tableaux suivants illustrent l'application des formules précédentes au terme **w (t_5)** sachant les termes originaux de la requête Q.

terme	***N***	***df***	***imp***	***idf***
t_1	23	12	23/12	5
t_2	23	9	23/9	3
t_3	23	7	23/7	8
t_4	23	5	23/5	6
t_3 t_2	23	6	/	8
t_4 t_3	23	9	/	10

A. Le premier tableau calcule la probabilité de $\mathbf{t_5}$ sachant le terme composé $\mathbf{T_j}$ de la requête, $\mathbf{T_j : t_1\ t_2}$

T_j	W	$P_{coo\_in}(W\|T_j)$ **(18)**	$P(T_j\|Q)$ (7)	$P_{coo\_in}(W\|t_k)$ (18)	$P(t_k\|T_j)$ (11)	$P_{coo}(W\|T_j)$ (9)
t_1t_2	t_5	0.7	2/4	$P_{coo\_in}(t_5\|t_1)$ = 0.1 $P_{coo\_in}(t_5\|t_2)$ = 0.2	$P(t_1\|t_1\ t_2)=\frac{23/12}{23/12+23/9}=0.42$ $P(t_2\|t_1\ t_2)=\frac{23/9}{23/9+23/12}=0.57$	0.5*0.7+ (1-0.5)*(0.1*0.42+ 0.2*0.57) =0.35+0.5*0.042+0.114=0.48

Tableau 3.1 : exemple de calcule de probabilité $P_{coo}(W|T_j)$

W : est un terme appartenant à l'ensemble **G** qui contient les termes à ajouter à la requête originale, **w= t_5**

α: est un facteur d'interpolation pour contrôler l'apport d'un terme composé par rapport à ses termes composant où **α** =0.5

N : est le nombre de documents dans la collection **C**. ***N*=23**

C(T_j) : est l'ensemble des termes simples qui composent le terme composé $\mathbf{T_j}$

B. Le deuxième tableau calcule la probabilité de $\mathbf{t_5}$ sachant le terme simple $\mathbf{t_i}$ de la requête originale, $\mathbf{t_i}$=t_3

t_i	W	$P_{coo\_in}(W\|t_i)$ = **(18)**	$P(t_i\|Q)$ (6)	$P(T_L)$ (15)	$P(t_i\|T_L)$ = (15)	$P(T_L/t_i)$ = **(14)**	$P_{coo}(W\|C(t_i))$= **(13)**	$P_{coo}(W\|t_i)$= (12)
t_3	t_5	0.3	2/4	$P(t_3t_2)$= $\frac{1/8}{1/8+1/10}$ =0.54	$\frac{23/7}{23/7+23/9}$ =0.56	0.56*0.54= 0.30	$P_{cooin}(t_5\|t_3t_2)$=0.4 $P_{cooin}(t_5\|t_3t_2)$=0.3 **$P_{coo}(W\|C(t_i))$=** 0.4*0.30+ 0.3*0.18 =0.17	0.2*0.3+ (1-0.2)* 0.17= 0.19
				$P(t_4t_3)$= $\frac{1/10}{1/8+1/10}$ =0.45	$\frac{23/7}{23/7+23/5}$ = 0.41	0.41*0.45 =0.18		

Tableau 3.2 : exemple de calcule de probabilité $P_{coo}(W|T_i)$

β: est un facteur d'interpolation pour contrôler l'apport d'un terme simple par rapport aux termes composés auxquels il appartient où **β** =0.2

C(t_i) est l'ensemble des termes composés auxquels le terme t_i appartient

$\boldsymbol{T_L}$ **:** est le terme composé où $\mathbf{t_i}$ appartient, $\boldsymbol{T_L}$ =t_3t_2, $\boldsymbol{T_L}$ =t_4t_3.

Pour calculer $\mathbf{P(W|\vartheta_Q^{coo})}$**,** on fait la somme des deux probabilités : **$P_{coo}(W|t_i)$ et $P_{coo}(W|t_j)$**

$$\mathbf{P(W|\vartheta_Q^{coo})= P_{coo}(W|t_i) * P(t_i|Q) + P_{coo}(W|T_j)* P(T_j|Q)}$$

$$= 0.19*2/4+0.48*2/4 =0.33$$

4. Conclusion

L’expansion de requêtes est une étape importante dans la recherche d’information. Elle consiste principalement à enrichir la requête initiale de l’utilisateur avec de une information supplémentaire, de façon à ce que le système propose des résultats plus appropriés pour satisfaire les besoins de ce dernier.

Dans ce chapitre on a décrit une nouvelle approche pour l’expansion de la requête basée sur les termes composés dans le modèle de langage et on a choisi la relation de cooccurrence pour extraire les nouveaux termes à ajouter à la requête initiale.

Chapitre IV : Expérimentations et évaluations

1. Introduction

Après avoir expliqué dans le chapitre précédent l'approche à implémenter, nous allons à présent présenter les résultats du travail que nous avons réalisé en passant d'abord par la présentation de l'environnement de développement, que nous avons utilisé, ensuite une petite description de l'architecture générale de notre application, puis nous présentons notre expérimentation qui porte sur la reformulation de requête basée sur les termes composés, qui considère la requête comme un ensemble de termes composés et un ensemble de termes simples, testée sur une collection de test TREC AP88.

2. Environnement de développement

Dans nos expérimentations on a utilisé la plate-forme Terrier [3] qui a pour objectif de faciliter la recherche d'information et implémenter pour cela un certain nombre de fonctionnalités de recherche qui offre un large choix pour le développement de nouvelles applications et pour les testes en recherche d'information (RI). Terrier est développé entièrement en java. Le choix de java c'est imposé vu que l'Open Source de terrier est entièrement écrit en java

Notre approche fonde sur la reformulation de requête avec des termes composés. Pour déterminer dans les documents les termes qui cooccurrent avec la plupart des termes de la requête et de calculer la cooccurrence entre termes on a utilisé Ngram Statistics Package (Text::NSP) qu'est un ensemble de modules Perl.

2.1. le langage java :

Java est un langage de programmation à usage général, évolué et orienté objet dans la syntaxe est proche du C, apparu en fin 1995 et développé par SUN Microsystems. Il permet d'exécuter des programmes au travers d'une machine virtuelle (JVM).

- **Les caractéristiques du langage java :**

Java possède de nombreuses caractéristiques qui en font un des langages de haut succès.

- **java est interprété :** le fichier source est compilé (traduit) en pseudo code (Byte-code) puis exécuté par un interpréteur java (JVM) et non pas directement par le processeur de la machine.

- ✓ **Java est indépendant de toute plate-forme :** java a été conçu pour maitre en œuvre des applications susceptibles de s'exécuter sur n'importe qu'elle plate-forme, tel que une application java fonctionne sur tout environnement (Unix, Windows…) disposant d'une JVM (Java Vertual Machine).
- ✓ **Java est orienté objet :** java est un pur langage orienté objet dont on ne manipule pas des procédures et des fonctions, mais des objets qui s'échangent des messages.
- ✓ **Java doté en standard de bibliothèque de classe :** java dispose d'une bibliothèque très impressionnante qui inclut la connectivité des bases de données, la conception GUI, les E/S et la programmation multithread (multitâche) …etc.
- ✓ **Java assure la gestion de la mémoire : java** assure la location de la mémoire à un objet automatiquement à sa création, comme il récupère la mémoire occupé par des objets inutilisables, après leurs destructions par garbage collector (Ramasse Miettes).

➢ **Présentation de NetBeans :**

Dans notre travail on a utilisé NetBeans (version 7.0.1) qu'est un environnement de développement intégré (IDE) pour java, placé en Open Source par Sun 2000**.** En plus de java, NetBeans peut supporter d'autres langages comme PHP, XML**…**etc. il est conçu en java, il permet d'écrire, compiler et d'exécuter des programmes. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projet multi-langage, …etc). NetBeans est gratuit et disponible sous (Windows, Linux, Solaris…), son installation nécessite l'installation de la JDK (Java Développement Kit).

2.2. La plateforme terrier

Terrier, TeRabyte RetrIEver a été développé par le département informatique de l'université de Galasgow, crée spécialement pour la recherche d'information. Il est distribué en Open Source entièrement écrit en java. C'est une plate forme qu'est destinée à l'indexation de volumes importants de documents en plein-texte : jusqu'à 25 millions de documents. Selon l'étude comparative sur les systèmes de recherche d'information en open source effectuée par

Ricardo Baeza-Yates [4], terrier fait partie des trois meilleurs systèmes dans un environnement java. Les deux autres sont MG4J et Lucene.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requêtes basé sur le Framework DFR (Divergence From Randomness) [5]. Comme touts les moteurs de recherche terrier possède les principes 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.

2.2.1. Le processus d'indexation de terrier :

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

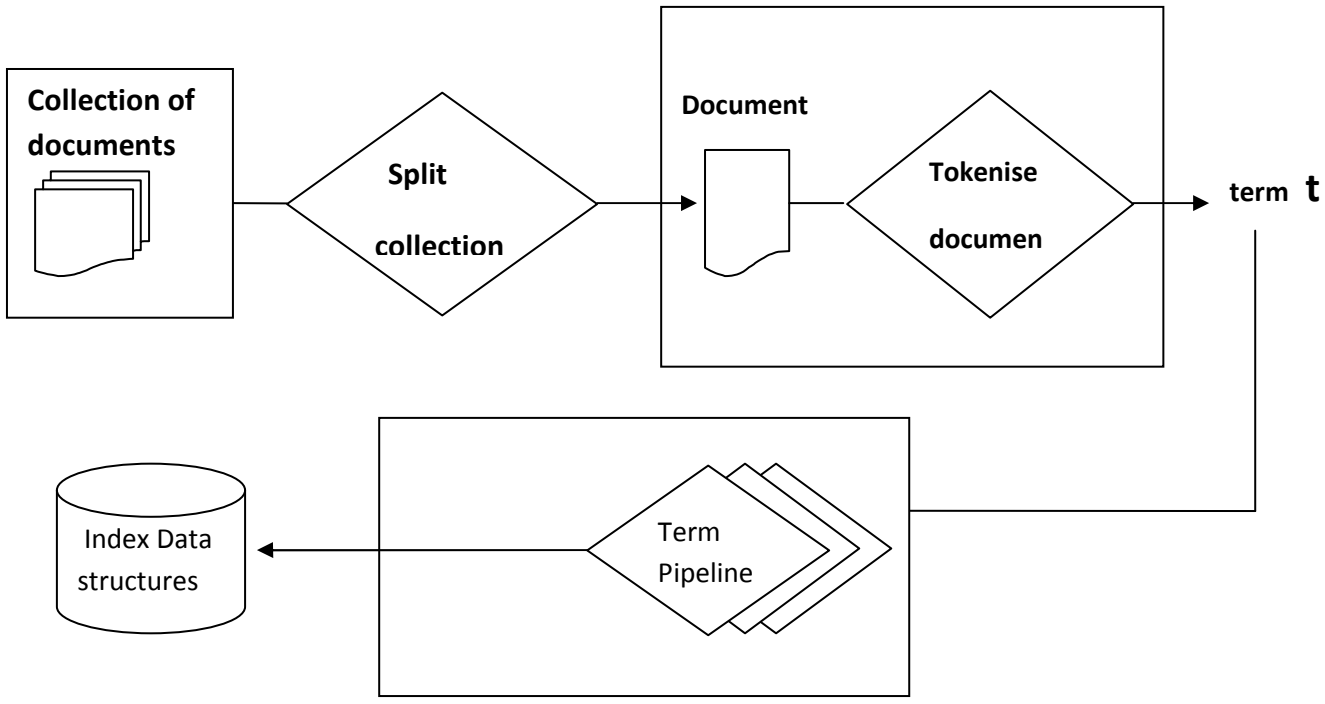


Figure 4.1 : présentation du processus d'indexation de terrier.

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

Le résultat de l'indexation consiste en un index constitué des structures de données suivantes : Inverted File, lexicon file, Direct Index, Document Index (présentés en détail en annexe 1).

2.2.2. Le processus de recherche de terrier :

Un des principaux objectifs de terrier est de faciliter la recherche d'information. Terrier implémente pour cela certain nombre de fonctionnalités de recherche qui offre un large choix pour le développement de nouvelles applications et pour les testes en RI. En effet, Terrier offre un grand choix de modèles pondération [6], il propose aussi un langage de requête avancée [7]. Une autre fonction de recherche très importante intégrée dans Terrier est l'automatisation de l'expansion de requête.

Dans notre cas le module qui nous intéresse est la classe QueryExpansion qui se trouve dans le module Quering et l'objet Weighting Model qui se trouve dans le module matching. La figure présente un aperçu de processus de recherche dans Terrier.

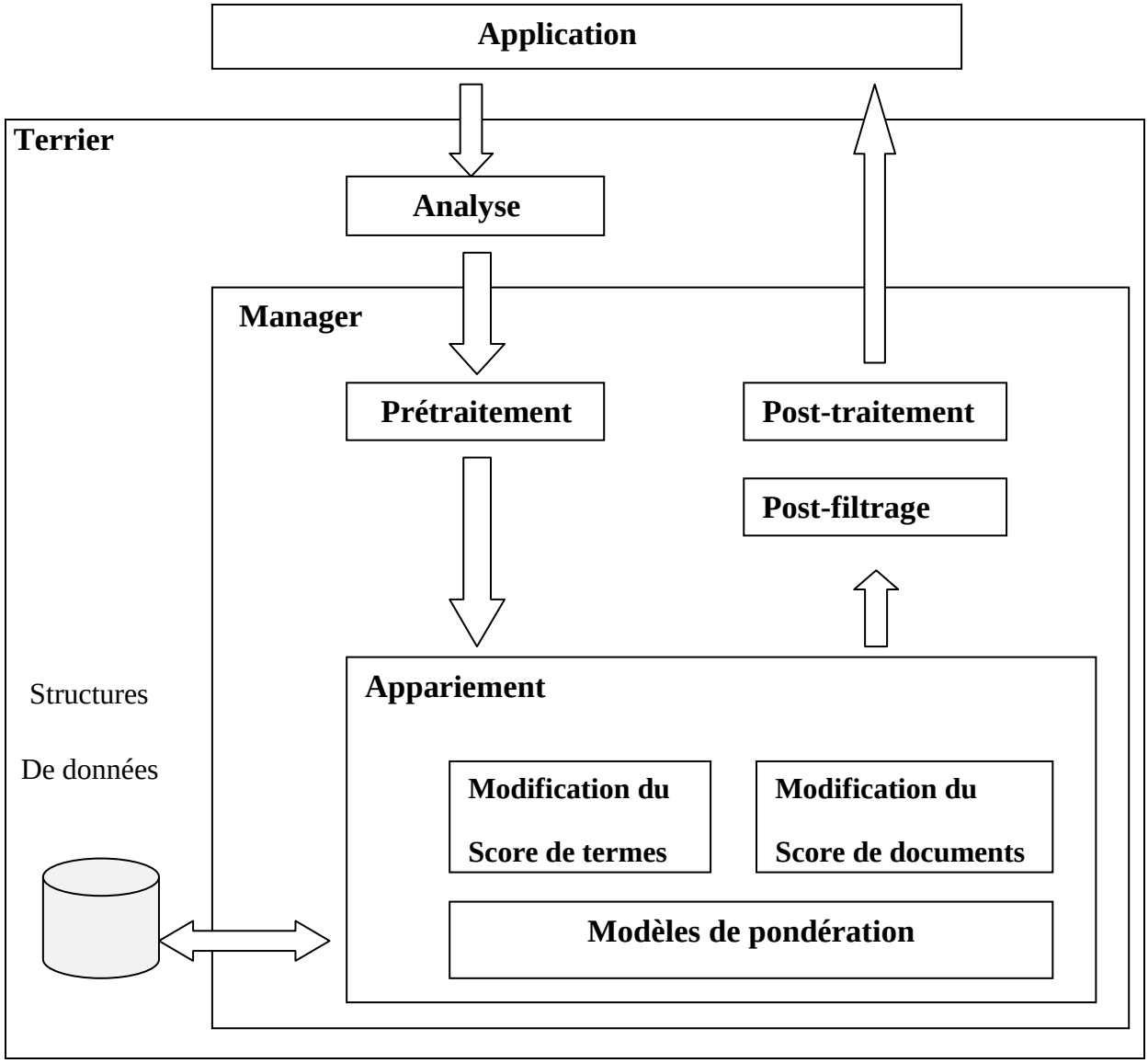


Figure 4.2 : L'API de recherche de terrier

Nous présentons une description détaillée l'API de recherche de terrier en annexe.

2.3. Présentation de Ngram Statistics Package (Text::NSP)

Ngram Statistics Package (Text::NSP) est un ensemble de modules Perl. Il a pour objectif de calculer les nombres d'occurrence des Ngram (sequence de N tokens) dans un texte, et identifie ceux qui ont une certaine signification supplémentaire par l'intermédiaire d'un certain

nombre de différentes mesures d'association. Les fonctions offertes par NSP sont exécutées avec des lignes de commandes.

2.3.1. Ngrams:

Ngram est une séquence de n tokens. NSP utilise des expressions régulières de Perl pour définir les tokens dans un texte qui seront délimités par le symbole "<>". Après avoir définit les tokens, les Ngrams seront construits en assemblant N tokens, par exemple "intelligence<>artificielle<>?<>" est un trigram dont les tokens sont " intelligence " et " artificielle " et "?". NSP construit habituellement les Ngrams avec des tokens contigus. Comme il peut aussi les former avec des tokens non contigus, mais il faut d'abord définir une fenêtre de taille K qu'est une séquence de K tokens contigus, où la valeur de k est supérieure ou égale à la valeur de N de Ngrams. Un Ngram peut être formé de n token quelconque, et chaqu'un de ces tokens doit appartenir à la fenêtre de taille k définie. Les n tokens qui apparaissent dans le Ngramm doivent être exactement dans le même ordre dont ils se produisent dans la fenêtre

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

Why s the stock falling ?

Les bigrams formés des tokens précédents sont:	Les trigrams formés avec une fenêtre de taille 4 sont comme suit:
why<>s<> s<>the<> the<>stock<> stock<>falling<> falling<>?<>	why<>s<>the<> why<>s<>stock<> why<>the<>stock<> s<>the<>stock<> s<>the<>falling<> s<>stock<>falling<> the<>stock<>falling<> the<>stock<>?<> the<>falling<>?<> stock<>falling<>?<>

Tableau 4.1 : exemple de construction des bigrams et de trigrams

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

2.3.2. Le module count.pl:

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

La valeur de N par défaut est égale à 2, comme elle peut être définie par l'utilisateur. Exemple :

count.pl --ngram 3 output.txt input.txt

Où : --ngram 3 pour spécifier qu'on cherche les n-grammes de taille 3.

output.txt : le fichier résultat.

input.txt : le fichier d'entrée.

La valeur de la fenêtre peut être aussi définie par l'utilisateur. Exemple :

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

Où : --window 3 pour spécifier la valeur de la fenêtre de taille 3.

le contenu de fichier ountput.txt est comme suit :

```
7
    line<>of<>2 5 4
    of<>text<>3 2 2
    second<>line<>1 1 3
    line<>and<>1 5 1
   first<>line<>1 1 3
   third<>line<>1 1 3
```

Figure 4.3 : exemple d'un fichier ountput.txt

Dans la première ligne, le numéro 7 indique qu'il y a 7 Bigrams dans le fichier texte en entrée. Les autres lignes sont constitués des Bigrams et des listes de nombres. Le premier nombre de la liste dans chaque ligne désigne le nombre de fois le Bigram occure dans le texte en entré ; le deuxième nombre dénote dans combien de Bigrams le premier token occure à la première position ; ainsi le troisième nombre dénote dans combien de Bigrams le deuxième token occure à la deuxième position.

Prenant comme exemple la première ligne : line<>of<> 2 5 4

- Le Bigram est constitué des deux tokens **line** et **of**.
- Les deux tokens apparaissent deux fois ensemble dans le texte.
- Le token **line** apparait 5 fois à la première position dans le texte.
- Le token **of** apparait 4 fois à la deuxième position dans le texte.

2.3.4. Le programme statistic.pl :

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

3. architecture logicielle de notre approche

Pour la réalisation de notre travail, nous avons développé plusieurs étapes. Ces étapes sont structurées et illustrées dans la figure suivante (**figure 4.4)**

3.1. Architecture générale

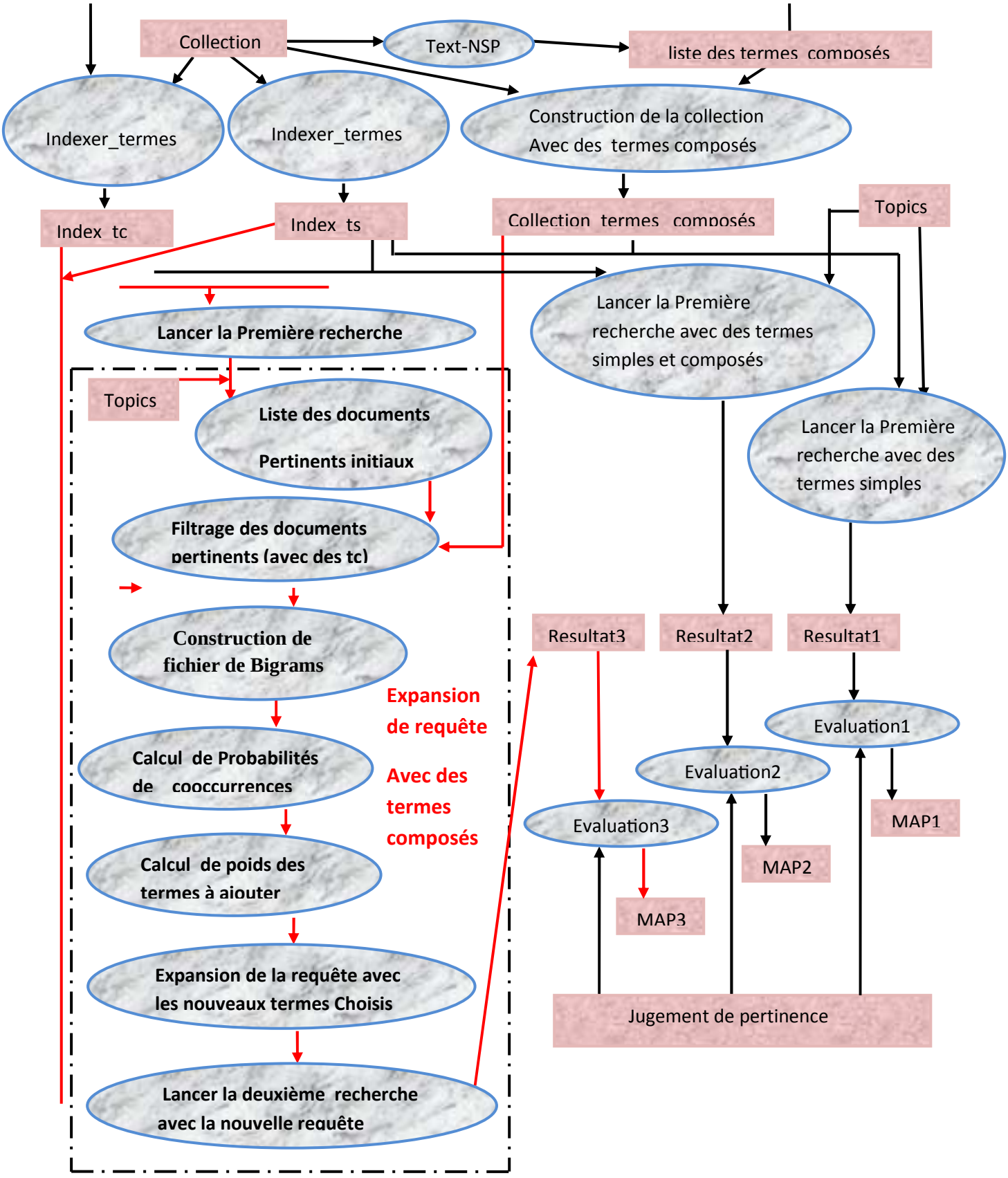


Figure 4.4: Schéma générale de notre de l'application

Pour chacune des étapes de notre approche illustrées dans le schéma précédent nous avons utilisé plusieurs types de données décrites comme suit :

3.2. Données utilisés:

Afin d'évaluer la performance de notre approche, on a utilisé d'une collection test qui consiste en un ensemble de documents, un ensemble de requêtes représentant un ensemble de thèmes (topics) et un jugement de pertinence entre ces requêtes et ces documents :

- **Collection** : on a choisit la collection AP88 contenant 79919 documents et de taille 243Mo
- **Topics 251-300** : fichier contient 50 topics, définissant les requêtes**.** Seul le titre des topics est utilisé.

  Exemple d'une requête :

<top>

<num> Number: 251

<title> Exportation of Industry

<desc> Description:

Documents will report the exportation of some part of U.S. Industry to another country.

<narr> Narrative:

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

</top>

Figure 4.5 : Structure d'une requête.

- **Liste des termes composés** : construite à partir de la collection en faisant appel au module count.pl de Text-NSP. Un terme composé est retenu dans la liste si sa fréquence dans la collection est supérieur à 20.
- **Un fichier de jugement de pertinence :** une liste des documents pertinents et non pertinents pour chaque requête (réponses idéales)

Nous décrivons ci-dessous les différentes étapes de notre approche

3.3. Les étapes de mise en œuvre de notre approche :

- **Etape1 : Construction de la collection :** permet de construire la collection avec des termes composés (termes simples) en utilisant la liste qui contient les termes composés.
  - **Resultat1 :** collection avec des termes composés.

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

period_time quang inmat return_famili ho_chi chi_minh minh_citi south_vietnames capit saigon amnesti appar part_effort communist_parti parti_chief nguyen_van linh heal intern_division improv vietnam imag_abroad

Figure 4.6 : extrait de contenu d'un document avec des termes composés

- **Etape2 : Indexation de la collection :** cette étape a pour objectif d'identifier les termes (simples et composés) sensés représenter au mieux le contenu d'un document. Deux types de termes sont successivement identifiés à l'issue de cette étape : les termes simples et les termes composés.

  - *Indexation de la collection en termes simples :* pour ce type d'indexation on a utilisé les classes BasicIndexer et TRECIndexing existantes par défaut dans Terrier.

    Résultat 2.1 : index terme simple.

  - *Indexation de la collection en termes composés :* comme terrier est un logiciel qui offre la possibilité d'extension sur le processus d'indexation ; on a ajouté une classe à l'application pour indexer les documents avec des termes composés.

    Résultat 2.2 : index avec termes composés.

- **Etape 3 : Lancement de la première recherche :** cette étape consiste à lancer la première recherche, dont on a deux types :

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

  - **Résultat 1 :** obtention de résultat (fichier .RES) pour l'ensemble de requêtes afin d'évaluer cette recherche.

  B. recherche_ terme composé : pour lancer cette recherche on a utilisé une autre classe qui traite la requête originale comme un ensemble de termes simples et de termes composés et donne un score au document qui vérifie la requête.

  L'une des caractéristiques principales de terrier est son extensibilité. comme terrier ne possède pas un modèle de pondération qui prend en considération les termes composés, on a ajouté un modèle de pondération qui permet d'assigner un score à chaque terme simple et terme composé de la requête dans le document.

  - **Résultat 2 :** il crée un fichier résultat (fichier .RES) qui permet par la suite de calculer la précision moyenne(MAP)

- **Etape4 : Expansion de requête à base des termes composés :** La classe qui permet d'exécuter ce processus est Query Expansion dans terrier (expansion automatique de requête) qui permet l'expansion de la requête avec des nouveaux termes extraits de N premiers documents pertinents, ainsi que la repondération des termes de la requête.

Comme notre approche se fonde sur la reformulation de requête à base des termes composés, et comme terrier est caractérisé par sa possibilité de modification, on a modifié Query Expansion qu'est initialement intégrée dans terrier

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

1. **Récupération de l'ensemble de N documents pertinents :** dans cette étape on récupère les documents pertinents retournés de la première recherche dont leurs contenus sont des termes simples puis par la suite on a filtré ces documents

dans la collection construite avec des termes composés à l'étape1 ce qui donne par la suite des documents pertinents dont leur continu c'est des termes simples ou composés.

2. **Construction de fichier de Bigrams :** Après avoir récupérer l'ensemble des documents pertinents, en les faisant passer à text-NSP (count.pl) qui retourne un fichier texte contenant tout les bi-grammes et leurs fréquences de cooccurrences.

Voici un extrait de contenu de résultat retourné par count.pl :

```
involv_hundr<>million_dollar<>2 18 18
compani<>industri<>2 45 18
japan<>plan_end<>2 35 18
bc_japan<>japan<>2 18 36
mai<>investig<>2 9 45
hundr_million<>million_dollar<>2 18 18
```

Figure 4.7 : extrait de contenu de résultat retourné par count.pl

Chaque linge est constitué de bi-gramme et une liste de nombres. Le premier nombre de la liste dans chaque linge désigne le nombre de fois le bi-gramme occure dans le texte en entrée ; le deuxième nombre dénote dans combien de b-grammes le premier token occure à la première position; ainsi le troisième nombre dénote dans combien de bi-gramme le deuxième token occure à la deuxième position.

3. **Calcul des probabilités de cooccurrence :** Dans cette étape, on a utilisé le résultat retourné par text-NSP (count.pl) pour calculer les probabilités de cooccurrence de chaque bi-gramme (w, w_j), en utilisant la formule (18) de chapitre III:

Exemple : soit la première ligne de l'extrait de fichier de bigrams précedent

W= involv_hundr , w_j = million_dollar

Pour calculer la probabilité de cooccurence entre W et w_j on appliqué la formule (10) :

$$\boldsymbol{P_{coo}(w/w_j)} = \frac{\boldsymbol{Count(w,w_j)}}{\sum_{\boldsymbol{w_i}} \boldsymbol{Count(w,w_i)}}$$

$$\boldsymbol{P_{coo}(w/w_j)} = \frac{2}{18} = 0.1$$

4. **Reformulation de requête avec des nouveaux termes** : avant de reformuler la requête originale il faut d'abord choisir les termes à ajouter :

   - **Choix des nouveaux termes :**pour choisir les termes à ajouter on a utiliser la méthode présenté en chapitre 3.

Après avoir calculé la probabilité pour chaque terme de la requête originale et de la requête de cooccurrence, on a trié les nouveaux termes selon leurs poids croissants, ensuite on a choisi seulement les **N** premiers termes à ajouter. On a tester plusieurs valeurs pour N et la meilleure est 30.

❖ **Etape 5 : lancement de la deuxième recherche avec la nouvelle requête :** une fois le modèle de la requête est construit une nouvelle recherche est effectuée

4. Evaluation de l'approche d'expansion de requête à base des termes composés :

Pour évaluer la performance de l'approche d'expansion de requête à base des termes composés définit dans le chapitre précédant, nous avons réalisé une série d'expérimentation dont le but est de comparer le résultat de recherche de documents pertinents avec la requête reformulée à base des termes composés par rapport à l'utilisation de la requête originale sous la plateforme Terrier.

On a comparé notre approche à deux types de recherche :

- ❖ Recherche avec la requête originale représentée avec des termes simples, en utilisant la formule (21) du chapitre III .

- Recherche avec la requête originale basée sur des termes composés, en utilisant le modèle qui considère la requête comme un ensemble de termes simples et composés ajouté à notre application.

4.1. Paramètres de modèles de notre approche :

On a estimé les différents paramètres de modèle présenté dans la section précédente d'une manière empirique de façon à optimiser la valeur de MAP. Ces paramètres sont :

- Le paramètre **μ** du modèle Dirichlet est fixé à 2500.
- Les paramètres de modèle de document de termes simples et termes composés (sans expansion de requête) :
  - Le paramètre **ω** de la formule [20] qui contrôle l'apport de modèle de document de termes simples par rapport au modèle de document de termes composés ; la valeur de ce paramètre est fixée à 0.5.
  - Le paramètre **μ** de la formule **[21]** est fixé à 2000
  - Le paramètre **μ** de la formule **[22]** est fixé 500
- Les paramètres de modèle de document de termes simples et termes composés (avec expansion de requête) :
  - Le paramètre **ω** de la formule [20] qui contrôle l'apport de modèle de document de termes simples par rapport au modèle de document de termes composés ; la valeur de ce paramètre est fixée à 0.3.
  - Le paramètre **μ** de la formule **[21]** est fixé à 2000
  - Le paramètre **μ** de la formule **[22]** est fixé à 2500.
- Les paramètres de modèle de la requête :
  - Le paramètre φ de la formule **[1]** qui contrôle l'apport de modèle de la requête original par rapport au modèle considérant la relation de cooccurrence ; la valeur de ce paramètre est fixée à 0.4.
  - Le paramètre α de la formule **[8]** qui permet de contrôler l'apport d'un terme composé par rapport à ses termes composants. La valeur de ce paramètre est fixée à 0.6
  - Le paramètre β de la formule **[13]**, permettant de contrôler l'apport d'un terme simple par rapport aux termes composés auxquels il appartient. La valeur de ce paramètre est fixée à 0.3.

- Le paramètre **F** de la formule **[10]** ; représentant la taille de la fenêtre pour estimer la fréquence de cooccurrence de couple. La valeur de ce paramètre est fixée à 20.
- Le paramètre d'identification d'un terme composé ; un terme composé est identifié si sa fréquence dans la collection dépasse un seuil fixé à 25.
- le nombre de documents d'où on extrait les termes à ajouter. est fixé à 4.

4.2. Résultats d'évaluation :

Le tableau suivant présente les résultats obtenus pour l'ensemble des requêtes tests. Les résultats montrent que notre approche d'expansion de requête à base des termes composés offre une précision moyenne plus importante que les deux autres recherches

Information	Recherche avec termes simples **(A)**	Recherche avec termes composes sans expansion **(B)**	Recherche après expansion de requête avec des termes composés **(C)**	Taux d'amélioration **(B)** et **(A)**	Taux d'amélioration **(B)** et **(A)**
Nombre de requêtes	48	48	48		
Nombre de doc sélectionné	50000	50000	50000		
Nombre de doc pertinents	1672	1672	1672		
Nombre de doc Pertinents sélectionnés	722	753	867		
Précision Moyenne	0.1729	0.1865	0.2175	+7.86%	+16.62%
R Précision	0.1843	0.2057	0.2144		

Tableau 4.2 : résultat d'évaluation de l'approche d'expansion à base des termes composés sous la plateforme de RI terrier.

En analysant le tableau 4.2, on remarque que :

A. Les précisions moyennes des trois recherches sont différentes :

- la précision moyenne (MAP) de la recherche avec termes simples est : MAP=0.1729.
- la précision moyenne (MAP) avec de la recherche à base des termes composés sans expansion de requête est : MAP=0.1865.
- la précision moyenne (MAP) de la recherche à base des termes composés avec expansion de requête est : MAP=0.2175.

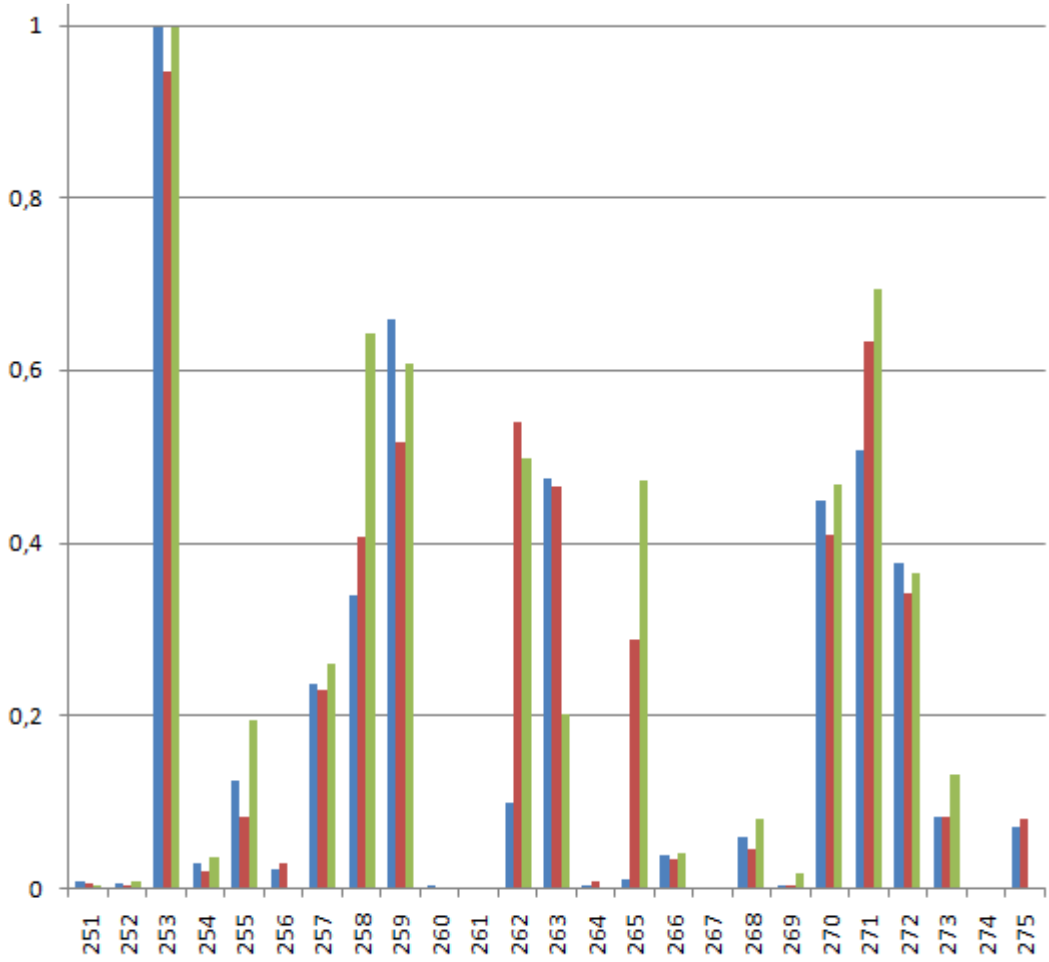
B. le nombre de documents pertinents retournés de la recherche à base des termes composés sans expansion de requête est amélioré par rapport au nombre retourné de la recherche avec termes simples et le nombre de documents pertinents retournés de la recherche à base des termes composés avec expansion de requête est amélioré par rapport au nombre retourné de la recherche avec termes composés sans expansion de requête.

D'après les résultats obtenus de la comparaison qu'est faite entre chaque deux recherche (la recherche avec termes simples avec la recherche à base des termes composés sans expansion et la recherche à base des termes composés sans expansion avec la recherche à base des termes composés avec expansion de requête) par rapport à leurs précisions , on peut dire que :

- la recherche à base des termes composés est améliorée par rapport à la recherche avec termes simples, avec un taux d'amélioration de +7.86%.
- la recherche à base des termes composés avec expansion de requête est améliorée par rapport à la recherche à base des termes composés sans expansion, avec un taux d'amélioration de +7.86%.

4.3. Résultats d'évaluation requête par requête :

Pour avoir une vision et une analyse plus claire et plus profonde on a effectué l'évaluation requête par requête les deux graphes suivants illustrent cette analyse :


1
0,8
0,6
0,4
0,2
0
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275

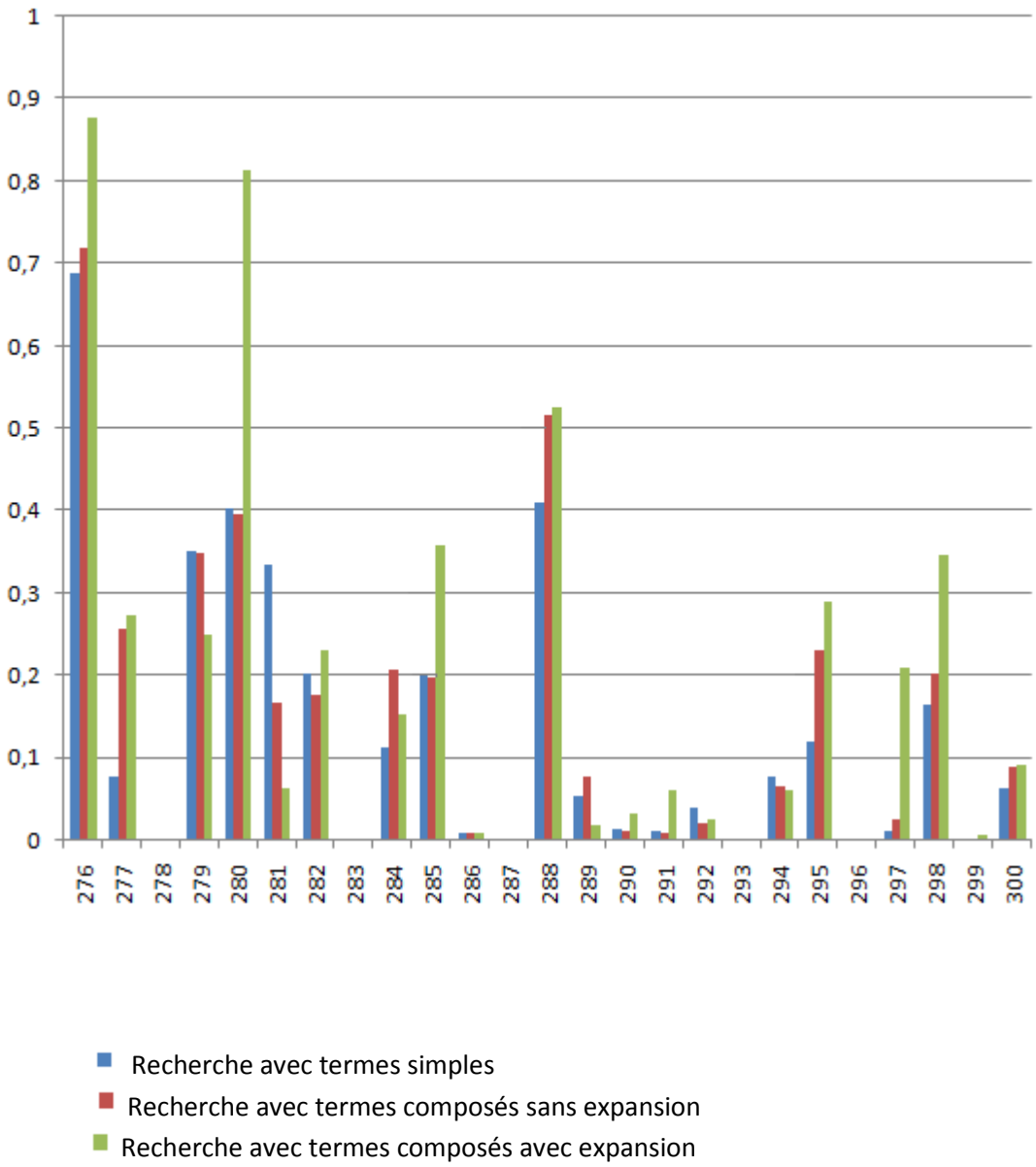


Figure 4.8 : Graphes représentant les résultats d'évaluation requête par requête pour les trois recherches (simple, termes composés sans expansion et termes composés avec expansion)

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

Num requête	Recherche termes simples	Recherche termes composés sans expansion	Amelioration
251	0,0066	0,0057	-
252	0,0045	0,0038	-
253	1	0,9484	-
254	0,0292	0,019	-
255	0,124	0,082	-
256	0,0227	0,0299	+
257	0,2359	0,2293	-
258	0,3398	0,4075	+
259	0,6589	0,5183	-
260	0,0035	0,001	-
261	0,0005	0,0007	+
262	0,0985	0,54	+
263	0,4758	0,466	-
264	0,0024	0,0085	+
265	0,0101	0,2886	+
266	0,037	0,0346	-
267	0	0,0011	+
268	0,0585	0,0451	-
269	0,0017	0,0023	+
270	0,4499	0,4096	-
271	0,5082	0,6354	+
272	0,3779	0,341	-
273	0,0835	0,0819	-
274	0	0	0
275	0,0704	0,0806	+
276	0,6885	0,72	+
277	0,0765	0,2579	+
278	0	0	0
279	0,35	0,35	0
280	0,4018	0,3971	-
281	0,3333	0,1667	-
282	0,2005	0,1762	-
283	0	0	0
284	0,1124	0,2074	+
285	0,3367	0,1983	-
286	0,0084	0,01	+
287	0	0	0
288	0,5355	0,5164	-
289	0,0523	0,0772	+
290	0,012	0,012	0
291	0,0098	0,0095	-
292	0,0388	0,0201	-
293	0,0016	0,0027	+

294	0,076	0,0662	-
295	0,2322	0,2308	-
296	0	0	0
297	0,01	0,0261	+
298	0,1629	0,2357	+
299	0,0008	0,0023	+
300	0,0622	0,0896	+

Tableau 4.3: Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec des termes composés sans expansion

Num requête	**Recherche termes composés sans expansion**	**Recherche termes composés avec expansion**	Amelioration
251	0,0057	0,005	-
252	0,0038	0,0104	+
253	0,9484	1	+
254	0,019	0,0372	+
255	0,082	0,1955	+
256	0,0299	0,0003	-
257	0,2293	0,2624	+
258	0,4075	0,6443	+
259	0,5183	0,6102	+
260	0,001	0,0016	+
261	0,0007	0,0005	-
262	0,54	0,5	-
263	0,466	0,2044	-
264	0,0085	0,0003	-
265	0,2886	0,4752	+
266	0,0346	0,0418	+
267	0,0011	0,0016	+
268	0,0451	0,0819	+
269	0,0023	0,0198	+
270	0,4096	0,4699	+
271	0,6354	0,6969	+
272	0,341	0,3663	+
273	0,0819	0,1325	+
274	0	0	0
275	0,0806	0,0009	-
276	0,72	0,8762	+
277	0,2579	0,2718	+
278	0	0	0
279	0,35	0,25	-

280	0,3971	0,8131	+
281	0,1667	0,0625	-
282	0,1762	0,2294	+
283	0	0	0
284	0,2074	0,1519	-
285	0,1983	0,3583	+
286	0,01	0,0092	-
287	0	0	0
288	0,5164	0,526	+
289	0,0772	0,0176	-
290	0,012	0,0311	+
291	0,0095	0,06	+
292	0,0201	0,0239	+
293	0,0027	0,001	-
294	0,0662	0,0608	-
295	0,2308	0,2897	+
296	0	0	0
297	0,0261	0,2096	+
298	0,2357	0,3447	+
299	0,0023	0,0056	+
300	0,0896	0,0902	+

Tableau 4.4 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche avec des termes composés sans expansion et de la recherche avec des termes composés avec expansion

5. Conclusion

L'évaluation est une étape importante dans le domaine de la Recherche d'Information du fait qu'elle permet de juger de la pertinence des documents restitués par le Système de Recherche d'Information et donc d'évaluer les performances d'un SRI

A la suite de ce chapitre, nous avons abordé les différentes expérimentations que nous avons réalisées, et cela en passant d'abord par une présentation de notre environnement, nous avons ensuite détaillé l'architecture générale de notre application et finalement nous avons rapporté les résultats de nos expérimentations qui montrent l'intérêt de l'approche d'expansion de requête basé sur les termes composés.

Conclusion générale

Conclusion générale

L'analyse des travaux de recherche nous a permet de constater qu'un des obstacles à toute recherche d'information est de trouver, à partir d'une requête donnée, des documents qui répondent aux besoins de l'utilisateur.

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

Nous avons vu que le problème majeur de la traduction de requêtes est l'ambiguïté des termes sources. La plupart des systèmes de recherche d'informations (RI) utilisent des mots simples pour représenter des documents et des requêtes. Il est reconnu depuis longtemps que cette représentation est insuffisante à cause de l'ambiguïté des mots isolés et du non prise en compte des relations entre les mots. Pour remédier à ce problème nous avons tenté d'exploiter les avantages de la relation de cooccurrence statistique entre termes ainsi que des mots composés. Naturellement il faut choisir un modèle de RI qui supporte une telle représentation. Notre choix s'est fixé sur les modèles de langue étant données les performances qu'ils ont montrées.

En ce qui concerne l'expansion de requête par ajout des termes composés, nous avons pu répondre à la question majeure posée, à savoir : «Comment extraire les meilleurs termes composés (simples) a partir des documents pertinents retournés de la première recherche.

Les expérimentations et évaluations que nous avons réalisées en utilisant la collection AP88 sur la plateforme de recherche d'information Terrier ont permet de valider notre approche. Plus précisément, les résultats obtenus mettent en exergue l'intérêt de l'approche d'expansion de requête à base des termes composés.

Comme perspective de notre travail, nous proposons de valider nos résultats en utilisant d'autres collections de teste. Comme deuxième perspective, nous pensons que l'utilisation des termes composés pour reformuler la requête de l'utilisateur peut encore évoluer vers plus de performance. Pour cela il serait intéressant de poursuivre ce projet en

introduisant une autre méthode de choix de termes composés dans un document en prenant en considération l'importance d'un terme composé dans un titre par rapport à un autre qui est dans un sous titre ou bien dans un paragraphe.

Nous envisageons tous cela dans nos futurs travaux de recherche qu'ils soient d'ordre individuels, professionnels ou académiques.

Pour finir, ce projet de fin d'études nous a permis de bien assimiler les bases de la recherche d'information, de cerner ses mécanismes particulièrement le processus de reformulation de la requête qui est une étape cruciale dans tous processus de recherche d'information. On a aussi beaucoup appris sur les schémas de pondération, leurs performances dans un SRI ainsi que les outils qu'ils utilisent pour juger la pertinence d'un document vis-à-vis d'une requête.

Bibliographie

Bibliographie

[B. Croft, D. Harper, 79]: B. Croft and D. Harper. *Using probabilistic models of information without relevance information*. Journal of Documentation., 35(4) :285,295, 1979.

[Bai et al., 05] : Bai, J., Song, D., Bruza, P., Nie, J. Y. and Cao, G. *Query expansion using term relationships in language models for information retrieval*. *ACM CIKM 2005:* Proceedings of international conference on Information and knowledge management, p. 688-695.

[Baziz M, 05] : Mustapha Baziz, *« indexation conceptuelle guidée par ontologie pour la recherche d'information »*, thèse de doctorat de l'université Paul Sabatier (France), décembre, 2005.

[Beigbeder et al, 05] : Amélie Imafouo, Michel Beigbeder : *Scalability Influence on Retrieval Models: An Experimental Methodology*. ECIR 2005: 388-402.

[Berger et Lafferty, 99]: A. Berger and J. Lafferty, *Information Retrieval as Statistical Translation*, Research and Development in Information Retrieval, Proc. ACM-SIGIR'99, pp. 222-229, 1999.

[Boughanem et al., 04] : Mohand Boughanem, Wessel Kraaij, Jian-Yun Nie, *Modèles de langue pour la recherche d'information* . In Les systèmes de recherche d'informations, pages 163–182. Hermes-Lavoisier. 2004.

[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

[Buckley et al., 95]: Chris Buckley, Gerard Salton: *Optimization of Relevance Feedback Weights*. SIGIR 1995: 351-357

[Cobb & Grefenstette, 93]: Cobb H. Et Grefenstette J. J: *Genetic Algorithms for Tracking Changing Environnements*, ICGA5 pp 523-530,1993

[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 et Lafferty, 02] : W.B. Croft and J. Lafferty (2002). *Language Models for Information Retrieval*. Kluwer Int. Series on Information Retrieval, Vol. 13, Kluwer Academic Publishers.

[D. Buscaldi, P. Rosso, 05] : D. Buscaldi, P. Rosso, M. Montes-y-Gómez: *Context Expansion with Global Keywords for a Conceptual Density-Based WSD*. CICLing, pages 263-266, 2005.

[D. Harman, 88]: D. Harman. *Towards interactive query expansion*. In 11th Annual International ACM SIGIR Conference on Research and developement in Information Retrieval, pages 321,331, 1988.

[Deerwester et al., 90]: Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas and Richard A. Harshman, 1990. *"Indexing by Latent Semantic Analysis"*. In Journal of the American Society of Information Science, Vol. 41:6, 391-407.

[E. Voorhees, 94]: E. Voorhees. *Query expansion using lexical-semantic relations*. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 61.69, Dublin, Ireland, 1994. Springer-Verlag New York, Inc.

[Efthimiadis, 96]: F. Efthimiadis *Query Expansion: Annual Review on Technology*. ARIST, Volume 31, pages: 121-187, 1996.

[Fluhr et al, 85] : C. Flhur, F.Debili *Interrogation en langue Naturelle de données textuelles et factuelles* Intelligent Multimadia Information System and Management (RIAO), Grenoble (France), pages : 548-556, 1985.

[G, Cao, Nie, 05] : G.Cao, J. Y. Nie and J. Bai (2005). *Integrating Term Relatioships into Language Models*. In proceedings of the 28h ACM SIGIR Conference on Research and Development in RI, pp. 298305.

[G. Brajnik, S. Mizzaro, C. Tasso. 96]: G. Brajnik, S. Mizzaro, , and C. Tasso. *Evaluating user interfaces to information retrieval systems: a case study on user support*. In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 128.136, Zurich, 1996.

[G. Miller, 95]: G. Miller. Wordnet : *a lexical database for english. Commun. ACM*, 38(11) :39.41, 1995.

[G. salton, M. McGill, 84]: G. salton and M. McGill, *Introduction to modern information retrieval*. McGraw Int.Book Co. 1984.

[Grossman et al, 98]: D. A. Grossman, O. Fieder, *Information retrieval: Algorithms and heuristics*. Kluwer Academic Publishers, 1998.

[H. Peat & P. Wilett, 91]: PEAT H. J. & WILLETT P. (1991). *The Limitations of Term Cooccurrence Data for Query Expansion in Document Retrieval Systems*. Journal of the American Society for Information Science, 42(5), 378–383.

[Haines & Croft, 93]: D. Haines and W.B. Croft. *Relevance feedback and inference network.* In 16th Annual International ACM SIGIR Conference on Research and developement in Information Retrieval, pages 2, 11, 1993.

[Harman, 92]: D. Harman, *Relevance feedback revisited*. Proceedings of ACM SIGIR 92, pages: 125-132, 1992

[I. Ruthven, M. Lalmas, 03]: I. Ruthven and M. Lalmas. *A survey on the use of relevance feedback for information access systems.* Knowl. Eng. Rev., 18(2) :95{145, 2003.

[Ide, 71]: E Ide. *New experiments in relevance feedback.* In The SMART retrieval system - experiments in automatic document processing., pages 337{354,1971.

[J.J. Rocchio, 71] : J.J. *Rocchio. Relevance feedback in information retrieval*. In The SMART retrieval system-experiments in automatic document processing, pages: 313,323. Prentice Hall Inc, 1971.

[Khan, 00]: Latifur R. Khan, *Ontology-based Information Selection, Phd Thesis*, Faculty of the Graduate School, University of Southern California. August 2000.

[L. Tamine et al, 00] L. Tamine, M. Boughanem, *An Improved genetic algorithm to query optimization*. Proceedings of ACM-CIKM 2000, pages : 45-52, 2000.

[L. Tamine, 97] : L. Tamine : *Reformulation Automatique de Requête basée sur l'Algorithmique Génétique*, Actes du Congrès Inforsid, pp 643-662, Toulouse Juin 1997

[Lafferty et al , 01] : J. Lafferty and C. Zhai (2001). *Document Language Models,Query Models, and Risk Minimization for Information Retrieval***.** In Proceedings of the 24th ACM SIGIR Conference. on Research and Development in IR, pp.111-119.

[Lafferty et *al.*, 01]: Lafferty, J. and Zhai, C. *Document language models, query models, and risk minimization for information retrieval*. In SIGIR 2001: Proceedings of the 24th annual international ACM.

[Lavrenko,01]: V. Lavrenko and W.B. Croft, *Relevance-based Language Models*, Research and Development in Information Retrieval, Proc ACM-SIGIR'2001, pp. 120-127, 2001.

[Lelu et al, 92]: A. Lelu, C. François, *Information retrieval based on a neural unsupervised extraction of thematic fuzzy clusters*. Les réseaux Neuromimétiques et leurs applications (Neuro Nîmes), pages : 93-104, 1992.

[Lv et *al.*, 09] : Yuanhua Lv and ChengXiang Zhai. *Positional language models for information retrieval*. In SIGIR '09, pages 299{306, 2009.

[M. Mitra, A. Singhal, 98]: M. Mitra, A. Singhal, and C. Buckley. *Improving automatic query expansion.* In Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval., pages 206{214. Melbourne, 1998.

[MacFarlane et al, 00]: Andy MacFarlane, Julie A. McCann, Stephen E. Robertson: *Parallel Search Using Partitioned Inverted Files*. SPIRE 2000: 209-220.

[N. J. Belkin, C. Cool, 96]: N. J. Belkin, C. Cool, J. Koenemann, K. Bor Ng, and S. Park. *Using relevance feedback and ranking in interactive searching.* In Proceedings of the Fourth Text Retrieval Conference (TREC-4)., pages 181{210, 1996.

[N. J. Belkin, C. Cool, 97]: N. J. Belkin, A. Cabezas, C. Cool, K. Kim, K. B. Ng, S. Park, R. Press-man, S. Rieh, P. Savage, and H. Xie. *Rutgers interactive track at trec-5.* In Proceedings of the Sixth Text Retrieval Conference (TREC-5)., pages257{266, 1997.

[Pont and Croft , 98]: Jay M. Ponte and W. Bruce Croft, *A Language Modeling Approach to Information Retrieval*. Research and Development in Information Retrieval, Proc. ACM-SIGIR, pp. 275-281, 1998.

[Ponte et al., 98]: Jay M. Ponte and W. Bruce Croft, *A Language Modeling Approach to Information Retrieval.* Research and Development in Information Retrieval, Proc. ACM-SIGIR, pp. 275-281, 1998.

[Ponte et *al.*, 98]: Ponte, J. and Croft, W.B. *A language modeling approach to information retrieval*. ACM SIGIR 1998: Proceedings of the 21th annual international ACM SIGIR conference on Research and development in information retrieval. p. 275-281.

[Porter, 80]: M. Porter. 1980. *An algorithm for suffix stripping*. Program, 14(3):130-137, July, 1980.

[R. G. Sumner, K. Yang, 98]: R. G. Sumner, K. Yang, R. Akers, and W. M. Shaw. *Interactive retrieval using iris: Trec-6 experiments*. In Proceedings of the Sixth Text Retrieval Conference (TREC-6)., pages 711{734, 1998.

[Rijsbergen, 79]: Rijsbergen C. V., *"Information retrieval", In Information retrieval experiments*. Butterworths, London, 2nd edition, 1979.

[Robertson & al, 95]: S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. *Okapi at trec-3.* In Proceedings of the Third Text Retrieval Conference (TREC-3), pages 109, 126, 1995.

[Robertson et al, 76] : S. Robertson, EA. Sparck Jones, *Relevance weighting of search terms*, Journal of American Society of Information Science, JASIS, 27(3), pages: 129-146, 1976

[Robertson et al., 97]: S. E. Robertson and S. Walker. *On relevance weights with little relevance information.* In Proceedings of the 20th annual international ACM SIGIR

conference on Research and development in information retrieval, pages 16–24. ACM Press, 1997.

[Roget, 88]: P. Roget: *Roget's II the New Thesaurus*, Houghton Mifflin Company, Boston, USA, 1988

[S L'HADJ L, 09]: SAID L'HADJ L, *Recherche Conceptuelle d'Information Modèle d'Indexation Mixte : concepts-mots* : 2008,2009(mémoire magister ESI, ALGER)

[S L'HADJ L, 94] : Ihadjaden M., *Conception, réalisation et évaluation d'un système de recherche et de catégorisation automatique d'information textuelle sur Internet*, Thèse de l'université ParisIV, 1994.

[Salton et al., 68] : Gerard Salton, Michael Lesk: *Computer Evaluation of Indexing and Text Processing*. J. ACM 15(1): 8-36 (1968).

[Salton, 71]: G. Salton, *Automatic processing of foreign language documents.*, Prentice-Hall, Englewood Cliffs, Nj, 1971.

[salton, 83]: SALTON G., MCGILL M.J., *Introduction to Modern Information Retrieval* , *Mc Graw-Hill Computer Science Series*, 1983.

[Salton, 89]: G. Salton, *Automatic text processing: The transfromation, analysis and retrieval of information by computer*. Addison-Wesley publishing, pages : 85-92, 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.

[Signore et al, 92]: O. Signore, A.M.Garibald, M.Greco *Proteus: a concept browsing interface towards conventional Information Retrieval System* Database and Expert System Applications (DEXA), pages : 149-154, 1992.

[Singhal, 97]: A. K. Singhal, *Term weighting revisited*, PHD of Cornell University 1997.

[Soule Dupuy, 90] : Chantal Soule-Dupuy. *Systèmes de recherche d'information : le système Videotex Infodiab. Mecanismes d'indexation et d'interrogation*. Thèse de Doctorat, Université Paul Sabatier, Toulouse, France.

[Stanly & Goodman, 98] : Stanley F. Chen & Joshua Goodman. *An Empirical Study of Smoothing Techniques for Language Modeling.* Computer Science Group.Harvard University. Cambridge, Massachusetts, 1998

[Suy & Lang, 94]: I. Syu & S. D Lang: *A Competition-Based Connexionist Model for Information Retrieval,* Intelligent Multimedia Information Systems and Management (RIAO), New York, Vol 1. pp 248-265,1994

[Tebri, 04] :Tebry Hamid .*Formalisation et spécification d'un système de filtrage incremental* , 2004.

[Van Rijsbergen, 77]: C. J. Van Rijsbergen, *A theoritical basis for use of co-occurrence data information retrieval*, Journal of Documentation, 33(2), pages : 106-119, 1977.

[Voorhees, 99]: E. M. Voorhees *TREC Overview*. *In the proceedings of the seventh Text retrieval Conference TREC 7*, pages: 58-66, 1999.

[Walker, S. E. Robertson, 97]: S. Waller, S. E. Robertson, M. Boughanem, G. J. F. Jones, K. Sparck Jones. *Okapi at TREC-6 automatic and ad hoc, VLC routing, filtring and QSDR*. Proceeding of TREC-6, 1997.

[Xu & Croft, 96]: J. Xu & W.B. Croft: *Query Expansion Using Local and Global Document Analysis*. In Proc. ACM SIGIR Annual Conference on Research and Development, Zurich, 1996

[Yannick, 02]: Yannick Estève. *Thèse sur l'intégration de sources de connaissance pour la modélisation stochastique du langage, appliqué à la parole continue dans un contexte de dialogue oral Homme-Machine*, 2002.

[Yates, 99]: R. B.Yates, R. Neto, *Modern Inforation Retrival*. ACM Press, Addison Wesley, 1999.

[Yuanhua, 09] : Yuanhua Lv and ChengXiang Zhai. *A comparative study of methods for estimating query language models with pseudo feedback*. In CIKM '09, pages 1895{1898, 2009.**)**

[Zhai et *al.*, 01] : C. Zhai and J. Lafferty (2001). *Model-based Feedback in the Language Modeling Approach to Information Retrieval*. In Proceedings of the 10th International Conference on Information and Knowledge Management, pp.403-410.

Sites web:

[1] http://www.iro.umontreal.ca/~nie/IFT6255/Introduction.pdf

[2] http://www-csli.stanford.edu/~schuetze/information-retrieval-book.html

[3] http://terrier.org/

[4] http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf

[5] http://terrier.org/docs/v2.2.1/dfr_description.html

[6] http://terrier.org/docs/v2.2.1/configure_retrieval.html

[7] http://terrier.org/docs/v2.2.1/querylanguage.html

Annexe

La plateforme de RI Terrier 2.1

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

1. Introduction :

Terrier [1] est l'abréviation de TéRabyte RetrIEveR est un logiciel libre et open source entièrement écrit en java, il est classé dans la ligné des SRI, sa création fut par le département d'informatique de l'université de Glasgow du Royaune-Uni. Il a été conçu pour fonctionner sur la plupart des systèmes d'exploitation actuels tels que Windows ou Linux.

Son fonctionnement repose sur différentes méthodes d'indexations, plusieurs modèles de pondération et il permet la reformulation de la requête.

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

Nous pouvons différencier les deux principales étapes qui caractérisent le système Terrier :

- ***L'étape d'indexation*** *:* Permet l'extraction des termes des déférents documents du corpus (basic indexed unit).
- ***L'étape de recherche*** : permet de générer des résultats aux requêtes formulées par les utilisateurs.

---

[1] :http://www.terrier.org

2. Installation de Terrier :

2.1. Préalablement :

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 [2] .

2.2.. Installation :

Après avoir téléchargé une copie de Terrier version 2.1 sur la page d'accueil du projet Terrier, créer un nouveau répertoire et dézippé Terrier dans ce dernier [3].

3. La structure des répertoires de Terrier :

Terrier contient un ensemble de répertoires et ils 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).
- scr\ : contient le code source de Terrier.
- var/index : contient les structures de données : fichier inverse, fichier lexicon, index direct ,Document Index.
- var/résultats : contient les résultats de la recherche.
- licenses/ : contient les informations sur la licence des différents composants inclus dans Terrier.

---

[2] :http//www.oracle.com/technetwork/javase/downloads/index.html

[3] : httpp//www.terrier.org

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 [4].
- ***Interactive Terrier :*** permet une recherche interactive en exécutant le script interactive_terrier. C'est un moyen facile tester Terrier [5].
- ***Desktop Terrier :*** c'est une interface graphique pour la plateforme de RI Terrier.

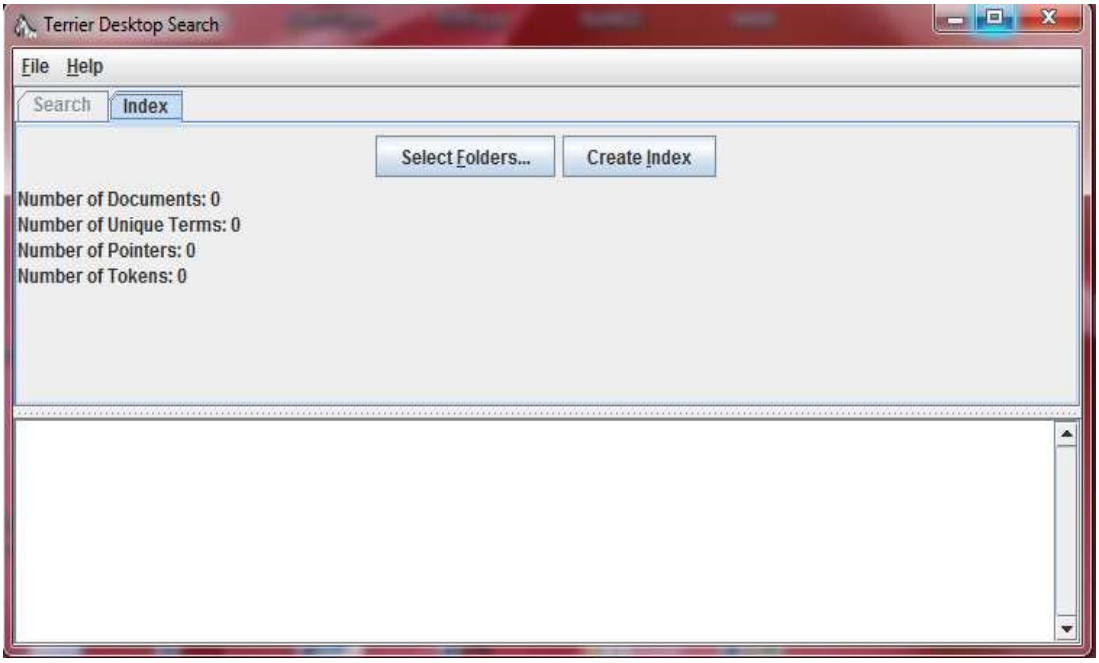


Figure.1 : Présentation de l'interface Desktop Terrier.

[4] : http://trec.nost.gov/

[5] :http://www.terrier.org/docs/v2.2.1/javdoc/uk/gla/terrier/applications/InteractiveQuerying.html

5. L'API d'indexation de Terrier :

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

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

La figure.2 ci-dessous présente les différentes étapes du processus d'indexation dans Terrier.

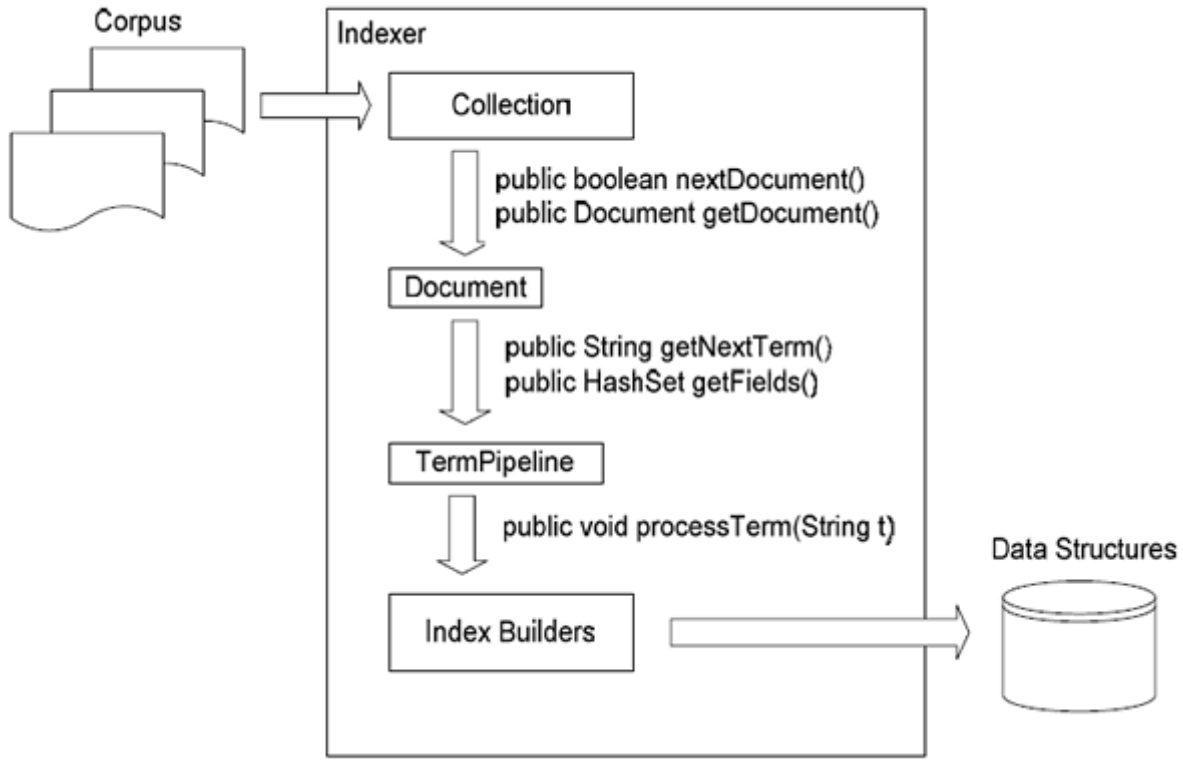


Figure.2 : présentation du processus d'indexation dans Terrier .

Le processus d'indexation est présenté dans les étapes suivantes :

5.1 Traitement sur la collection :

La composante qui 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. Utilisées généralement par Terrier pour intégrer de nouvelles sources de données (nouveaux corpus). Elle permet de parcourir un ensemble de documents en faisant appel à sa méthode nextDocument().

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

Après la sélection de la collection sur laquelle on fait la recherche, cette collection sera ensuite divisée en document particulier pour leur effectuer un autre traitement.

5.2. Traitement sur les documents :

Document c'est une interface qui se trouve dans le package uk.ac.gla.terrier.indexing. 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, MSExcelDocument, etc.

Pour chaque document de la collection, un processus visant à extraire les termes du document « tokénisation » :

- **Tokénisation :**

Le processus de tokénisation fait l'analyse lexicale des documents, il reconnait les mots dans le texte, comme il offre des possibilités de traitement sur les chiffres, les traits d'union, les signes de ponctuation…etc. Ce processus retourne en résultat l'ensemble de termes du document qu'on peut adopter comme termes d'indexation.

Tous les termes (token) obtenus de l'étape précédente (tokénisation) seront traités par un processus dit Term-Pipline qui lui-même se subdivise en deux sous processus.

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 extraits des documents. Cette étape se subdivise en deux sous processus.

5.3.1. Elimination des mots vides(stopword-removal) :

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

5.3.2. La normalisation (stemming) :

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

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

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

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 :BasicIndexer, BlockIndexer.

5.4.1. Construction de l'index :

Le but de cette étape est d'effectuer plusieurs traitements afin de parvenir à la création de structures d'index. La base de ces fichiers est le Lexicon qui est composé du terme et sa fréquence d'occurrence. Le stockage des différentes statistiques se fait sur les quatre principaux types de fichier :

- Document Index (data.docid): détient les informations à propos des documents tels que leur identificateur, leur taille….etc
- Vocabulary/Lexicon (data.lex): sauvegarde le vocabulaire.
- Inverted Index (data.if): index inverse.
- Direct Index (tada.df): index direct.

Pour les deux derniers nous les expliquerons un peu plus dans ce qui suit:

5.4.1.A) Index direct :

L'index direct enregistre pour un document les termes qui apparaissent dans ce dernier. Il est souvent utilisé pour la reformulation de la requête, la classification et comparaison des documents.

5.4.1.B) Index inversé :

L'index inversé contrairement à l'index direct enregistre pour un terme les documents dans lesquels il apparait, il contient aussi la position de chaque terme et sa fréquence dans ces documents.

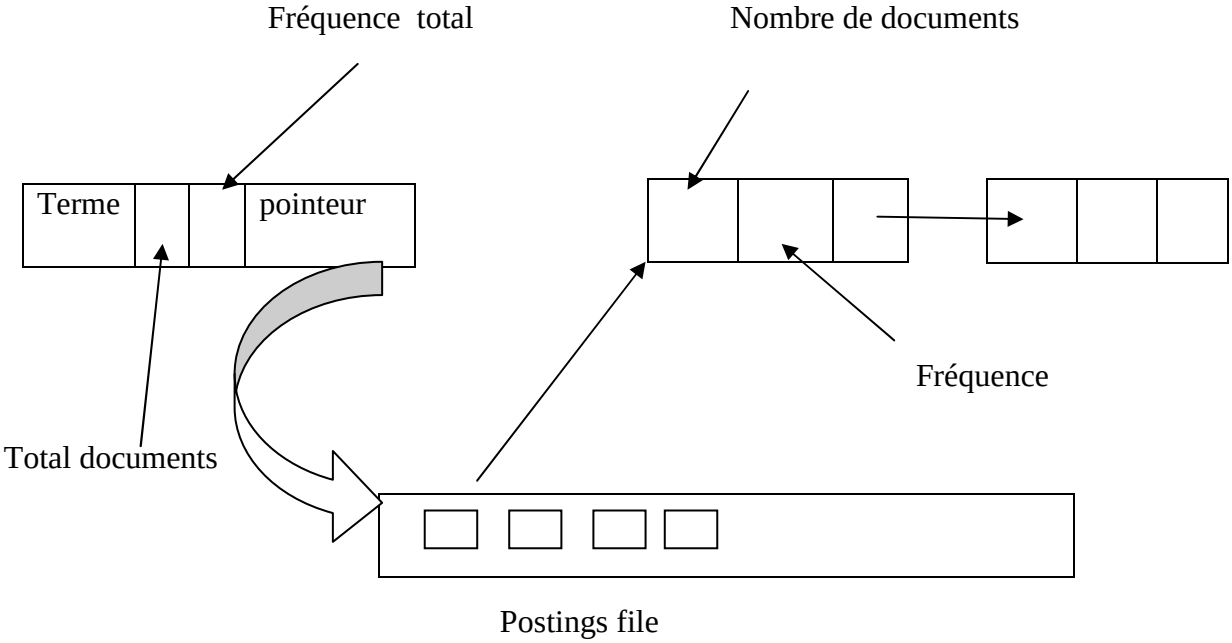


Figure.3: Composants de l'index inversé**.**

Pour construire ces fichiers terrier dispose des deux méthodes définies ci-dessous :

5.4.1 .B-1) Méthode à deux phases :

Elle s'effectue comme suit :

- Cette méthode commence par l'enregistrement des identificateurs pour chaque terme du document :

  -Ecrire l'index direct pour chaque document.

- Inversion :

  -Construire les posting-list pour chaque terme du document dans la mémoire.

  -Inverser ces listes une fois qu'on connait pour chaque terme les documents aux quels il appartient.

  Cette méthode est très couteuse.

5.4.1 .B-2) Méthode à une seul phase :

Elle s'effectue comme suit :

- Analyse la collection, construire l'index inversé dans la mémoire.

-En cas de saturation de la mémoire enregistrer dans le disque.

- Fusionner les fichiers inversés crées.

6. L'API de recherche de Terrier :

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

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

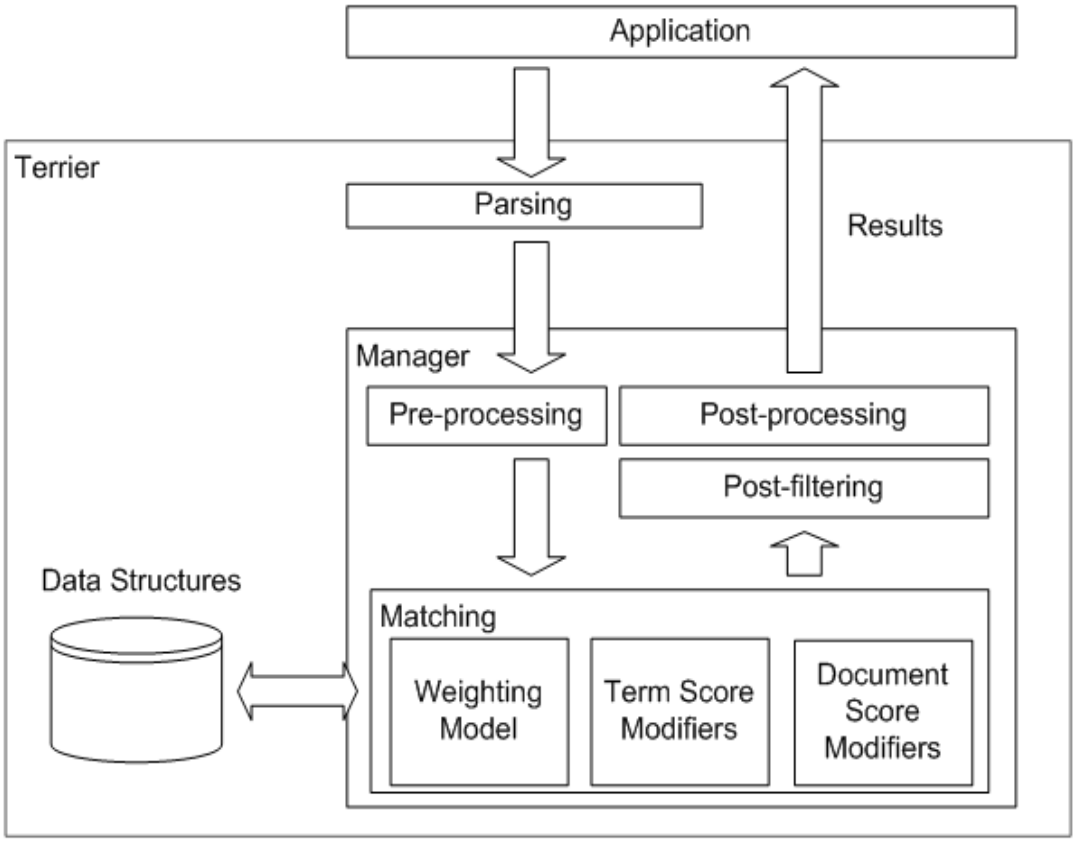


Figure.4: Présentation du processus de recherche de Terrier .

6.1. Query :

C'est l'entrée que l'application offre à Terrier (voir la **Figure 4**) .le Module Matching est responsable de déterminer quel document correspond à une requête spécifique et attribut un score au document qui respect la requête.

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

Terrier offre un support pour différents types de requête :

- *SingleTermQuery :* modèle de requête avec un seul terme
- *MultiTermQuery :* modèle de requête qui contient plus d'un terme.
- *FileQuery* : modèle de requête qualifié par un champ (field).

6.2. Manager :

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

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

Pre-processing, Matching, Post-processing, Post-filtering.

6.3. Pré-processing (prétraitement):

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

6.4. Matching:

Le composant Matching est responsable de déterminer des documents correspondent à la requête spécifiée et donne un score au document qui vérifie la requête. Il utilise le *weighting models* pour assigner un score à chaque mot de la requête dans le document. Terrier contient un nombre important de modèle de pondération. parmi ces modèles nous pouvons citer les suivants :

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

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

6.5. Post-processing/filtering (Post-traitement/Filtrage):

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

6.5.1. Post-traitement :

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

- **La reformulation de la requête :**

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

  Terrier déploie différents modèles de reformulation de la requête, il permet aussi l'extension et l'ajout d'autres modèles.

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

6.5.1. Post- Filtrage :

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

6.6. Résultat :

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

7. Modification Terrier :

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

Terrier permet la modification du code source pour intégrer tout changement nécessaire .Pour que toutes modifications soient prises en compte, il est nécessaire de recompiler tout le code source de Terrier.

8. Utilisation de Batch(TREC) Terrier :

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

8.1. Indexation :

1- Aller dans le répertoire où Terrier est installé en utilisant la commande *cd* :

C:\Master>**cd** terrier

2-Initialisation de Terrier pour l'indexation d'une nouvelle collection TREC.

C:\Master\terrier\bin>**trec_setup** <le chemin absolu du répertoire contenant les documents à indexer >

3-Maintenant nous pouvons indexer la collection TREC on utilisant l'option –i (index)

C:\Master\terrier\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 et Evaluation :

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

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

```
#add the qrels files to use for evaluation
c:\Master\terrier\6.txt
```

Figure.5 : Exemple d'un fichier trec.qrels

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

```
#add the topic files to use for querying
G:\topics.251-300
```

Figure.6 : Exemple d'un fichier trec.topics.list.

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

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

4-Pour lancer la recherche dans Terrier il suffit d'exécuter la commande :

C:\Master\terrier\bin>**trec_terrier –r**

5-Les résultants de la recherché peuvent être évalués en exécutant la commande :

C:\Master\terrier\bin>**trec_terrier –e**