Université Mouloud MAMMERI de TIZI-OUZOU

Faculté de Génie Electrique et d'Informatique

Département Informatique

2ème Année Master Académique

Option : Les systèmes informatiques

Mémoire de fin d'études

Thème : *Implémentation d'une méthode de calcul de pertinence à priori d'un document*

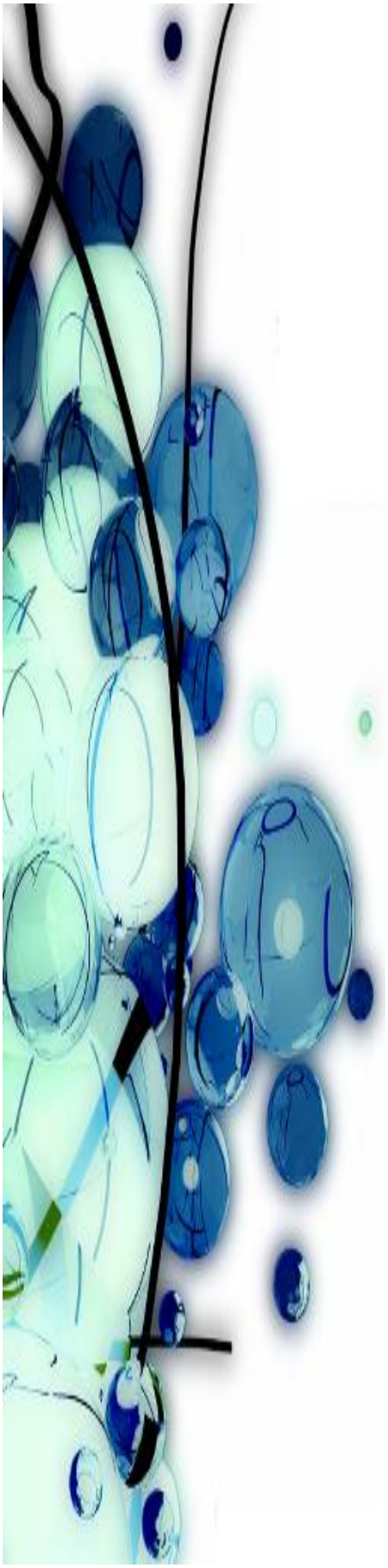
Réalisé par :

***KHELIFA** Souad.*

Dirigé par :

Mr A.HAMMACHE.

2011-2012



Remerciements

Je tiens à exprimer toute ma profonde gratitude et mes sincères remerciements à mon promoteur Mr HAMMACHE pour le temps qu'il m'a consacré tout au long de la réalisation de ce projet, ses précieux conseils m'ont été d'une grande aide.

Je tiens également à remercier tous les membres du jury d'avoir accepté de me faire l'honneur de juger ce travail.

Je remercie ma famille et mes amis (es) sans qui je n'aurai avancé chaque jour pour aboutir à la réalisation de mon travail.

KHELIFA Souad

Table des matières

Chapitre II : Modélisation de langue et pertinence à priori de documents en RI sur le Web

Chapitre III : Description de l'approche et Implémentation

Introduction générale

Introduction générale

L'information joue certainement un rôle essentiel dans la société d'information d'aujourd'hui et la croissance exponentielle de sa volumétrie et de son nombre potentiel d'utilisateurs entrainent de nouveaux défis scientifiques dans tous les domaines dont la tâche principale est la gestion de l'information. La Recherche d'Information (RI) est sans conteste l'un des domaines les plus concernés. En effet, l'objectif principal de la RI est de fournir des modèles techniques et des systèmes pour stocker, organiser des masses d'informations et sélectionner dans ces masses celles qui répondent à certains critères.

D'énormes efforts ont été déployés pour développer des approches et des techniques permettant de retrouver l'information voulue effectivement et efficacement à partir de vastes collections de données textuelles. Cependant, en raison de la surabondance de l'information d'une part et de sa large accessibilité à travers notamment le Web, d'autre part, leur mise en œuvre est confrontée à de nouveaux problèmes. En effet, retrouver au sein d'un corpus de documents volumineux et hétérogène, les seuls documents qui répondent précisément aux besoins des utilisateurs est devenu difficile car cette croissance accentue le retour de documents non pertinents pour l'utilisateur final.

Parmi les limites des systèmes actuels de RI est la non prise en compte de toutes les dimensions d'un document lors du processus de l'indexation et de la recherche. Ce qui engendre beaucoup de bruits en réponse à une requête d'un utilisateur. Parmi ces dimensions on peut citer : la taille d'un document, la structure des liens, le type d'URL d'un document,…etc. Dans notre cas, la caractéristique qu'on va utiliser est le score de dissemblance d'un document par rapport à une collection de documents c'est-à-dire qu'un document qui est dissemblable au reste des documents de la collection est à priori plus pertinent. Par la suite, on va formaliser cette hypothèse dans le cadre du modèle de langage.

L’objectif de notre projet est d’implémenter une méthode pour calculer la pertinence à priori d’un document par rapport à une collection de documents. Pour ce faire, on a souhaité le partager en trois chapitres :

- ✓ Dans le premier chapitre, on va présenter les différents concepts de base et principaux modèles de la Recherche d’Information.

- ✓ Dans le second chapitre, on va définir la modélisation de langage et la notion de calcul de pertinence de documents dans la Recherche d’Information sur le Web.

- ✓ Dans le troisième chapitre, on exposera notre approche ainsi que son implémentation et évaluation sur deux collections de tests AP88 et WSJ90-92.

On terminera par une conclusion et les perspectives possibles envisagées dans le cadre de ce travail.

Chapitre I

Concepts de base et principaux modèles de la Recherche d'Information

I.1 Introduction :

Le monde assiste depuis ces dernières décennies, à une production massive d'informations dans tous les domaines d'intérêt. De multiples directions de recherche ont tenté de mettre en œuvre des processus automatiques d'accès à l'information. L'objectif est d'exploiter au mieux les bases volumineuses de ces informations.

Un Système de Recherche d'Information (SRI) nécessite la combinaison de modèles et d'algorithmes permettant ainsi la représentation, le stockage, la recherche et la visualisation des informations. L'objectif principal de ce système est de mettre en œuvre un processus de comparaison entre les besoins de l'utilisateur et les documents d'une collection dans le but de retrouver ceux qui sont pertinents.

L'élaboration d'un mécanisme de recherche d'information pose alors des problèmes liés tant à la représentation qu'à la localisation de l'information pertinente. En effet, la recherche d'information induit un processus d'inférence véhiculé par l'objet de la requête, en se basant sur une description structurelle des unités d'information.

I.2 Les concepts de base de la recherche d'information :

La Recherche d'Information [Rijsbergen, 1979] [Grossman & Frieder, 1998] [Salton, 1971] [Baeza-Yates & B. A. Ribeiro-Neto, 1999] est traditionnellement définie comme l'ensemble des méthodes, procédures et techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur.
Gérer des textes implique stocker, rechercher et explorer des documents ou parties de documents.

Historiquement, la gestion des documents concernait surtout des spécialistes : bibliothécaires, documentalistes ou conservateurs. Ceux-ci devaient d'une part stocker les documents et en assurer la pérennité, et d'autre part en rendre l'accès possible.
Avec l'explosion de la quantité d'informations mises à disposition du grand public et des entreprises, notamment au travers d'Internet, l'automatisation du stockage et de la consultation de l'information est devenue un besoin. Le développement d'outils et de méthodes pour gérer ces quantités d'informations est bien plus qu'un besoin, une nécessité. Les outils les plus utilisés pour l'accès à l'information sur le web sont les moteurs de recherche tels que Google, Yahoo,…etc.
Ce sont les SRI (Systèmes de Recherche d'Information) qui assurent le stockage et permettent la consultation des informations.

I.2.1 Le processus de recherche d'information et des SRI :

Le processus de Recherche d'Information a pour but la mise en relation des informations disponibles d'une part, et les besoins de l'utilisateur d'autre part. Ces besoins sont traduits de façon structurée par l'utilisateur sous forme de requêtes.

La mise en relation des besoins utilisateurs et des informations est effectuée grâce à un Système de Recherche d'Information (SRI), dont le but est de retourner à l'utilisateur le maximum de documents pertinents par rapport à son besoin (et le minimum de documents non-pertinents).

Du côté du système, le processus de RI est composé de plusieurs fonctions :

- L'indexation des documents et des requêtes.
- La mise en correspondance requête-documents avec un ordonnancement des documents quand le modèle le permet.
- La restitution des documents reconnus pertinents par rapport à la requête.

Du point de vue utilisateur, le processus de recherche est composé de deux fonctions principales :

- L'interrogation du système par une requête.
- L'analyse des documents restitués par le système et une éventuelle reformulation de requête.

Schématiquement, le processus de recherche d'information peut être représenté comme suit :

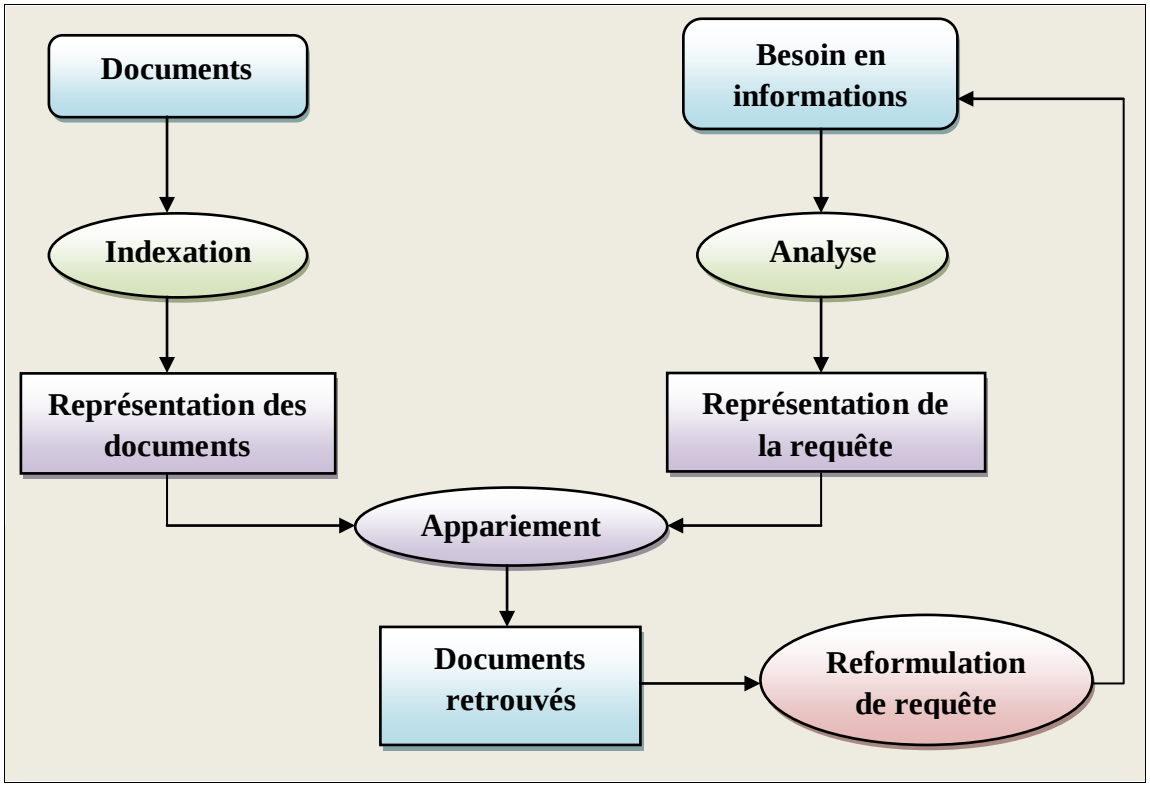


Figure I.1 : Processus en U de recherche d'information [Belkin et al., 1992]

La figure I.1 permet de dégager les principaux concepts suivants :

I.2.1.1 Les documents :

Le document constitue le potentiel d'informations élémentaire d'une base documentaire.

I.2.1.2 Le besoin en informations :

Un besoin en information d'un utilisateur est exprimé par une requête, c'est elle qui initie le processus de recherche. La requête doit contenir les concepts clés du besoin et les relations entre ces concepts. Elle est issue d'une analyse conceptuelle du besoin d'information qui est effectuée dans l'esprit de l'utilisateur de façon plus ou moins précise. En effet, l'utilisateur fait face à un « problème de vocabulaire » quand il tente de traduire son besoin d'information en une requête. Les mots sont souvent polysémiques et les concepts peuvent être décrits par un grand nombre de mots. Une fois formulée, la requête peut avoir la forme d'une expression en langue naturelle, ou encore d'une liste de concepts avec éventuellement un degré d'importance associé, ou encore une formule logique de concepts coordonnés par des opérateurs logiques. Une fois la requête exprimée, il est nécessaire de lui donner une forme utilisable par un SRI pour entamer le processus de recherche. Cela est réalisé par le processus d'analyse ou d'indexation.

I.2.1.3 Représentation des documents et des requêtes :

La représentation des documents et des requêtes est supportée par un ensemble de règles et notations permettant la traduction d'une requête ou d'un document à partir d'une description brute vers une description structurée. Ce processus de conversion est appelé « Indexation ».

L'indexation est un processus permettant d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique. Le résultat de l'indexation constitue le descripteur du document ou de la requête.

Le descripteur est une liste de termes ou groupes de termes significatifs pour l'unité textuelle correspondante, généralement assortis de poids représentant leur degré de représentativité du contenu sémantique de l'unité qu'ils décrivent. Les descripteurs des documents (mots, groupe de mots) forment le langage d'indexation. L'indexation est une étape primordiale dans la RI. De sa qualité, dépend en partie la qualité des réponses du système. Conscients de son importance, et soucieux de bien la réaliser, les développeurs des SRI ont proposé plusieurs manières de procéder.

Les principales sont l'indexation manuelle et l'indexation automatique. Elles sont définies comme suit :

- **Indexation manuelle :**

Chaque document est analysé par un spécialiste du domaine ou par un expert documentaliste.

L'indexation manuelle a pour avantage d'assurer une meilleure correspondance entre les documents et les termes choisis par les indexeurs pour les représenter (termes d'indexation), ce qui implique une meilleure précision dans les documents que le système de RI retourne en réponses aux requêtes des utilisateurs. L'inconvénient, est qu'elle exige un effort intellectuel (en temps et en nombres de personnes). De plus, un degré de subjectivité lié au facteur humain fait que pour un même document, des termes différents peuvent être sélectionnés par des indexeurs différents. Il peut même arriver qu'une personne, à des moments différents, indexe différemment le même document.

- **Indexation automatique :**

Elle reconnaît des chaînes de caractères constitutives de mots non vides. Elle détecte automatiquement les termes les plus représentatifs du contenu du document.

L'indexation automatique est sans doute celle qui a été le plus étudiée en recherche d'information, étant donnée sa faculté d'automatisation du processus d'indexation. Elle comprend un ensemble de traitements sur les documents. On y distingue : l'extraction automatique des descripteurs, l'utilisation d'un anti-dictionnaire pour éliminer les mots vides, la lemmatisation, le repérage de groupes de mots, la pondération des mots avant de créer l'index.

- ✓ **Extraction des descripteurs :**

Comme nous l'avons déjà évoqué, dans un SRI, le processus d'indexation consiste à extraire à partir du texte des documents, les termes clés appelés aussi "descripteurs". Les descripteurs représentent l'information atomique d'un index. Ils sont censés indiquer de quoi parle le document [Salton, 1983]. On parle aussi d'unités élémentaires (en anglais "tokens") [Gaussier et al., 2003]. Le but étant de les choisir de manière à ce que l'index (qui réduit la représentation) perde le moins d'informations sémantiques possibles. Les descripteurs peuvent être :

- Les *mots simples* du texte du document en soustrayant les mots vides.

- Les *lemmes* ou les *racines* des mots extraits.

- Les *concepts* qui sont des expressions contenant un ou plusieurs mots.

Ces concepts peuvent être écrits de manière libre par l'utilisateur ou, ce qui est souvent le cas, choisis dans une liste de concepts (on parle alors de vocabulaire contrôlé). Cette liste peut être décrite dans un thésaurus, une ontologie, une hiérarchie de concepts, etc.

- Les *N-grammes* qui sont une représentation originale d'un texte en séquence de N caractères consécutifs. On trouve des utilisations de bi-grammes et trigrammes dans la recherche documentaire (ils permettent de reconnaître des mots de manière approximative et ainsi de corriger des flexions de mots ou même des fautes de frappe ou d'orthographe). Ils sont aussi fréquemment utilisés dans la reconnaissance de la langue d'un texte [Harbeck et al., 1999], [Dunning, 1994].

- Les *contextes* : dans ce cas les descripteurs peuvent êtres des termes n'apparaissant pas explicitement dans le texte du document mais ayant un lien sémantique et/ou de cooccurrence avec les mots du document. L'ajout de termes reliés sémantiquement aux concepts décrits dans le document est déterminé par un calcul de proximité ou de similarité sémantique [Khan, 2000]. Ceci est rendu possible notamment grâce à la disponibilité de plus en plus croissante d'ontologies et autres ressources sémantiques externes. L'analyse de cooccurrences des mots dans un corpus est utilisée dans le cas de "Latent Semantic Indexing" [Deerwester et al., 1990]. Les documents et leurs mots sont alors représentés sur d'autres dimensions où les mots apparaissant dans un même contexte sont supposés proches sémantiquement.

- Les *groupes de mots* : un groupe de mots est souvent plus riche sémantiquement que les mots qui le composent pris séparément. En effet, à titre d'exemple, le terme composé "pomme de terre" est plus précis que "pomme" et "terre" pris isolément. Cet argument a conduit à considérer les groupes de mots comme unité de base dans le langage d'indexation.
Le problème qui se pose dans ce cas est le repérage des groupes de mots à associer. Trois approches existent dans la littérature :

- **Approches basées sur la cooccurrence** : Les groupes de mots sont extraits en utilisant la cooccurrence [Fagan, 1987] [Croft, 1991] [Rijsbergen, 1977] [Turpin et al., 1999] [Salton, 1983] [Robertson, 1977] [Lauer, 1995] [Alvarez et al., 2003], en partant de l'hypothèse que des termes (souvent réduits à deux ou trois mots) qui apparaissent ensemble dans le texte sont susceptibles de représenter un concept. De la même manière que pour les mots simples, le principe de pondération locale et globale (Tf * Idf) est appliqué pour éliminer les "groupes de mots vides" ("dans ce cas", "pour cela",...) étant donné qu'ils ne font pas partie du contenu réel du document et ne contribuent pas à distinguer les documents.

- **Approches linguistiques** : Ces approches [Sheridan et al., 1992] se basent sur une analyse syntaxique partielle ou l'utilisation de patrons (templates) syntaxiques pour détecter les termes composés [Bourigault, 1996] [Aussenac-Gilles et al., 2000] [David et al., 1996] [Sabah et al., 2000] [Jacquemin, 2001] [Church et al., 1990] [Jones et al., 2002]. Malgré les nombreuses études consacrées à ce problème, il n'existe pas encore, à notre connaissance, une méthode effective qui permette de distinguer les termes des non termes d'un point de vue syntaxique.

- **Approches mixtes** [Mitra et al., 1997] [Liu et al., 2004] [Daille, 1996] [Strzalkowski et al., 1995] [Smeaton, 86] [Salton et al., 89] : Ces approches combinent les deux informations (syntaxe et cooccurrence) pour améliorer la précision dans la détection des termes composés.

- ✓ **Elimination des mots vides :**

Un problème majeur de l'indexation est d'éviter les mots vides (pronom personnel, prépositions, articles, mots mathématiques (appartenir, contenir, inclure.)….etc) et choisir seulement les termes significatifs qui représentent au mieux un document donné.
Afin d'éliminer ces mots de force, on utilise une liste, appelée " stoplist " (ou parfois anti-dictionnaire) qui contient tous les mots qu'on ne veut pas garder. Une autre méthode consiste à éliminer les mots qui dépassent un certain nombre d'occurrences dans la collection.

- ✓ **La lemmatisation :**

L'idée qui conduit à utiliser la lemmatisation est de pouvoir indexer un ensemble de mots par un seul mot qui représente le même concept.
En effet, on remarque que beaucoup de mots ont des formes différentes, mais leur sens reste le même ou très similaire et notamment dans le cas des mots conjugués. Ces mots ont la même racine (lemme). Ainsi, on arrive à éliminer les terminaisons des mots, et garder seulement la racine, on a donc une forme identique pour eux. Plusieurs méthodes sont utilisées : la troncature, variétés de successeurs, méthode de n-gramme…etc.

- ✓ **La pondération des termes :**

La pondération d'un terme d'indexation est l'association de valeurs numériques à ce terme de manière à représenter son pouvoir de discrimination pour chaque document de la collection. Cette caractérisation est liée au pouvoir informatif du terme pour le document donné.

Quand un SRI retourne un document à l'utilisateur, celui-ci en tire de l'information. Cette information a un sens pour un utilisateur donné, il se peut que la même information sémantique puisse prendre une importance plus ou moins grande, susciter un intérêt plus ou moins vif selon les individus et les circonstances. L'information a donc un sens pour un utilisateur donné qui lui attribue une certaine valeur.

Par la suite, l'information a une certaine force : en effet l'énoncé d'un évènement probable informe peu voire pas (« demain il fera jour »), par contre l'annonce d'un évènement improbable suscite plus d'intérêt (« demain il fera beau »). Autrement dit, plus un phénomène est probable moins il est informant. Ainsi, à une information sont attachés un sens et une probabilité qui permet de quantifier, de mesurer son contenu d'informations.

Dans la langue naturelle, chaque signe (lettre, phonème, mot, catégorie grammaticale,…) revient avec une fréquence stable, donc prévisible [Guiraud, 1967].

Plusieurs distributions aléatoires rencontrées dans le langage sont de la même forme que celles analysées par les théoriciens de l'information. L'existence de telles distributions a été relevée en 1916 par le sténographe français J.B. Estoup qui a montré que si les mots d'un texte sont rangés par ordre de fréquence décroissante, la fréquence du second terme apparaissant dans cette liste est la moitié du premier et la fréquence du 3ème est le tiers de celle du premier. Cette constatation a donné naissance vingt ans plus tard à la célèbre loi de distribution de Zipf [Zipf, 1949] qui dit que la fréquence d'un mot est inversement proportionnelle à son rang dans la liste des termes classés par fréquence décroissante ou encore que le produit de la fréquence de n'importe quel mot par son rang est constant. Cette loi est écrite sous la forme :

$$rang \times fréquence \approx constante \qquad (1)$$

Mandelbrot a montré dans [Mandelbrot, 1965] que la formule de Zipf devait être aménagée en :

$$(rang + b)^{a} \times fréquence = constante \qquad (2)$$

Où :
b correspond à l'aplatissement du sommet de la courbe rang×fréquence qui devient négligeable quand le rang croît. *b* indique qu'un terme peu fréquent est plus important qu'un terme très fréquent. *a* est un indice légèrement supérieur à 1.

La relation entre fréquence et rang permet de choisir les termes représentatifs. Luhn dans [Luhn, 1955] a montré que la fréquence d'apparition d'un terme dans un texte en langue naturelle est caractéristique de son pouvoir de représentation du contenu de ce texte. Le pouvoir de représentation d'un terme est parfois nommé l'informativité du terme. Cette notion fait référence à la quantité de sens qu'un mot porte. Un terme très fréquent dans la collection (fréquence absolue) est peu discriminant car il est restitué dans de nombreux documents et inversement un terme très peu fréquent dans un texte a peu d'influence sur le processus de recherche car il n'est pas représentatif du contenu sémantique de ce texte.

Le rapport entre rang×fréquence et importance du terme est représenté par les courbes suivantes :

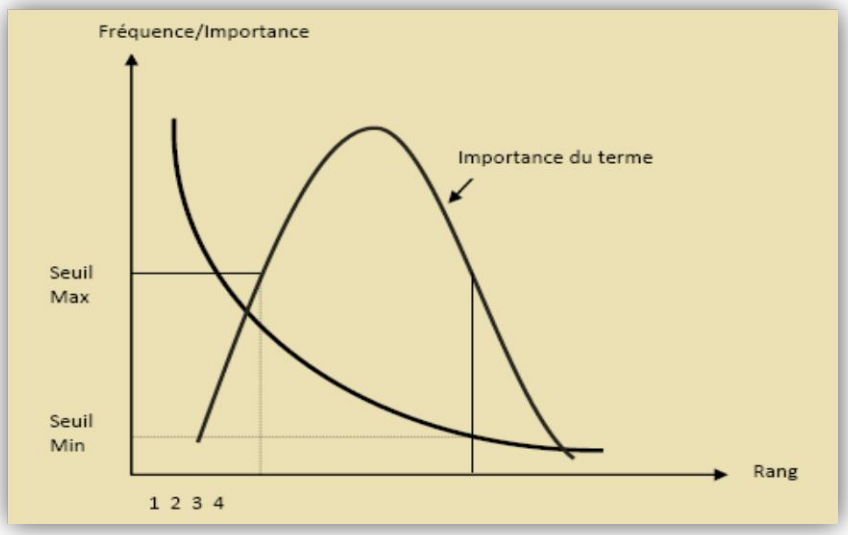


Figure I.2 : Rapport entre la fréquence d'un terme et son importance
[Y. Champclaux, 2009]

Les termes représentatifs vont être triés en définissant un seuil maximum en dessous duquel les termes à rang×fréquence élevé ne sont pas conservés et un seuil minimum au dessus duquel les termes à rang×fréquence faible ne sont pas conservés non plus.
La fréquence relative d'un terme dans un document est représentative du pouvoir de représentation du terme pour le document, au même temps, la fréquence absolue d'un terme dans la collection est caractéristique du pouvoir de discrimination du terme pour les documents. Il est donc important de prendre en compte la fréquence relative et la fréquence absolue d'un terme lors de sa pondération. La pondération c'est l'association d'une valeur appelée poids à un terme.

Pour associer un poids à un terme, on peut procéder de différentes manières :

- 0 ou 1 : exprime la présence (1) ou l'absence (0) d'un terme dans le document.
- *tf* : *term-frequency* est la fréquence du terme dans le document, c'est-à-dire, le nombre d'occurrences d'un terme dans le document.
- *idf* : *Inverse of Document Frequency* est la fréquence absolue inverse. C'est un facteur qui varie inversement proportionnel au nombre *n* de documents où un terme apparaît dans une collection de *N* documents.

La fréquence absolue inverse est égale à [Salton, 1987] :

$$idf = \log (N/n) \qquad (3)$$

Avec *N* le nombre total de documents dans la collection et *n* le nombre de documents où le terme apparaît.

Le poids d'un terme *j* dans le document *i* s'écrit alors généralement [Sparck Jones, 1972] :

$$poids_i(j) = tf_{ij} \times idf_j \qquad (4)$$

Où tf_{ij} est la fréquence d'apparition du terme *j* dans le document *i* et idf_j est la fréquence absolue inverse du terme *j* dans la collection. Ainsi le poids d'un terme augmente si celui-ci est fréquent dans le document et décroît si celui-ci est fréquent dans la collection.

La formule *tf*×*idf* fournit une bonne représentation du poids pour les corpus dont les documents sont de taille homogène c'est-à-dire composés de documents de tailles similaires. Dans le cas de corpus non homogènes, il peut être intéressant de procéder à une normalisation du poids. Pour cela une normalisation est incorporée à la formule du poids [Salton, 1987]:

$$poids_i(j) = tf_{ij} \times idf_j \,/\, \sum_{k=1}^{t} tf_{ik},\, idf_k \qquad (5)$$

$$poids_i(j) = tf_{ij} * idf_j \,/\, \sqrt{\sum_{k=1}^{t} (tf_{ik}, idf_k)^2} \qquad (6)$$

Avec $\sum_k tf_{ik}$, idf_k la somme des poids des termes du document *j* et *t* le nombre de termes dans le document.

- ✓ **Construction de l'index :**

Le résultat d'une indexation donne un ensemble de termes et leurs pondérations pour chaque document comme suit :

$$d_j \rightarrow \{\ldots(t_i, a_{ij})\ldots\}$$

Avec *t* le terme d'indice i dans le vocabulaire et a_{ij} son poids dans le document d_j.

Avec cette structure, il est facile de trouver les termes inclus dans un document. Cependant, étant donné une requête contenant quelques termes, il est plus intéressant de retrouver les documents correspondant à chacun de ces termes. Pour cela un fichier inversé est construit avec la structure suivante :

$$t_i \rightarrow \{\ldots(d_j, a_{ij})\ldots\}$$

I.2.1.4 L'appariement document / requête :

Le processus d'appariement requête-document est le noyau d'un système de recherche d'information. Il permet d'associer à chaque document une valeur de pertinence vis à vis d'une requête. Les documents ayant une pertinence positive sont sélectionnés.
La mesure de pertinence est calculée à partir d'une fonction de similarité, notée *RSV* (*Q*,*d*) (*Retrieval Status Value*), *Q* étant une requête et *d* un document.

Elle tient compte des poids des termes déterminés en fonction d'analyses statiques et probabilistes. Notons que ce processus est étroitement lié aux représentations des documents et des requêtes.

En effet, si l'opération d'indexation est la même dans la plupart des modèles de recherche d'information, ces derniers diffèrent souvent par rapport aux fonctions utilisées pour la mesure des poids et pour l'appariement requête-document.

La notion de pertinence est souvent difficile à appréhender, on parle souvent de deux types de pertinence :

- la pertinence utilisateur : le document est jugé pertinent par l'utilisateur en fonction de son besoin en information,
- la pertinence système : le document est jugé pertinent par le SRI pour une requête sur la base de la fonction de pertinence.

Les mesures d'évaluation (I.4) permettent de comparer ces deux pertinences.

I.2.1.5 Mécanismes de reformulation de requête :

La qualité d'un SRI dépend de sa capacité à retrouver des documents pertinents pour l'utilisateur. Elle est donc liée à l'indexation (au choix des termes par le système) et au choix des termes que l'utilisateur a fait pour formuler sa requête. C'est pour résoudre les problèmes liés à un mauvais choix de termes et pour pallier les lacunes de l'indexation qu'ont été introduits (pour la première fois dans [Rijsbergen, 1979]) les mécanismes de reformulation de requête.

La reformulation consiste à réajuster les poids des termes de la requête ou à rajouter des termes reliés à ceux de la requête initiale. Elle peut être manuelle (avec intervention de l'utilisateur) ou automatique. Parmi les méthodes de reformulation de requête, nous pouvons citer la réinjection de pertinence ou relevance feedback. Elle consiste à exploiter l'information que le jugement de pertinence de l'utilisateur fournit sur les documents initialement restitués par le système en lien avec sa requête. La reformulation peut se faire sans expansion de la requête par simple repondération des termes (en maximisant le poids des termes apparaissant dans des documents jugés pertinents et en minimisant le poids des termes présents dans des documents reconnus non pertinents par l'utilisateur) ou par ajout des termes à la requête originale.

L'ajout d'un terme peut se faire de plusieurs manières : soit par une interaction entre l'utilisateur et le système soit de façon automatique, on parle alors de réinjection de pertinence aveugle. Le système considère les premiers documents comme pertinents. Salton et Buckley proposent dans [Salton et al., 1990] une comparaison de différentes méthodes de réinjection de pertinence, il en ressort que la méthode Ide Dec-Hi obtient les meilleurs résultats en termes de précision devant la méthode de Rocchio [Rocchio, 1971] et devant les approches de réinjection de pertinence probabilistes. Selberg propose dans [Selberg, 1997] une description des méthodes de réinjection de pertinence dans les approches classiques de RI (booléenne, vectorielle et probabiliste).

I.3 Les modèles de recherche d'information :

L'indexation choisit les termes pour représenter le contenu d'un document ou d'une requête, le modèle permet de donner une interprétation des termes choisis pour représenter le contenu d'un document. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit deux fonctions :

- La première est de créer une représentation interne pour un document ou pour une requête basée sur ces termes,
- La seconde est de 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 lui qui détermine le comportement clé d'un SRI.

La figure I.3 présente une classification des principaux modèles utilisés en recherche d'information. Les trois principales classes ou modèles de recherche d'information sont :

- ✓ Les modèles basés sur la théorie des ensembles.
- ✓ Les modèles algébriques.
- ✓ Les modèles probabilistes.

- Les modèles basés sur la théorie des ensembles: Cet ensemble de modèles se base sur la théorie des ensembles, ainsi une requête est représentée par un ensemble de termes séparés par des opérateurs logiques (OR, AND, NOT). Le document est, quant à lui, représenté par une liste de mots clés. Ces modèles permettent d'effectuer des opérations d'union, d'intersection et de différence lors de l'interrogation. Le modèle le plus connu et le plus simple de cette catégorie est le modèle booléen. On y retrouve également le modèle booléen étendu et le modèle basé sur les ensembles flous.

- Les modèles algébriques: Ces modèles regroupent tous les modèles de RI qui utilisent une représentation vectorielle des documents et des requêtes [Piwowarski, 2003] dans lesquels la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. La similarité est calculée algébriquement en se basant sur la représentation du document et de la requête. Le représentant le plus connu de cette catégorie est le modèle vectoriel. Plusieurs modèles s'inspirant du modèle vectoriel ont été proposés dans le domaine de la RI : le modèle vectoriel généralisé [Wong et al., 1985], le modèle connexionniste [Boughanem, 1992] [Mothe, 1994] [Kwok, 1989] et le modèle LSI (Latent Semantic Indexing) [Deerwester & al., 1990].

- Les modèles probabilistes: Ces modèles se basent sur la théorie des probabilités. La pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

On distingue le modèle probabiliste de base [Maron & al., 1960], le modèle inférentiel bayésien [Turtle, 1991] et le modèle de langage [Ponte & Croft, 1998].

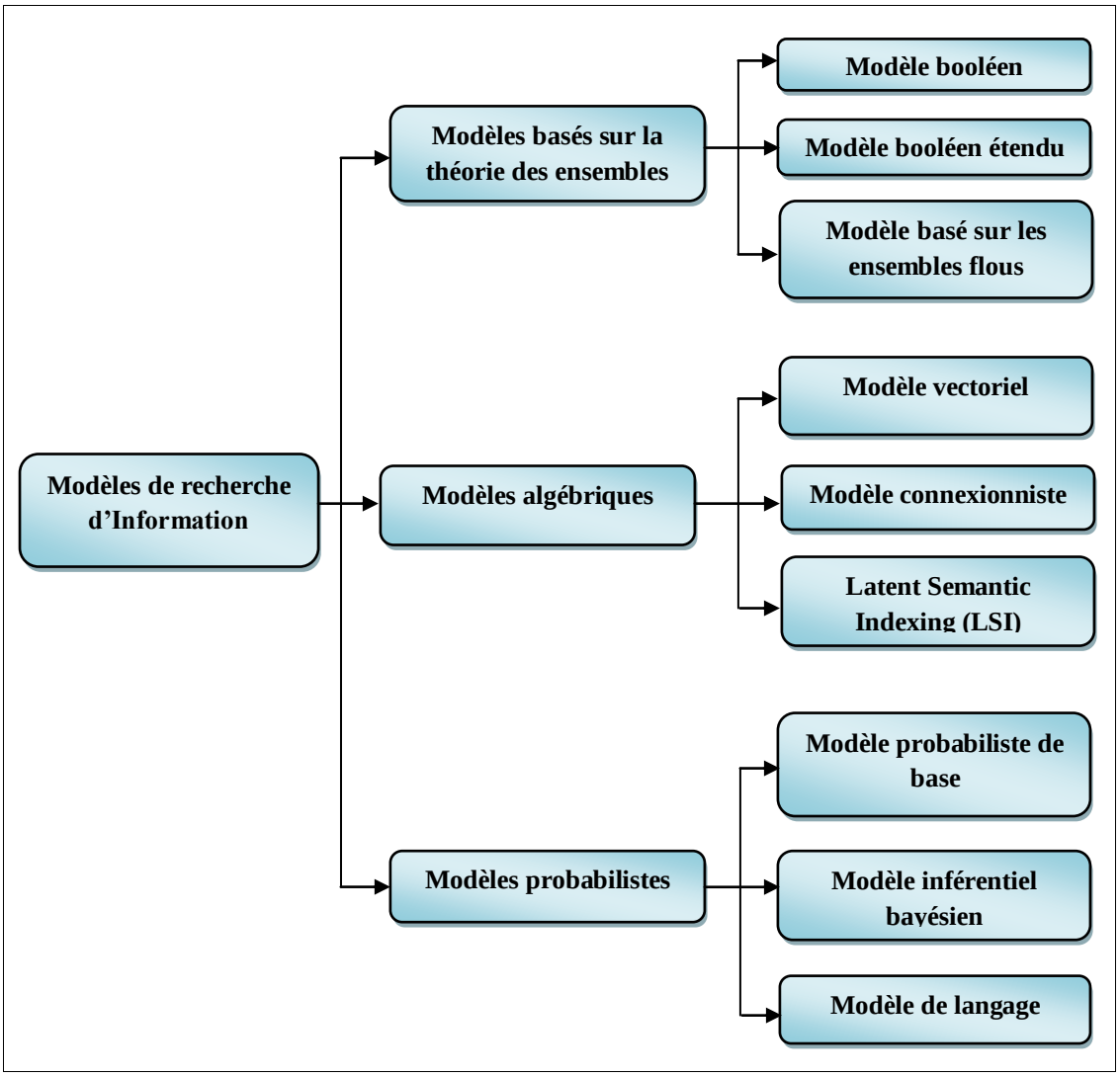


Figure I.3 : Taxonomie des modèles de Recherche d'Information [K.Amrouche, 2008]

I.3.1 Les modèles basés sur la théorie des ensembles :

I.3.1.1 Le modèle booléen :

Le modèle booléen propose la représentation d'une requête sous forme d'une formule logique. Les termes d'indexation sont reliés par des connecteurs logiques *ET*, *OU* et *NON*.

Le processus de recherche mis en œuvre par le système consiste à effectuer des opérations sur des ensembles de documents définis par l'occurrence ou absence de termes d'indexation afin de réaliser un appariement exact avec l'équation de la requête. De manière formelle, le modèle de recherche booléen est défini par un quadruplet *(T, Q, D, F)*
Où :
T: Ensemble des termes d'indexation
Q : Ensemble de requêtes booléennes
D : Ensemble des documents de la collection
F : Fonction présence définie par : D x Q → {0 ,1}

F (d,t) = 1 si t occurre dans D
= 0 sinon

Sur la base de cette fonction, on calcule la ressemblance relativement à la forme de la requête comme suit :

Formulation booléenne	Formulation d'évaluation
F(dk , ti et tj)	Min (F(dk , ti) , F(dk , tj)) = F(dk , ti) * F(dk , tj)
F(dk , ti ou tj)	Max (F(dk , ti) , F(dk , tj)) = F(dk , ti) + F(dk , tj) - F(dk , ti) * F(dk , tj)
F(dk , Non ti)	1 - F(d, ti)

Tableau I.1 : Calcul de la ressemblance relative à la forme de la requête [L.Tamine, 2000]

Le modèle booléen présente le principal avantage de simplicité de mise en œuvre.

Toutefois, il présente les principaux inconvénients suivants :
- les formules de requêtes sont complexes, non accessibles à un large public,
- la réponse du système dépend de l'ordre de traitement des opérateurs de la requête,
- la fonction d'appariement n'est pas une fonction d'ordre (les documents retournés ne sont pas ordonnés),
- les modèles de représentation des requêtes et documents ne sont pas uniformes. Ceci rend le modèle inadapté à une recherche progressive.

I.3.1.2 Le modèle booléen étendu :

Le modèle booléen étendu [Fox, 1983] [Salton, 1989] complète le modèle de base (modèle booléen) en associant des poids d'indexation à chaque terme d'une requête et d'un document, ceci permet au SRI de supporter un appariement approché et de mesurer un score de pertinence.

Considérons un ensemble de termes $t_1,\ldots,t_n$ et soit wd_{ij} le poids du terme t_i dans le document $D_j = (wd_{1j},\ldots,wd_{nj})$, avec $1 \leq i \leq n$ et $0 \leq wd_{ij} \leq 1$. La similarité entre le document D_j et une requête Q_k décrite sous une forme conjonctive ou disjonctive est donnée comme suit :

$$\textbf{Opérateur OR :} \quad RSV(D_j, Q_k) = \left[\frac{\sum_{j=1}^{n} \sum_{i=1}^{n} wq_{ik}^{P} \, wd_{ij}^{P}}{\sum_{i=1}^{n} wq_{ik}^{P}} \right]^{1/P} \qquad (7)$$

$$\textbf{Opérateur AND :} \quad RSV(D_j, Q_k) = \left[\frac{\sum_{j=1}^{n} \sum_{i=1}^{n} wq_{ik}^{P} \, (1 - wd_{ij}^{P})}{\sum_{i=1}^{n} wq_{ik}^{P}} \right]^{1/P} \qquad (8)$$

Où :
p est une constante $0 \leq p \leq \infty$ et wq_{ik} le poids du terme t_i dans la requête Q_k.

I.3.1.3 Le modèle basé sur les ensembles flous :

La théorie des ensembles flous est due à Zadeh [Zadeh, 1965]. Elle est basée sur l'appartenance probable, et non certaine, d'un élément à un ensemble. Un ensemble flou est formellement décrit comme suit :

$$EF = \{(e_1, f_{EF}(e_1)) , \ldots . (e_n, f_{EF}(e_n))\}$$

Où :
e_i : Elément probable de E.

f_{EF}: $E \rightarrow [0, 1]$
$e_i \rightarrow f_{EF}(e_i)$ = degré d'appartenance de e_i à E.

Les opérations de base sur les ensembles flous sont alors définies comme suit :

- **Intersection :** $f_{A \cap B}(e_i) = \text{Min}(f_A(e_i), f_B(e_i)) \quad \forall e_i \in E$

- **Union :** $f_{A \cup B}(e_i) = \text{Max}(f_A(e_i), f_B(e_i)) \quad \forall e_i \in E$

- **Complément :** $f_{A'}(e_i) = 1 - f_B(e_i) \quad \forall e_i \in E$ avec $A' = \{x \in A \text{ et } x \notin B\}$

Une extension du modèle booléen basée sur les ensembles flous a été proposée par Salton [Salton, 1989]. L'idée de base est de traiter les descripteurs de documents et de requêtes comme étant des ensembles flous.

L'ensemble flou des documents supposés pertinents à une requête est obtenu en suivant les étapes suivantes :
- Pour chaque terme t_i de Q_k, construire l'ensemble flou D_t des documents contenant ce terme.
- Effectuer sur les ensembles D_t, les opérations d'intersection et union selon l'ordre décrit dans l'expression de Q_k relativement aux opérateurs *ET* et *OU* respectivement.
- Ordonner l'ensemble résultat de la précédente opération selon le degré d'appartenance de chaque document à l'ensemble associé à chaque terme.

I.3.2 Les modèles algébriques :

I.3.2.1 Le modèle vectoriel :

Le modèle vectoriel est un modèle algébrique où l'on représente les documents et les requêtes par des vecteurs dans un espace multidimensionnel dont les dimensions sont les termes issus de l'indexation [Salton, 1983]. Comme on l'a vu précédemment, la création de l'index implique le parcours de la collection, la recherche des termes pertinents, le traitement lexical des termes retenus et enfin l'analyse statistique de la distribution de ces termes dans les documents et dans la collection pour leur attribuer un poids. Ainsi, les documents et la requête sont représentés comme des vecteurs dans le repère des termes. La comparaison de la requête au document est effectuée en comparant leurs vecteurs respectifs. On ramène ainsi une proximité sémantique à une mesure de distance géométrique. Soit R l'espace vectoriel défini par l'ensemble des termes: <t1, t2, …, tn>, un document *d* et une requête *q* peuvent être représentés par des vecteurs de poids comme suit:

$$d \rightarrow <w_{d1}, w_{d2}, \dots, w_{dn}>$$
$$q \rightarrow <w_{q1}, w_{q2}, \dots, w_{qn}>$$

Où :
w_{di} et w_{qi} correspondent aux poids du terme t_i dans le document d_i et dans la requête q et n correspond au nombre de termes du vocabulaire d'indexation.

Étant donnés ces deux vecteurs, leur degré de correspondance est déterminé par leur similarité. Plusieurs mesures peuvent être utilisées pour déterminer cette similarité :

Mesure	Notation vectorielle	Notation ensembliste
Produit scalaire	$\mathbf{Sim_0(d,q) = \sum_i w_{di} \times w_{qi}}$	$\mathbf{\lvert d \cap q \rvert}$
Cosinus	$\mathbf{Sim_1(d,q) = \frac{\sum_i w_{di} \times w_{qi}}{\sqrt{\sum_i w_{di}^2 \times \sum_i w_{qi}^2}}}$	$\mathbf{\frac{\lvert d \cap q \rvert}{\sqrt{\lvert d \rvert} \times \sqrt{\lvert q \rvert}}}$

Coefficient de Dice	$\mathbf{Sim_2(d,q) = \dfrac{\sum_i w_{di} \times w_{qi}}{\frac{1}{2}\left((\sum_i w_{di}^2) + (\sum_i w_{qi}^2)\right)}}$	$\mathbf{\dfrac{2 \times \lvert d \cap q \rvert}{\lvert d \rvert + \lvert q \rvert}}$
Mesure de Jacard	$\mathbf{Sim_3(d,q) = \dfrac{\sum_i w_{di} \times w_{qi}}{\sum_i w_{di}^2 + \sum_i w_{qi}^2 - \sum_i w_{di} \times w_{qi}}}$	$\mathbf{\dfrac{\lvert d \cap q \rvert}{\lvert d \cup q \rvert}}$
Mesure de recouvrement	$\mathbf{Sim_4(d,q) = \dfrac{\sum_i w_{di} \times w_{qi}}{\min(\sum_i w_{di}, \sum_i w_{qi})}}$	$\mathbf{\dfrac{\lvert d \cap q \rvert}{\min(\lvert d \rvert, \lvert q \rvert)}}$

Tableau I.2 : Mesures et similarités [Y. Champclaux, 2009]

Les documents ayant les plus hauts degrés de correspondance sont retournés en réponse à la requête.

Voici un exemple qui illustre un des intérêts de l'approche vectorielle, à savoir ramener un problème complexe de comparaison de documents à un problème de comparaison de mesures de similarité ou de distances.

Soit l'espace des termes rencontrés pendant l'indexation : {*modèle, graphe, similarité*},
Soit le document d1 représenté par le vecteur : {(*modèle*, 1), (*graphe*, 2)},
Soit le document d2 représenté par le vecteur : {(*modèle*, 2), (*graphe*, 1), (*similarité*, 2)},
Soit la requête q représentée par : {(*graphe*, 1), (*similarité*, 2)}.
On représente les vecteurs d1, d2 et q dans le repère des termes :

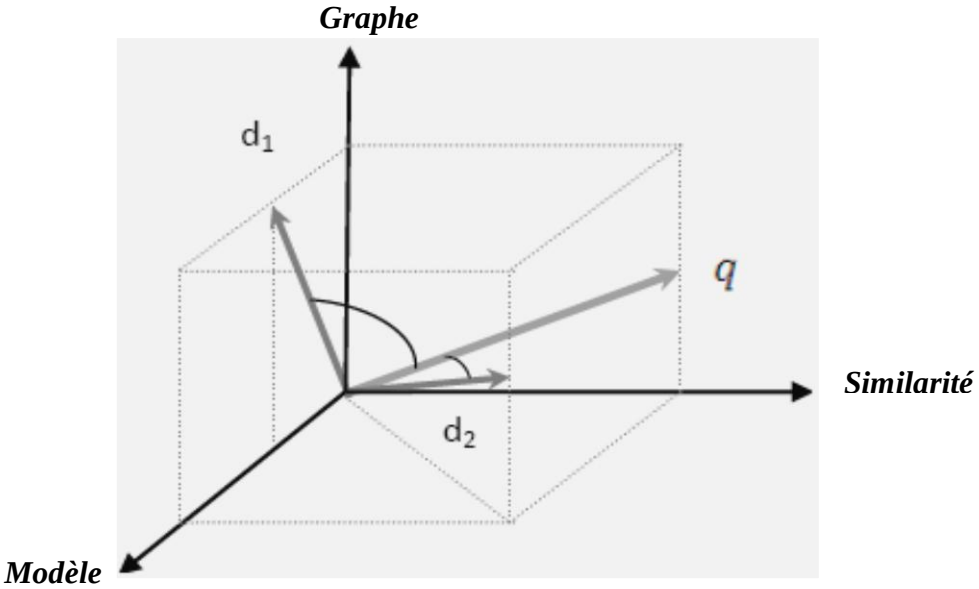


Figure I.4 : Vecteurs documents et vecteur requête dans l'espace des termes

En RI, dans les modèles à espace vectoriel, il est d'usage de représenter les documents, la requête et les termes d'indexation dans une matrice dont chaque ligne représente un document (la requête étant considérée comme un document) et chaque colonne représente un terme de l'index. Ainsi les documents et la requête de l'index peuvent être représentés comme suit :

Termes d'indexation / Documents	Graphe	Modèle	Similarité
d1	1	2	0
d2	1	2	2
q	1	0	2

Tableau I.3 : Représentation matricielle de documents [Y. Champclaux, 2009]

Ce type de représentation dit matricielle est très utile dès lors que l'on quitte l'espace à trois dimensions.

D'une manière générale, les résultats de recherche tendent à prouver que les systèmes de recherche vectoriels sont plus performants en terme de précision que les systèmes de recherche booléens [Turtle et al., 1991].

I.3.2.2 Le modèle connexionniste :

Les systèmes de recherche d'informations basés sur le modèle connexionniste [Kwok, 1989], [Boughanem, 1992], [Crestani, 1995] utilisent les fondements des réseaux de neurones tant pour la modélisation des unités textuelles que pour la mise en œuvre du processus de recherche d'information. Ce modèle peut être vu comme un modèle vectoriel récurrent non linéaire, les neurones formels représentent des objets de la recherche d'information.

L'idée de base est que la recherche d'information soit un processus associatif bien représenté par les mécanismes de propagation et d'activation des réseaux de neurones. Par ailleurs, les capacités d'apprentissage de ces modèles peuvent permettre d'obtenir des systèmes de recherche d'information adaptatifs.

Sur la base de l'architecture du réseau qu'ils utilisent, les systèmes de recherche d'information connexionnistes peuvent être répartis en deux catégories : les modèles à auto-organisation et les modèles à couches.

- **Les modèles à auto-organisation** : les systèmes de cette catégorie sont basés sur le modèle des cartes topologiques de Kohonen [Kohonen, 1989]. L'apprentissage du réseau permet d'effectuer la classification des documents à partir de leurs descriptions initiales.

- **Les modèles à couches** : ces modèles sont largement utilisés en recherche d'information. Certains travaux sont basés sur le modèle probabiliste [Kwok, 1989], [Belew, 1989], [Crestani et al, 1994] et [Kwok, 1995], d'autres utilisent le modèle vectoriel. Le réseau est généralement construit à partir des représentations initiales de documents et informations descriptives associées. La recherche d'information est fondée sur un processus d'activation/propagation depuis les neurones descriptifs de la requête. La pertinence des documents est alors mesurée grâce à leur niveau d'activation.

La notion de réseau, généralement, est très intéressante pour représenter les différentes relations et associations qui existent entre les termes et les documents. Ceci est d'autant plus vrai quand ces relations sont évaluées. Ces dernières sont de trois types :

- ✓ Relations entre termes : synonymie, voisinage, etc.
- ✓ Relations entre documents : similitude, référence, etc.
- ✓ Relations entre termes et documents : fréquence, poids, etc.

PIRCS [Kwok et al, 1999] est un exemple de système de recherche d'information basée sur l'approche connexionniste. La pertinence d'un documents est *RSV (q,d)*. Il s'agit d'une combinaison de deux valeurs de pertinence. La première est produite par le document, la deuxième par la requête.

Ces valeurs sont obtenues par un processus de propagation de signaux dans un réseau de neurones. Etant donnée une requête *q* et un document *d*, nous avons alors :

$$RSV\,(q,d) = \alpha RSV_d + (1-\alpha) RSV_q \qquad (9)$$

Où:
RSV_d : représente la pertinence focalisée sur le document,
RSV_q : représente la pertinence focalisée sur la requête.
$\alpha \in [0,1]$.

L'avantage principal de l'utilisation des réseaux de neurones est la propriété d'apprentissage.

I.3.2.3 Le modèle LSI (Latent Semantic Indexing) :

Le modèle LSI a pour objectif fondamental [Dumais, 1994] d'aboutir à une représentation conceptuelle des documents où les effets dus à la variation d'usage des termes dans la collection sont nettement atténués. Ainsi, les documents qui partagent des termes coocurrents ont des représentations proches dans l'espace défini par le modèle.

La base mathématique du modèle LSI est la décomposition par valeur singulière SVD de la matrice Terme-Document. La SVD identifie un ensemble utile de vecteurs colonnes de base de cette matrice. Ces vecteurs de base couvrent le même espace de vecteurs associé à la représentation des documents, car ils sont obtenus par rotation (multiplication par une matrice orthogonale) des vecteurs d'origine.

$$\boldsymbol{X = T_0 S_0 D_0^T} \qquad \textbf{(10)}$$

Où :
X : Matrice Terme-Document,
T_0 : Matrice avec colonnes orthonormées qui couvre l'espace des colonnes de X,
D_0 : Matrice avec colonnes orthonormées qui couvre l'espace des lignes de X,
S_0 : Matrice diagonale formée des valeurs singulières qui résultent de la normalisation de T_0 et D_0.

La propriété de la SVD pour le modèle LSI est qu'en raison du tri des valeurs singulières dans l'ordre décroissant, la meilleure approximation de X peut être calculée comme suit :

$$\boldsymbol{X = \sum_{i=1}^{k} T_{0\times 1i} S_{0\times 1i} D_{0\times i}} \qquad \textbf{(11)}$$

La matrice T_0 a deux principales propriétés [Oard, 1996] :

- Chaque paire de représentations de documents obtenue par combinaison linéaire des lignes de T_0 a la même valeur de degré de similitude que les représentations associées classiques Document-Termes.

- La suppression des composants de faible poids dans un vecteur ligne améliore l'ordre lors du calcul de similitude requête-document.

On passe ainsi d'une représentation de documents à base de termes à une représentation à base de concepts dans un espace de dimension plus réduite.

La méthode LSI a été appliquée dans la collection TREC pour la tâche de croisement des langues [Deerwester & al, 1990]. L'application de la méthode dans un corpus parallèle a permis d'identifier les principaux composants de l'espace vectoriel, associés à chaque langue, produisant ainsi une représentation unifiée de documents écrits dans différentes langues. En utilisant des requêtes en anglais, la méthode sélectionne en début de liste les versions traduites en français dans 92% des cas.

Outre l'accroissement des performances dues à son utilisation [Deerwester & al, 1990] [Dumais, 1994], le modèle LSI présente un intérêt majeur dans l'introduction de la notion de concept en recherche d'information à travers l'utilisation de la théorie relative à la décomposition par valeurs singulières.

I.3.3 Les modèles probabilistes :

I.3.3.1 Le modèle probabiliste de base :

Le modèle de recherche probabiliste utilise un modèle mathématique fondé sur la théorie des probabilités [Robertson & Sparck Jones, 1976]. Le processus de recherche se traduit par calcul de proche en proche, du degré ou probabilité de pertinence d'un document relativement à une requête. Pour ce faire, le processus de décision complète le procédé d'indexation probabiliste en utilisant deux probabilités conditionnelles :
$P(w_{ji}/Pert)$: Probabilité que le terme t_i apparaisse dans le document D_j sachant que ce dernier est pertinent pour la requête.
$P(w_{ji}/Nonpert)$: Probabilité que le terme t_i apparaisse dans le document D_j sachant que ce dernier n'est pas pertinent pour la requête.

Si on suppose l'indépendance des variables documents ″pertinents″ et ″non pertinents″, la fonction de recherche peut être obtenue en utilisant la formule de Bayes.

Soit D_j $(t_1, t_2, \ldots, t_N)$
Où :
$t_i = 1$ si le terme t_i indexe le document D_j,
$t_i = 0$ sinon.

$$\mathbf{P(pert/D_j)} = \frac{\mathbf{P(D_j/pert) * p(pert)}}{\mathbf{p(Dj)}} \quad \text{et} \quad \mathbf{P(Nonpert/D_j)} = \frac{\mathbf{P(D_j/Nonpert) * p(Nonpert)}}{\mathbf{p(Dj)}} \qquad (12)$$

Où:
$P(Pert/D_j)$ est la probabilité de pertinence du document D_j sachant sa description.
$P(D_j) = p(D_j/pert) * p(pert) + p(D_j/Nonpert) * p(Nonpert)$
$P(D_j/pert)$ (respectivement$P(D_j/Nonpert)$) est la probabilité d'observer le document D_j, sachant qu'il est pertinent (respectivement non pertinent).

Si l'on considère l'indépendance des termes :
$P(pert/D_j) = p(t_1/pert) * p(t_1) \ldots p(t_N/pert) * p(t_N)$
$P(Nonpert/D_j) = p(t_1/Nonpert) * p(t_1) \ldots p(t_N/Nonpert) * p(t_N)$

Où :

$$p(t_1/pert) = \frac{r_i}{R} \quad \text{et } p(t_1/Nonpert) = \frac{m_i - r_i}{M - R},$$

R étant le nombre de documents pertinents pour une requête, r_i le nombre de documents pertinents dans lesquels le terme t_i apparaît, M le nombre total de documents dans la collection et m_i le nombre total de documents dans lesquels le terme t_i apparaît.

Pour caractériser l'occurrence des termes d'indexation dans les documents, on utilise une loi de distribution du type loi de Poisson. Cette occurrence est alors déduite de l'étude d'un échantillon de documents. Pour la restitution, les documents sont rangés en fonction de $P(pert/t_i)$. Il résulte du principe d'ordonnancement probabiliste que cet ordonnancement est optimal. C'est-`a-dire que, quelque soit le nombre de documents pertinents restitués, le pourcentage de documents restitués qui sont effectivement pertinents est maximisé [Robertson & Sparck Jones, 1976].

Les modèles probabilistes sont plus efficaces que les modèles booléens (appariement exact). Ils ont une base théorique saine et sont indépendants du domaine d'application [Callan, Croft & Harding, 1992]. Un obstacle majeur avec les modèles de recherche d'information probabilistes est de trouver des méthodes pour estimer les probabilités utilisées pour évaluer la pertinence qui soient théoriquement fondées et efficaces au calcul [Crestani & van Rijsbergen, 1998]. Pour des raisons de simplicité, l'hypothèse de l'indépendance des termes est utilisée en pratique pour implémenter ces modèles.

I.3.3.2 Les réseaux inférentiels bayésiens :

Un réseau inférentiel bayésien [Turtle et al, 1990] [Turtle et al, 1991] [Turtle et al, 1992] [Turtle, 1991a] [Callan, 1996] est un graphe de dépendances, orienté et acyclique. Dans ce graphe, les nœuds représentent des variables propositionnelles (ou également des constantes) et les arcs des liens de dépendances entre les nœuds. Ainsi, si la proposition représentée par le nœud *p* cause ou implique la proposition représentée par le noeud *q*, on trace alors un arc de *p* vers *q*.

Dans le contexte de la recherche d'information, les nœuds et les arcs sont définis comme suit :

- ✓ **Les nœuds** : représentent des concepts, des groupes de termes ou des documents.
- ✓ **Les arcs** : représentent les dépendances entre termes et entre termes et documents.

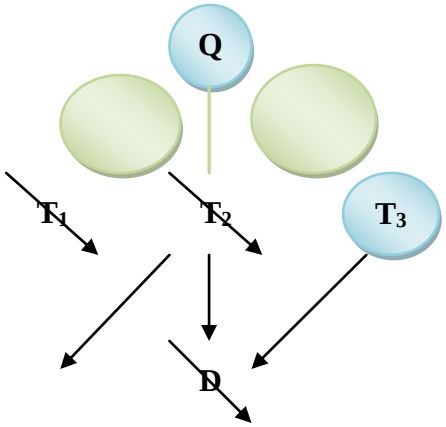


Figure I.5 : Modèle de réseau bayésien simple [N. Nassr, 2002]

Le réseau inférentiel représenté ci-dessus illustre le réseau de Turtle [Turtle et al, 1990] de pertinence d'un document vis à vis d'une requête composée de trois termes. Les nœuds de la requête représentent des variables aléatoires ayant pour valeur 0 ou 1.
L'évènement " la requête est accomplie " ($Q = 1$) est réalisé si le sujet lié à un terme ($T1 = 1$, $T2 = 1$ ou $T3 = 1$), ou si une combinaison de ces évènements sont vrais. Les trois sujets sont inférés par l'évènement " le document est pertinent" ($D = 1$). Par l'enchainement de règles de probabilités, la probabilité jointe des autres nœuds du graphe est :

$$P(D,T1,T2,T3,Q) = P(D)P(T1|D)P(T2|D,T1)\ P(T3|D,T1,T2)P(Q|D,T1,T2,T3) \qquad (13)$$

La direction des arcs indique les relations de dépendance entre les variables aléatoires. L'équation (13) devient :

$$P(D,T1,T2,T3,Q) = P(D)P(T1|D)P(T2|D)P(T3|D)P(Q|T1,T2,T3) \qquad (14)$$

La probabilité de réalisation de la requête $P(Q = 1|D = 1)$ peut être utilisée comme score de rangement des documents :

$$P(Q = 1|D = 1) = \frac{P(Q = 1, D = 1)}{P(D = 1)} = \frac{\sum P(D = 1, T_1 = t_1, T_2 = t_2, T_3 = t_3, Q = 1)}{P(D = 1)}$$

Le modèle nécessite la connaissance de $P(D = [0|1])$, $P(T_i = [0|1]|D = [0|1])$, $P(Q = [0|1]|(T_1,T_2 \ldots T_n) \in \{0,1\}^n)$, cette dernière est la plus difficile à trouver car le nombre de probabilités à spécifier augmente exponentiellement avec le nombre de termes dans la requête.

Inquery est un exemple de système de recherche d'information basé sur l'approche bayésienne [Allan et al 1998]. La fonction de pondération utilisée dans Inquery exprime le degré de croyance à un terme t dans le document d :

$$w_{t,d} = 0.4 + 0.6 \frac{t f_{t,d}}{t f_{t,d} + 0.5 + 1.5 \frac{length(d)}{avglen}} \frac{\log \frac{N + 0.5}{n_t}}{\log (N+1)}$$

Où:
$tf_{t,d}$: fréquence du terme t dans le document d,
$length(d)$: longueur du document d,
$avglen$: longueur moyenne d'un document,
N : nombre de documents dans la collection,
n_t : nombre de documents contenant le terme t.

Le modèle présente l'intérêt de considérer la dépendance entre termes mais engendre une complexité de calcul importante. Cependant, l'utilisation des réseaux inférentiels présente quelques inconvénients. En effet, l'estimation de $P(T_i|D)$ n'est pas spécifiée. Les approches nécessitent l'utilisation des probabilités bayésiennes à la place. La probabilité bayésienne d'un évènement est le degré de croyance humaine à cet évènement. Le calcul des probabilités demande un temps d'ordre exponentiel du nombre de termes de la requête.

I.3.3.3 Le modèle de langage :

Dans les modèles de recherche classique, on cherche à mesurer la similarité entre un document *d* et une requête *q* ou à estimer la probabilité que le document répond à la requête ($p(d_i|q)$). 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 sont 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. La requête est alors inférée par l'utilisateur à partir de ce(s) document(s). Un document n'est pertinent que si la requête utilisateur ressemble à celle inférée par le document. On cherche alors à estimer la probabilité que la requête soit inférée par le document $p(q|d_i)$.

En se basant sur le principe d'indépendance des termes (l'apparition d'un terme n'influe pas la probabilité d'appartenance d'un autre terme dans le document ou dans la requête), $p(q|d_i)$ peut être réécrite comme suit :

$$p(q|d_i) = \prod_{k=1}^{n} p(q_k|d_i) \qquad (15)$$

Où *n* est le nombre de termes dans la requête et q_k est un terme de la requête ($1 \leq k \leq n$).

Cette équation, introduite par Ponte et Croft [Ponte et al, 1998] a été reprise par Berger et Lafferty [Berger et al, 1999] pour formaliser l'effet de dérivation des requêtes à partir des documents. Ces deux approches seront présentées plus en détails dans le chapitre II.

I.4 L'évaluation des systèmes de recherche d'information :

Depuis la naissance du domaine de la RI, l'évaluation des modèles et méthodes proposés a toujours été un centre d'intérêt important. Pour cela des mesures de qualité des systèmes de recherche ainsi que des ensembles de données ont été développés afin de tester ces systèmes sur une base commune. La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses que l'utilisateur espère. Plus les réponses du système correspondent à celles que l'utilisateur espère, meilleur est le système.

Pour réaliser une telle évaluation, une expérimentation qui utilise les éléments suivants doit être établie:

- ✓ Un ensemble de documents (Collection).
- ✓ Un ensemble de requêtes (Topics).
- ✓ La liste des documents pertinents pour chaque requête (Qrels).
- ✓ Des mesures et des critères quantifiables (MAP, p@5, p@10,..etc).

I.4.1 Les notions de base :

Consécutivement au traitement d'une requête par le système, les documents de la collection forment deux partitions selon deux caractéristiques :

- ✓ Les documents restitués et les documents non restitués.
- ✓ Les documents pertinents et les documents non pertinents.

La figure I.6 représente les partitions de la collection lors d'une interrogation :

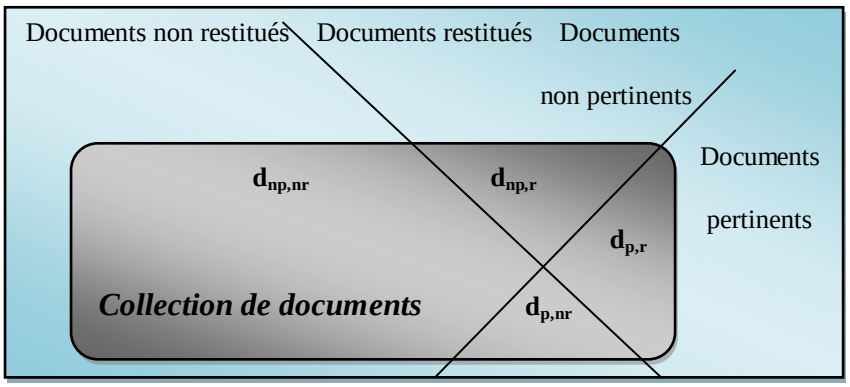


Figure I.6 : Représentation des partitions de la collection lors d'une interrogation [Y. Champclaux, 2009]

Où :
$d_{np,nr}$ représente les documents non pertinents non restitués,
$d_{np,r}$: les documents non pertinents restitués (bruit),
$d_{p,r}$: les documents pertinents restitués,
$d_{p,nr}$: les documents pertinents non restitués (silence).

Pour mesurer les performances qualitatives des SRI, on procède à la comparaison de combinaisons des ensembles de documents pertinents, de documents non pertinents, de documents restitués et de documents non restitués sur l'ensemble des requêtes. Il existe à cet effet de nombreuses mesures, chacune mettant en évidence telle ou telle propriété du système.

De nombreuses mesures existent, nous avons retenu les mesures suivantes :

- La précision et le rappel,
- La précision exacte,
- La mesure F,
- La précision moyenne.

I.4.2 La précision et le rappel :

Le rappel mesure la proportion de documents pertinents restitués parmi tous les documents pertinents disponibles. Si le rappel vaut 1 c'est que les documents pertinents disponibles ont tous été restitués par le système, inversement si le rappel vaut 0 c'est qu'aucun document pertinent n'a été restitué. Cette mesure permet aussi de déterminer le silence, c'est-à-dire la proportion de documents pertinents non trouvés.

La précision mesure la proportion de documents pertinents restitués parmi tous les documents restitués. Elle mesure la capacité du système à trouver exclusivement des documents pertinents. La précision vaut 1 quand tous les documents restitués sont pertinents. Elle vaut 0 si aucun des documents restitués n'est pertinent. Cette mesure détermine également le bruit, c'est-à-dire la proportion de documents non pertinents restitués par le système.

$$\textit{Précision} = \frac{d_{p,r}}{d_{p,r} + dn_{p,r}} \qquad \textit{Rappel} = \frac{d_{p,r}}{d_{p,r} + d_{p,nr}} \qquad (16)$$

La valeur de ces taux est influencée par le processus d'indexation. En effet, plus l'indexation est exhaustive, plus le taux de rappel est potentiellement important : toutes les informations susceptibles d'être pertinentes peuvent être restituées, mais certaines ne seront pas pertinentes pour l'utilisateur. Symétriquement, plus l'indexation est spécialisée, plus le taux de précision est élevé, mais cela induit le risque d'avoir un taux de rappel faible : les informations restituées seront pertinentes, mais d'autres informations pertinentes ne seront pas restituées. Un système qui aurait 100 % à la fois pour la précision et pour le rappel signifie qu'il a trouvé tous les documents pertinents et rien que les documents pertinents. En pratique, cette situation n'arrive pas. Le plus souvent, il est possible d'obtenir un taux de précision et de rappel aux alentours de 30 %, ce qui implique que le problème auquel répond la RI est non trivial.

Les deux métriques ne sont pas indépendantes: quand l'une augmente, l'autre diminue. Il est facile d'avoir 100% de rappel: il suffit de donner tous les documents disponibles comme réponse à chaque requête. Cependant, la précision dans ce cas serait très basse. De même, il est possible d'augmenter la précision en donnant très peu de documents en réponse, mais le rappel en souffrira. Il faut donc utiliser les deux métriques conjointement. Le but d'un SRI est d'améliorer la précision sans trop sacrifier le rappel et vice-versa.

Il est intéressant pour étudier la qualité de l'ordonnancement des scores de similarités obtenus par chaque document avec la requête, de regarder la précision P_n ou le rappel R_n du sous ensemble des documents constitués des *n* premiers du retour. Ces deux mesures se notent respectivement P@n et R@n. Ainsi, il est utile d'examiner la précision à 10 documents restitués si l'on s'intéresse à la capacité du système de restituer des documents pertinents en tête de liste (ce qui est une préoccupation traditionnelle des utilisateurs des moteurs de recherche). La précision à 5, 10, 30, … documents restitués présente néanmoins des limites : par exemple si une requête donnée a seulement 8 documents pertinents, et que le SRI restitue bien ces 8 documents en tête de liste, le SRI aura une précision à 10 documents restitués égale à 0,8, ce qui n'illustre pas que tous les documents pertinents disponibles ont été trouvés. De plus, dans cet exemple, une précision à 10 documents restitué égale à 0,8 ne permet pas de déterminer où se situent les deux documents non pertinents parmi les dix restitués. Pour pallier ce défaut, soit on regarde la précision à différents *n*, soit on utilise la R-précision.

I.4.3 La précision exacte ou R-précision :

La R-précision est la précision à *n* quand *n* est égal au nombre total de documents pertinents. Cette mesure est plus réaliste pour l'étude de l'ordonnancement en tête de liste, mais pour l'obtenir, il est nécessaire de connaître au préalable le nombre de documents pertinents disponibles dans le corpus pour une requête donnée. Une R-précision de 1.0 signifie une précision et un rappel optimaux.

I.4.4 La mesure F :

Examiner la succession des précisions à *n* ou des rappels à *n* permet d'évaluer un ordonnancement dans son ensemble et ainsi de déterminer telle ou telle propriété de l'algorithme étudié. Il peut être intéressant d'avoir une valeur synthétisant ces deux mesures. On voudrait pouvoir maximiser la précision et le rappel, mais comme on l'a vu ces deux mesures évoluent souvent de façon opposée. La précision est globalement décroissante au fur et à mesure que le SRI restitue des documents, alors que le rappel est globalement croissant. On peut choisir la mesure F comme valeur synthétique exploitant la précision et le rappel. Elle est calculée comme suit :

$$F = (2 \times Rappel \times Précision) / (Rappel + Précision) \qquad (17)$$

Pour évaluer l'ordonnancement, il est possible de calculer la mesure F_n à chaque rang *n :*

$$F@n=2 \times R@n \times P@n / (R@n+P@n) \qquad (18)$$

I.4.5 La courbe précision-rappel :

Une autre manière de représenter le couple précision-rappel en fonction de *n* (nombre de documents restitués) est de représenter pour chaque *n*, le nuage des points des couples précision-rappel. Ainsi, en reliant les points par un segment, la courbe précision-rappel a l'allure suivante :

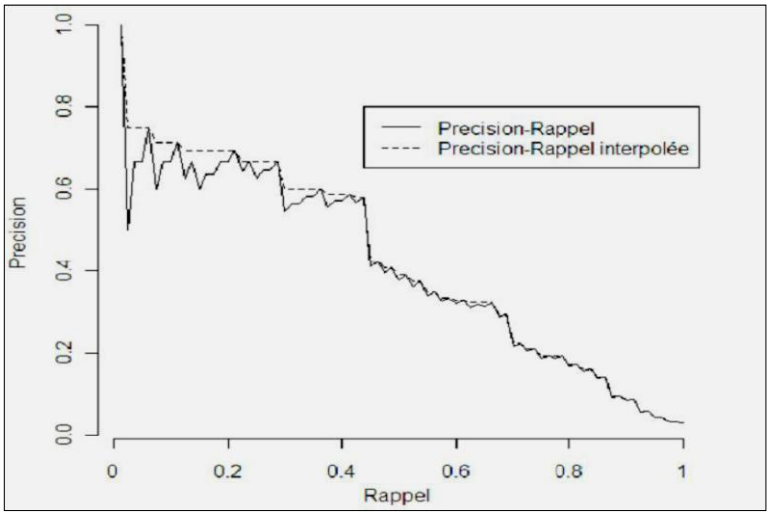


Figure I.7 : Courbe précision-rappel [Y. Champclaux, 2009]

Cette courbe globalement décroissante est dentelée, la précision décroit au fur et à mesure que des documents non pertinents sont restitués, le rappel ne varie pas, puis un document pertinent est restitué, alors la précision et le rappel augmentent, puis continuent de croître tant que les documents restitués sont pertinents et décroissent sinon. Plus cette courbe décroît tardivement, meilleur est l'algorithme à l'origine de l'ordonnancement étudié.

Les corpus d'études sont la plupart du temps fournis avec un ensemble de requêtes, et il peut être intéressant pour comparer deux SRI de disposer des mesures moyennes reflétant le comportement général d'une méthode pour l'ensemble des requêtes d'un corpus donné. A partir de courbes précision-rappel de requêtes différentes, il n'est pas possible de calculer une courbe moyenne, la série des valeurs de rappel étant à priori distinctes d'une courbe à l'autre.

En utilisant la technique de [Yang, 1999], il est possible de transformer la courbe précision-rappel en une courbe interpolée continue entre 0 et 1, à partir de laquelle il est possible d'extraire onze valeurs de précision pour onze valeurs de rappel fixées : le rappel varie de 0 à 1 par pas de 0,1. Le rappel varie à chaque fois qu'un document pertinent est restitué et ne varie plus tant que les documents restitués sont non pertinents. A un rappel donné, il est donc possible d'avoir plusieurs précisions. Pour réaliser la courbe en escalier, on associe la précision maximale à un rappel donné parmi les couples rappel-précision pour ce rappel.

Une fois cette opération effectuée sur chaque courbe précision-rappel de chaque requête du corpus étudié, une courbe précision-rappel est obtenue en faisant la moyenne des précisions pour chacune des onze valeurs de rappel fixées.

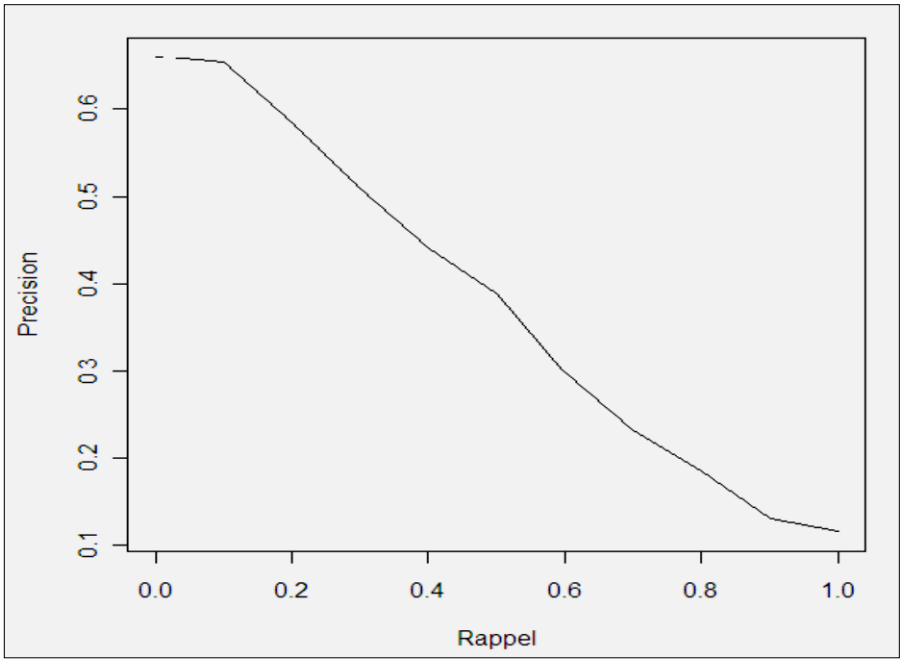


Figure I.8 : Moyenne des courbes de précisions à 11 points de rappel [Y. Champclaux, 2009]

L'aspect en dents-de-scie a disparu, l'analyse du comportement du SRI est facilitée. La moyenne des précisions à 11 points de rappel mesure la qualité d'un algorithme d'ordonnancement sur une collection de test.

I.4.6 La précision moyenne :

La précision moyenne est une mesure de performance globale. La précision moyenne est une moyenne de précision sur un ensemble de points de rappel.

$$MAP = \frac{1}{n} \sum_{i=1}^{N} p(i) * R(i) \qquad (19)$$

Avec : - *R(i)* = 1 si le i^{ème} document restitué est pertinent,
- *R(i)* = 0 si le i^{ème} document restitué est non pertinent
- *p(i)* la précision à *i* documents restitués.
- *n* le nombre de documents pertinents restitués.
- N le nombre total de documents.

La MAP est la moyenne des précisions à *i* pour chaque *i* tel que le i^{ème} document est pertinent. Comme la courbe précision a onze points de rappel, la précision moyenne décrit la performance globale d'un système. Elle synthétise en une valeur la qualité d'un système. Elle présente l'avantage par rapport à la courbe précision à 11 points de rappel d'être calculée sans interpolation.

I.4.7 Les autres critères d'évaluation :

L'étude de [Baccini et al., 2010] a montré que sur l'ensemble des mesures de RI existantes, un nombre restreint (5 ou 6) de ces mesures permet d'avoir une vue claire des propriétés d'un système donné.

Il ne faut pas perdre de vue que d'autres critères peuvent être mis en avant pour évaluer un système de recherche par exemple le temps de réponse, l'effort fourni par l'utilisateur, la présentation du résultat, éventuellement la taille de l'index, … De nombreuses recherches portent sur la notion d'utilité et font référence au point de vue utilisateur. Il y a d'autres critères d'évaluation comme le ratio de couverture qui fait référence aux documents restitués connus et reconnus pertinents par l'utilisateur, le ratio de nouveauté qui fait référence aux documents restitués reconnus pertinents par l'utilisateur et qui lui étaient inconnus jusque là, l'effort de rappel qui correspond au rapport entre le nombre de documents que l'utilisateur souhaitait retrouver et le nombre de documents lus pour les trouver, ...etc.

I.5 Les compagnes et collections de tests :

Depuis l'apparition de l'expression « Recherche d'Information » dans le mémoire de fin d'étude de Calvin Mooers en 1950, le monde de la RI n'a cessé de développer des outils et méthodes de recherche et, en parallèle des campagnes d'évaluation pour tester les méthodes et outils développés.

Les objectifs des campagnes d'évaluation sont les suivants : encourager la RI sur de grandes collections fermées, développer la communication entre l'industrie, l'université et l'état en mettant en place un forum ouvert pour faciliter les échanges d'idées sur la recherche, augmenter la vitesse de transfert de la technologie du laboratoire de recherche aux enseignes commerciales, rendre disponible et accessible des techniques d'évaluation appropriées pour les industriels et les académiciens.

Les principes de bases de l'évaluation en RI ont été mis en place depuis les années soixante pour juger de l'efficacité des systèmes et ainsi faire évoluer la performance des mêmes systèmes, technologiquement mais également par rapport aux attentes des utilisateurs.

Dans ce qui suit, on évoquera les trois corpus : Cranfield, CISI, et le corpus TREC ad hoc 1998.

I.5.1 Le projet Cranfield :

Cyril Cleverdon [Cleverdon, 1962], libraire du « Cranfield College Of Aeronautics », est à l'origine de deux contributions majeures : Cranfield 1 et Cranfield 2.

Le projet Cranfield 1 (1958-1962), vise à tester l'efficacité de différentes façons d'indexer et de rechercher des documents. Le but est de comparer quatre méthodes d'indexation, sur une base commune constituée d'un ensemble d'articles scientifiques et de rapports (1800 documents au total). Les expérimentateurs ont formé un ensemble de 1200 requêtes en demandant aux auteurs de documents faisant référence dans le domaine (les documents sources) de formuler le besoin d'information qui a motivé l'écriture de l'article. Les systèmes étaient alors évalués par rapport à leur capacité à retrouver les documents sources. Retrouver les documents source correspond à maximiser le rappel. La façon de créer les requêtes ainsi que la méthode des documents sources ont été critiqués, néanmoins cette expérimentation a posé les bases de la construction de collections de test et de la création de critères d'évaluation.

Une deuxième expérimentation a été proposée : Cranfield 2 a permis quelques avancées importantes. Le corpus de test est plus petit que son prédécesseur : 1400 documents et 225 requêtes. Les documents pertinents sont simplement ceux que les experts ont jugés comme tel. C'est dans cette expérimentation qu'est apparu pour la première fois l'usage des courbes de précision-rappel lors de l'évaluation.

I.5.2 Le corpus CISI :

Le corpus CISI est comme le corpus Cranfield, un corpus de petite taille : 1460 documents et 120 requêtes.

I.5.3 TREC (Text REtrieval Conference) :

La campagne TREC est une série d'évaluations annuelles des méthodes et outils pour la RI. TREC est un projet international initié au tout début des années quatre-vingt dix par le NIST (National Institute of Standard and Technology) dans le but de proposer des moyens homogènes d'évaluation de systèmes documentaires sur des collections de documents conséquentes. Il est aujourd'hui co-sponsorisé par le NIST, l'ITL (Information Technology Laboratory Retrieval group) et l'IARPA (Intelligence Advanced Research Projects Activity) (ex-DARPA (Defense Advanced Research Projects Agency)).

Les conditions de participation sont les suivantes : le NIST diffuse courant décembre, un appel à participation qui explique dans les grandes lignes les objectifs et le déroulement du projet pour l'année à venir. Les demandes de participation doivent être déposées en janvier, aussi bien pour les anciens participants que pour les nouveaux.

Les demandes d'intégration à TREC sont étudiées par un comité de programme qui se prononce en février. La participation à la conférence annuelle elle-même est soumise à l'envoi au NIST de résultats.

Les pistes explorées évoluent au fil des années, elles reflètent les intérêts du moment : en 1995, les deux tâches principales étaient celle de routage et celle de recherche ad hoc ; les tâches secondaires étaient la recherche multilingue, le filtrage, la fusion de bases de données, la tâche interactive et la tâche de « confusion » (recherche d'erreur). En 2008, les tâches proposées ont été : la tâche Blog, la tâche Légale, la tâche Recherche d'Information Chimique, la tâche Entreprise, la tâche de Réinjection de pertinence, la tâche Entité, la tâche Web et la tâche « million de requêtes ». Certaines tâches ne sont plus au goût du jour, il en est ainsi pour la tâche Nouveauté, la tâche Question/Réponse, la tâche Spam, …etc.

I.5.4 Autres campagnes d'évaluation :

D'autres campagnes d'évaluation ont vu le jour :

- Les campagnes NTCIR (NII-NACSIS Test Collection for IR Systems) : Apparues en 1999, les ateliers d'évaluation du NTCIR (Research Center for Information Resources) sont conçus dans le but d'améliorer tous les domaines de l'accès à l'information y compris la recherche d'information, la production de résumés, l'extraction terminologique, etc. La collection test utilisée comprend des textes publiés en 1998 et 1999, en chinois traditionnel, en coréen, en japonais et en anglais.

- Les campagnes CLEF (Cross-Language Evaluation Forum) : Projet européen d'évaluation des SRI qu'ils soient monolingues ou multilingues de langue européenne. Ce projet a vu le jour en l'an 2000. CLEF propose des tâches principales (les tâches monolingues, bilingues, multilingues et de recherche dans un domaine spécifique) et des tâches additionnelles dont le but est d'identifier les nouveaux besoins et les nouvelles exigences afin d'acquérir de nouvelles méthodes pour l'évaluation des SRI monolingues ou multilingues.

- Les campagnes FIRE (Forum for Information Retrieval Evaluation) : Apparues en 2008, ces campagnes ont pour but de fournir des collections de grande échelle pour l'évaluation de la recherche d'information en langues indiennes (Bengali, Hindi, Marathi, Tamil).

Les campagnes d'évaluation sont un élément incontournable de la RI, elles fournissent des outils d'évaluation des systèmes, elles permettent la comparaison de systèmes et elles définissent des cadres d'études utiles aux chercheurs.

I.6 Conclusion :

Ce premier chapitre a porté essentiellement sur une présentation des concepts de base inhérents au domaine de la Recherche d'Information ainsi que sur ses modèles les plus connus. Aussi, on a évoqué les mesures ainsi que les collections (compagnes) d'évaluation des Systèmes de Recherche d'Information.

Ce domaine, qui est en pleine évolution, vise à répondre à un certain nombre d'objectifs qui peuvent se résumer dans la phrase suivante : Retrouver automatiquement et rapidement les informations pertinentes à un besoin particulier d'un utilisateur, à partir d'une grande masse d'informations. Dans ce cadre, plus les réponses du système correspondent à celles de l'utilisateur, mieux est le système.

Le modèle de langage convient bien à la recherche d'information. En s'appuyant sur des bases mathématiques, il permet d'analyser et de modéliser d'énormes masses de documents. Il permet également de combiner différentes représentations d'un document comme l'intégration des connaissances à priori de la pertinence d'un document.

Le chapitre qui va suivre portera sur la présentation du modèle de langage et la notion de calcul de pertinence de documents dans le domaine de la Recherche d'Information sur le Web.

Chapitre II

Modélisation de langue et pertinence à priori de documents en RI sur le Web

II.1 Introduction :

La plupart des moteurs de Recherche d'Information (RI) sur le web privilégient la minimisation du temps de réponse par rapport à la qualité des documents retournés à l'utilisateur. En effet, ces derniers délivrent des résultats massifs en réponse aux requêtes des utilisateurs, qui génèrent ainsi, une surcharge informationnelle dans laquelle il est difficile de distinguer l'information pertinente de l'information secondaire ou même du bruit.

L'une des raisons qui a engendré ceci est la non prise en compte de toutes les caractéristiques d'un document web dans les processus d'indexation et de recherche. Par exemple : la taille du document, son type d'URL ou bien le score de dissemblance du document vis-à-vis d'une collection de documents c'est-à-dire qu'un document est à priori pertinent s'il est dissemblable aux autres documents de la collection.

Les modèles de langage offrent un cadre probabilistique pour la description du processus de la RI. Les résultats obtenus ont montré des performances équivalentes voire supérieures à celles des modèles classiques (vectoriel, probabiliste).

Le modèle de langage convient bien à la recherche d'information sur le web et s'appuie sur des bases mathématiques qui permettent d'analyser et de modéliser d'énormes masses de documents (le web). De plus, il permet de combiner différentes représentations d'un document comme l'intégration des connaissances à priori de la pertinence d'un document web.

II.2 La Recherche d'Information et le Web :

Devant la révolution du WEB, plusieurs travaux ont été menés afin de faciliter l'accès aux informations disponibles dans ce gigantesque espace d'information. Les moteurs de recherche représentent une aide inestimable pour la recherche et l'accès aux documents du WEB. Ils sont très souvent placés en tête de liste des sites les plus visités, ils sont utilisés par 85 % des utilisateurs [Schwartz, 1998] comme premier outil de recherche d'informations.
On peut dire que la plupart des moteurs de recherche actuels ne tiennent pas compte des spécificités du WEB et sont basés sur des modèles de RI qui ont été développés pour des documents textuels classiques [Rijsbergen, 1979] [Salton, Yang & Yu, 1975].

Actuellement, la taille du Web est estimée à plusieurs milliards de pages. Cette collection possède des caractéristiques (présentées dans le tableau II.1) particulières : les données sont hétérogènes et distribuées sur plusieurs sites, de nouveaux médias autres que le texte sont accessibles (image, vidéo et son) et les documents sont liés par des liens hypertextes. Ces nouvelles caractéristiques exigent de nouveaux critères de pertinence d'un document par rapport à une requête. L'information disponible dans la collection ne se résume plus au contenu textuel et les traitements d'extraction des index doivent tenir compte de ces aspects multimédias et hypertextuels.

La pertinence, basée sur la correspondance entre les mots clés de la requête et le texte du document, semble insuffisante car elle n'utilise pas l'information portée par les liens. Il faut noter aussi qu'il est difficile pour l'utilisateur de traduire un besoin d'information par une requête : il doit s'adapter à un langage artificiel d'interrogation, même s'il est très simple. Il doit également se faire une image mentale (fausse parfois) sur comment le système produit ses résultats avec une série de mots composés par quelques opérateurs.

De ce fait, la plupart des moteurs de recherche considèrent que le Web est un ensemble de documents atomiques et indépendants, dont la granularité est celle d'une page HTML.
Ils ne tiennent pas compte du fait que le Web est avant tout un ensemble de documents liés par des liens hypertextes et que les pages HTML indexées indépendamment les unes des autres perdent leur contexte. Il existe des travaux qui ont intégré les liens : PageRank, Hits et Salsa.

Sur le Web, on ne peut plus créer une collection statique. La collection (qui est le Web au complet) est une collection gigantesque qu'il est impossible de couvrir au complet. Bien que nous puissions employer des robots logiciels pour explorer automatiquement le Web à la découverte de nouveaux documents, nous ne pouvons pas affirmer que nous avons une bonne couverture de tout le Web. Un système de RI ou un moteur de recherche retrouve en général toujours beaucoup de documents. Parmi ces documents retrouvés, certains sont pertinents, mais noyés parmi beaucoup d'autres documents non pertinents. Plus notre collection contient des documents, plus ce problème devient aigu. Il est de plus en plus demandé que la recherche soit plus précise, même si on doit accepter que certains documents pertinents ne soient pas retrouvés. Cette problématique a motivé les recherches sur la question/réponse : plutôt que de proposer un ensemble de documents dans lesquels l'utilisateur peut trouver une réponse, on tente d'identifier directement la réponse [Voorhees & Tice, 2000].

L'utilisation des langues différentes pose un autre problème. Or, la pertinence d'un document est souvent indépendante de la langue utilisée. Ainsi, nous avons besoin d'outils pour la recherche d'information translinguistique ou multilingue. Les grands problèmes à traiter sont notamment la traduction de la requête et le regroupement des réponses en différentes langues dans une seule liste (si on utilise toujours une liste ordonnée pour présenter les résultats). Pour ces raisons, différentes méthodes ont été développées afin d'utiliser les liens dans le processus de RI sur le Web.

Le tableau II.1 qui suit présente les différentes caractéristiques du contenu Web.

Type de l'autorité	Type du site	Type d'information contenue sur la page	Type de page
Institution	Site vitrine	Auto-descriptive	Page d'accueil
Entreprise	Site de recherche		Portails (liens externes)
Association	Site de ressources	Non auto-descriptive	Index (liens internes)
Personne individuelle	Services web		Page de contenu (texte > lien)

Tableau II.1 : Caractéristiques du contenu Web [S. Karbasi, 2007]

Le schèma suivant reprend les étapes de la RI sur le Web :

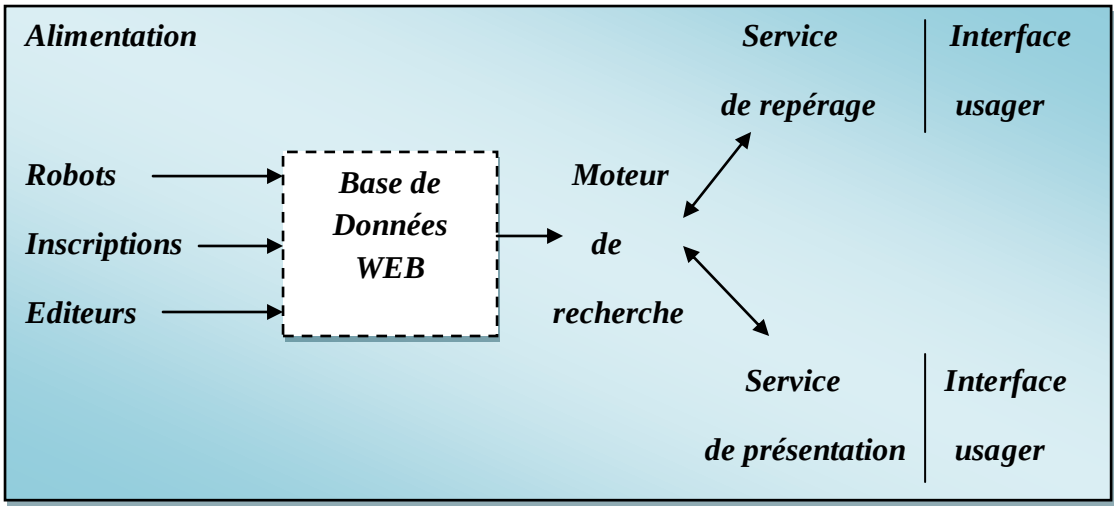


Figure II.1 : Recension et repérage d'information sur le Web [Carmel, 2000]

II.2.1 Propriétés des informations du Web :

L'utilisateur a donc accès, au travers du Web, à un nombre important de documents contenant des informations aussi diverses que abondantes. Cependant, le Web a des limites qui lui sont inhérentes. Ces limites sont présentées dans [Baeza-Yates & Ribeiro-Neto, 1999] :

- **La non persistance de l'information :**

Le Web possède une dynamique très importante et l'information naît, évolue et disparaît rapidement. Il a été estimé que 40% des informations disponibles sur le Web changent tous les mois [Kahle, 1997],

- **L'instabilité de l'information :**

Le Web repose sur une architecture informatique qui peut connaître diverses pannes ou dysfonctionnements. De ce fait, l'information n'est pas accessible de façon permanente et il se peut qu'à tout moment celle-ci ne soit plus accessible,

- **Le manque de qualité de l'information :**

Le Web est un média ouvert, dans le sens où il n'y a pas d'organisme régulateur des contenus disponibles. De ce fait, les informations disponibles sont souvent sujettes à des problèmes de véracité, de fautes de langages ou erreurs typographiques,

- **La redondance d'information :**

Une expérimentation [Shivakumar & Garcia-Molina, 1998] réalisée à partir d'une collection de 24 millions de pages Web montre que plus de 30% de l'information est redondante (page Web miroir par exemple). Cette proportion peut être encore plus importante si l'on considère une redondance sémantique ou partielle des informations,

- **L'hétérogénéité de l'information :**

Sur le Web cohabitent des informations dans des médias différents (image, son, ...etc.), des formats différents (mp3, jpeg, ...etc.) et même des langues différentes (anglais, français, ...etc.),

- **Le volume d'information disponible :**

L'utilisateur a du mal à se repérer, à identifier les documents intéressants au sein de cette masse informationnelle qui évolue sans cesse. De plus, ce volume d'informations très important implique que la couverture du Web par les outils de recherche est assez faible.

La plupart de ces problèmes sont difficilement améliorables de façon automatique (stabilité, hétérogénéité de l'information). Certains d'entre eux sont, en effet, intrinsèques à la nature humaine (contenu inexact ou mal formé des documents par exemple).

II.2.2 Types de Recherche d'Information :

Il existe deux types de Recherche d'Informations : la recherche par interrogation et la recherche par navigation. Le tableau II.2 ci-après visualise la différence entre ces deux types de recherche.

Critères	Recherche par interrogation	Recherche par navigation
Nature de l'information	Homogénéité : sources, informations, domaines, description, présentation, recherche...etc.	Hétérogénéité : informations, images, vidéos…etc.
Structuration de l'information	Information non structurée.	Information semi structurée.
Validité de l'information	Informations doublement validées : sources connues, responsabilité du professionnel de la documentation.	Informations non validées : sources non connues, qualification sous la responsabilité de l'utilisateur.
Cohérence de l'indexation	Indexation par un langage documentaire homogène, adapté à la base et au domaine.	Pas de langage d'indexation précis.
Représentation des documents	-Normalisée : notices textuelles, de description bibliographique. -Format unique de représentation des Informations.	Non normalisée : multiplicité des formats, liberté des auteurs...etc. - page(s) HTML, - «chapeau» + document en format PDF (ou DOC).
Volume d'informations	-Limité (quelques centaines de milliers de documents), maitrise par le responsable du système. -Croissance linéaire de la base.	-Quantité énorme d'informations; volume redondant, non maitrisé ; pas de filtrage, -Croissance exponentielle du web.
Enrichissement de l'information	Ajout de documents, mais stabilité du contenu et des notices des documents.	Information très changeante, modifications du contenu et de l'organisation des sites.
Responsabilité de l'alimentation du système	Base documentaire alimentée par les professionnels de l'information : souci de cohérence de l'information.	Sites web alimentés par les webmasters: souci de présentation, de diffusion et d'impact de l'information.

Outils de recherche	Outil d'interrogation unique (logiciel documentaire), adapté a la base et au langage documentaire.	-Des outils de recherche adaptés selon les types de recherche. -Mélange de tous les modes de recherche.
Qualité de la recherche	Pour une recherche par mots-clés : - Dépend directement de la qualité de l'indexation manuelle. - Retrouve exactement les notices contenant les mots-clés demandés.	Dépend : - de la couverture du domaine par des sites accessibles en recherche, - de l'expertise de l'utilisateur, - des performances de l'outil. Recherche retourne la réponse qui ressemble le plus à la question posée : recherche par approximations successives.
Multilinguisme	Pas à la charge de l'utilisateur : pas de prise en compte de la langue des documents dans la recherche.	A la charge de l'utilisateur: choix de la langue des documents dans le moteur, tri...etc.

Tableau II.2 : Recherche par interrogation et Recherche par navigation

II.2.3 Les outils de Recherche d'Information sur le web :

- **Les moteurs de recherche :**

Un moteur de recherche est une application permettant de retrouver des ressources (pages web, forums Usenet, images, vidéo, fichiers,…etc.) associées à des mots quelconques. Certains sites web offrent un moteur de recherche comme principale fonctionnalité ; on appelle alors « Moteur de recherche » le site lui-même (par exemple, Google Video est un moteur de recherche vidéo).

Un moteur de recherche est un outil de recherche sur le web constitué de « robots », appelés *bots*, *spiders*, *crawlers* ou agents qui parcourent les sites à intervalles réguliers et de façon automatique (sans intervention humaine, ce qui les distingue des annuaires) pour découvrir de nouvelles adresses (URL). Ils suivent les liens hypertextes (qui relient les pages les unes aux autres) rencontrés sur chaque page atteinte. Chaque page identifiée est alors indexée dans une base de données, accessible ensuite par les internautes à partir de mots-clés.

Les moteurs de recherche ne s'appliquent pas qu'à Internet : certains moteurs sont des logiciels installés sur un ordinateur personnel. Ce sont des moteurs dits *desktop* qui combinent la recherche parmi les fichiers stockés sur le PC et la recherche parmi les sites Web. **Exemples :** Exalead Desktop, Google Desktop, Copernic Desktop Search,…etc.

- **Les méta-moteurs :**

Les méta-moteurs présentent des stratégies de recherche beaucoup plus hétérogènes que ce que l'on peut trouver dans les moteurs de recherche classiques. Cependant, tous ont pour point commun d'utiliser les résultats produits par ces mêmes moteurs de recherche.

La plupart de ces outils ne font que trier les liens récupérés de plusieurs sources en utilisant leurs propres algorithmes de détection de pertinence. Dans ce cas, leur but est de tirer parti de la complémentarité et de la spécialisation de plusieurs outils de recherche. Dans ce cas ces outils se distinguent par les fonctions de tri utilisées qui sont propres à chaque méta-moteur.

Cependant, il existe d'autres méta-moteurs que l'on peut qualifier de plus évolués. Ils ne se contentent pas de compiler les résultats d'autres moteurs de recherche mais parcourent également Internet à la recherche de documents pertinents. **Exemple :** Ixquick, Scroogle et Seeks.

- **Les outils de recherche de type répertoire :**

Ce sont des outils qui réfèrent à une base de données et dont les fonctions de repérage reposent sur une classification hiérarchique par sujet, le plus souvent qui permet la recherche par navigation de catégorie en sous catégorie. Dans ces outils, la recherche par mot-clé est également possible, la recherche s'effectuant alors dans les catégories et les titres des pages elles-mêmes. **Exemples :** Yahoo, la toile du Québec.

Les outils de type répertoire sont intéressants dans la mesure où ils permettent la recherche par navigation. Cependant, le fait qu'un même site soit classé sous de multiples catégories peut engendrer une confusion.

II.2.4 Problèmes de recherche sur le Web :

Les difficultés qui peuvent être rencontrées lors du processus de recherche par interrogation nécessitent un approfondissement. Les problèmes liés à la tâche de recherche sont les suivants:

- **La couverture du Web :**

Les outils de recherche n'indexent qu'une partie limitée du Web [Lawrence & Giles, 1999] [Notess, 2002]. Une première raison à cette restriction provient du nombre très important de documents disponibles. Une seconde raison est l'incapacité qu'ont les robots d'indexation à indexer les documents du Web invisible. En effet, ces documents ne sont accessibles généralement que par le biais de formulaires que le robot ne peut pas automatiquement remplir [Seltzer, 1997]. Les robots ne se concentrent donc que sur le Web visible (portion limitée du Web global).

- **Le chevauchement des bases de documents des différents outils de recherche :**

Le chevauchement entre les bases d'indexation des différents outils de recherche est relativement faible [Notess, 2000], ce qui signifie que chacun des outils a préalablement indexé des documents différents et que pour une même requête il va retourner des documents différents des autres outils. Par exemple, l'étude de Notess [Notess, 2000] indique, suite à une expérimentation sur quatorze moteurs de recherche, que plus de 35% des résultats n'a été retrouvé que par un seul moteur.

- **La mise à jour des bases d'indexation des outils de recherche :**

Un facteur important dont pâtissent les outils de recherche est l'évolution rapide des documents. Par ce fait, les outils de recherche n'ont pas une base d'indexation à jour et ils proposent à l'utilisateur un grand nombre d'URLs de documents déplacés, supprimés voire obsolètes.

- **La présentation des résultats à l'utilisateur :**

Même s'ils ne couvrent qu'une partie du Web, les outils de recherche indexent plusieurs millions de documents. De ce fait, suite à une recherche, l'utilisateur se retrouve souvent avec des milliers de documents pertinents (du point de vue système) pour ses besoins. Or, les outils actuels utilisent communément des listes de résultats pour présenter ces derniers. Ce mécanisme ne s'avère pas adapté à la compréhension du résultat dans sa globalité car une liste de résultats ne présente que quelques dizaines de résultats par page. Outre ces difficultés de recherche liées à la technologie, il existe les difficultés d'ordre général liées à la gestion et à l'organisation des résultats retrouvés qui ont une incidence sur la RI.

II.3 Algorithmes de recherche pour le web :

Les méthodes qui ont été proposées pour extraire de l'information à partir des liens de navigation, traitent le web comme un graphe orienté dont les nœuds sont les pages HTML, et les arcs sont les liens de navigation entre les pages. Elles ne font pas de distinction de types entre les pages ou entre les liens. Pourtant il y a une différence entre une page contenant un livre et une page contenant quelques paragraphes, entre une page de contenu textuel et une page d'index ne contenant que des liens, entre une page personnelle et une page professionnelle,…etc. De même pour les liens, il y a une différence entre un lien intra-page et un lien inter-page, entre un lien de publicité et un lien vers un texte,…etc.

II.3.1 L'algorithme HITS (Hyperlinked Induced Topic Search) :

Kleinberg fut l'un des premiers à s'intéresser aux propriétés de connectivité du graphe représentatif du web et de son apport dans la détection de la pertinence d'une page à une requête [Kleinberg, 1999]. Quelques constatations simples sont à l'origine de ses travaux dans ce domaine.

Dans cette approche, deux types de pages sont identifiés en fonction de la nature de leurs connexions avec les autres documents. On retrouve d'une part les pages qui semblent être très importantes qui jouent le rôle d'autorité sur un sujet donné et d'autre part les documents possédant un grand nombre de liens vers des pages faisant autorité sur un sujet. On distingue ainsi les pages *autorités* ayant un grand nombre de liens entrants et les pages *hubs* ayant un grand nombre de liens sortants et regroupant les autorités d'un même sujet. Le but de l'algorithme HITS est de déterminer les hubs et les autorités qui renforcent leurs relations mutuellement sur un sujet donné. Ainsi, Kleinberg dénombre les bons hubs comme des pages pointant vers beaucoup de bonnes autorités et les bonnes autorités comme des pages pointées par beaucoup de bons hubs.

Cette dénotation de bons hubs et de bonnes autorités fait apparaître une troisième catégorie de pages ayant un grand nombre de liens entrants provenant de documents n'ayant aucune particularité. Ces pages, que nous nommons pages indépendantes, sont considérées comme universellement populaires et n'apportent que peu ou pas d'intérêt [Chakrabarti et al., 1999]. Par exemple, la page de Google est extrêmement référencée mais n'apporte que peu d'intérêt sur la plupart des sujets rencontrés où une publicité aura de nombreux liens vers elle. Cette situation est illustrée dans la figure II.2.

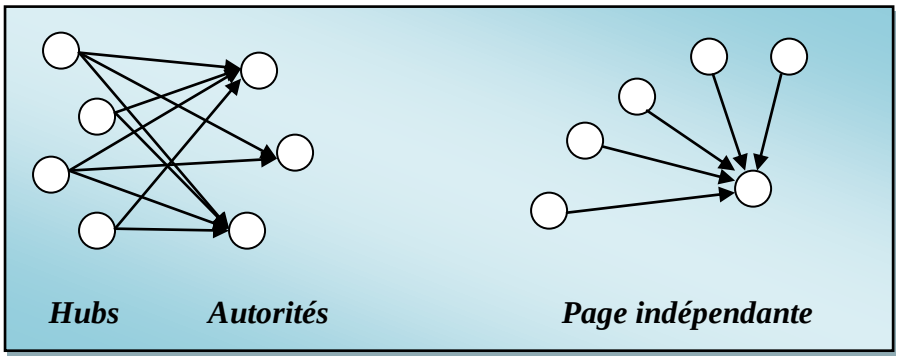


Figure II.2 : Répartition des pages en bon Hubs, bonnes autorités et pages indépendantes

Une justification intuitive de l'autorité conférée à une page en fonction de la structure des liens l'entourant peut être donnée en considérant qu'un fort taux de jugement humain entoure l'ajout d'un lien hypertexte dans un document. En quelque sorte, l'auteur du document estime que la page vers laquelle il construit un lien évoque un sujet similaire à son souhait et paraît intéressante.

Pour déterminer les hubs et les autorités d'un sujet σ donné, l'algorithme HITS se base sur un sous-graphe du web S_σ qui doit répondre aux conditions suivantes :

- ✓ S_σ est relativement petit,
- ✓ S_σ est riche en pages pertinentes,
- ✓ S_σ contient la totalité (ou la plupart) des plus importantes autorités.

En gardant S_σ petit, l'application d'algorithmes non triviaux peut s'effectuer sans s'occuper du temps de calcul nécessaire à la réalisation de la tâche. Les deux derniers points nous permettent de nous assurer d'avoir de bonnes chances de déterminer les bonnes autorités correspondant à la requête σ.

La mesure « *Authority* » est calculée comme suit :

$$a(p) = \sum_{q \epsilon S\sigma,\ p \to q} h(q)$$

La mesure « *Hub* » est calculée comme suit :

$$h(p) = \sum_{q \epsilon S\sigma,\ q \to p} a(q)$$

Où : p ϵ S_σ et $q \to p$ est la condition « q contient un lien pointant vers p ».

II.3.2 L'algorithme PageRank :

L'idée principale de cette méthode est de simuler le comportement d'un internaute naviguant de manière aléatoire sur le web. La probabilité qu'il visite une page donnée est d'autant plus grande que cette page soit pointée par beaucoup d'autres pages au travers de leurs liens hypertextes. En considérant qu'une page confère une certaine autorité à une autre page en établissant un lien vers elle, la probabilité de passage de cet internaute aléatoire sur une page indique le degré de pertinence de ce document.

Il existe deux manières d'accéder à une page Web. On peut d'une part l'atteindre directement en connaissant son adresse et d'autre part suivre un lien hypertexte d'un autre document. Le calcul du PageRank - et donc de la pertinence - d'une page intègre ces deux éléments au travers d'une probabilité d. d représente en quelque sorte la probabilité que l'internaute aléatoire s'ennuie sur une page et décide de choisir une autre page au hasard.

L'équation suivante permet de calculer le PageRank pour tous les nœuds du graphe représentant le web.

$$PR(p_j)_t = (1 - d) + d \sum_{\substack{i=1 \\ pi \to pj}}^{n} \frac{PR(p_i)_{t-1}}{C(p_i)} \qquad (20)$$

Où :
$PR(p_j)_t$ représente la valeur du PageRank à l'itération t pour la page p_j,
$C(p_i)$ est défini comme le nombre de liens sortants de la page p_i.

Le paramètre *d* prend ses valeurs dans l'intervalle [0−1] et est généralement placé à $d = 0.85$ d'après des études statistiques menées par Lary Page dans [Page et al., 1998].
Ce dernier paramètre permet de faire converger l'algorithme de manière plus ou moins rapide. En effet, plus *d* est élevé, plus l'effet de l'ajout d'un lien entrant vers une page est accru et plus celui-ci se propagera dans toutes les pages d'un même site.

Le PageRank forme ainsi une distribution de probabilités des pages Web. Le calcul peut s'effectuer de manière itérative et converge vers une valeur asymptotique de manière assez rapide. En effet, le calcul effectué dans [Page et al., 1998] sur un graphe de 26 millions de nœuds en considérant d = 0.85, converge en seulement 52 itérations.

II.3.3 L'algorithme Salsa :

Un algorithme alternatif, SALSA, a été proposé par Lempel et Moran en 2000 [Lempel et Moran, 2000]. Son but est similaire à celui de l'algorithme HITS décrit dans la section II.3.1 : il cherche à déterminer les meilleurs pages correspondant à un sujet donné en les caractérisant en hubs et autorités.

La méthode choisie par les auteurs pour résoudre ce problème est toutefois différente de celle utilisée par Kleinberg dans l'algorithme HITS. Elle consiste à parcourir au hasard les nœuds du graphe du web en suivant les liens les reliant avec une distribution de probabilité uniforme. Les pages les plus visitées correspondent alors aux solutions recherchées. Intuitivement, les pages faisant office d'autorités sur un sujet donné sont visibles - liées - depuis beaucoup de pages dans le sous-graphe étudié. Ainsi, en parcourant le graphe au hasard des liens, la probabilité de visiter une page autorité est grande.

Cette idée de détermination stochastique de l'intérêt d'une page se rapproche de celle utilisée par Brin et Page dans le calcul du PageRank mais diffère dans la séparation des résultats en hubs et autorités pour un domaine donné. La détermination des probabilités d'appartenance de chaque page du graphe à l'une des deux catégories possibles se fait à l'aide de chaînes de Markov.

La première étape consiste à construire un sous-graphe du web correspondant à une requête donnée, sur lequel s'effectuera la recherche de pages pertinentes. Cette construction se base sur celle utilisée dans HITS et requiert les mêmes propriétés que celles décrites dans la section II.3.1.

Afin de déterminer le potentiel de hub et d'autorité de chaque page, il faut déterminer la probabilité de visite de ces pages en suivant les liens du graphe de manière aléatoire. Le parcours des arcs dans le sens initial nous permet de déterminer les potentiels d'autorités des pages : on mesure ici la probabilité qu'une page soit pointée par beaucoup d'autres pages. Le parcours du graphe en suivant les arcs dans leur sens inverse nous donne les potentiels de hub de chaque page : on mesure, dans ce cas, la probabilité qu'une page pointe vers beaucoup d'autres pages.

Deux chaînes de Markov sont alors analysées : une chaîne de hubs et une chaîne d'autorités. Les états de transitions de ces chaînes sont générés en effectuant le parcours aléatoire du graphe. Mais, contrairement à un parcours classique des liens, deux liens hypertextes sont traversés : (1) en choisissant selon une probabilité uniforme d'aller sur une page pointée par la page actuelle, et (2) en choisissant selon une probabilité uniforme d'aller sur une page pointant vers la page actuelle.

Les potentiels d'autorités sont alors définis comme étant la distribution stationnaire de la chaîne de Markov effectuant d'abord un pas (1) puis un pas (2), tandis que les potentiels de hubs sont définis comme la distribution stationnaire de la chaîne effectuant d'abord un pas (2) puis un pas (1).

Formellement, les deux matrices de transition des deux chaînes de Markov sont définies ainsi :
- La matrice déterminant les potentiels de hub H :

$$h_{i,j} = \sum_{k:k\,\epsilon S(i)\cap S(j)} \frac{1}{|S(i)|} \cdot \frac{1}{|E(k)|} \qquad (21)$$

- La matrice déterminant les potentiels d'autorité Ã :

$$\tilde{a}_{i,j} = \sum_{k:k\,\epsilon E(i)\cap E(j)} \frac{1}{|E(i)|} \cdot \frac{1}{|S(k)|} \qquad (22)$$

Où : *E(i)* décrit tous les nœuds du graphe pointant vers le nœud i, et donc les pages que nous pouvons atteindre depuis la page *i* en suivant un lien dans le sens inverse, *S(i)* décrit tous les nœuds que nous pouvons atteindre depuis le nœud *i* en suivant un lien de *i*.
Une probabilité de transition $\tilde{a}_{i,j} > 0$ indique qu'une certaine page *k* pointe à la fois vers les pages *i* et *i* et que, de plus, la page *j* peut être atteinte depuis la page *i* en deux pas : en parcourant le lien de la page *k* vers *i* dans le sens inverse et en suivant ensuite le lien de la page *k* vers la page *j*.

II.4 La modélisation de langue en Recherche d'Information :

II.4.1 Idée de base :

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

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

$$P(s) = \prod_{i=1}^{l} P(m_i \mid m_1 \ldots m_{i-1}) \qquad (23)$$

Si on utilise la règle de chaîne en théorie de probabilité pour calculer cette probabilité, il y a souvent trop de paramètres (c'est-à-dire $P(m_i|m_1 \ldots m_{i-1})$) à estimer, et ceci est souvent impossible à réaliser. Ainsi, dans les modèles de langue utilisés en pratique, des simplifications sont souvent faites. En général, on suppose qu'un mot m_i ne dépend que de ses *n*-1 prédécesseurs immédiats, c'est-à-dire :

$$P(m_i \mid m_1 \ldots m_{i-1}) = P(m_i \mid m_{i-n+1} \ldots m_{i-1}) \qquad (24)$$

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

Uni-gramme : $$P(s) = \prod_{i=1}^{l} P(m_i) \qquad (25)$$

Bi-gramme : $$P(s) = \prod_{i=1}^{l} P(m_i \mid m_{i-1}) = \prod_{i=1}^{l} (P(m_{i-1}m_i) / P(m_{i-1})) \qquad (26)$$

Tri-gramme : $$P(s) = \prod_{i=1}^{l} P(m_i \mid m_{i-2}m_{i-1}) = \prod_{i=1}^{l} (P(m_{i-2}m_{i-1}m_i) / P(m_{i-2}m_{i-1})) \qquad (27)$$

Ce que l'on doit estimer sont les probabilités $P(m_i)$ (uni-gramme), $P(m_{i-1}m_i)$ (bi-gramme) et $P(m_{i-2}m_{i-1}m_i)$ (tri-gramme) pour la langue. Cependant, il est difficile d'estimer ces probabilités pour une langue dans l'absolu. L'estimation ne peut se faire que par rapport à un corpus de textes *C*. Si le corpus est suffisamment grand, on peut faire l'hypothèse qu'il reflète la langue en général. Ainsi, le modèle de langue peut être approximativement le modèle de langue pour ce corpus – $P(\bullet|C)$. Selon les fréquences d'occurrence d'un n-gramme α, sa probabilité $P(\alpha|C)$ peut être directement estimée comme suit :

$$P(\alpha) = \frac{|\alpha|}{\sum_{\alpha j \in C} |\alpha_j|} = \frac{|\alpha|}{|C|} \qquad (28)$$

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

Exemple : Supposons un petit corpus contenant 10 mots, avec les fréquences montrées dans le tableau qui suit :

Mot	le	un	prof	ML	dit	aime	De	langue	modèle	RI
Fréquence	3	2	2	1	2	1	4	2	1	2
P(•|C)	0.15	0.1	0.1	0.05	0.1	0.05	0.2	0.1	0.05	0.1

Tableau II.3 : Exemple d'estimation de vraisemblance maximale

En utilisant l'estimation de vraisemblance maximale, nous obtenons les probabilités illustrées dans le tableau (remarque : la fréquence totale de mots dans ce corpus est $|C| = 20$).

En utilisant ces probabilités estimées, nous pouvons par exemple, calculer la probabilité de construire la séquence *s* = « le prof aime le ML » dans cette langue comme suit:

$$P(s) \approx P(s|C) = P(\textbf{le}|C) * P(\textbf{prof}|C) * P(\textbf{aime}|C) * P(\textbf{le}|C) * P(\textbf{ML}|C) = 0.15*0.1*0.05*0.15*0.05$$

II.4.2 Lissage :

Plus le corpus utilisé pour ces estimations est grand, plus on peut espérer obtenir des estimations de probabilité justes. Cependant, quelle que soit la taille du corpus d'entraînement, il y a toujours des mots ou des séquences de mots absents du corpus. Pour ces mots ou séquences de mots, leur estimation de probabilité est 0. La conséquence de cette probabilité nulle est qu'on attribuera une probabilité nulle à toute séquence de mots ou des phrases contenant un mot ou un n-gramme non rencontré dans le corpus. Par exemple, la phrase *s* = « **le prof dit non** » aura une probabilité $P(s|C) = 0$. En d'autres termes, les modèles ainsi construits ne sauraient reconnaître que les phrases dont les n-grammes sont tous apparus dans le corpus. Ce sont donc des modèles très limités.

Afin de généraliser les modèles, on voudrait assouplir cette attribution systématique de probabilité nulle aux mots ou séquences de mots non rencontrés. Cette procédure qui consiste à attribuer une probabilité non-nulle à ces éléments est appelée le lissage. Le lissage peut aussi être vu comme une façon d'éviter le surentraînement d'un modèle sur un corpus, et de doter le modèle d'une plus grande capacité de généralisation.

Le principe du lissage peut être résumé ainsi : Au lieu de distribuer la totalité de masse de probabilité sur les n-grammes vus dans le corpus d'entraînement, on enlève une partie de cette masse et la redistribue aux n-grammes non vus dans le corpus. De cette façon, les n-grammes absents du corpus vont recevoir une probabilité non-nulle.

Sur la façon d'enlever et de redistribuer une partie de masse de probabilité, il y a une série de méthodes proposées dans la littérature. [Stanly & Goodman, 1998] constituent un bon état de l'art des méthodes de lissage.

Les techniques de lissage les plus connues sont récapitulées dans le tableau II.3 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-gramme α, C : correspond au corpus. → Pas de performance sur les corpus de petite taille.
Lissage de Good-Turing	$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_i \in C} \lvert \alpha_i \rvert}$ $r^* = (r+1)\frac{n_{r+1}}{n_r}$	r : fréquence d'occurrence d'un n-gramme α, n_r : nombre de n-gramme apparus r fois, → Recommandé pour les n-gramme de faibles fréquences, car l'estimation GT est instable pour les n-grammes de grandes fréquences.
Lissage de Backoff	$P_{Katz}(m_i/m_{i-1}) = \begin{cases} P_{GT}(m_i/m_{i-1}) & si \lvert m_{i-1}m_i \rvert > 0 \\ \alpha(m_{i-1}) P_{Katz}(m_i) & sinon \end{cases}$ Où : $\alpha(m_{i-1}) = \frac{1 - \sum_{mi : \lvert mi\text{-}1mi \rvert > 0} P_{GT}(m_i/m_{i-1})}{1 - \sum_{mi : \lvert mi\text{-}1mi \rvert > 0} P_{ML}(m_i)}$	$\alpha(m_{i-1})$: paramètre de normalisation, m_i , m_{i-1} : n-grammes observés d'ordre i et i-1, → Utilise un modèle d'ordre inférieur aux n-grammes dont la fréquence est nulle.
Lissage par interpolation	$P_{JM}(m_i/m_{i-1}) = \lambda_{mi-1} P_{ML}(m_i/m_{i-1}) + (1-\lambda) P_{JM}(m_i)$ En RI, cette méthode prend un autre sens : $(1-\alpha)P(w/d) + \alpha P(w/C)$	α : 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.
Lissage de 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 II.4 : Les différentes techniques de lissage

II.4.3 Principes du modèle de langue en RI :

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

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

$$\boldsymbol{Score\ (Q, D) = P(Q \mid M_D)} \qquad \boldsymbol{(29)}$$

De façon générale, une requête peut être vue comme une suite de mots : $Q = t_1 t_2 \ldots t_n$. Nous avons donc :

$$\boldsymbol{Score\ (Q, D) = P(t_1 t_2 \ldots t_n \mid M_D)} \qquad \boldsymbol{(30)}$$

Ainsi formulé, le problème de la RI devient similaire à la modélisation statistique de langue. Il est donc possible d'utiliser ces techniques développées pour cette dernière en RI.
Le principe que nous venons de décrire est celui que la plupart des modèles de langues en RI utilisent. Mais il y a également quelques autres variantes.

Nous pouvons aussi procéder dans le sens inverse : On tente de créer un modèle de langue pour la requête, et de déterminer le score d'un document $D = t_1 t_2 \ldots t_m$ par la probabilité que le document peut être généré par ce modèle :

$$\boldsymbol{Score\ (Q, D) = P(t_1 t_2 \ldots t_m \mid M_Q)} \qquad \boldsymbol{(31)}$$

Cependant, dans cette formulation, les documents longs et contenant des mots fréquents vont être favorisés.

Afin de résoudre ce problème, nous pouvons utiliser la loi de Bayes :

$$\boldsymbol{P\ (M_Q \mid D) = \frac{P(M_Q)\ P(D \mid M_Q)}{P(D)}} \qquad \boldsymbol{(32)}$$

On suppose ensuite que $P(D) \approx P(D|C)$ et que $P(M_Q)$ est une constante c. Ainsi :

$$P(M_Q \mid D) \approx \frac{cP(D \mid M_Q)}{P(D \mid C)} \qquad (33)$$

Dans cette nouvelle formule, nous tenons compte de la longueur du document et des mots contenus dans le document, en ajoutant le facteur $P(D|C)$.

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

Finalement, on peut aussi voir la relation entre une requête et un document pertinent comme celle d'une traduction : un document pertinent est une « traduction » de la requête; et on tente de déterminer la probabilité de cette traduction.

Dans ce qui va suivre, nous présentons en détails les méthodes spécifiques pour implanter ces principes.

II.4.4 Approches de modélisation de langage pour la RI :

II.4.4.1 La RI comme génération de la requête par le document :

a. Le modèle de Ponte et Croft :

[Ponte & Croft, 1998] supposent qu'une requête n'est pas créée de façon aléatoire, mais l'utilisateur a une idée (modèle) sur le document idéal D et son modèle M_D.

À partir de M_D, l'utilisateur choisit (génère) les termes pour constituer sa requête. Ainsi, la requête devrait être générée à partir d'un document pertinent ou de son modèle. La probabilité de cette génération est considérée comme le score du document :

$$Score\ (D, Q) = P(Q \mid M_D) \qquad (34)$$

Remarques :

- Une requête peut être considérée comme une suite de mots : $Q = t_1, t_2, \ldots t_n$.
- Comme dans la plupart des modèles de langues utilisés en RI, la simplification suivante a été faite : on suppose que les mots dans une requête sont indépendants.
- Dans le modèle de Ponte et Croft, ils utilisent le modèle de Bernoulli multiple, c'est-à-dire que non seulement les mots présents dans la requête, mais aussi ceux absents de la requête, sont pris en compte. Dans la plupart des autres modèles, on utilise plutôt le modèle multinomial, où on ne considère que les mots présents dans la requête.

Supposons, sans perdre la généralité, que la requête Q est composée de l'ensemble de mots $t_1, t_2, \ldots t_n$, et que les mots $t_{n+1}, t_{n+2}, \ldots t_l$ sont absents de la requête. $P(Q|M_D)$ peut être ré-exprimée comme suit :

$$P(Q \mid M_D) = P(t_1, t_2 \ldots t_n \mid M_D) \times P(\neg\, t_{n+1}, \neg\, t_{n+2}, \ldots \neg\, t_l \mid M_D) \qquad (35)$$

$$= \prod_{i=1}^{n} P(t_i \mid M_D) \times \prod_{i=n+1}^{l} P(\neg\, t_i \mid M_D)$$

$$= \prod_{t_i \epsilon Q} P(t_i \mid M_D) \times \prod_{t_i \notin Q} (1 - P(t_i \mid M_D))$$

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

$$P_{PC}(t_i \mid M_D) = \begin{cases} P_{ML}(t_i \mid D)^{1-\hat{R}(ti,D)} \times P_{avg}(t_i)^{\hat{R}(ti,D)} & si\ tf(t_i,D) > 0 \\ P_{ML}(t_i \mid C) & sinon \end{cases} \qquad (36)$$

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

Dans cette formule, on note deux éléments principaux $P_{ML}(t_i \mid D)$ et $P_{ML}(t_i \mid C)$. Il y a aussi quelques autres éléments ajoutés pour tenir compte de certaines particularités de la RI : $P_{avg}(t_i)$ est la probabilité moyenne du terme t_i dans les documents qui le contiennent ; $\hat{R}(t_i,D)$ est une fonction de « risque » déterminée comme suit :

$$\hat{R}(t,D) = \frac{1}{1+\bar{f}_t} \times \left(\frac{\bar{f}_t}{1+\bar{f}_t} \right)^{tf(t,D)} \qquad (37)$$

Où $\bar{f}_t$ est la fréquence moyenne du mot t dans les documents qui le contient. L'élément $P_{avg}(t_i)^{R(ti,D)}$ est ajouté pour contrer le « risque » de l'estimation $P_{ML}(t_i \mid D)$ en tenant compte de $P_{avg}(t_i)$ calculée sur le corpus. Ceci ressemble à un lissage par interpolation.

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

b. Le modèle de Hiemstra et al. et de Miller et al. :

Hiemstra et al. [Hiemstra, 1998] utilisent le même principe de génération de la requête par le document. La formule que Hiemstra propose est la suivante :

$$Score\ (D, Q) = P(D \mid Q) = P(D \mid t_1, t_2 \ldots t_n) \qquad (38)$$

$$= P(D)\ \frac{P(t_1, t_2 \ldots t_n \mid D)}{P(t_1, t_2 \ldots t_n)}$$

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

$$Score\ (D,Q) = c\ P(D) \prod_{t \epsilon Q} P(t_i \mid D)^{tf(ti|Q)} \qquad (39)$$

Pour estimer $P(t_i \mid D)$, Hiemstra utilise une approche d'interpolation :

$$P(t_i \mid D) = \alpha P_{ML}(t_i \mid D) + (1 - \alpha)\ P_{ML}(t_i \mid C) \qquad (40)$$

L'estimation de vraisemblance maximale de $P_{ML}(t_i \mid D)$ et $P_{ML}(t_i \mid C)$ sont respectivement :

$$\frac{tf\ (t_i, D)}{|\ D\ |} \quad \text{et} \quad \frac{df\ (t_i)}{\sum_{tj \epsilon V} df\ (t_j)}$$

Où $df\ (t_i)$ est le nombre de documents contenant t_i et V est le vocabulaire d'indexes.

En utilisant le logarithme, on obtient :

$$\log\ (P(t_1 t_2 \ldots t_n \mid D)) = \log_{t \epsilon Q} \prod P(t_i \mid D)$$

$$= \sum_{i=1}^{n} \log\ (\alpha P_{ML}(t_i \mid D) + (1 - \alpha)\ P_{ML}(t_i \mid C)) \qquad (41)$$

$$= \sum_{i=1}^{n} \log \left(1 + \frac{\alpha P_{ML}(t_i \mid D)}{(1 - \alpha)\ P_{ML}(t_i \mid C)} \right) + \sum_{i=1}^{n} \log\ ((1 - \alpha)\ P_{ML}(t_i \mid C))$$

En principe, le paramètre α peut bien dépendre du document ou de la requête. Ainsi, une façon plus sophistiquée consiste à estimer une valeur de α en utilisant un processus d'optimisation automatique telle que la maximisation de l'espérance (EM).

La probabilité à priori *P(D)* d'un document *D* est estimée comme suit :

$$P(D) = \frac{|D|}{|C|} \qquad (42)$$

Où : |*D*| est la taille du document et |*C*| est la taille de la collection.

En somme, on a :

$$\log Score(Q,D) = \sum_{t \epsilon Q} \log\left(1 + \frac{\alpha \times tf(t,D) \sum_{t_i \epsilon V} df(t_i)}{(1-\alpha)\,|D|\,df(t)}\right) + \log \frac{|D|}{|C|} \qquad (43)$$

On pourrait estimer la probabilité à priori conditionnée sur d'autres propriétés des documents, par exemple la forme de l'URL ou le nombre des liens. Cette approche s'est avérée très efficace dans une application RI particulière - la recherche des pages d'entrée (*entry page*) [Kraaij, 2002].

Miller et al. [Miller, 1998] [Miller, 1999] utilise une formulation similaire, à quelques détails près. La plus grande différence entre le modèle de Miller et celui de Hiemstra est la façon d'implanter le modèle : Miller utilise un modèle de Markov caché à deux états comme suit :

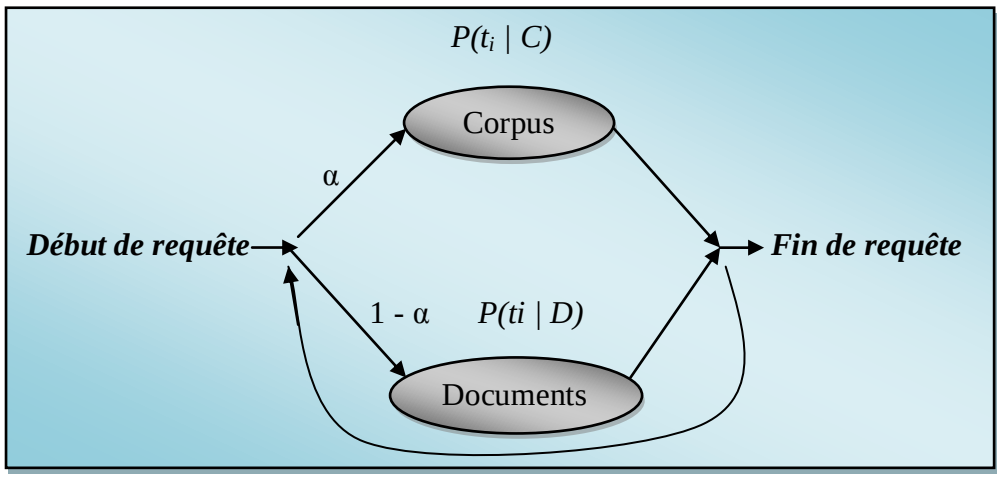


Figure II.3 : Modèle de Markov caché à deux états [Miller, 1998]

Ce modèle correspond à la formule suivante :

$$P(Q \mid D \text{ est pertinent}) = \prod_{t_i \epsilon Q} \alpha P_{ML}(t_i \mid D) + (1 - \alpha)\, P_{ML}(t_i \mid C) \qquad (44)$$

Ce qui est similaire au modèle de Hiemstra.

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

II.4.4.2 La RI comme ratio de vraisemblance :

Ng [Ng, 1999] a proposé une autre variante de modèle de langue basée sur le ratio de vraisemblance. Au lieu d'estimer la probabilité de pertinence d'un document pour une requête, Ng propose d'utiliser la vraisemblance comme critère de classement. L'idée qu'il avance est la suivante : Si la vraisemblance d'un document augmente après que la requête soit soumis, le document a une plus forte probabilité d'être utile qu'un autre document dont la vraisemblance ne change pas ou décroît.

Ng propose la formule suivante pour mesurer cette quantité :

$$LR(D,Q) = \frac{P(D \mid Q)}{P(D)} = \frac{P(Q \mid D)}{P(Q)} \qquad (45)$$

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

$$P(Q) = \prod_{t \epsilon Q} P_{GT}(t) = \prod_{t \epsilon Q} P_{GT}(t \mid C) \qquad (46)$$

Pour $P(t \mid D)$, l'estimation Good-Turing est mal adaptée à cause de la longueur limitée du document. Ainsi, Ng utilise l'interpolation suivante :

$$P(Q \mid D) = \prod_{t \epsilon Q} P(t \mid D) = \prod_{t \epsilon Q} \alpha P_{ML}(t \mid D) + (1 - \alpha)\, P_{GT}(t \mid C) \qquad (47)$$

La formule du score de document est donc la suivante :

$$Score(D,Q) = \sum_{t \epsilon Q} log\left(\frac{\alpha P_{ML}(t \mid D) + (1 - \alpha)\, P_{GT}(t \mid C)}{P_{GT}(t \mid C)} \right) \qquad (48)$$

Le paramètre α est estimé avec l'algorithme EM.

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

$$Score\ (D,Q) = \sum_{t \epsilon Q} log\left(1 + \frac{\alpha P_{ML}(t \mid D)}{(1-\alpha)\ P_{GT}\ (t)}\right) + \sum_{t \epsilon Q} log\ (1-\alpha) \qquad (49)$$

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

II.4.4.3 Le modèle basé sur l'entropie croisée :

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

Reprenons l'équation (45), on suppose comme d'habitude, l'indépendance des termes, en considérant une fonction logarithmique et après quelques transformations, cette équation peut s'écrire de la façon suivante :

$$LR(Q \mid D) = log\ \frac{P(Q \mid M_D)}{P(Q \mid M_C)} \qquad (50)$$

Supposons une distribution multinomiale de termes dans le document et dans le corpus, on a :

$$P(Q \mid M_D) = \frac{\mid Q \mid !}{\prod_{t_i \epsilon Q} tf\ (t_i,Q)\ !} \prod_{t_i \epsilon Q} P(t \mid D)^{\,tf(ti,Q)}$$

$$P(Q \mid M_C) = \frac{\mid Q \mid !}{\prod_{t_i \epsilon Q} tf\ (t_i,Q)\ !} \prod_{t_i \epsilon Q} P(t \mid C)^{\,tf(ti,Q)} \qquad (51)$$

Où | *Q* | est la taille de la requête (le nombre d'occurrences de mots).

Ainsi, on a :

$$LR(Q \mid D) = \sum_{i=1}^{n} tf\,(t_i,Q) \times log\ \frac{\alpha P(t_i \mid M_D) + (1-\alpha)\,P(t_i \mid M_C)}{P(t_i \mid M_C)} \qquad (52)$$

Où $P(Q|M_C)$ est la probabilité générative de la requête étant donné un modèle de langage estimé sur un corpus C et $tf\,(t_i, Q)$ est la fréquence du terme 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 la longueur de la requête, ils peuvent être facilement normalisés en les divisant par la longueur de la requête (comme la longueur est constante, elle n'intervient pas dans le score). La forme normalisée de l'équation (52) peut donc s'exprimer comme suit :

$$NLR\,(Q \mid D) = log\ \frac{P(Q \mid M_D)}{P(Q \mid M_C)}$$

$$= \sum_{i=1}^{n} \frac{tf\,(t_i,Q)}{\sum_i tf\,(t_i,Q)} \times \log \frac{\alpha P(t_i \mid M_D) + (1-\alpha)\,P(t_i \mid M_C)}{P(t_i \mid M_C)} \qquad (53)$$

L'étape suivante est de considérer le rapport $\frac{tf\,(t_i,Q)}{\sum_i tf\,(t_i,Q)}$ comme une estimation du maximum de vraisemblance de la distribution de probabilité représentant la requête $P(t_i|M_Q)$. L'équation (53) peut être réinterprétée comme la relation entre les deux distributions de probabilité $P(t|M_D)$ *et* $P(t|M_Q)$, normalisées par la troisième distribution $P(t|M_C)$:

$$NLR\,(Q \mid D) = \sum_{i=1}^{n} P(t_i \mid M_Q) \times \frac{P(t_i \mid M_D)}{P(t_i \mid M_C)} \qquad (54)$$

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

$$NLR\,(Q \mid D) = H(X \mid C) - H(X \mid D) \qquad (55)$$

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

L'entropie croisée permet donc de mesurer en moyenne l'écart entropique sur le fait que le modèle de document correspond (suit) bien à la distribution probabiliste de la requête. D'autres formes de l'entropie croisée ont été étudiées, on y trouve notamment celle basée sur l'information de Kullback-Leibler dans [Lavrenko, 2001].

Remarquons que dans l'entropie croisée, on voit apparaître implicitement le modèle de langue de la requête M_Q. Si on l'utilise seul (ou pour remplacer le modèle de langue du document), l'approche souffrira encore plus du problème de l'information limitée pour la construction du modèle. Dans le cas de la RI pour des requête ad hoc (comme sur le Web), typiquement, la requête est très courte. Cette approche est donc impraticable. Dans le contexte de « topic tracking » où la « requête » - un texte – est plus longue, on peut davantage utiliser cette approche. Cependant, sa performance n'est pas très bonne. Dans l'approche d'entropie croisée, comme celle utilisée dans le modèle de pertinence [Lavrenko, 2001], le modèle de langue de la requête n'est pas utilisé seul, mais ensemble avec le modèle du corpus et le modèle du document, dans un cadre qui tente de modéliser la pertinence.

Remarquons aussi que le principe de comparaison du modèle du document et le modèle de la requête est aussi utilisé dans le cadre de « minimisation de risque » [Lefferty, 2001] [Zhai, 2002]. La décision pour retrouver un document est basée sur une fonction de perte comparant les deux modèles. Ils ont ensuite proposé un modèle de langue à deux étapes : une première étape pour lisser le modèle de document (pour le problème de clairesemence de données), et une deuxième étape pour tenir compte de la pertinence des termes dans la requête (le modèle de requête). Les techniques spécifiques utilisées pour les deux étapes sont respectivement le lissage Dirichlet et le lissage par interpolation.

II.4.4.4 La RI comme traduction statistique :

Une idée toujours dans la même veine que les approches basées sur les modèles de langue considère que la RI est un processus de traduction particulier [Berger & Lafferty, 1999] : On considère qu'un besoin d'information est un message à transmettre. Ce message est d'abord formulé par une requête. Le processus de recherche est considéré comme un processus de traduction, qui traduit la requête en un document. Plus cette relation de traduction est probable, plus le document peut être un document pertinent à la requête. Ainsi, on tente de déterminer les documents qui maximisent la probabilité de traduction $P(D|Q)$.

a. Les travaux de Berger et Lafferty :

Il s'agit d'une adaptation (translation) du modèle $t(q_i|w)$ pour apparier un terme du document w à celui d'une requête [Berger & Lafferty, 1999].

En réalité, la probabilité de translation $t(q_i|w)$ décrit le degré d'association entre le mot de la requête *qi* et le mot du document *w*.

Avec cette translation, le modèle document / requête s'écrit:

$$P(q \mid d) = \prod_{i=1}^{n} \sum_{w} t(q_i \mid w)\, P(w \mid d) \qquad (56)$$

Même si leur modèle est plus général que les autres modèles de langue, en pratique, il est difficile de déterminer la probabilité de translation $t(q_i|w)$. Pour résoudre ce problème, Berger et Lafferty ont généré une collection de données d'apprentissage et estimé les cooccurrences de q_i et *w*.

b. Les travaux de Lafferty et Zhai :

[Lafferty & Zhai, 2001] ont tenté de résoudre le problème cité précédemment d'une façon différente. Ils ont développé un modèle plus général de translation de mots avec les chaînes de Markov. Ils utilisent un chemin aléatoire pour avoir la probabilité de translation $t(q_i|w)$ à partir d'un ensemble de documents dans la collection.

L'approche de traduction statistique présente un intérêt particulier d'un point de vue théorique [Boughanem & al., 2004], parce qu'elle intègre la *polysémie* et la *synonymie* dans le modèle de RI en modélisant explicitement les relations entre les termes.

Bien que cette classe de modèles soit toujours en cours de développement et d'investigation, les travaux et les expérimentations effectués dans ce thème ont montré que les modèles de langage fournissent un cadre intéressant et prometteur pour appréhender la RI. De plus, même si différents modèles ont été proposés, l'idée principale reste effectivement la suivante : « Calculer la probabilité que la requête soit générée par le modèle de langage du document ». Ce point est évidemment partagé par l'ensemble des modèles.

Une analyse comparative des différentes approches proposées montre que les premières d'entre elles ont omis la notion de pertinence. Une justification possible consiste à considérer $P(Q|D)$ comme une version probabiliste de l'implication logique. Cette implication a été mise en évidence par Van Rijsbergen [Rijsbergen, 1986] qui a effectivement démontré que la recherche d'information peut être vue comme le problème de calculer la probabilité qu'un document implique la requête : $P(D \rightarrow Q)$. Ceci étant, les expérimentations effectuées dans ces modèles montrent qu'ils permettent effectivement de « capturer » la pertinence.

II.5 La pertinence à priori d'un document :

Selon les approches qu'on vient de présenter, les propriétés (taille de document, nombre de liens entrants, etc.) indépendantes des requêtes peuvent être utilisées pour conditionner la probabilité à priori de pertinence d'un document. Si la probabilité à priori de pertinence *P(D)* n'est pas conditionnée par l'une de ces propriétés alors cette probabilité représente la probabilité de prélever un document de la collection, par conséquent tous les documents sont équiprobables dans la collection et la probabilité à priori de pertinence du document peut être ignorée lors du classement des documents.

Par contre, si la probabilité à priori est conditionnée par l'une de ces caractéristiques alors les documents de la collection n'ont pas la même probabilité à priori et les documents ne sont pas équiprobables. Par exemple, si la caractéristique utilisée est le score de popularité du document alors un document populaire est plus probable d'être pertinent qu'un document moins populaire.

Plusieurs caractéristiques ont été utilisées pour estimer la probabilité à priori d'un document comme : la longueur du document, la structures des liens, le rapport Information/Bruit, type d'URL, le facteur temps. Elles expriment qu'un document est plus probable parce qu'il est plus long, plus populaire, plus récent, ou contient plus d'informations que de bruits.

- **La taille du document :** la probabilité à priori est proportionnelle à la taille du document, telle que :

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

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

Un document plus long tend à contenir plus d'informations et par conséquent il est plus probable d'être pertinent. Les résultats obtenus avec l'utilisation de cette caractéristique ont été mixtes selon la collection utilisée [Kraaij, Westerveld & Hiemstra, 2002], [Miller, Leek & Schwartz, 1999].

- **La structure des liens :** les documents populaires ou les plus cités tendent à être plus pertinents. La méthode utilise généralement le nombre de liens entrants, i.e. :

$$P(D) = \frac{n(l, D)}{\sum_{D'} n(l, D')} \qquad (57)$$

Où : $n(l, D)$ est le nombre de liens entrants.

- **Rapport Information/Bruit :** il est défini comme le rapport entre la taille de document après prétraitement (élimination des mots vides et des balises Html) et la taille de document sans prétraitement [Zhu & Gauch, 2000] :

$$P(D) = \frac{L_{token}}{L_{document}} \qquad (58)$$

Tel que : L_{token} est la taille de document après le prétraitement et $L_{document}$ est la taille de document avant le prétraitement. Ainsi, un document avec moins de mots vides et peu de balises Html produit un haut rapport Information/Bruit, ce qui signifie que le document est de « bonne » qualité.

- **Type d'URL du document :** Kraaij, Westerveld et Hiemstra dans [Kraaij, Westerveld & Hiemstra, 2002] ont utilisé le type d'URL pour estimer la probabilité qu'une page soit une page d'entrée.

$$P(D) = P(PE \mid URLtype(D) = t_i) = \frac{c\,(PE, t_i)}{c\,(t_i)} \qquad (59)$$

Où *URLtype* est le type d'URL de document D, $c(PE, t_i)$ est le nombre de documents de type d'URL « t_i » qui sont des pages d'entrées « PE » pour un site web obtenu à partir des évaluations de pertinence, c (t_i) est le nombre de documents de type d'URL « t_i ».

Quatre types d'URL ont été définis :

- ✓ **Racine :** elle contient le nom de domaine seulement, par exemple : www.sigir.org.
- ✓ **Sous-racine :** elle contient le nom de domaine suivi d'un seul répertoire, par exemple : www.sigir.org/sigirlist/.
- ✓ **Chemin (répertoire) :** il contient le nom de domaine suivi d'un ou de plusieurs répertoires, par exemple : www.sigir.org/sigirlist/issues/.
- ✓ **Fichier :** tout URL se terminant avec un nom de fichier autre qu'index.html.

- **La structure du web :** les caractéristiques utilisées jusqu'ici dépendent du document web (page) seulement. Or, une page web fait partie en général d'un site qui fait partie du web. L'idée explorée ici est l'utilisation des caractéristiques du site web qui contient la page concernée pour conditionner la probabilité de pertinence à priori de la page.

Sous l'hypothèse que dans la plupart des cas les auteurs des pages web référencent la page principale « site » au lieu de référencer la page exacte « la page concernée », l'utilisation du nombre de liens entrants ou de facteurs (comme le Page Rank) ne reflète pas l'importance de la page web dans l'espace web. Autrement dit, on doit assigner plus de confiance aux pages provenant de sites importants « intéressants » que celles provenant de sites moins importants ou même des sites spams.

Pour cela, on introduit le facteur importance de site web « page principale » dans l'évaluation de l'importance d'une page.
Avec la notation suivante où :
Nb1 : nombre de liens pointant le site « page principale »,
Nb2 : nombre de liens pointant la page concernée (p),
N : le nombre de pages dans le site contenant la page (p), (ce nombre peut être considéré comme constant).

La formule suivante proposée, exprime la probabilité à priori de pertinence d'une page, qui intègre l'importance du site qui la contient :

$$P(D)= C[a((Nb1) / N) + (1- a) Nb2] \qquad (60)$$

Tel que C est une constante et α est compris entre 0 et 1.

II.6 Conclusion :

Dans ce chapitre, on a pu voir comment s'effectue la Recherche d'Information sur le web en utilisant certains outils et algorithmes de recherche. On a pu voir également comment s'effectue la modélisation de langue en Recherche d'Information de part ses principes et ses approches, du modèle initialement proposé dans ce domaine [Ponte & Croft, 1998] jusqu'aux modèles développés récemment. Cette étude nous a permis de constater l'engouement de la Recherche pour les modèles de langue.

Elle nous a permis aussi de constater que les approches de modélisation de langage présentées dans ce chapitre, permettent d'intégrer des informations pour conditionner la probabilité à priori de pertinence d'un document sur le web et cela en se basant sur l'hypothèse suivante : « Un document est à priori pertinent s'il est "dissemblable" aux autres documents de la collection ».

Dans le chapitre qui suit, on va voir comment implémenter une approche, pour calculer la pertinence à priori d'un document, basée sur le calcul de la dissemblance de ce dernier par rapport à tous les documents de la collection.

Chapitre III

Description de l'approche et Implémentation

III.1 Introduction :

Après avoir présenté les notions de base de la Recherche d'Information dans le premier chapitre et détaillé les différentes approches de modélisation de langage dans ce domaine dans le deuxième chapitre, on exposera notre approche ainsi que son implémentation et évaluation dans ce dernier chapitre.

On va commencer par une description de notre approche qui sera illustrée par un petit exemple illustratif. Suite à ça, on va effectuer quelques présentations des différents outils utilisés pour l'implémentation de notre approche et on terminera par la présentation des résultats d'évaluation de cette dernière.

III.2 Description de l'approche implémentée :

Pour calculer la pertinence d'un document vis-à-vis d'une requête, le modèle de langage propose la formule suivante :

$$RSV(q,d) = P(d) \prod_{i=1}^{n} p(t_i|d) \qquad (61)$$

Où : *P(d)* est la probabilité à priori de pertinence du document *d*.

Notre approche consiste à implémenter une nouvelle méthode pour calculer la pertinence à priori d'un document dans une collection de documents.

Notre approche est basée sur l'hypothèse suivante : « Un document est à priori pertinent s'il est "dissemblable" aux documents de la collection ». On peut expliquer cette hypothèse par le fait qu'un document qui traite plusieurs thématiques tend à être "semblable" à beaucoup de documents. Un tel document, on peut le qualifier d'un document générique et il ne répond pas à un besoin précis d'un utilisateur.

Pour calculer cette dissemblance d'un document, on a utilisé les deux facteurs *Tf* et *Idf* :

- ✓ *Tf (Term Frequency)* est simplement le nombre d'occurrences d'un terme dans le document considéré. L'idée sous-jacente à *Tf* est la suivante : plus un terme est fréquent dans un document plus il est important dans le document.

- ✓ *Idf (Inverted Document Frequency)* est une mesure de l'importance du terme dans l'ensemble du corpus qui vise à donner un poids plus important aux termes les moins fréquents, considérés comme plus discriminants.

Elle consiste à calculer le logarithme de l'inverse de la proportion des documents du corpus qui contiennent le terme :

$$Idf_i = log \frac{|D|}{|\{d_j : t_i \epsilon d_j\}|} \qquad (62)$$

Avec $|D|$ qui représente le nombre total de documents dans le corpus et $|\{d_j : t_i \epsilon d_j\}|$ qui représente le nombre de documents où le terme t_i apparaît. L'idée sous-jacente à *Idf* est : plus un terme est peu fréquent dans la collection plus il est important dans le document.

Le score de similarité qu'on emploie, dans notre cas, est issu de la formule du Cosinus :

$$Sim\ (D_j,D_i) = \frac{\sum_i w_{Dj} \times w_{Di}}{\sqrt{\sum_i w_{Dj}^2 \times \sum_i w_{Di}^2}} \qquad (63)$$

Pour arriver à notre hypothèse de dissemblance, on utilisera la formule suivante :

$$P(D_j) = log \left(\sum_{i=1}^{n} 1 / Sim\ (D_j,D_i) + 1\right) \qquad (64)$$

Où : *n* est le nombre de documents dans la collection et $Sim(D_j,D_i)$ est la formule *(63)*.

Par la suite, on va comparer notre approche à une méthode répandue qui utilise la taille d'un document comme évidence de pertinence à priori du document et qui est donnée par la formule suivante :

$$P(D_j) = log\ (|D_j|) \qquad (65)$$

Où : $|D_j|$ représente la taille du document *Dj*.

Exemple d'illustration :

Soit les trois documents suivants :

D1 = {java, classe, polymorphisme}

D2 = {java, C, prolog, programmation}

D3 = {C, fonction}

Pour confirmer notre hypothèse, D2 est à priori non pertinent car il traite le langage Java et le langage C.

III.3 Tests et évaluation de l'approche :

III.3.1 Collections de tests utilisées :

Les collections de tests utilisées dans notre projet sont : la collection AP88 (Associated Press newswire, 1988) et la collection WSJ90-92 (Wall Street Journal, de 1990 à 1992), les statistiques sur ces deux collections sont décrites dans le tableau III.1.

Collection	Nombre de documents dans la collection	Nombre de termes dans la collection	Taille moyenne d'un document
AP88	79919	110868	144
WSJ90-92	74520	101428	146

Tableau III.1 : Description des collections de tests AP88 et WSJ90-92

Dans notre cas, on n'utilise qu'un nombre restreint de documents de ces deux collections (3877 documents pour la collection AP88 et 4973 documents pour la collection WSJ90-92) pour l'évaluation de notre approche car le temps nécessaire pour calculer la probabilité de tous les documents d'une collection est trop long.

Pour les 3877 documents de la collection AP88 ou bien les 4973 documents de la collection WSJ90-92, le temps nécessaire pour le calcul de probabilité est de l'ordre de 2 jours.

III.3.2 Outils de développement utilisés :

Les outils utilisés pour implémenter notre approche sont la plateforme de RI Terrier et l'environnement de développement NetBeans.

III.3.2.1 Présentation de la plateforme Terrier :

Terrier (Terabyte Retriever) a été développé par le département informatique de l'université de Glasgow en l'an 2000. C'est un logiciel open source entièrement écrit en Java. Il est utilisé avec succès pour la recherche Ad hoc, la recherche web et la recherche inter-langages dans des environnements distribués et centralisés.

Terrier est une plateforme flexible dédiée au développement rapide des applications de la Recherche d'Information. Elle fournit divers modèles de pondération de documents et d'expansion de requêtes basés sur le Framework DFR (Divergence From Randomness)[1].

Comme tous les moteurs de recherche, Terrier possède les principales facettes suivantes :

- ✓ **L'indexation :** permet l'extraction des termes des différents documents du corpus (basic indexed unit).

- ✓ **La recherche et l'évaluation :** permettent de générer des résultats aux requêtes formulées par les utilisateurs.

a. Le processus d'indexation dans Terrier :

L'indexation d'une collection de documents dans Terrier est un processus qui se déroule en quatre étapes comme suit :

- ✓ **Splitter la collection de documents :** consiste à parcourir l'ensemble du corpus et d'envoyer chaque document à l'étape suivante.

- ✓ **Extraction des termes :** consiste à analyser chaque document reçu et en extraire les différents termes.

- ✓ **Traitement des termes extraits :** consiste, en autre, en l'élimination des mots vides et la lemmatisation des termes.

- ✓ **Construction de l'index.**

---

[1] http://terrier.org/docs/v2.2.1/dfr_description.html

Le procédé d'indexation dans Terrier est représenté dans la figure III.1 suivante :

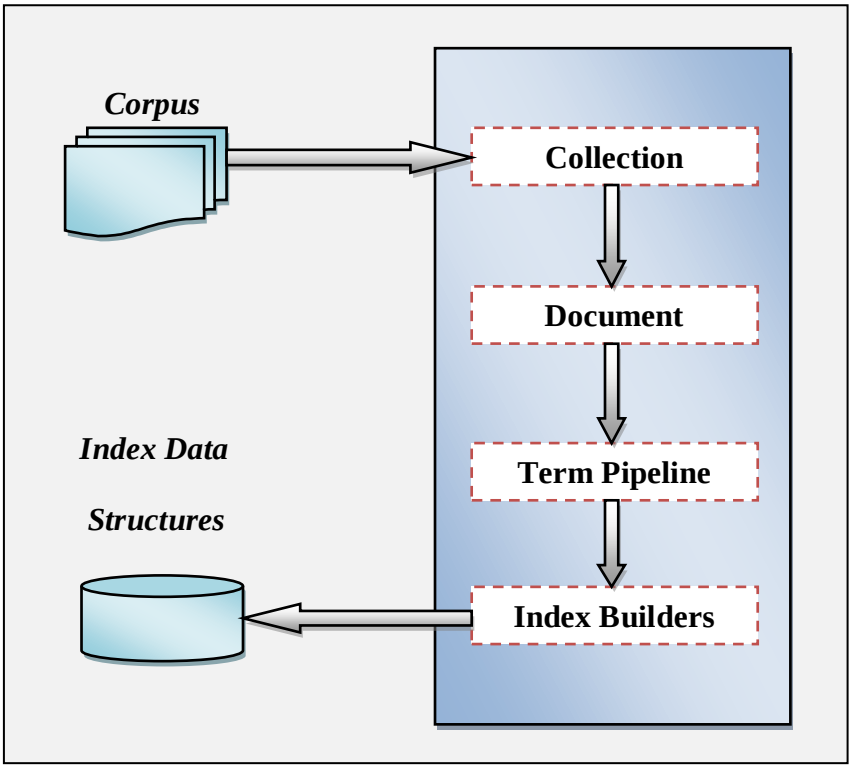


Figure III.1 : Procédé d'indexation dans Terrier

Le résultat de l'indexation consiste en un index constitué des structures de données suivantes : Inverted File, Lexicon File, Direct Index et Document Index (présentés en détails en Annexe).

Les modules Collection, Document, Term Pipeline et Index Builders font partie de l'API d'indexation de Terrier. Ils sont présentés plus en détails en annexe.

b. Le processus de recherche dans Terrier :

Un des principaux objectifs de Terrier est de faciliter la recherche d'information. Terrier implémente pour cela un certain nombre de fonctionnalités de recherche qui offrent un large choix pour le développement de nouvelles applications et pour les tests en RI. En effet, Terrier offre divers modèles de pondération et propose un langage de requête avancé.

Une autre fonction de recherche très importante intégrée dans Terrier est l'expansion de requête automatique.

La figure III.2 suivante représente le procédé de recherche dans Terrier.

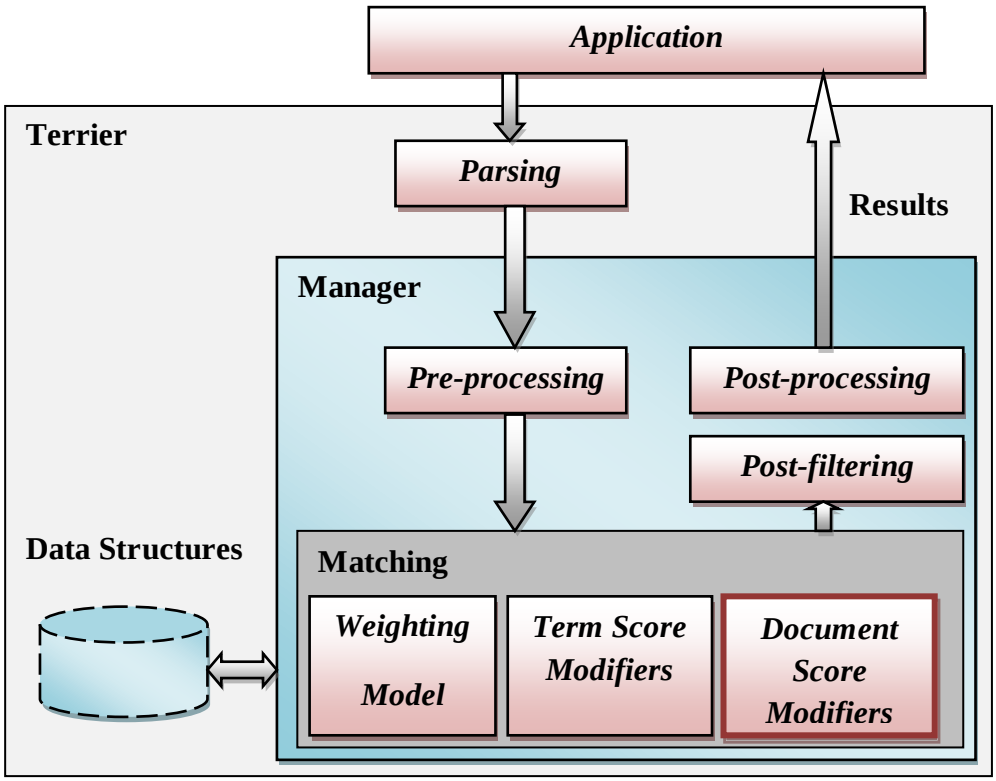


Figure III.2 : Procédé de recherche dans Terrier

Le processus de recherche de Terrier est constitué d'un ensemble de modules : Query, Manager et Matching qui seront présentés plus en détails en annexe.

Le module auquel on s'intéresse dans notre approche est le module Matching et plus précisément l'interface *DocumentScoreModifiers*.

DocumentScoreModifiers est l'interface qui devrait être implémentée par chaque classe qui assigne ou modifie le score d'un document.

Dans notre cas, le score final d'un document est :

$$Score\_Final\ (D) = Score\_Initial\ (D) + (w * Score\_de\_Similarité\ (D)) \qquad (66)$$

Où : *Score_Initial (D)* est le score du document D sans utiliser la probabilité de pertinence à priori, *Score_de_Similarité (D)* est le score du document en utilisant la probabilité de pertinence à priori et $w \in [0,1]$.

III.3.2.2 Présentation de l'environnement de développement NetBeans :

a. Définition et caractéristiques du langage de programmation Java :

- **Définition :**

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. Ses caractéristiques ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisé pour le développement d'applications d'entreprise et mobiles.

- **Caractéristiques :**

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- ✓ **Java est interprété :** le source est compilé en pseudo code ou bytecode puis exécuté par un interpréteur Java : la JVM (Java Virtual Machine).

- ✓ **Java est indépendant de toute plateforme :** il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine.

- ✓ **Java est orienté objet :** comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

- ✓ **Java est multitâche** : il permet l'utilisation de threads qui sont des unités d'exécutions isolées. La JVM, elle même, utilise plusieurs threads.

b. Définition de NetBeans :

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML.

Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, éditeur graphique d'interfaces et de pages Web).

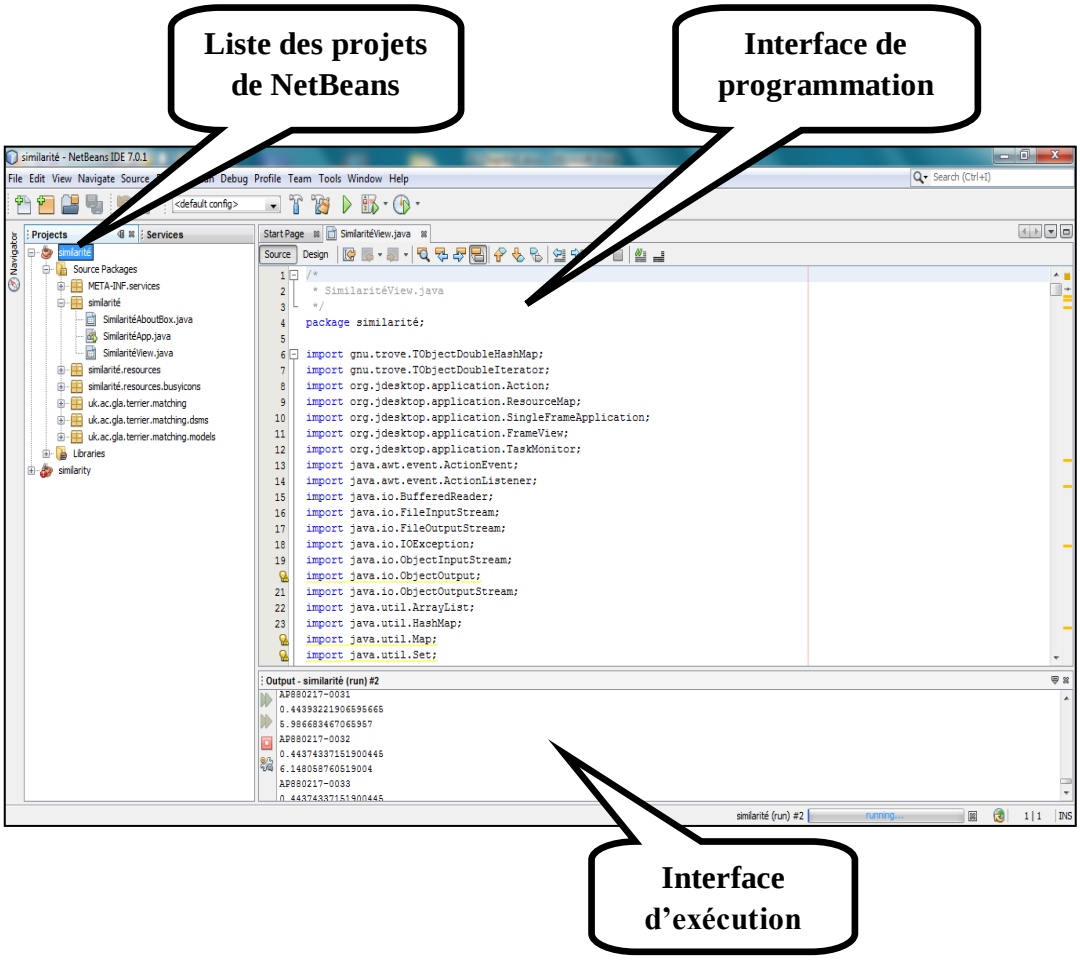


Figure III.3 : L'environnement de développement NetBeans

Conçu en Java, NetBeans est disponible sous plusieurs systèmes d'exploitation notamment Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit (JDK) est requis pour les développements en Java.

III.3.3 Architecture générale de l'approche :

L'architecture générale de notre approche est représentée dans la figure III.4. Elle se déroule en trois parties fondamentales.

- ✓ **Une partie Interface :** constitue l'interface d'accueil à partir de laquelle s'effectuent les différentes fonctions d'exécution.

- ✓ **Une partie Application :** contient les processus d'indexation des documents, le calcul du score de similarité et le processus d'évaluation.

- ✓ **Une partie Stockage :** contient les fichiers de stockages : le fichier contenant le résultat de l'indexation, le fichier contenant le résultat du calcul du score de similarité et le fichier contenant le résultat de l'évaluation.

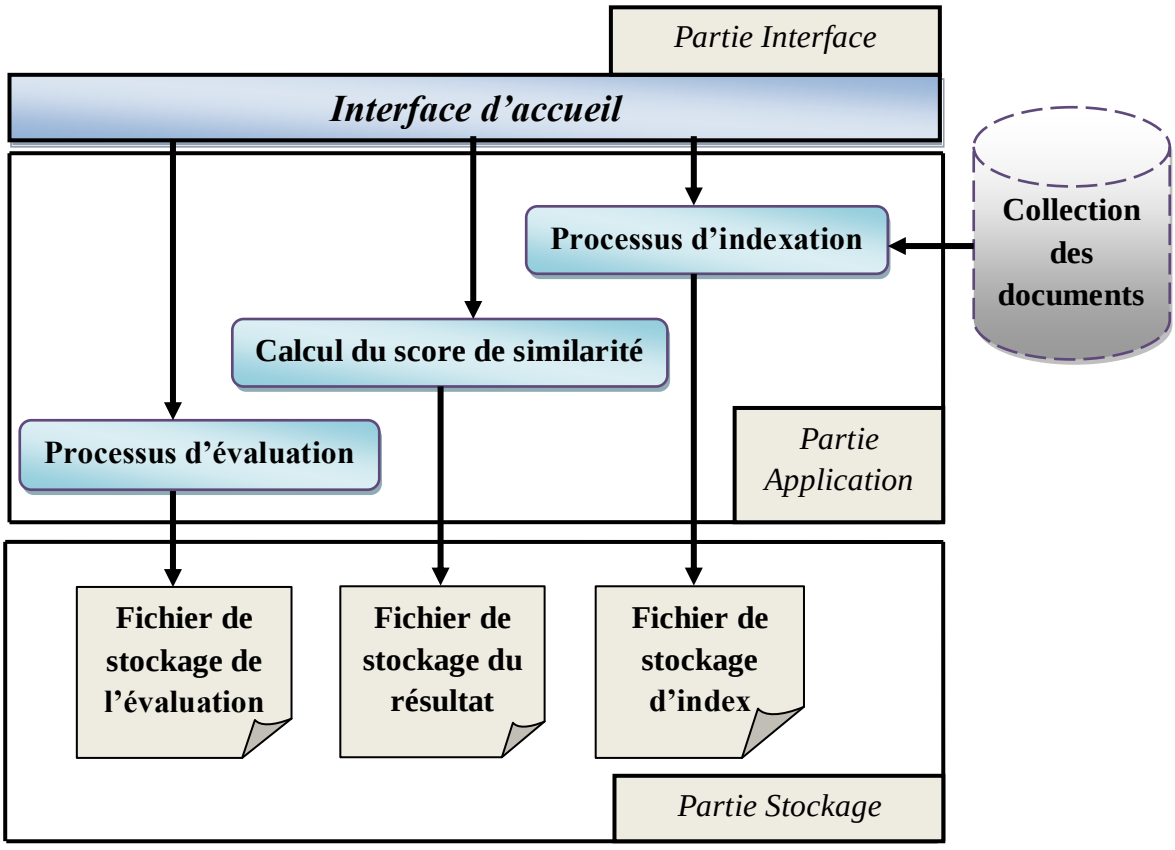


Figure III.4 : Architecture générale de l'approche

III.3.4 Réalisation de l'approche :

Pour réaliser notre approche, on commence, dans un premier temps, par intégrer les différentes fonctions utiles pour effectuer les processus d'indexation, de recherche et d'évaluation dans Terrier. A savoir la collection utilisée, le jugement de pertinence système, Terrier properties,…etc.

Puis, dans un second temps, on importera les classes compilées de Terrier et les différentes librairies externes utilisées par Terrier dans NetBeans. Quand aura programmé notre approche, on obtiendra une interface d'accueil qui sera décrite dans la section suivante.

III.3.4.1 L'interface d'accueil :

L'interface d'accueil de notre projet est présentée dans la figure suivante :

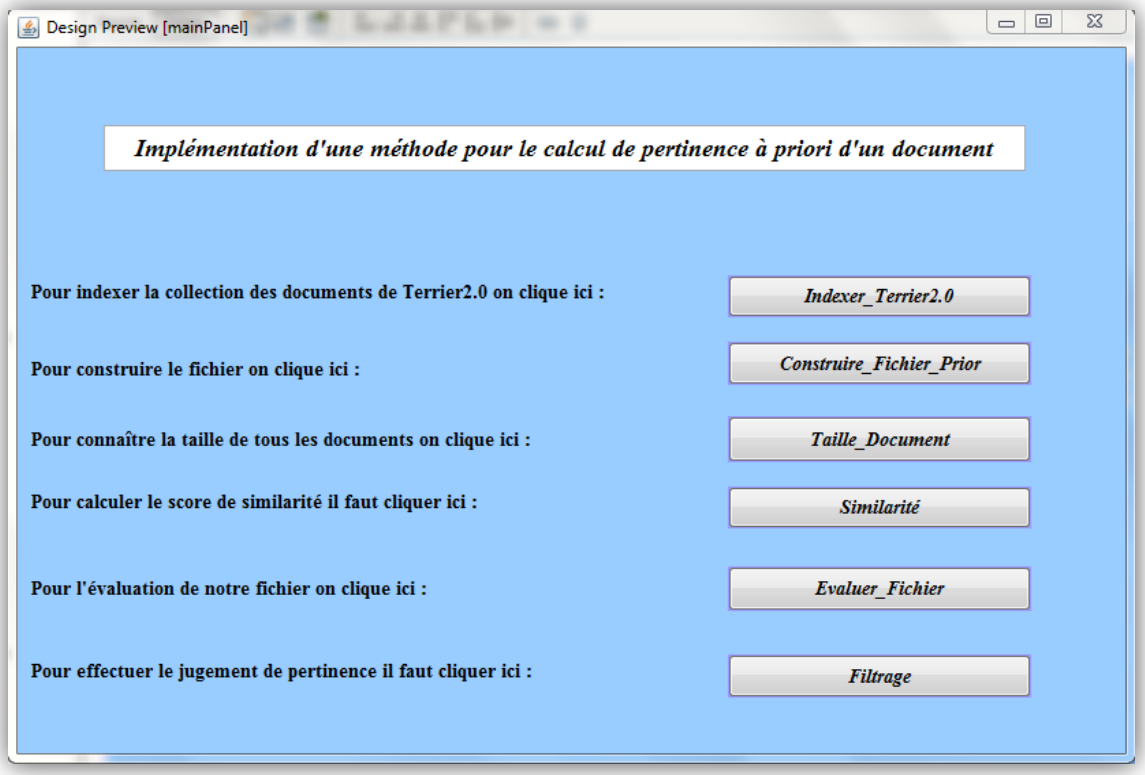


Figure III.5 : Interface d'accueil

La figure précédente montre les différentes fonctions utilisées pour réaliser notre approche :

- ✓ ***Indexer_Terrier2.0 :*** permet de parcourir toute la collection de documents (AP88 ou WSJ90-92) qui se trouve dans la version 2.0 de la plateforme Terrier, crée un index pour la collection et stocke la liste des index dans le fichier *C:\Master\similarity_wsj\terrier\var\index.*

- ✓ ***Construire_Fichier_Prior :*** permet de créer le fichier dans lequel sera stocké le résultat du calcul du score de similarité.

- ✓ ***Taille_Document :*** permet de nous retourner l'identifiant de chaque document avec sa taille dans la collection utilisée.

- ✓ ***Similarité :*** permet d'effectuer le calcul du score de similarité et stocke le résultat dans le fichier créé précédemment.

- ✓ ***Evaluer_Fichier :*** permet d'effectuer l'évaluation du fichier créé et stocke le résultat dans le dossier *C:\Master\similarity_wsj\terrier\var\results*.

- ✓ ***Filtrage :*** permet d'effectuer le jugement de pertinence de la recherche utilisateur par rapport à la recherche système.

Le résultat obtenu du calcul du score sera stocké dans un fichier qu'on nommera *sim.txt* en utilisant la collection de tests AP88 et *wsj.txt* en utilisant la collection de tests WSJ90-92.

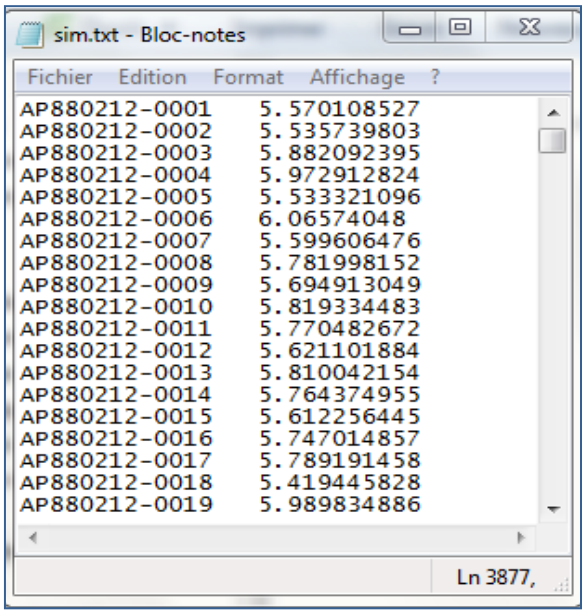


Figure III.6 : Extrait du fichier sim.txt

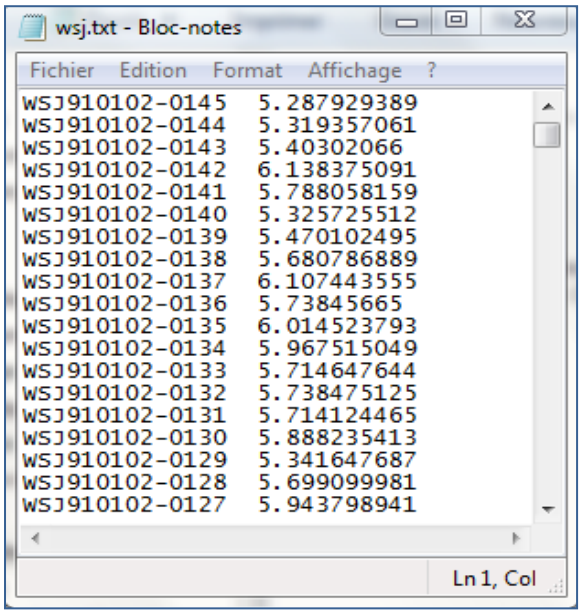


Figure III.7 : Extrait du fichier wsj.txt

L'évaluation de ces deux fichiers *sim.txt* et *wsj.txt* se déroule de la même manière et est présentée dans la section suivante.

III.3.4.2 Résultats d'évaluation :

a. Résultats d'évaluation sur la collection AP88 :

- **Evaluation du score de similarité :**

Avant d'évaluer le fichier *sim.txt*, il est nécessaire d'effectuer les trois étapes suivantes :

- ✓ On spécifie le fichier de jugement de pertinence dans le fichier *C:\Master\similarity_wsj\terrier\etc\trec.qrels.*

- ✓ On spécifie les fichiers qui contiennent les requêtes dans le fichier *C:\Master\similarity_wsj\terrier\etc\trec.topics.list*. Dans notre cas, les requêtes sont issues des topics numérotés par 251-300 de la collection TREC.
  Exemple :

```
<top>

<num> Number: 266
<title> Topic: Professional Scuba Diving

<desc> Description:
Where and for what purpose is scuba diving done
professionally?

<narr> Narrative:
What are some jobs that involve scuba (self-
contained
underwater breathing apparatus) diving?  Where do
scuba divers
work frequently?  Who hires scuba divers?

</top>
```

- ✓ On spécifie le modèle de pondération à utiliser dans le fichier *C:\Master\similarity_wsj\terrier\etc\trec.models*. Dans notre cas, on a utilisé le modèle de langage (Lissage de Dirichlet).

On peut maintenant évaluer notre fichier *sim*.txt. Pour cela, il suffit de cliquer sur le bouton *Evaluer_Fichier* dans l'interface d'accueil.

On obtient un fichier *.res* qui sera stocké dans le dossier *C:\Master\similarity_wsj\terrier\var\results*. On effectue l'évaluation par rapport à chaque valeur de $w \in [0.1, 1]$ qui représente un poids dans le score final (formule *(66)*). La moyenne des précisions moyennes est de 0.1030.

Les valeurs de la précision moyenne pour chaque valeur de *w* sont obtenues à l'aide de la commande :

> > *./bin/trec_terrier –e*

Les résultats de l'évaluation pour les différentes valeurs de *w* sont représentés dans le tableau suivant :

w	***Précision***
0.0	0.1030
0.1	0.1034
0.2	0.1033
0.3	0.1033
0.4	0.1031
0.5	0.1031
0.6	0.1030
0.7	**0.1035**
0.8	0.1030
0.9	0.1015
1	0.1020

Tableau III.2 : Résultats d'évaluation

Comme on peut le remarquer dans le tableau précédent, la méthode qu'on a utilisée améliore la précision des résultats obtenus pour les valeurs de $w = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.7\}$. La meilleure valeur de *w* qui donne une meilleure précision moyenne est $w=0.7$.

On remarque aussi que pour les valeurs de $w = \{0.6, 0.8, 0.9, 1\}$, notre méthode n'améliore pas la précision des résultats obtenus.

Pour approfondir notre analyse sur les résultats obtenus, le tableau III.3 suivant nous permet de comparer les précisions des requêtes 251-300 sans utiliser la probabilité à priori de pertinence (*w=0.0*) avec les précisions des requêtes 251-300 pour la valeur de *w=0.7* qui donne la meilleure précision moyenne en utilisant notre méthode à savoir *0.1035*.

Numéro de requête	**Précision**	
	w = 0.0	**w = 0.7**
251	0,0005	0,0005
252	0	0
253	1	1
254	0	0
255	0	0
256	0,0099	0,0095
257	0,05	0,0625
258	0	0
259	0	0
260	0,0833	0,1111
261	0,1111	0,1111
262	0	0
263	0	0
264	0,0156	0,0155
265	0,0169	0,0179
266	0,0417	0,0417
267	0	0
268	0	0
269	0,0134	0,0127
270	0	0
271	0,5	0,5
272	0	0
273	0,0649	0,0737
274	0	0
275	0	0
276	0	0
277	0,2159	0,2433
278	0	0
279	0	0
280	0	0
281	0	0
282	0,0659	0,0689
283	0	0
284	0,0119	0,0119
285	0,2807	0,2007
286	0	0
287	0	0

288	0,1667	0,1667
289	0	0
290	0	0
291	0	0
292	0	0
293	0	0
294	0,3333	0,3333
295	0	0
296	0	0
297	0,0065	0,0085
298	0	0
299	0	0
300	0	0

Tableau III.3 : Précisions des requêtes 251-300 pour w=0.0 et w=0.7

Pour avoir une vision plus claire de cette comparaison, les valeurs du tableau III.3 sont représentées dans les deux figures III.8 et III.9 suivantes :

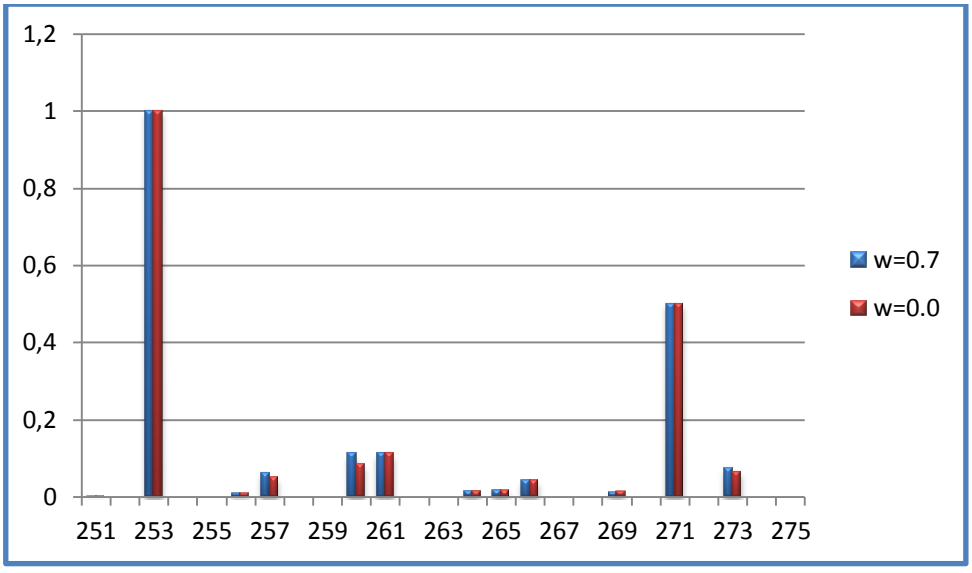


Figure III.8 : Précision des requêtes 251-275

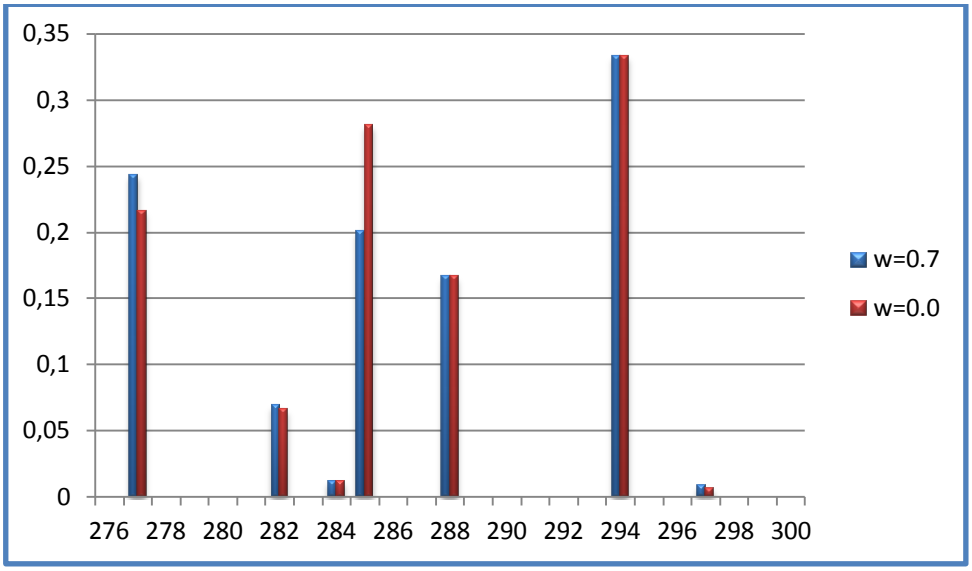


Figure III.9 : Précision des requêtes 276-300

On peut voir à partir des figures III.8 et III.9 précédentes que pour certaines requêtes telles que les requêtes 260, 265, 273, 277, 282 et 297 la précision a augmenté pour la valeur *w=0.7* par rapport à *w=0.0* contrairement aux requêtes 256, 264, 269 et 285 pour lesquelles elle a diminué ou encore les requêtes 251-255, 258, 259, 261-263, 266-268, 270-272, 274-276, 278-281, 283, 284, 286-296, 298-300 où la précision ne change pas.

L'augmentation de la précision pour les requêtes 260, 265, 273, 277, 282 et 297 nous permet de déduire que la méthode qu'on a utilisée est une bonne méthode pour avoir des résultats précis dans le domaine de la recherche.

Dans la section qui va suivre, on va calculer la taille de documents qu'on utilisera comme facteur de pertinence à priori pour comparer les résultats que retournera cette méthode avec les résultats qu'on a obtenu en utilisant notre méthode.

- **Evaluation en utilisant la taille des documents comme probabilité à priori de pertinence :**

Pour calculer la taille de chaque document de la collection utilisée, il suffit de cliquer sur le bouton *Taille_Document* dans l'interface d'accueil.

Les résultats de l'évaluation pour les différentes valeurs de *w* seront comme suit :

w	*Précision*
0.0	0.1030
0.1	**0,1004**
0.2	0,0993
0.3	0,0987
0.4	0,0983
0.5	0,0980
0.6	0,0980
0.7	0,0979
0.8	0,0976
0.9	0,0972
1	0,0968

Tableau III.4 : Résultats d'évaluation

Les résultats présentés dans le tableau III.4 ci-dessus nous montrent clairement que considérer la taille de documents comme un facteur de pertinence à priori n'est pas une méthode qui améliore la précision des résultats. Les valeurs de la précision moyenne, pour chaque valeur de *w*, sont toutes en dessous de la moyenne des précisions moyennes *w=0.1030*.

Pour approfondir notre analyse sur les résultats obtenus, on présentera dans le tableau III.5 les précisions des requêtes 251-300 sans utiliser la taille des documents comme facteur de pertinence à priori (*w=0.0)* et les précisions des requêtes 251-300 pour la valeur *w=0.1* qui donne la précision moyenne la plus importante des précisions à savoir 0.1004, qu'on va comparer par la suite.

Numéro de requête	**Précision**	
	w = 0.0	**w = 0.1**
251	0,0005	0,0005
252	0	0
253	1	1
254	0	0
255	0	0
256	0,0099	0,01
257	0,05	0,05
258	0	0

259	0	0
260	0,0833	0,0833
261	0,1111	0,1111
262	0	0
263	0	0
264	0,0156	0,0157
265	0,0169	0,0167
266	0,0417	0,0417
267	0	0
268	0	0
269	0,0134	0,0136
270	0	0
271	0,5	0,5
272	0	0
273	0,0649	0,0649
274	0	0
275	0	0
276	0	0
277	0,2159	0,2092
278	0	0
279	0	0
280	0	0
281	0	0
282	0,0659	0,0659
283	0	0
284	0,0119	0,0111
285	0,2807	0,2667
286	0	0
287	0	0
288	0,1667	0,1111
289	0	0
290	0	0
291	0	0
292	0	0
293	0	0
294	0,3333	0,3333
295	0	0
296	0	0
297	0,0065	0,006
298	0	0
299	0	0
300	0	0

Tableau III.5 : Précisions des requêtes 251-300 pour w=0.0 et w=0.1

Pour avoir une vision plus claire de cette comparaison, les valeurs du tableau III.5 seront présentées dans les deux figures III.10 et III.11 suivantes :

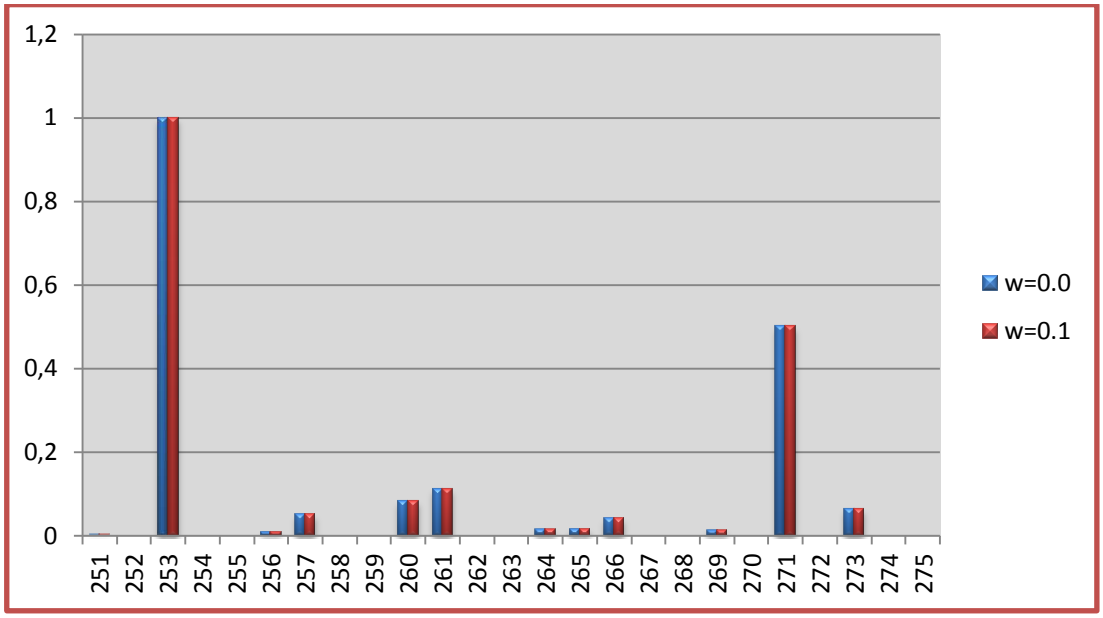


Figure III.10 : Précision des requêtes 251-275

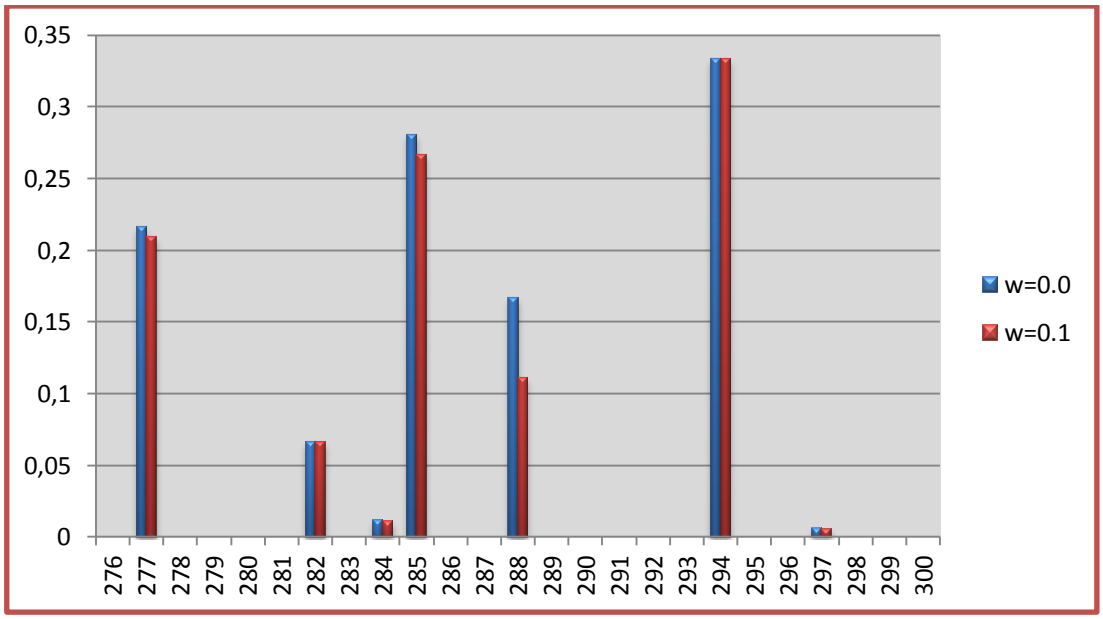


Figure III.11 : Précision des requêtes 276-300

A comparer les valeurs de précision des requêtes pour les valeurs de *w* dans les deux figures III.10 et III.11 précédentes, on remarque qu'il n'y a pas d'amélioration dans les résultats par rapport à la moyenne des précisions moyennes. Les valeurs de précision des requêtes pour *w=0.1* sont soit moindres, soit égales mais pas au dessus des valeurs de précision des requêtes pour *w=0.0*.

Sur ce dernier point, on peut déduire que « Taille des documents » ne peut être un facteur de pertinence à priori concernant la collection de tests AP88 car il n'améliore pas la précision des résultats dans la recherche.

b. Résultats d'évaluation sur la collection WSJ :

- **Evaluation du score de similarité :**

L'évaluation du fichier *wsj.txt* se déroule exactement de la même façon que celle du fichier *sim.txt* décrite précédemment.

Les résultats de l'évaluation pour les différentes valeurs de *w* sont représentés dans le tableau suivant :

w	***Précision***
0.0	0.1479
0.1	**0.1474**
0.2	0.1472
0.3	0.1465
0.4	0.1460
0.5	0.1433
0.6	0.1431
0.7	0.1432
0.8	0.1428
0.9	0.1425
1	0.1407

Tableau III.6 : Résultats d'évaluation

Les résultats présentés dans le tableau III.6 ci-dessus nous indiquent que la méthode qu'on a utilisée n'améliore pas la précision des résultats concernant la collection WSJ. Les valeurs de la précision moyenne, pour chaque valeur de *w*, sont toutes en dessous de la moyenne des précisions moyennes *w=0.1479*.

Pour approfondir notre analyse sur les résultats obtenus, le tableau III.7 présente les précisions des requêtes 251-300 sans utiliser la probabilité à priori de pertinence (*w=0.0*) et les précisions des requêtes 251-300 pour la valeur de *w=0.1* qui donne la précision moyenne la plus importante des précisions à savoir 0.1474, qu'on va comparer par la suite.

Numéro de requête	**Précision**	
	w = 0.0	**w = 0.1**
251	0,03	0,0251
252	0	0
253	0	0
254	0,2953	0,2954
255	0,0103	0,0104
256	0	0
257	0	0
258	0	0
259	0	0
260	0	0
261	0	0
262	0	0
263	0	0
264	0	0
265	0	0
266	0	0
267	0	0
268	0,006	0,0058
269	0	0
270	0	0
271	0	0
272	0,5	0,5
273	0	0
274	0,1112	0,1022
275	0	0
276	0	0
277	0	0
278	0	0
279	0	0
280	0,0141	0,0139
281	0	0
282	0,0909	0,0909
283	0	0
284	0	0
285	1	1
286	0,002	0,002
287	0	0

288	0	0
289	0,1126	0,1159
290	0,0147	0,0149
291	0,0226	0,0236
292	0,25	0,25
293	0,0364	0,0361
294	0,0492	0,0494
295	0,0159	0,0159
296	0	0
297	0,0588	0,0588
298	0	0
299	0	0
300	0,4856	0,4849

Tableau III.7 : Précisions des requêtes 251-300 pour w=0.0 et w=0.1

Pour avoir une vision plus claire de cette comparaison, les valeurs du tableau III.7 seront présentées dans les deux figures III.12 et III.13 suivantes :

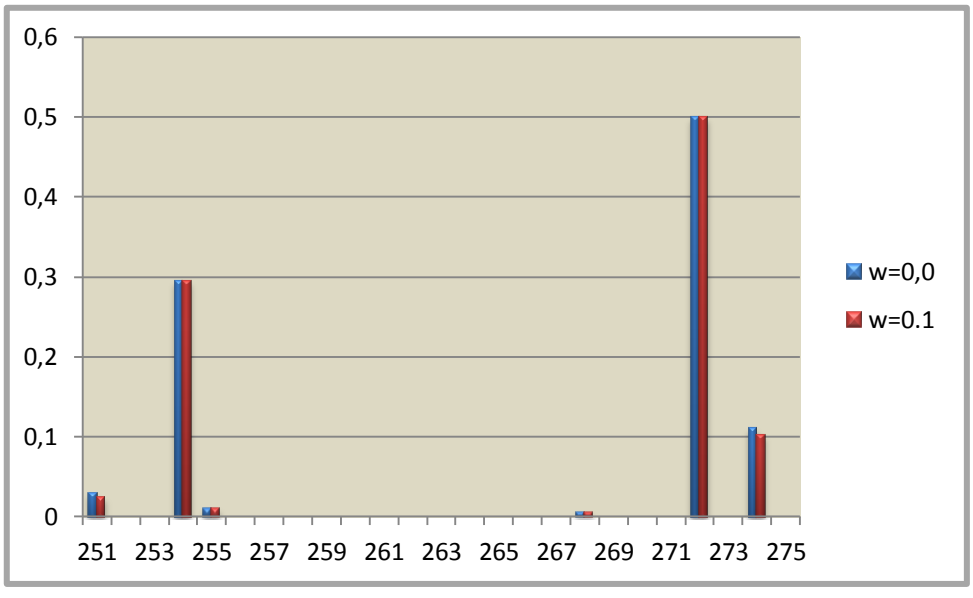


Figure III.12 : Précision des requêtes 251-275

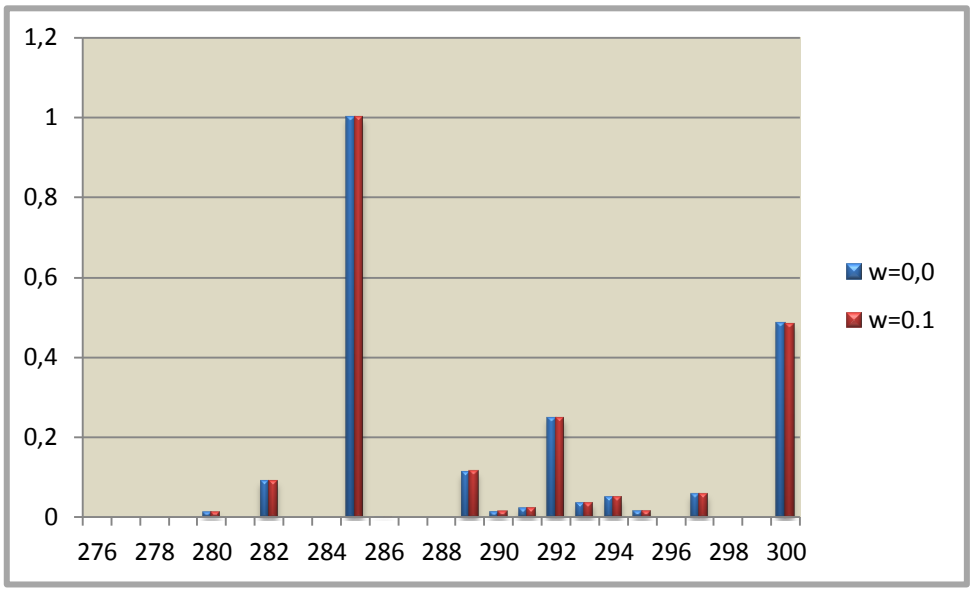


Figure III.13 : Précision des requêtes 276-300

D'après les résultats présentés dans les deux figures précédentes, on peut remarquer qu'il n'y a pas d'amélioration dans les résultats par rapport à la moyenne des précisions moyennes car pour certaines requêtes telles que 251, 268, 274, 280 la précision a diminué et pour d'autres requêtes telles que 254, 272, 285, 300, elle n'a pas changé.

Ceci nous amène à dire que la méthode utilisée n'est pas une bonne méthode pour le calcul de pertinence à priori concernant les documents de la collection WSJ.

- **Evaluation en utilisant la taille des documents comme probabilité à priori de pertinence :**

Comme pour la collection de tests AP88, on va calculer la taille des documents de la collection WSJ qu'on utilisera comme facteur de pertinence à priori pour comparer les résultats que retournera cette méthode avec les résultats qu'on a obtenus en utilisant notre méthode.

Les résultats de l'évaluation pour les différentes valeurs de *w* sont présentés dans le tableau suivant :

w	***Précision***
0.0	0.1479
0.1	0.1480
0.2	0.1475

0.3	0.1465
0.4	0.1465
0.5	0.1467
0.6	**0.1505**
0.7	0.1498
0.8	0.1427
0.9	0.1384
1	0.1383

Tableau III.8 : Résultats d'évaluation

Les valeurs de précisions représentées dans le tableau précédent démontrent qu'en utilisant la taille des documents comme facteur de pertinence à priori améliore la précision des résultats obtenus pour les valeurs de $w = \{0.1, 0.6, 0.7\}$. La meilleure valeur de w qui donne une meilleure précision moyenne est $w=0.6$.

On remarque aussi que pour les valeurs de $w = \{0.2, 0.3, 0.4, 0.5, 0.9, 0.8, 1\}$, cette méthode n'améliore pas la précision des résultats obtenus.

Pour approfondir notre analyse sur les résultats obtenus, on présentera dans le tableau III.9 les précisions des requêtes 251-300 sans utiliser la taille des documents comme facteur de pertinence à priori ($w=0.0$) et les précisions des requêtes 251-300 pour la valeur $w=0.6$ qui donne la meilleure des précisions moyennes à savoir 0.1505, qu'on va comparer par la suite.

Numéro de requête	**Précision**	
	w = 0.0	**w = 0.6**
251	0,03	0,0292
252	0	0
253	0	0
254	0,2953	0,2924
255	0,0103	0,0114
256	0	0
257	0	0
258	0	0
259	0	0
260	0	0
261	0	0
262	0	0
263	0	0

264	0	0
265	0	0
266	0	0
267	0	0
268	0,006	0,0071
269	0	0
270	0	0
271	0	0
272	0,5	0,5
273	0	0
274	0,1112	0,1273
275	0	0
276	0	0
277	0	0
278	0	0
279	0	0
280	0,0141	0,0137
281	0	0
282	0,0909	0,0909
283	0	0
284	0	0
285	1	1
286	0,002	0,002
287	0	0
288	0	0
289	0,1126	0,0999
290	0,0147	0,0139
291	0,0226	0,0194
292	0,25	0,3333
293	0,0364	0,0348
294	0,0492	0,0533
295	0,0159	0,0141
296	0	0
297	0,0588	0,0556
298	0	0
299	0	0
300	0,4856	0,4625

Tableau III.9 : Précisions des requêtes 251-300 pour w=0.0 et w=0.6

Pour avoir une vision plus claire de cette comparaison, les valeurs du tableau III.9 sont présentées dans les deux figures III.14 et III.15 suivantes :

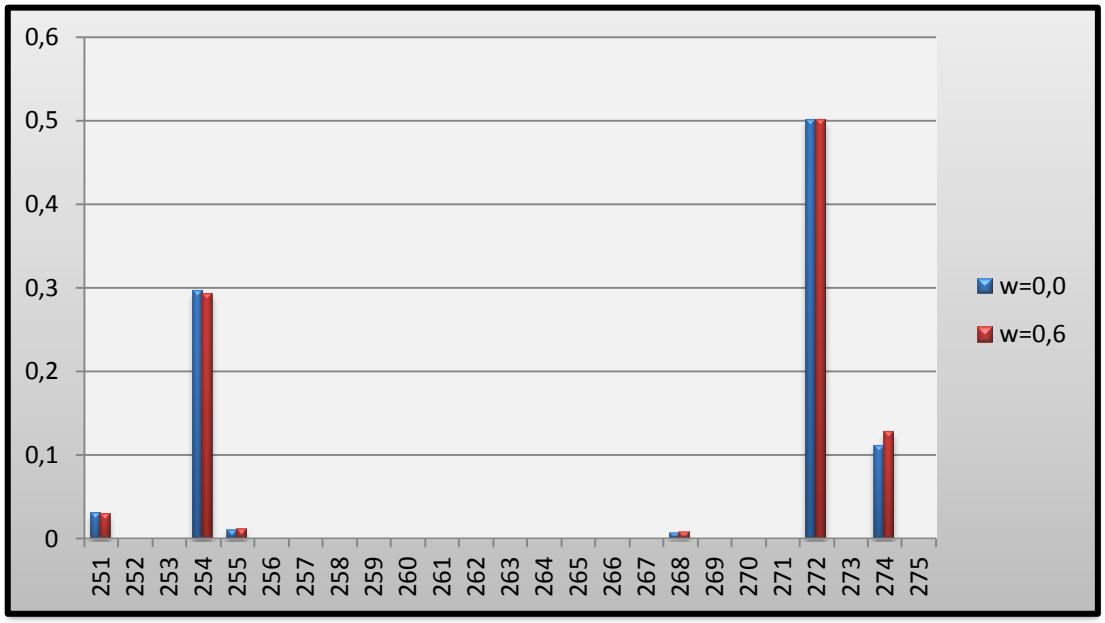


Figure III.14 : Précision des requêtes 251-275

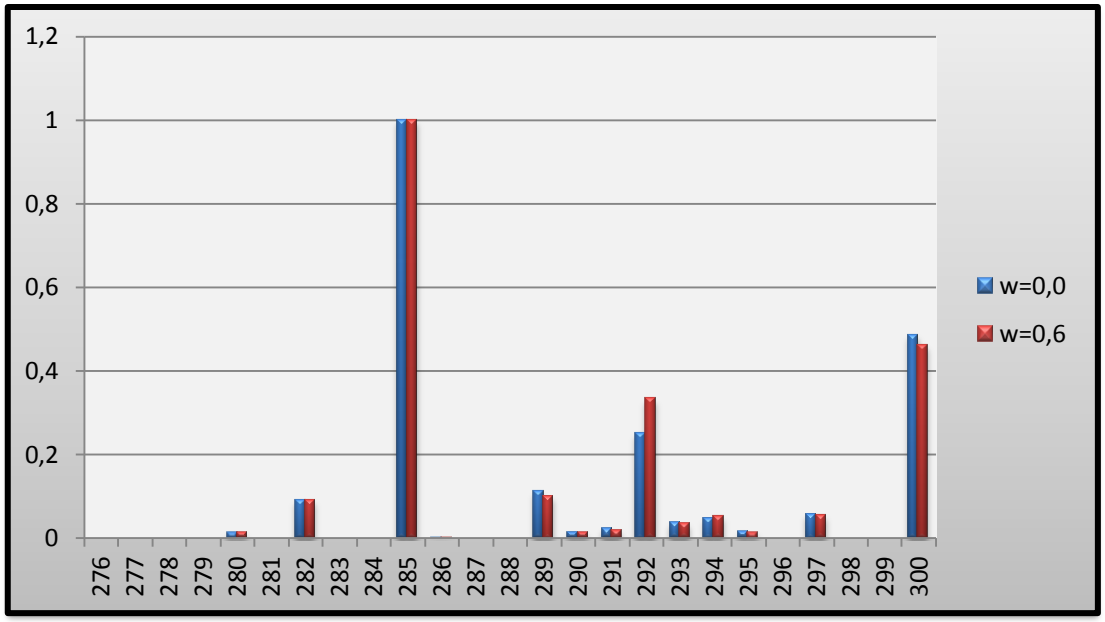


Figure III.15 : Précision des requêtes 276-300

A partir des figures III.14 et III.15 précédentes, on peut remarquer que pour certaines requêtes telles que les requêtes 255, 268, 274, 292 et 294 la précision a augmenté pour la valeur *w=0.6* par rapport à *w=0.0* contrairement aux requêtes 251, 254, 280, 289, 290, 291, 293, 295, 297 et 300 pour lesquelles elle a diminué ou encore les requêtes 252, 253, 256-267, 269-273, 275-279, 281-288, 296, 298 et 299 où la précision n'a pas changé .

L'augmentation de la précision pour les requêtes 255, 268, 274, 292 et 294 nous permet de déduire que l'utilisation de la taille des documents comme facteur de pertinence à priori concernant la collection des documents WSJ est une bonne méthode pour avoir des résultats précis dans le domaine de la recherche.

c. Comparaison entre les deux collections AP88 et WSJ90-92 :

A partir des résultats obtenus concernant les deux collections de tests AP88 et WSJ90-92, on a pu remarquer que la précision des résultats est meilleure en utilisant la méthode qu'on a proposée qu'en utilisant la taille des documents comme facteur de pertinence à priori pour la collection AP88. Alors que pour la collection WSJ90-92, c'est l'inverse.

On ne peut donc rien conclure. Afin de tirer des conclusions, il faudrait tester notre approche sur tous les documents des deux collections AP88 et WSJ et sur d'autres collections de tests.

III.4 Conclusion :

Dans cette dernière partie de notre projet, on a donné une description de notre approche et détaillé son implémentation en présentant le langage de programmation utilisé Java et l'environnement de développement NetBeans. On a également présenté la plateforme Terrier de part ses processus d'indexation et de recherche.

Pour finir, on a expliqué comment on a réalisé notre approche et donné les différents résultats de l'évaluation pour les deux collections de tests utilisées.

Conclusion générale

Conclusion générale

Bilan du travail réalisé

Dans ce travail, on a étudié le problème de pertinence des documents dans le domaine de la Recherche d'Information. On a proposé une méthode pour pallier à ce problème qui consiste à calculer la pertinence à priori d'un document par rapport à une collection de documents en se basant sur le principe de la dissemblance.

On a commencé notre travail par la présentation des différents concepts de base de la Recherche d'Information ainsi que ses principaux modèles dans le premier chapitre.

Ensuite, dans le second chapitre, on a présenté la modélisation de langage en expliquant ses différentes approches et la notion de calcul de pertinence de documents dans le domaine de la Recherche d'Information sur le Web.

Enfin, on a exposé notre approche ainsi que son implémentation et évaluation dans le troisième et dernier chapitre. Cela nous a permis de nous familiariser avec les différents outils utilisés pour implémenter notre approche notamment la plateforme de RI Terrier ainsi que l'environnement de développement NetBeans.

On a utilisé le score de dissemblance d'un document vis-à-vis d'une collection de documents pour calculer la probabilité à priori de pertinence et ce dans le cadre du modèle de langage. On a appliqué cette méthode sur deux collections de tests AP88 et WSJ et d'après les résultats obtenus, on n'a pas pu tirer des conclusions sur la résolution de notre méthode au problème exposé précédemment. Pour cela, il faudrait la tester sur l'ensemble des deux collections AP88 et WSJ et sur d'autres collections de tests.

Perspectives

Les perspectives qu'on peut tirer de notre travail sont :

- ✓ Tester la méthode de dissemblance qu'on a proposée sur d'autres types de collections et d'autres types de requêtes.

- ✓ Obtention de meilleurs résultats que ceux obtenus avec la méthode qu'on a proposée en se basant sur une autre méthode du calcul de score de similarité, par exemple la mesure de Jacard ou le coefficient de Dice.

ANNEXE

La plateforme de Recherche d'Information Terrier :

1. Définition :

Terrier (Terabyte Retriever) est un projet qui a été initié à l'Université de Glasgow en l'an 2000, dans le but de fournir une plate-forme flexible pour le développement rapide des applications à grande échelle de Recherche d'Information.

C'est un logiciel open source entièrement écrit en java. Il est utilisé ave succès pour la recherche Ad hoc, la recherche web et la recherche inter-langage dans des environnements centralisés et distribués.

2. Caractéristiques de Terrier :

La plateforme Terrier possède un certain nombre de caractéristiques comme suit :

- ✓ **Efficace :** Terrier peut indexer des corpus volumineux de documents, et fournit plusieurs stratégies d'indexation.

- ✓ **Flexible :** Terrier est idéal pour exécuter des expériences de recherche documentaire. Il peut indexer et exécuter une série d'expériences de recherche pour tous les tests des collections TREC.

- ✓ **Multilingue :** Terrier emploie le codage UTF intérieurement et peut supporter des corpus écrits dans des langues autres que l'anglais.

- ✓ **Extensible :** Terrier suit une architecture plugin et est facile à étendre pour développer de nouvelles techniques de recherche.

- ✓ **Interactif :** Les résultats de recherche vus dans une application de recherche de bureau sont à portée de main, ou en ligne à partir d'une interface Web JSP.

3. Installation de Terrier :

Pour pouvoir utiliser Terrier, il est nécessaire d'installer une JRE. La JRE ou la JDK peuvent être téléchargées sur le site de Java[1].

Après avoir téléchargé une copie de Terrier 2.0 sur la page d'accueil du projet Terrier[2], on crée un nouveau répertoire et dézippe Terrier dans ce dernier.

---

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.htm

[2] http://www.terrier.org

4. La structure des répertoires de Terrier :

Terrier contient un ensemble de répertoires qui 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 (stopwords list).
- ✓ **src** : contient le code source de Terrier.
- ✓ **var\index** : contient les structures de données d'index : inverted index, direct index, lexicon et document index.
- ✓ **var\results** : contient les résultats de la recherche.
- ✓ **licenses** : contient les informations sur la licence des différents composants inclus dans Terrier.

5. Le processus d'indexation dans Terrier :

La figure a ci-dessous décrit le procédé d'indexation dans Terrier. Terrier est conçu pour permettre beaucoup de différentes manières d'indexer un corpus de documents.
L'indexation est un processus à quatre étapes, et à chaque étape, des plugins peuvent être ajoutés pour changer le procédé d'indexation. Ces quatre étapes sont :

- ✓ Extraction de l'objet *Document* de la *Collection* qui est générée par le corpus reçu en entrée par Terrier.
- ✓ Parcours de chaque document de la collection pour en extraire les termes ainsi que les informations relatives.
- ✓ Traitement des termes extraits en utilisant le *Term Pipeline*.
- ✓ Construction de l'index via l'utilisation de la classe *Index Builders*.

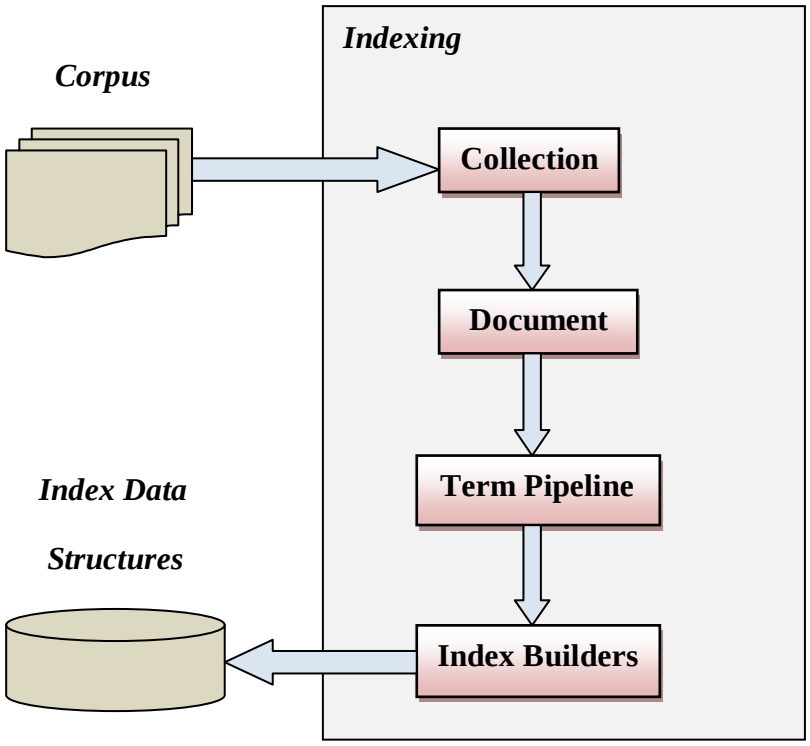


Figure a : Processus d'indexation dans Terrier

5.1 Collection :

Collection est la composante qui englobe le concept fondamental de l'indexation dans Terrier. 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*. Elle est utilisée par Terrier pour intégrer de nouvelles sources de données et cela en ajoutant une nouvelle classe java qui implémente cette interface. Dans le cas qu'on a traité, on a utilisé la classe TRECCollection qui permet de traiter les collections de type TREC (à savoir AP88 et WSJ9092).

Collection permet de parcourir un ensemble de documents et de renvoyer un objet Document en faisant appel à sa méthode *nextDocument( )*.

5.2 Document :

C'est une interface qui se trouve dans le package *uk.ac.gla.terrier.indexing*. Cette composante englobe le concept de Document. Les classes qui implémentent cette interface s'occupent de parcourir les documents et d'en extraire les différents termes.

Terrier fournit divers analyseurs syntaxiques de documents. Ceux-ci incluent la capacité d'indexer des documents HTML, des documents avec des textes simples, Microsoft Word, des documents Excel et Powerpoint,…etc. Pour ajouter un support d'un autre format de fichier à Terrier, le développeur doit seulement ajouter un autre document plugin qui peut extraire les termes à partir du document.

5.3 Term Pipeline :

C'est une interface qui se trouve dans la package *uk.ac.gla.terrier.terms*. Cette composante reçoit l'ensemble des termes extraits des documents.

Chaque terme extrait à partir d'un document possède trois propriétés fondamentales :

- ✓ La forme textuelle réelle du terme.
- ✓ La position à laquelle le terme survient dans le document.
- ✓ Les champs dans lesquels le terme survient (des champs peuvent être arbitrairement définis par le document plugin, mais sont typiquement employés pour définir que le terme survient dans des balises HTML).

Term Pipeline est considérée comme étant un pipeline qui traite et transforme chaque terme.

5.4 Index Builders :

Cette dernière composante est responsable de construire et d'écrire l'index dans la structure de données appropriée. Terrier offre deux types d'indexeurs : BasicIndexer et BlockIndexer.

5.5 Les structures d'index :

Un index de Terrier se compose de 4 structures de données principales :

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

Le tableau 1 suivant présente le contenu de ces différentes structures d'index.

Structure d'index	Contenu
Lexicon	- Term (le nom du terme). - Term id (l'identificateur du terme). - Document Frequency (la fréquence de document du terme). - Term Frequency (la fréquence globale du terme dans la collection). - Byte offset in inverted file (la position dans le fichier inverted file). - Bit offset in inverted file (la position dans le fichier inverted file).
Inverted Index	- Document id (l'identificateur du document correspondant). - Term Frequency (la fréquence globale du terme dans la collection). - Fields (# of fields bits) (les champs). - Block Frequency (la fréquence dans le bloc). - [Block id] (l'identificateur du bloc).
Document Index	- Document id (l'identificateur du document). - Document Length (la longueur du document). - Document Number (le numéro du document). - Byte offset in direct file (la position dans le fichier direct file). - Bit offset in direct file (la position dans le fichier direct file).
Direct Index	- Term id (l'identificateur du terme). - Term Frequency (la fréquence globale du terme dans la collection). - Fields (# of fields bits) (les champs). - Block Frequency (la fréquence dans le bloc). - [Block id] (l'identificateur du bloc).

Tableau 1 : Les structures d'index et leurs contenus

L'indexation dans Terrier peut être configurée en changeant les propriétés appropriées dans le fichier *etc\terrier.properties.* Chacune des étapes citées précédemment possède ses propres propriétés.

6. Le processus de recherche dans Terrier :

Un des objectifs principaux de Terrier est de faciliter la recherche dans le domaine de la Recherche d'Information. La figure b présente le procédé de recherche dans Terrier.

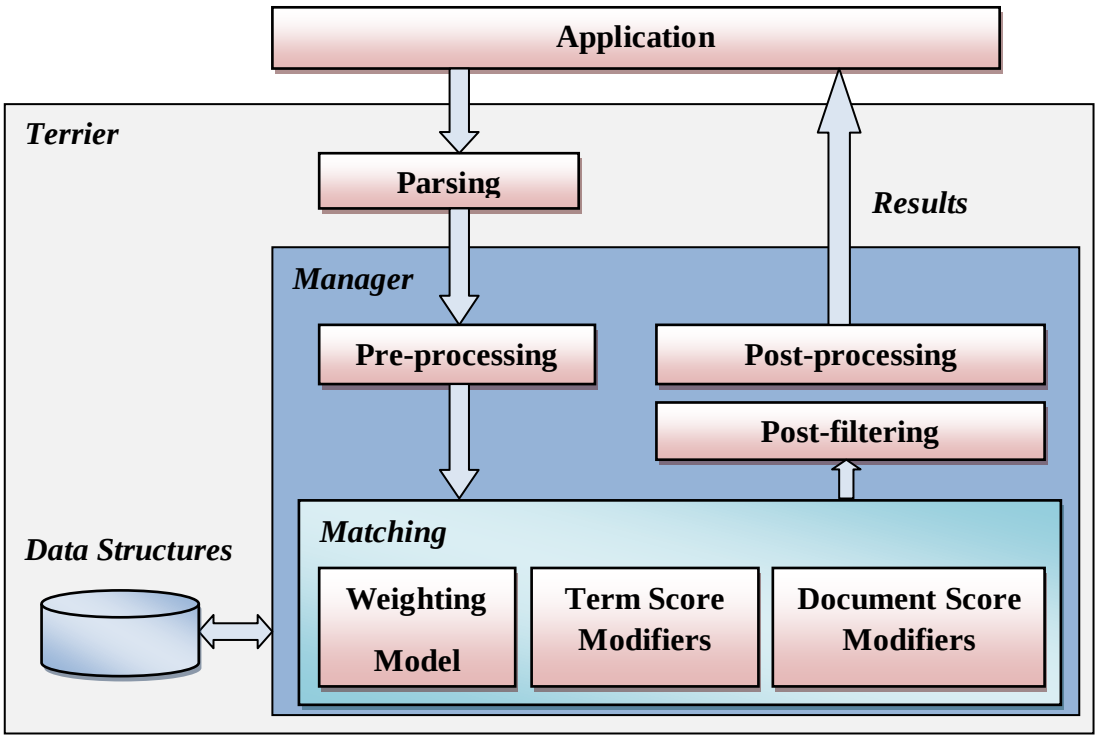


Figure b : Processus de recherche dans Terrier

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

6.1 Query :

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

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

Terrier offre un support pour différents types de requêtes : *SingleTermQuery*, *MultiTermQuery*, *FieldQuery*.

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 l'analyseur syntaxique approprié. Ensuite il crée un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaires et en spécifiant un modèle de pondération.

Puis il crée un fichier résultat *.res* où il associe à chaque requête les documents qui lui sont pertinents. Le résultat de chaque requête est donné par le second niveau de gestion.

Le second niveau de gestion est responsable de l'ordonnancement et de la coordination des opérations principales de haut niveau sur une seule requête. Ces opérations sont : Pre-processing, Matching, Post-processing et Post-filtering.

6.3 Matching :

Le composant Matching est responsable de déterminer qui des documents correspond à la requête spécifiée et donne un score au document qui vérifie la requête. Il utilise des modèles de pondération (*Weighting Models*) tels que TF-IDF, BM25, pour assigner un score à chaque mot de la requête dans le document. Terrier fournit un nombre important de modèles de pondération qui offrent des performances robustes.

- *DocumentScoreModifier* : modifie le score des documents en fonction de leurs propriétés.

- *TermScoreModifier* : modifie le score des documents en fonction de la position des termes.

7. Utilisation de Batch (TREC) Terrier :

7.1 Indexation :

1. Tout d'abord, on va dans le répertoire où Terrier est installé en utilisant la commande *cd* :

> > cd **terrier**

2. Ensuite, on initialise Terrier pour l'indexation d'une nouvelle collection TREC :

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

3. Enfin, on peut indexer la collection TREC comme suit :

> > **.\bin\trec_terrier –i**

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

7.2 Recherche et évaluation :

Avant d'effectuer toute recherche ou évaluation, il est nécessaire de réaliser les trois étapes suivantes :

1. Spécifier le fichier de jugement de pertinence dans le fichier *etc\trec.qrels* comme suit :

```
#add the qrels files to use for evaluation
Terrier.home\qe.qrels
```

Figure c : Exemple d'un fichier trec.qrels

2. Spécifier les fichiers contenant les requêtes (topic files) dans le fichier *etc\trec.topics.list* :

```
#add the topic files to use for querying
Terrier.home\etc\qe.topics.list
```

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

3. Spécifier le modèle de pondération (exemple : TF_IDF, BM25,…etc) à utiliser dans le fichier *etc\trec.models*.

```
#Add the weighting models to use for retrieval
#The possible values are the following classnames:
#uk.ac.gla.terrier.matching.models.BM25
#uk.ac.gla.terrier.matching.models.DFR_BM25
#uk.ac.gla.terrier.matching.models.TF_IDF
#uk.ac.gla.terrier.matching.models.BB2
#uk.ac.gla.terrier.matching.models.IFB2
#uk.ac.gla.terrier.matching.models.In_expB2
```

```
#uk.ac.gla.terrier.matching.models.In_expC2
#uk.ac.gla.terrier.matching.models.InL2
#uk.ac.gla.terrier.matching.models.PL2
#If you enter a not fully-qualified name of a
#class, then the default namespace
#uk.ac.gla.terrier.matching.models is prepended
#to the class name.
#uk.ac.gla.terrier.matching.models.InL2
```

Figure e : Exemple d'un fichier trec.models

Après que ces étapes soient effectuées, la recherche ou l'évaluation peuvent être lancées.

4. Pour lancer la recherche dans Terrier, il suffit d'exécuter la commande :

> > **.\bin\trec_terrier –r**

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

> > **.\bin\trec_terrier –e**

Liste des tableaux

Chapitre I :

Chapitre II :

Chapitre III :

Table des figures

Chapitre I :

Chapitre II :

Chapitre III :

Références Bibliographiques

Références Bibliographiques

[Allan et al, 1998] : J. Allan, J. Callan, M. Sanderson, J. Xu, S. Wegmann, *INQUERY at TREC-7*. Proceedings of TREC-7, pages : 201-216, 1998.

[Alvarez et al., 2003] : Alvarezc.,Langlaisp.J.Y- Nie. Word Pairs in Language Modeling for Information Retrieval. Rapport interne, RALI. (2003).

[K.Amrouche, 2008] : Karima AMROUCHE**,** Thèse de Doctorat en Informatique. Option : Système d'Information. Thème **:** Passage à l'échelle en Recherche d'Information : Méthode d'élagage pour la réduction de l'espace de recherche. Institut National de formation en Informatique (I.N.I), Oued-Smar Alger (2007/2008).

[Aussenac-Gilles et al., 2000] : Aussenac-Gilles N., Biébow B., Szulman N., Revisiting Ontology Design: a method based on corpus analysis. Knowledge engineering and knowledge management: methods, models and tools, Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Management. Juan-Les-Pins (F). Oct 2000. R Dieng and O. Corby (Eds). Lecture Notes in Artificial Intelligence Vol 1937. Berlin: Springer Verlag. pp. 172-188. 2000.

[Baccini et al., 2010] : Alain Baccini, Sébastien Déjean, Nongdo Désiré Kompaore, Josiane Mothe. *Analyze criteria of evaluation of the systems of search for information.* In: *Technology and Science Data processing*, Hermès Science Publications, 2010.

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

[Belew, 1989] : R.K.Belew, *Adaptative Information Retrieval : using a connectionist representation to retrieve and learn about document* In proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston (USA), pages : 11-20, 1989.

[Belkin et al., 1992] : N. J. Belkin, W. B. Croft, *Information retrieval and information filtering: Two sides of the same coin?*. CACM, pages : 29-38, 1992.

[Berger & Lafferty, 1999]: 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, 1992] : Boughanem M., "Systèmes de recherche d'informations : d'un modèle classique à un modèle connexionniste", Thèse de doctorat, Université Paul Sabatier, Toulouse III, 1992.

[Boughanem & al., 2004] : 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.

[Bourigault, 1996] : Bourigault D. (1996) : " Lexter, a Natural Language Processing Tool for Terminology Extraction ". Proceedings of Euralex'96, Göteborg University, Department of Swedish, 1996, pp. 771-779.

[Callan, Croft & Harding, 1992] : J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of DEXA 92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[Callan, 1996] : J. P. Callan, *Document filtering with inference networks*. Proceedings of ACM SIGIR 96, pages : 262-269, 1996.

[Carmel, 2000] : Carmel L., " Recherche d'information", Janvier 2000, http://www.esi.umontreal.ca/~carmellu/BLT6057/notes3.htm.

[Chakrabarti et al., 1999] : S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, et A. Tomkins. Hypersearching the web. Scientific American, 280 :54–60, June 1999.

[Y.Champclaux, 2009] : Yaël CHAMPCLAUX, Thèse de Doctorat en Informatique. Thème: Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information. Université Paul Sabatier Toulouse III (2009).

[Church et al., 1990] : K. Church and P. Hanks. Word association norms, mutual information and lexicography. In proceedings of the 28th Annual Meeting of the Association for Computational Linguistics. Pages 76-83. 1990.

[Cleverdon, 1962] : Cleverdon, C. W. *Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems.* Cranfield, U.K.: College of Aeronautics. Aslib Cranfield Research Project, 1962.

[Crestani, 1995]: Crestani F. *Implementation and evaluation of a relevance feedback device based on neural networks*. In J. Mira and J. Cabestany, editors, From Natural to Artificial neural Computation: International Workshop on Artificial Neural Networks, vol. 930 of Lecture Notes in Computer Science, p. 597–604. Springer-Verlag, Malaga, Spain, June 1995.

[Crestani et al, 1994] : F.Crestani, C.J Van Rijsbergen *Probability Kinematics in information retrieval* In proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, Seatle Washington, pages : 291- 299, 1994.

[Crestani & van Rijsbergen, 1998] : F. Crestani and C. J. van Rijsbergen. A study of probability kinematics in information retrieval. In *ACM Trans. Inf. Syst.*, volume 16(3), pages 225–255, 1998.

[Croft, 1991] : W. Bruce Croft: Editorial. ACM Trans. Inf. Syst. 9(3): 185 (1991).

[Daille, 1996] : B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. The balancing act combining symbolic and statistical approaches to language, MIT Press, Cambridge, Massachusetts pages 49-66, 1996.

[David et al., 1996] : David A. Evans and Chengxiang Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, pages 17--24, Santa Cruz, CA, 1996.

[Deerwester et al., 1990] : 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.

[Dumais, 1994] : S. Dumais : *Latent Semantic Indexing (LSI), TREC3 report.* In Proceedings of the 3rd Conference on Text Retrieval Conference, 1994.

[Dunning, 1994] : Ted Dunning. 1994. Statistical identification of language. Computing Research Laboratory Technical Memo MCCS 94-273, New Mexico State University, Las Cruces, New Mexico.

[Fagan, 1987] : Fagan, Joel L. 1987. Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-syntactic methods, PhD thesis, Dept. of Computer Science, Cornell University, Sept. 1987.

[Fox, 1983] : Fox E.A, "Extending the boolean and vector space models of information retrieval with P-norm queries and multiple concepts types", Phd thesis, Cornel University, Ithaca New York, 1983.

[Gaussier et al., 2003] : Eric Gaussier, Christian Jacquemin, & Pierre Zweigenbaum. *Traitement automatique des langues et recherche d'information*. In Éric Gaussier and Marie-Hélène Stefanini, editors, Assistance intelligente à la recherche d'informations, chapter 2, pages 71-96. Hermès-Lavoisier, Paris, 2003.

[Grossman & Frieder, 1998] : D. Grossman and O. Frieder. *Information Retrieval : Algorithms and Heuristics*. ISBN 0-7923-8271-4. Kluwer Academic Publishers, 1998. Second Edition 2004 by Springer Publishers.

[Guiraud, 1967] : Les structures étymologiques du lexique français, Larousse, Paris, 1967.

[Harbeck et al., 1999] : Stefan Harbeck, Uwe Ohler, Elmar Nöth, Heinrich Niemann: Information Theoretic Based Segments for Language Identification. TSD 1999: 187-192.

[Hiemstra, 1998]: Djoerd Hiemstra, A linguistically motivated probabilistic model of information retrieval, dans Christos N and Stephanides C. (eds), *Proc. European Conference of Digital Library (ECDL98),* Sept. 1998, Springer Verlag.

[Jacquemin, 2001] : Jacquemin C. (2001). Spotting and Discovering Terms through Natural Language Processing. Cambridge, Mass : MIT Press.

[Jones et al., 2002] : Steve Jones, Gordon W. Paynter: Human evaluation of Kea, an automatic keyphrasing system. JCDL 2001: 148-156.

[Kahle, 1997] : B. Kahle. Archiving the internet, extended version of the article "preserving the internet". *Scientific American*, March 1997.

[S.Karbasi, 2007] : Soheila KARBASI, Thèse de Doctorat en Informatique. Thème : Pondération des termes en Recherche d'Information: Modèle de pondération basé sur le rang des termes dans les documents. Université Paul Sabatier de Toulouse (2007).

[Khan, 2000] : Latifur R. Khan, *Ontology-based Information Selection, Phd Thesis,* Faculty of the Graduate School, University of Southern California. August 2000.

[Kleinberg, 1999] : Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. J. ACM, 46(5) :604–632, 1999.

[Kohonen, 1989]: T.Kohonen, *Self-Organisation and Associative Memory* 3rd Edition, Springer Verlag, Berlin, pages : 80-89, 1989.

[Kraaij, Westerveld & Hiemstra, 2002] : The importance of prior probabilities for entry page search. ACM SIGIR Conference on Research and Bibliography and Development in Information Retrieval. Tampere, Finland, (2002) 27–34.

[Kwok, 1989] : Kwok K., "A neural network for probabilistic information retrieval", In Proceedings of ACM SIGIR, pages 21–30, 1989.

[Kwok, 1995] : K.L. Kwok, *A network approach to probabilistic information retrieval*. ACM Transactions on Information Systems, pages : 324-353, 1995.

[Kwok et al, 1999] : K. L. Kwok, L. Grunfeld, M. Chan, *TREC-8 Adhoc, query and filtering track experiments using PIRCS*. Proceedings of TREC-8, pages : 129-137, 2000.

[Lafferty & Zhai, 2001]: Lafferty, J. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. *In Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111-119.

[Lauer, 1995] : Lauer, Mark. 1995. *Corpus statistics meet the noun compound: some empirical results*. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pp. 47-54.

[Lavrenko, 2001]: V. Lavrenko and W.B. Croft, Relevance-based Language Models, *Research and Development in Information Retrieval, Proc ACM-SIGIR'2001*, pp. 120-127, 2001.

[Lawrence & Giles, 1999] : S. Lawrence and L. Giles. Accessibility of information on the web. *Revue Nature*, 400 :107–109, 1999.

[Lempel et Moran, 2000] : R. Lempel et S. Moran. The stochastic approach for linkstructure analysis (SALSA) and the TKC effect. j-COMP-NET-AMSTERDAM, 33(1–6) :387–401, June 2000.

[Liu et al., 2004] : Liu, S., Liu, F., Yu, C., and Meng, W. 2004. An effective approach to document retrieval via utilizing WordNet and recognizing phrases. In Proceedings of the 27th Annual international Conference on Research and Development in information Retrieval (Sheffield, United Kingdom, July 25 - 29, 2004). SIGIR '04. ACM Press, New York, NY, 266-272.

[Luhn, 1955] : Luhn, H.P. *A new method of recording and searching information*. American Documentation vol. 4:1; p. 14-16; 1955.

[Mandelbrot, 1965] : Mandelbrot, Benoît. Information Theory and Psycholinguistics. in B.B. Wolman and E. Nagel. *Scientific psychology*. Basic Books. 1965.

[Maron & al., 1960] : Maron, M. E., & Kuhn, J. L.. **"**On relevance, probabilistic indexing, and information retrieval**"**. Journal of the Association for Computing Machinery, 7(3), 216-244, 1960.

[Miller, 1998]: David R.H. Miller, Tim Leek and Richard M. Schwartz, BBN at TREC-7: Using Hidden Markov Models for Information Retrieval, *TREC7*, pp. 133-142, 1998.

[Miller, Leek & Schwartz, 1999] : David R. H. Miller and Tim Leek and Richard M. Schwartz, A Hidden Markov Model Information Retrieval System, *Research and Development in Information Retrieval Proc. ACM-SIGIR*, pp. 214-221, 1999.

[Mitra et al., 1997] : Mitra, M., C. Buckley, A. Singhal, and C. Cardie, 1997. An analysis of statistical and syntactic phrases. The RIAO 97 Proceedings, pp. 200-214. Montreal.

[Mothe, 1994] : J. Mothe : *Modèle Connexioniste pour la Recherche d'informations, Expansion Dirigée de Requêtes et Apprentissage*, Thèse de l'Université Paul Sabatier Toulouse, 1994.

[N.Nassr, 2002] : Nawel NASSR, Thèse de Doctorat en Informatique. Thème : Croisement de langues en Recherche d'Information : traduction et désambiguïsation de requêtes. Université Paul Sabatier de Toulouse III (2002).

[Ng, 1999] : Kenney Ng, A Maximum Likelihood Ratio Information Retrieval Model, *TREC8*, pp. 483-492, 1999.

[Notess, 2000] : G.R. Notess. Search engine statistics: database overlap, 2000. SearchEngineShowdown, http : //www.searchengineshowdown.com/stats/overlap.shtml.

[Notess, 2002] : G.R. Notess. Search engine statistics :database total size estimates, 2002. SearchEngineShowdown, http : //www.searchengineshowdown.com/stats/sizeest.shtml.

[Oard, 1996] : S. Dumais : *Latent Semantic Indexing (LSI), TREC3 report.* In Proceedings of the 3rd Conference on Text Retrieval Conference, 1994.

[Page et al., 1998] : Lawrence Page, Sergey Brin, Rajeev Motwani, et Terry Winograd. The pagerank citation ranking : Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[**Piwowarski, 2003] :** Piwowarski B. « Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information », Thèse de doctorat de l'université de PARIS 6, 2003.

[Ponte & Croft, 1998]: 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.

[Rijsbergen, 1977] : C. J. van Rijsbergen (1977). A theoretical basis for the use of co-occurrence data in information retrieval. Journal of Documentation, 33: 106-119.

[Rijsbergen, 1979] : C. J. van Rijsbergen *Information Retrieval*. Butterworths, London, 1979.

[Rijsbergen, 1986] : C. J. van Rijsbergen, A Non-Classical Logic for Information Retrieval, *The Computer Journal*, 29(6): 481-485, 1986.

[Robertson & Sparck Jones, 1976] : S.E Robertson & K. Sparch Jones : *Relevance Weighting for Search Terms*, Journal of The American Society for Information Science, Vol 27, N°3, pp 129-146, 1976.

[Robertson, 1977] : Robertson, S. E. (1977). The probability ranking principle in IR. Journal of Documentation, 33 (4), 294-304.

[Rocchio, 1971] : Rocchio, J., *Relevance feedback in information retrieval*. In Salton, G., editor, The SMART Retrieval System: Experiments in Automatic Document Processing, p. 313-323. Prentice-Hall, Englewood Cli s, NJ, USA. 1971.

[Sabah et al., 2000] : Gérard Sabah et Brigitte Grau, Compréhension automatique de textes, 2000, chap. 13, pp. 293-307, Ingéniérie des langues, sous la direction de J.M.Pierrel, Hermes.

[Salton, 1971] : G. Salton. *The SMART retrieval system : experiments in automatic document processing*. Prentice Hall, 1971.

[Salton, Yang & Yu, 1975] : G. Salton, C. S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science and Technology*, 26(1) :33–44, 1975.

[Salton, 1983] : Salton, G., & McGill, M.. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.

[Salton, 1987] : Gerard Salton, Chris Buckley, *Term Weighting Approaches in Automatic Text Retrieval*, Cornell University, Ithaca, NY, 1987.

[Salton et al., 1989] : Gerard Salton, Maria Smith: On the Application of Syntactic Methodologies in Automatic Text Analysis. SIGIR 1989: 137-150.

[Salton et al., 1990] : Gerard Salton, Chris Buckley: *Improving retrieval performance by relevance feedback*. JASIS 41(4): p. 288-297, 1990.

[Schwartz, 1998] : C. Schwartz. Web search engines. *Journal of the American Society for Information Science*, 49 :973–982, 1998.

[Selberg, 1997] : Selberg E. *Information Retrieval Advances using Relevance Feedback. UW Dept. of CSE General Exam. 1997.*

[Seltzer, 1997] : R. Seltzer. "altavista, understanding the limits of accuracy", internet search advantage, Novembre 1997.

[Sheridan et al., 1992] : Paraic Sheridan, Alan F. Smeaton: The Application of Morpho-Syntactic Language Processing to Effective Phrase Matching. Inf. Process. Manage. 28(3): 349-370 (1992).

[Shivakumar & Garcia-Molina, 1998] : N. Shivakumar and H. Garcia-Molina. Finding near- replicas of documents on the web. In *International workshop on the web and databases (WebDB)*, Valencia, Spain, March 27-28, 1998.

[Smeaton, 1986] : Alan F. Smeaton: Incorporating Syntactic Information into a Document Retrieval Strategy: An Investigation. SIGIR 1986: 103-113.

[Sparck Jones, 1972] : Sparck Jones, K. *A statistical interpretation of term specificity and its application.* In Retrieval Journal of Documentation 28:1; p. 11-21, march 1972.

[Stanly & Goodman, 1998] : Stanley F. Chen & Joshua Goodman. "An Empirical Study of Smoothing Techniques for Language Modeling" Computer Science Group.Harvard University. Cambridge, Massachusetts.

[Strzalkowski et al., 1995] : Strzalkowski T. et al. Natural language information retrieval. TREC 3 report, Harman D. (ed.), Overview of the Third Text REtrieval Confrerence (TREC3). NIST special publication, 1995.

[L.Tamine, 2000] : Lynda TAMINE, Thèse de Doctorat en Informatique. Thème : Optimisation de requêtes dans un système de Recherche d'Information : Approche basée sur l'exploitation de techniques avancées de l'algorithmique génétique. Université Paul Sabatier de Toulouse (2000).

[Turpin et al., 1999] : Andrew Turpin, Alistair Moffat: Efficient Approximate Adaptive Coding. Data Compression Conference 1997: 357-366.

[Turtle et al, 1990] : H. Turtle, W. B. Croft, *Inference networks for document retrieval*. Proceedings of ACM SIGIR 90, pages : 1-24, 1990.

[Turtle, 1991] : Turtle H., "Inference Networks for Document Retrieval", Phd Thesis University of Massachusetts, 1991.

[Turtle et al., 1991]: Turtle, H. and W. Croft. *Evaluation of an inference network-based retrieval model*. ACM Transactions on Information Systems 9(3), p. 187–222, 1991.

[Turtle et al, 1992] : H. R. Turtle et W. B. Croft, *Evaluation of an inference network-based retrieval model*. ACM Transactions in Information Systems 3, pages : 187-222, 1992.

[Voorhees & Tice, 2000] : E.M. Voorhees and D.M. Tice. The trec-8 question answering track evaluation. In *Proceedings of the 8th Text Retrieval Conference*, pages 83–105. Spec Pub 500-246, Washington DC : NIST, 2000.

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

[Yang, 1999] : Yiming Yang *An evaluation of statistical approaches to text categorization*. Information Retrieval, vol. 1(1-2): p. 69-90, 1999.

[Zadeh, 1965] : L.A. Zadeh, "Fuzzy sets", Information and control, p338-353, 1965.

[Zhai, 2002]: ChengXiang Zhai and John Lafferty, Two-Stage Language Models for Information Retrieval. Proc. *Research and Development in Information Retrieval, ACM-SIGIR,* pp. 49-56, 2002.

[Zhu & Gauch, 2000] : Incorporating quality metrics in centralized / distributed information retrieval on the World Wide Web. The 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, (2000) 288–295.

[Zipf, 1949] : G. Zipf. Human Behaviour and the Principle of Least Effort. Addison-Wesley, 1949.