

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE.
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE.

UNIVERSITÉ MOULOUD MAMMERI DE TIZI OUZOU.
FACULTÉ DE GÉNIE ELECTRIQUE ET DE L'INFORMATIQUE.
DÉPARTEMENT D'INFORMATIQUE.



Mémoire de fin d'étude

En vue d'obtention du diplôme de MASTER en informatique

Spécialité : Systèmes Informatiques (SI).

THEME :

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

Présenté par :

M^{elle} KADRI Ahlam.

Dirigé par :

M^r A. HAMMACHE.

Promotion 2011/2012.

Remercîment

Je remercie d'abord le bon dieu qui m'a aidé et qui m'a donné le courage et la volonté pour réaliser ce modeste travail.

Mes remerciements, les plus vifs, ma profonde gratitude et mes respects s'adressent à mon Promoteur Mr. Hammache Arezki pour avoir accepté de m'encadrer, pour les conseils et orientations tant précieux qu'il m'a prodigué durant ce travail.

Que les membres du jury trouvent ici mes plus sincères remerciements pour avoir accepté d'honorer par leur jugement notre travail.

Mes remerciements les plus chaleureux s'adressent à ma famille et surtout ma mère qui est la source de cette réussite et qui m'a soutenue et encouragé pour aller au bout de ce travail.

Et enfin, je réserve ici une place particulière pour remercier vivement toute ma section 2ème année Master, et tout ceux qui par un sourire, un geste ou une parole m'ont encouragé à mener à bien ce projet.



Dédicace

Je préserve tout ce travail au nom de notre dieu miséricordieux et de son prophète MOHAMED « sala alaho alih wa salam ».

Je dédie ce Modeste travail à la mémoire de mon père, puisse tu être fière de ta fille .

À Celle qui est la source de mon inspiration et de mon courage, à qui je dois de l'amour et de la reconnaissance, ma très chère mère.

A Mes chers frères « Ziad », « Mahdi » et « Rabah » que j'aime tellement.

A mes neveux et nièces : « Abdenour », « bilel », « samia » et « lyasmine »

A tous mes oncles et tantes, cousins et cousines maternels et paternels ainsi qu'à mes belles sœurs.

A mes copines de coeur : Safia, kahina, hayet, fadhila, amel, lila, lydia, linda, souad et karima.

Un grand merci a tout mes ami(e)s et tous ceux qui me connaissent.

Ah!am.



Sommaire

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

Chapitre 1 : la recherche d'information (RI)

I.1. introduction.....	2
I.2.Principe de la recherche d'information.....	2
I.3.Architecture générale d'un Système de Recherche d'Information (SRI).....	3
I.3.1. Processus d'indexation.....	4
I.3.1.1. Langage d'indexation.....	4
I.3.1.2. Les étapes de l'indexation automatique.....	5
I.3.2 Pertinence et appariement document-requête.....	13
I.3.3 Reformulation des requêtes	14
I.4. Les modèles de Recherche d'Information.....	14
I.4.1 Les modèles basés sur la théorie des ensembles.....	16
I.4.1.1 Le modèle booléen	16
I.4.1.2 Le modèle Booléen étendu	17
I.4.1.3 Le modèle booléen basé sur des ensembles flous	18
I.4.2 Les modèles algébriques	18
I.4.2.1 le modèle vectoriel	18
I.4.2.2 Le modèle connexionniste	21
I.4.2.3 Latent Semantic Indexing (LSI)	23
I.4.3. Les modèles probabilistes	26
I.4.3.1.Le modèle probabiliste.....	26
I.4.3.2 Réseau de Document (Document Network).....	27
I.4.3.3 Le modèle de langages.....	29
I.5. Evaluation d'un SRI	29
I.5.1 Mesures d'évaluation	29
I.5.1.1 Rappel et précision	29

I.5.1.2 la précision moyenne	33
I.5.1.3 Les mesures combinées	34
I.6. Collections de test	35
I.7.Conclusion.....	36

Chapitre 2 : La recherche d'information sur le web

II.1. Introduction	37
II .2. Les Concepts de base du web	37
II.3. Documents structurés	39
II.4. Le web et les documents structurés	41
II.4.1. Structure hiérarchique interne des pages web	41
II.4.2. Structure hiérarchique interne des sites web	41
II.5. Hypertextes	41
II.6. Recherche d'information sur internet	42
II.7. Types de recherche d'information sur le web	43
II.7.1. Recherche par Navigation	43
II.7.2. Recherche par interrogation	43
II.8. Les outils de la recherche d'information	43
II.8.1. les moteurs de recherche	44
II.8.3. L'annuaire	45
II.8.3. les méta- moteurs	46
II.9. Algorithme des moteurs de recherche d'information	47
II.9.1. Algorithme Hyperlink-Induced Topic Search (HITS)	48
II.9.2. Algorithme PageRank	49
II.9.3. Algorithme SALSA	50
II.11. La Modélisation de Langue en Recherche d'Information	50

II.12. RI comme génération de la requête par le document	52
II.12.1. Modèle de Ponte et Croft	52
II.12.2. Modèle de Hiemstra et al. Et de Miller et al.	53
II.13. Pertinence a priori d'un document	55
II.14. Conclusion	57

Chapitre 3 : Evaluations et expérimentations

III.1. Introduction	58
III.2. Approche à implémenté	58
III.3. Architecture Logicielle	59
III.3.1. Description des données utilisées	60
1) La collection .GOV	60
2) Les fichiers décrivant la structure des liens de la collection .GOV	61
III.4. L'environnement technique de développement	63
III.4.1. Présentation du matériel	63
III.4.2. L'environnement Logiciel	63
1) Présentation de la plateforme Terrier 3.0	64
2) Le langage de programmation utilisé (java)	67
3) Les principales raisons du succès de java	67
4) Présentation de NetBeans	67
5) Présentation PhpMyAdmin	69
III.5. Etapes d'implémentation de notre approche	70
III.6. Conclusion.....	83
Conclusion générale.....	84

Annexes

Annexe 1.....85

Annexe 2.....96

Bibliographie

Références bibliographiques.....105

Table des figures

Chapitre 1 : La recherche d'information (RI)

Figure 1: processus en U de la RI (... 4)

Figure 2 : Pouvoir discriminatoire des termes d'indexation (...8)

Figure 3 : Structure de fichier Inverse (... 13)

Figure 4 : Taxonomie des Modèles de Recherche d'Information (...15)

Figure 5 : Représentation des documents et requêtes dans un espace vectoriel (...19)

Figure 6: Un réseau de neurones à couches (... 22)

Figure 7: Calcul d'une valeur à l'intérieur d'un nœud du réseau de neurone (...23)

Figure 8: Représentation d'un espace vectoriel classique (1) et d'un espace Réduit LSI (2) (...25)

Figure 9 : Représentation d'un réseau d'inférence (...28)

Figure 10: Représentation schématique des zones de précision et de rappel (...30)

Figure 11 : Allure d'une courbe de rappel-précision (...31)

Figure 12 : Représentation des points de rappel-précision (...33)

Chapitre 2 : La recherche d'information sur le web

Figure 1 : Structure logique et structure physique d'un document (...40)

Figure 2. Structure hiérarchique d'une page Web (...41)

Figure 3 : Recension et repérage d'information sur le Web(...47)

Figure 4. Répartitions des pages en bon Hubs, bonnes autorités et pages indépendantes (...48)

Figure 5 : Modèle de Markov caché à deux états (...55)

Chapitre 3 : expérimentations et implémentations

Figure 1 : Schéma Architecture Logicielle (...60)

Figure 2 : Extrait du Fichier Links_id (...62)

Figure 3 : Exemple du fichier Url2id (...63)

Figure 4 : Présentation du processus d'indexation de Terrier(...65)

Figure 5 : Présentation du processus de recherche de Terrier (...66)

Figure 6 : environnement de développement NetBeans (...68)

Figure 7. Accueil de PhpMyadmin (...69)

Figure 8 : créer une table sous PhpMyadmin (...70)

Figure 9 : remplissage des champs de la table (...70)

Figure 10 : Schéma de L'application (...71)

Figure 11 : Programme 1 d'exécution Java (....73)

Figure 12 : extrait de la table de la base de données (....74)

Figure 13 : Exemple de fichier Prior (...75)

Figure 14 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (1) (...76)

Figure 15 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (2) (...76)

Figure 16 : Extrait de la table de la base de données (...78)

Figure 17 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (3) (...79)

Figure 18 : partie de la table de la base de données (..80)

Figure 19 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (4) (..81)

Figure 20: extrait de la table (....82)

Figure 21 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (5) (..83)

ANNEXE 1 : La plateforme Terrier

Figure 1. Présentation de l'interface Desktop Terrier (...87)

Figure 2. Présentation de l'interface Web de Terrier (..87)

Figure 3. Présentation du processus d'indexation de Terrier (..88)

Figure 4. Présentation du processus d'indexation dans Terrier (....89)

Figure 5 : Présentation du processus de recherche de Terrier (...92)

Liste des tableaux

Chapitre 1 : la recherche d'information (RI)

Tableau 1 : Signature des termes (...12)

Tableau 2 : Signature des documents (..12)

Tableau 3 : Exemple de valeur rappel-précision (...32)

Chapitre 2 : la recherche d'information sur le web

Tableau 1 : Comparaison entre annuaires et moteurs de recherche (..46)

Chapitre 3 : expérimentation et implémentations

Tableau 1 : les composants de la collection .GOV (...61)

ANNEXE 1 : La plateforme Terrier

Tableau.1: présentation des différentes structures Index (..91)

Tableau.2 : Comparaison des plates formes de RI : Lemure, Luence, Terrier (95)

ANNEXE 2 : les tableaux d'évaluations

Tableau 1 : Résultats d'évaluation de la formule (3)(..97)

Tableau 2 : résultats d'évaluation de la formule (4)(..97)

Tableau 3 : résultats d'évaluation de la formule (5) (..99)

Tableau 4 : résultats d'évaluation de la formule (6)(...102)

Tableau 5 : notation des résultats d'évaluation de la formule (7) (..104)

Introduction Générale :

La plupart des systèmes de recherche d'information (SRI) privilégient la minimisation de temps de réponses par apport à la qualité du document retournés à l'utilisateur.

En effet ces derniers délivrent des résultats massifs en réponses aux requêtes des utilisateurs, ce qui génère ainsi une surcharge informationnelle dans laquelle il est difficile de distinguer l'information pertinente de l'information secondaire ou même de bruit.

L'une des raisons qui a engendré ceci est la non prise en compte de toutes les caractéristiques d'un document dans le processus d'indexation et de recherche. En effet les SRI qui implémentent les techniques traditionnelles de la RI considèrent un document comme un ensemble de termes. Pour calculer la pertinence d'un document vis-à-vis d'une requête, des modèles de RI sont proposés, parmi ces modèles, le modèle de langage.

Le modèle langage est basé sur la théorie des probabilités, pour la description du processus de la RI, il a montré des résultats meilleurs que les modèles classiques et parmi ses propriétés : il permet la combinaison de plusieurs sources d'information parmi ces sources on peut citer la structure document, la sémantique (contenu) de document et la structure des liens.

Dans le cadre de notre travail nous combinons deux sources d'information le contenu et la structure des liens, et cela en utilisant le modèle de langage. Plus explicitement nous nous intéresserons à l'intégration de la pertinence a priori d'une page web tout en considérant les propriétés du site au quel elle appartient.

Nous avons testé notre approche sur une collection TREC la collection .GOV de taille 20GO. Pour l'implémentation et l'évaluation de l'approche nous avons utilisé la plateforme Terrier.

Nous avons organisé notre mémoire en trois chapitres :

- **Premier chapitre** : présente l'état de l'art de la recherche d'information classique.
- **Deuxième chapitre** : Décrit la recherche d'information sur le web, concepts, caractéristiques et utilisation, ainsi que la description des différentes caractéristiques utilisées pour estimer la probabilité a priori de pertinence d'un document.
- **Troisième chapitre** : comprend la présentation de l'approche a implémenté, puis l'implémentation de cette approche en exposant les résultats d'expérimentation obtenus.

On finalisera par une conclusion générale qui englobera nos perspectives futures.

I.1 Introduction

La difficulté d'évaluer l'information désirée a augmenté avec la croissance du volume des données disponibles. Dans le but d'exploiter au maximum ces informations mises aux services des utilisateurs tout en évitant la contrainte de surcharge d'information, un processus nommé Recherche d'information est alors engagé.

Ce dernier est réalisé par un système qui permet de retrouver les documents pertinents relatifs à une requête d'utilisateur, à partir d'une base de données volumineuse.

L'objectif de ce chapitre est de présenter les concepts de base de la RI. Dans la première section, nous décrivons le processus global de la RI, communément connu sous le nom du processus en U [Belkin, 92]. Nous donnerons l'utilité et l'importance des opérations qui composent ce processus.

Nous décrivons par la suite trois modèles connus de la RI, à savoir le modèle booléen, le modèle vectoriel et le modèle probabiliste. Nous présenterons ensuite les techniques utilisées pour l'évaluation des systèmes de recherche d'information.

I.2. Principe de la recherche d'information

Abrégée en RI (Recherche d'Information) ou IR (Information Retrieval), la recherche d'information est un domaine historiquement lié aux Sciences de l'information et à la bibliothéconomie qui ont toujours eu le souci d'établir des représentations des documents dans le but d'en récupérer des informations, à travers la construction d'index. Le terme de « recherche d'information » a été utilisé pour la première fois dans les années cinquante par Kelvin N. Mooers quand il travaillait sur son mémoire de maîtrise [Mooers, 48].

Un système de recherche d'information (RI) est un ensemble de programmes informatiques ayant pour but de satisfaire le besoin en information d'une requête utilisateur. Le rôle principal est donc de sélectionner les informations ou les documents les plus pertinents correspondant à ce besoin, et cela à partir d'une base de documents volumineuse.

Dans cette définition, il y a trois notions clés : documents, requête, pertinence.

- Un document peut être un texte, un morceau de texte, une page web, une image, etc. On appelle document toute unité qui peut constituer une réponse à une requête d'utilisateur.
- Une requête exprime le besoin d'information d'un utilisateur.
- La pertinence est une notion très importante qui détermine la correspondance ou le degré de relation entre le document et la requête.

La RI concerne donc, les mécanismes qui facilitent l'accès à une base d'informations. Il existe un grand nombre de modèles de recherche d'information, et ces modèles diffèrent principalement sur la façon dont les informations disponibles sont représentées, et sur la façon de représentation du besoin de l'utilisateur. La recherche d'information s'intéresse à la représentation, le stockage, l'organisation, l'acquisition, la recherche et la sélection d'information [Salton et al., 83].

I.3. Architecture générale d'un Système de Recherche d'Information (SRI)

Pour répondre aux besoins en information de l'utilisateur, un SRI met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenues dans un fond documentaire d'une part, et des besoins en information des utilisateurs d'autre part. Ces processus supposent que la collection de documents est unique. Un système de recherche d'information intègre trois fonctions principales représentées schématiquement par le processus en U de recherche d'information. La Figure 1 illustre l'architecture générale d'un système de recherche d'information. D'un côté, on a l'information accessible dans le système. Elle est en général le résultat de collecte de documents ou de sous collections de documents traitant divers domaines. D'un autre côté, on a le besoin en information exprimé par l'utilisateur, en général sous forme de requête, une fois stabilisé. Ensuite, l'information aussi bien que le besoin en information passe par des étapes de traitement pour être exploitables. Ces processus s'appuient sur un certain nombre de modèles permettant de sélectionner des informations pertinentes en réponses à une requête utilisateur. Il s'agit principalement du processus de représentation et du processus de recherche :

- **Processus de représentation** : un processus de représentation a pour rôle d'extraire d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux son contenu sémantique. Ce processus de conversion est appelé *indexation*. Le résultat de l'indexation constitue le *descripteur* du document ou de la requête, qui est une liste de *termes* significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids pour différencier leur degré de représentativité. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée *dictionnaire d'index* constituant le *langage d'indexation*.

- **Processus de recherche** : il représente le processus du noyau d'un SRI. Il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer. Il est utilisé pour la recherche d'informations proprement dite et est étroitement lié au modèle de représentation des documents et des requêtes. Ces modèles de recherche représentent ce qui diffère le plus entre les SRI. Ils sont inspirés de concepts mathématiques afin de pouvoir évaluer certaines relations, notamment la relation d'appariement entre la requête et les documents. La problématique majeure des SRI est de retrouver les quelques dizaines ou milliers de documents pertinents parmi des millions de documents. Cet écart de cardinalité rend cette tâche encore plus difficile.

En plus des étapes de représentation et de recherche, quelques systèmes peuvent supporter une étape supplémentaire de *reformulation de requêtes*. Cette étape a pour objectif d'améliorer les performances du SRI, donc la précision dans les réponses du système. Une des méthodes utilisées pour la reformulation de requêtes est la *réinjection de pertinence*, qui ajoute des termes issus des premiers documents restitués (supposés pertinents) pour améliorer la requête de l'utilisateur.

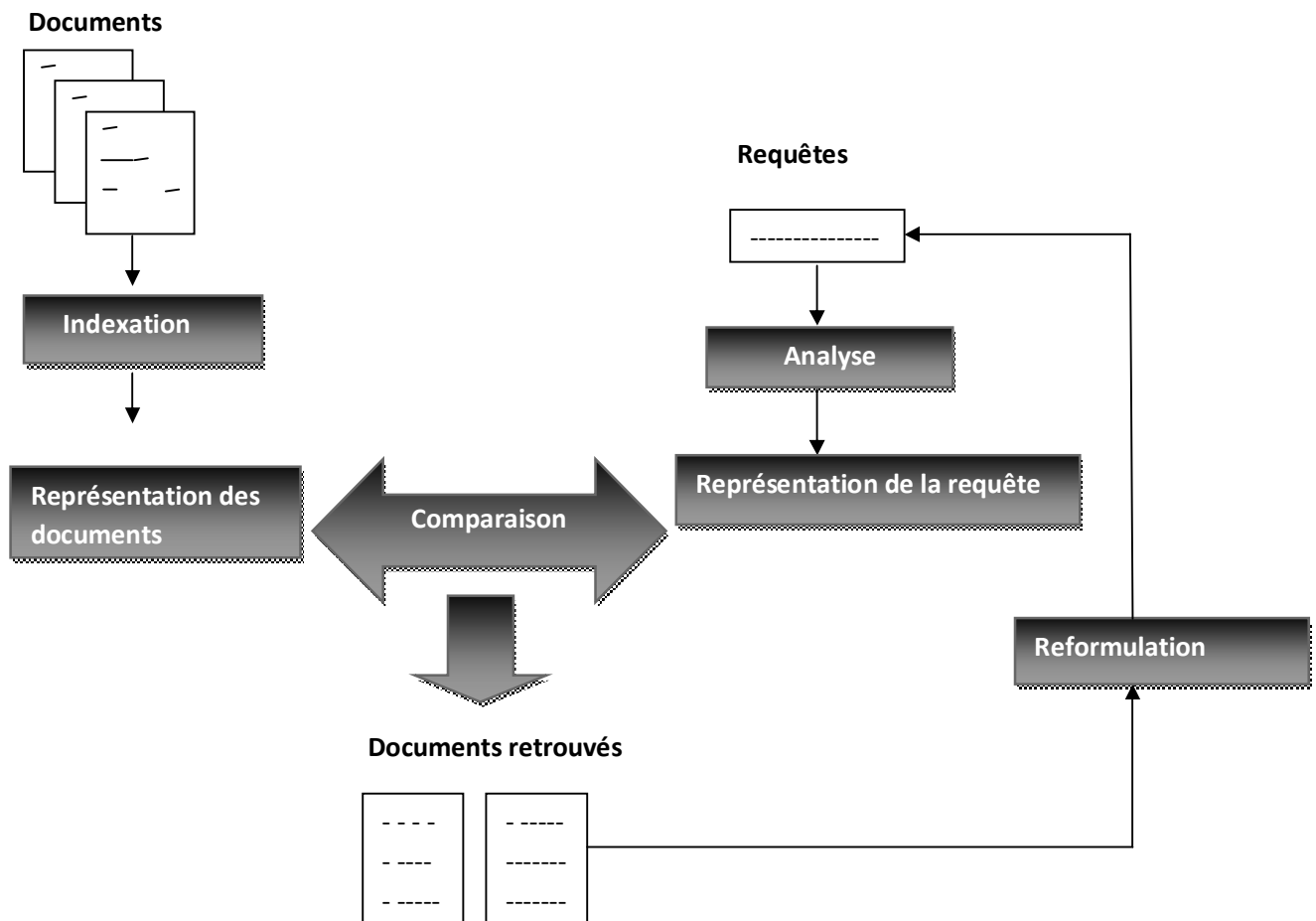


Figure 1 : processus en U de la RI.

I.3.1. Processus d'indexation

D'après Marie Gaëlle MONTEIL [MONTEIL, 95] l'indexation est l'opération qui vise à construire une structure d'index qui permet de retrouver très rapidement les documents incluant des mots demandés.

Cette étape consiste à analyser chaque document de la collection afin de créer un ensemble de mots-clés. Son objectif est de trouver les concepts les plus importants du document (ou de la requête), qui formeront le descripteur du document.

I.3.1.1. Langage d'indexation :

L'ensemble de tous les termes d'indexation constitue le langage d'indexation. Le langage d'indexation exprime le contenu sémantique des documents. Il doit offrir un compromis entre la capacité de la représentation, pour qu'elle puisse être traitée efficacement par un système informatique, et l'expressivité afin d'exprimer aussi fidèlement que possible le contenu des documents. Le langage d'indexation peut être libre ou contrôlé :

- **Langage d'indexation libre** : Ce langage est construit à partir des termes extraits de documents analysés.
- **Langage d'indexation contrôlé** : Ce langage est construit à partir d'un ensemble de termes préalablement définis et organisés généralement dans un thésaurus¹. Lorsqu'un document est analysé, on ne garde dans sa présentation que les mots clés appartenant à ce thésaurus.

L'indexation peut être [Salton et al 88] :

-**Manuelle** : Chaque document est analysé par un spécialiste du domaine ou par un documentaliste, qui se charge de recenser selon ses connaissances propres, les concepts dont traite un document et de les présenter à l'aide d'un langage libre ou contrôlé.

L'indexation manuelle à l'avantage d'assurer une meilleure correspondance entre les documents et les termes d'indexation choisis. Ceci a pour conséquence une meilleure précision dans les documents que le SRI retourne en réponses aux requêtes des utilisateurs.

L'inconvénient majeur de cette méthode est l'effort intellectuel qu'elle exige (en nombre 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 arrivé qu'une même personne, à des moments différents, indexe différemment le même document.

-**automatique** : L'indexation automatique consiste à simuler par une machine cette opération d'indexation, que ce soit dans sa méthode ou surtout dans ses résultats. Elle repose sur un ensemble des méthodes automatisées sur un document comme l'extraction automatique des mots des documents, l'élimination des mots vides, la lemmatisation (radicalisation), la pondération des termes et la création de l'index.

-**Semi-automatique**: Le choix final revient au spécialiste ou au documentaliste, qui intervient souvent pour établir des relations sémantiques entre mots-clés et choisir les termes significatifs (Synonymes....etc) à partir du thésaurus ou d'ontologie. Cette méthode est une combinaison des deux méthodes précédentes.

I.3.1.2. Les étapes de l'indexation automatique

Etape 1 : identification des termes d'indexations :

a. Extraction des termes d'index :

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

b. 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...) et choisir seulement les termes significatifs qui représentent au mieux un document donné. Afin d'éliminer ces mots, 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.

Etape 2 : Normalisation des termes d'indexation

Ce traitement consiste à envoyer pour un mot sa forme normalisée. Ainsi, dans l'index ne sont conservées que les formes normalisées des mots (généralement le masculin pour les noms, l'infinitif pour les verbes, le masculin singulier pour les adjectifs, ..). Ce qui offre un gain de place appréciable, mais surtout, une recherche efficace.

Le traitement associé se repose sur deux procédures :

1. La troncature :

Utilisée pour éliminer les variations morphologiques ou les variantes orthographiques des mots clés de l'index. Elle consiste à couper le mot à partir d'un rang précis, afin d'obtenir son radical. Pour la langue française, la troncature à sept caractères est adoptée.

2. 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.

Des expériences ont montré que la troncature et la lemmatisation améliorent significativement les performances pour les langues riches morphologiquement (ex. le français, l'italien, ..)[Gaussier et al., 97][Gaussier et al., 00].

Etape 3 : La pondération des termes

La pondération est l'une des fonctions fondamentales en RI. Elle est la clé de voute de la majorité des modèles et approches de RI proposés depuis les années 1960. L'idée derrière la pondération des termes d'indexation est d'affecter à chaque terme d'un document un poids, traduisant son importance dans le document, donc son degré d'informativité.

Si certaines méthodes proposent d'introduire des éléments linguistiques dans l'indexation des documents, la grande majorité des approches et systèmes opérationnels, se base sur les aspects statistiques. Ces méthodes tirent leur origine de la loi de Zipf et de la conjecture de Luhn.

ü *Loi de Zipf*

La loi de Zipf est une loi empirique énoncée en 1949 par G.K Zipf [Zipf, 1949]. Elle décrit la répartition statistique des fréquences d'apparition des différents éléments d'un ensemble, comme les mots d'un texte. Concernant le texte, selon Zipf, les mots dans les documents ne s'organisent pas de manière aléatoire mais suivant une loi inversement proportionnelle à leur rang. Le rang d'un mot est sa position dans la liste décroissante des fréquences des mots du corpus. Ainsi, la fréquence du second mot le plus fréquent dans le corpus est la moitié de celle du premier, la fréquence du troisième mot le plus fréquent, son tiers, etc. Formellement, cette loi s'exprime de la manière suivante :

$$\text{Fréquence} \times \text{Rang} \approx \text{Constante}$$

Cette égalité est "approximative" car à l'origine, la fonction de Zipf est exponentielle, et est indépendante des locuteurs, des types de textes et des langues.

Un autre concept a été introduit pour déterminer les mots qui représentent au mieux le contenu d'un document, il s'agit de la conjecture de Luhn.

ü *La conjecture de Luhn*

La conjecture de Luhn [luhn, 58] émet une hypothèse sur l'information contenue dans les termes (informativité) d'un document. L'informativité mesure la quantité de sens et le pouvoir d'expression des mots dans un texte en fonction de leur fréquence. Elle considère que les descripteurs non pertinents sont les descripteurs de rangs faibles (très fréquents), car ce sont des mots qui reviennent souvent, ils n'ont pas de pouvoir discriminant. Par exemple, dans un corpus de documents informatiques, le mot ordinateur est très fréquent et ne va pas permettre de différencier les documents entre eux. La conjecture de Luhn considère aussi ceux de rangs élevés (très rares), comme peu pertinents, en effet ces mots sont rares et donc peu utilisés. Les descripteurs qui sont pertinents sont les descripteurs de rangs intermédiaires. Cette conjecture est utilisée pour diminuer la taille des index des documents. Deux seuils de fréquence (seuil max et seuil min), sont fixés pour éliminer les termes dont le contenu informatif est jugé faible. Seuls les termes entre ces deux seuils sont alors pertinents pour représenter les documents. La correspondance entre l'informativité c'est-à-dire l'importance d'un terme ou le pouvoir expressif et la fréquence d'apparition dans un document est illustrée dans la Figure suivante :

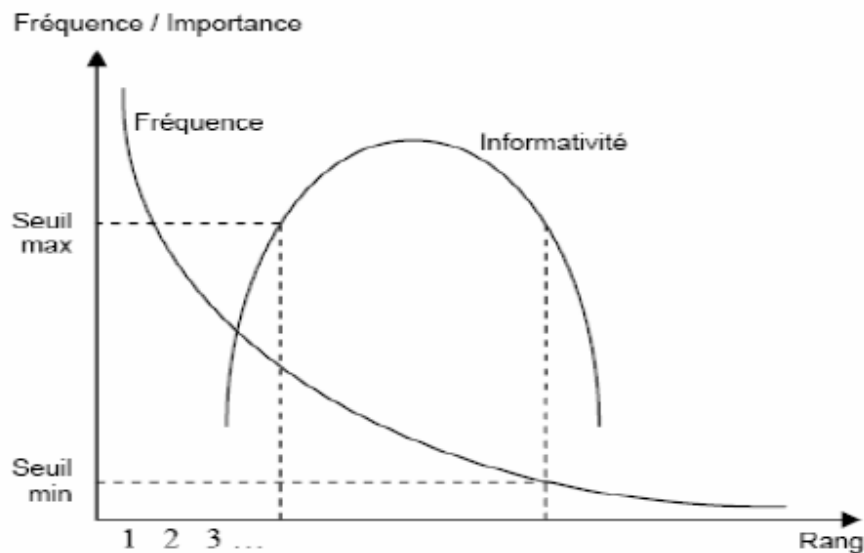


Figure 2 : Pouvoir discriminatoire des termes d'indexation [luhn, 58]

Ainsi, en choisissant les mots qui ont des fréquences entre les deux seuils, on espère obtenir les mots dont l'informativité est la plus élevée.

Dans un processus d'indexation, l'élément fondamental est la technique utilisée pour pondérer les termes [Robertson et al., 97],[Singhal et al., 97] ,[SparkJone, 71] ,[SparkJones, 79]. Le poids associé à un terme peut être soit sa fréquence d'occurrence (tf) ou bien une mesure dérivant de cette fréquence (fréquence normalisée par exemple), ou également le facteur $tf \times idf$ qui est mieux adapté et utilisé.

1. Approches basées sur la fréquence locale (tf) :

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

- *fonction brute de tf_{ij} (term frequency)* : correspond au nombre d'occurrences du terme t_i dans le document D_j ;
- *fonction binaire* : elle vaut 1 si la fréquence d'occurrence du terme dans le document est supérieure ou égale à 1, et 0 sinon;
- *fonction logarithmique* : combine tf_{ij} avec un logarithme, elle est donnée par : $a + \log(tf_{ij})$,
où a : est une constante.

Cette fonction a pour but d'atténuer les effets de larges différences entre les fréquences d'occurrence des termes dans le document.

- *fonction normalisée* : permet de réduire les différences entre les valeurs associées aux termes du document. Elle est donnée par la formule suivante :

$$0,5 + 0,5 \frac{tf_{ij}}{\max_{ti \in D_j}(tf_{ij})}$$

$\max_{ti \in D_j}(tf_{ij})$ La plus grande valeur de tf_{ij} des termes du document D_j .

2. Approches basées sur la fréquence globale (idf) :

La pondération d'un terme par discrimination reflète qu'un terme distingue bien un document des autres documents. C'est-à-dire, un terme qui a une valeur de discrimination élevée doit apparaître seulement dans un petit nombre de documents : un terme qui apparaît dans tous les documents n'est pas discriminant. La capacité de discrimination d'un terme est importante dans le choix des termes d'indexation à retenir.

L'idée est de garder seulement les termes discriminants, et éliminer ceux qui ne le sont pas. Les approches basées sur tf ainsi que ceux qui sont basées sur idf sont efficaces, par le choix des termes discriminants. La pondération est néanmoins communément basée sur la combinaison des deux mesures. Les raisons pour lesquelles ces deux paramètres ont l'intérêt de coexister est qu'ils sont très révélateurs de l'importance d'un terme, et aussi pour se livrer de l'obligation d'identifier des seuils et de migrer vers l'identification de l'importance d'un terme par une mesure plus statistique.

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

Ou

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

Ou, n_i est le nombre de documents contenant le terme t_i et N le nombre total de documents dans la collection. Du fait de cette double pondération (locale et globale), ces fonctions de pondération sont souvent référencées sous le nom de *TF*IDF*.

3. Approches basées sur $tf \times idf$:

La mesure *TF*IDF* est une bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles homogènes. Cette mesure a eu en revanche un succès très limité dans les corpus de tailles très variables. Dès TREC2, de nouvelles investigations ont été effectuées pour améliorer cette fonction de pondération [Declaris et al., 94]. Le problème posé est que les termes appartenant aux documents longs apparaissent très

fréquemment et l'emportent en poids sur les termes appartenant à des documents moins longs. Les documents longs auront alors plus de chance d'être sélectionnés. Les distorsions engendrées par cette hétérogénéité sont corrigées en effectuant une normalisation [Singhal et al., 95] [Robertson et al., 97]. Robertson et Sparck-Jones proposent de normaliser la fonction de pondération de la façon suivante :

$$w_{ij} = \frac{tf_{ij} \cdot (k_1 + 1)}{K_1((1-b) + (b \cdot \frac{dl_i}{\Delta l})) + tf_{ij}}$$

Où

w_{ij} est le poids du terme t_i dans le document D_j , K_1 contrôle l'influence de la fréquence du terme t_i dans le document D_j , sa valeur optimale dépend de la longueur et de l'hétérogénéité des documents dans la collection de documents, b est une constante appartenant à l'intervalle [0, 1] et contrôle l'effet de la longueur du document, dl_j est la longueur du document D_j , et Δl est la longueur moyenne des documents dans la collection entière.

Une autre fonction de pondération normalisée utilisée dans le système Inquiry [Callan et al., 92] est donnée comme suit :

$$wd_{ij} = 0.4 + 0.6 \frac{tf_{ij}}{tf_{ij} + 0.5 + 1.5 \frac{dl_{ij}}{\Delta l}} \cdot \frac{\log \frac{N+0.5}{n_i}}{\log(N+1)}$$

Le plus souvent, un système d'indexation avec un vocabulaire contrôlé travaille à partir d'une base de connaissances. Cette base de connaissances (la plupart du temps contextuelle) lui permet de "choisir" les descripteurs les plus appropriés en fonction du document analysé. Le choix du type de descripteur utilisé est primordial, et sera déterminant pour les performances de l'indexation.

Etape 4: créations de l'index

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants : les fichiers inverses ("inverted files"), les tableaux de suffixes ("suffix arrays") et les fichiers de signatures ("signature files").

Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications et sont utilisés dans la plupart des systèmes commerciaux. Dans cette structure, chaque document peut être représenté par une liste de mots clés qui décrivent le contenu du document pour la recherche (fichiers directs). Une recherche rapide, peut être réalisée si nous inversons ces mots

clés. Une remarque à noter est que les documents ne sont pas les seuls à être indexés : les requêtes sont également perçues comme des listes de mots-clés.

Ø Structure des fichiers index

Un élément important dans le domaine de la recherche d'information est la manière dont l'index doit être organisé (stocké) pour répondre efficacement et rapidement aux besoins d'information des utilisateurs. En effet, les structures d'index déterminent la méthode d'indexation et agissent directement sur les performances du système de recherche d'information. Nous nous baserons sur [Rijsbergen, 1979] pour présenter les plus importantes.

a. Fichier Séquentiel

Un fichier séquentiel est le moyen le plus simple de stocker un fichier de données puisque l'on stocke les enregistrements les uns à la suite des autres dans leur ordre d'insertion. Jusqu'au milieu des années 70, tous les systèmes textuels utilisaient les fichiers séquentiels du fait de l'utilisation de bandes magnétiques. Une requête sur un document consistait alors à parcourir toute la bande jusqu'à ce que l'on trouve le bon document, d'où une lenteur certaine du système malgré l'optimisation des algorithmes de recherche. L'amélioration du processus était bloqué par le matériel (bandes magnétiques) ce qui changea radicalement avec l'arrivée d'un nouveau système de stockage plus rapide : le disque dur.

b. Fichier de signature

Une signature de texte est une chaîne de n bits (chaîne binaire) dans laquelle les bits sont positionnés pour décrire le document, où n est appelé la longueur de signature. Le nombre de bits à 1 est appelé le poids de sa signature.

La chaîne de bit est créée en appliquant une opération de hachage sur tous les mots clés décrivant un document.

Les signatures résultantes du codage de documents sont stockées séquentiellement dans un fichier séparé (le fichier de signatures), qui est beaucoup plus petit que la collection origine et les recherches dans ce fichier sont beaucoup plus rapides.

Par exemple, on considère le terme t_1 représenté par les bits numéro 1 et 3 positionnés à la valeur 1 dans sa signature et le terme t_2 représenté par les bits numéro 2 et 4 positionnés à la valeur 1. Le troisième terme t_3 pourrait avoir des bits numéros 1 et 2 positionnés à la valeur 1, chaque terme a ainsi une signature unique.

Terme	Signature du terme
t ₁	0101
t ₂	1010
t ₃	0011

Tableau 1 : Signature des termes

On considère à présent le document d₁ contenant le terme t₁, le document d₂ contenant les termes t₁ et t₃ et le document d₃ contenant les termes t₁ et t₂. Le fichier de signature pour les trois documents est alors présenté dans le tableau 2.

Document	Signature du document
d ₁	0101
d ₂	0111
d ₃	1111

Tableau 2 : Signature des documents

Une fois qu'un fichier signature est créé, la réponse à une requête consiste à calculer une chaîne pour la requête et à la comparer à celle calculée pour chaque document.

Le principal inconvénient de cette méthode est le fait qu'un bit signale la présence d'un terme dans un fichier mais ne donne ni son emplacement, ni le nombre de fois qu'il apparaît dans ce fichier. De plus, l'utilisation de notion comme l'adjacence, la proximité ou des calculs de poids nécessitent une deuxième recherche. Néanmoins, les résultats obtenus par cette méthode sont relativement bons, tant du point de vue mise à jour, charge de la machine, stockage des données ou réponses aux requêtes.

c. Fichier inverse

La structure de fichier inverse est à la base de la plupart des systèmes de recherche d'information. Dans cette structure, chaque document peut être représenté par une liste de mots clés qui décrivent le contenu du document pour la recherche. Une recherche rapide peut être réalisée si nous inversons ces mots clé. Des mots clés sont stockés par exemple par ordre alphabétique dans le fichier index. Pour chaque mot clé, une liste de pointeurs aux documents de qualification dans les "fichiers postants" est maintenue. Cette méthode est utilisée dans la plupart des systèmes commerciaux [Amrouche, 08].

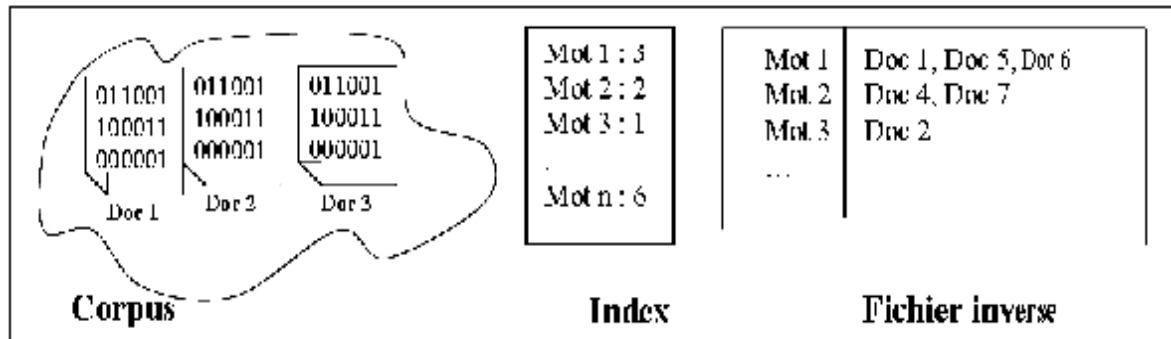


Figure 3 : Structure de fichier Inverse.

Un fichier inverse contient une série de couples pour chaque terme distinct dans la collection. La série de couples est de la forme <doc-id, tf>.

Le terme t_i qui existe dans j documents différents, aura une "liste posting" de la forme:

$t_i \rightarrow (d_1, tf_{i1}), (d_2, tf_{i2}), \dots, (d_j, tf_{ij}),$

Où d_j indique l'identificateur du document j et tf_{ij} indique la fréquence du terme i dans le document j .

Les fichiers inverses sont à l'origine de meilleures performances que les fichiers de signature en termes de temps de réponse. En effet, l'utilisation de la structure de fichier inverse rend la recherche d'un résultat plus rapide. Néanmoins, ce type de structure consomme énormément de place de stockage, les fichiers index étant parfois aussi gros que les fichiers de données, surtout dans le cas où les positions où apparaissent les mots clés dans les fichiers sont stockées. Les mises à jour sont aussi coûteuses puisqu'il faut refaire l'index à chaque nouvelle insertion.

Pour l'évaluation d'une requête, les modèles présentés en section 1.4 utilisent un fichier inverse.

Ainsi, à titre d'exemple, l'évaluation dans le modèle booléen se fera de la manière suivante:

- Pour chaque terme de la requête, retrouver les documents grâce au fichier inverse.
- Si deux termes sont reliés par ET, on prend l'intersection des ensembles. Dans le cas des ensembles flous, il faut calculer la pondération du document. Si deux termes sont reliés par OU, on utilise l'union et la nouvelle pondération est calculée.
- Une fois que tous les opérateurs sont évalués, on obtient un seul ensemble de documents avec leur pondération.

1.3.2 Pertinence et appariement document-requête

Le processus d'appariement document-requête permet de mesurer la pertinence d'un document vis-à-vis d'une requête. De manière générale, à chaque réception d'une requête, le système crée une représentation similaire à celle des documents, puis calcule un score de correspondance entre la représentation de chaque document et celle de la requête. Ce score traduit un degré de pertinence système. Cette dernière est supposée représenter le jugement de pertinence de l'utilisateur vis-à-vis du document. La valeur de pertinence système est calculée à partir d'une fonction de

similarité appelée RSV(Q, D) (Retrieval Status Value) où Q est une requête et D un document de la base.

Cette mesure tient en compte des poids des termes déterminés généralement en fonction d'analyses statistiques et probabilistes. On parle souvent de la pertinence utilisateur et la pertinence système. La première correspond au jugement de l'utilisateur sur la réponse en document rendu par le système et la deuxième est une mesure d'évaluation de la similarité entre le document et la requête.

Un problème fondamental est que la pertinence système est indépendante de l'utilisateur et du contexte de la recherche.

I.3.3 Reformulation des requêtes

L'utilisateur exprime son besoin en information sous forme d'une requête afin de trouver des résultats qui l'intéressent. Cependant, le SRI lui rend parfois des résultats qui ne lui conviennent pas. L'augmentation continue du volume des bases documentaires rend quasi impossible le fait de retrouver des informations pertinentes en utilisant la requête initiale de l'utilisateur. Pour cela, une étape de reformulation de la requête est souvent utilisée dans l'espoir de retrouver plus de documents pertinents. Ce processus permet de générer une requête plus adéquate que celle initialement formulée par l'utilisateur.

I.4. Les modèles de Recherche d'Information

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

- Créer une représentation interne pour un document ou une requête basée sur les termes de l'indexation.
- Définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de similarité (mesure de pertinence).

Il existe un grand nombre de modèles de recherche d'information, et ces modèles diffèrent principalement par la façon dont les informations disponibles sont représentées, et par la façon de formaliser la mesure de pertinence.

La figure 4 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

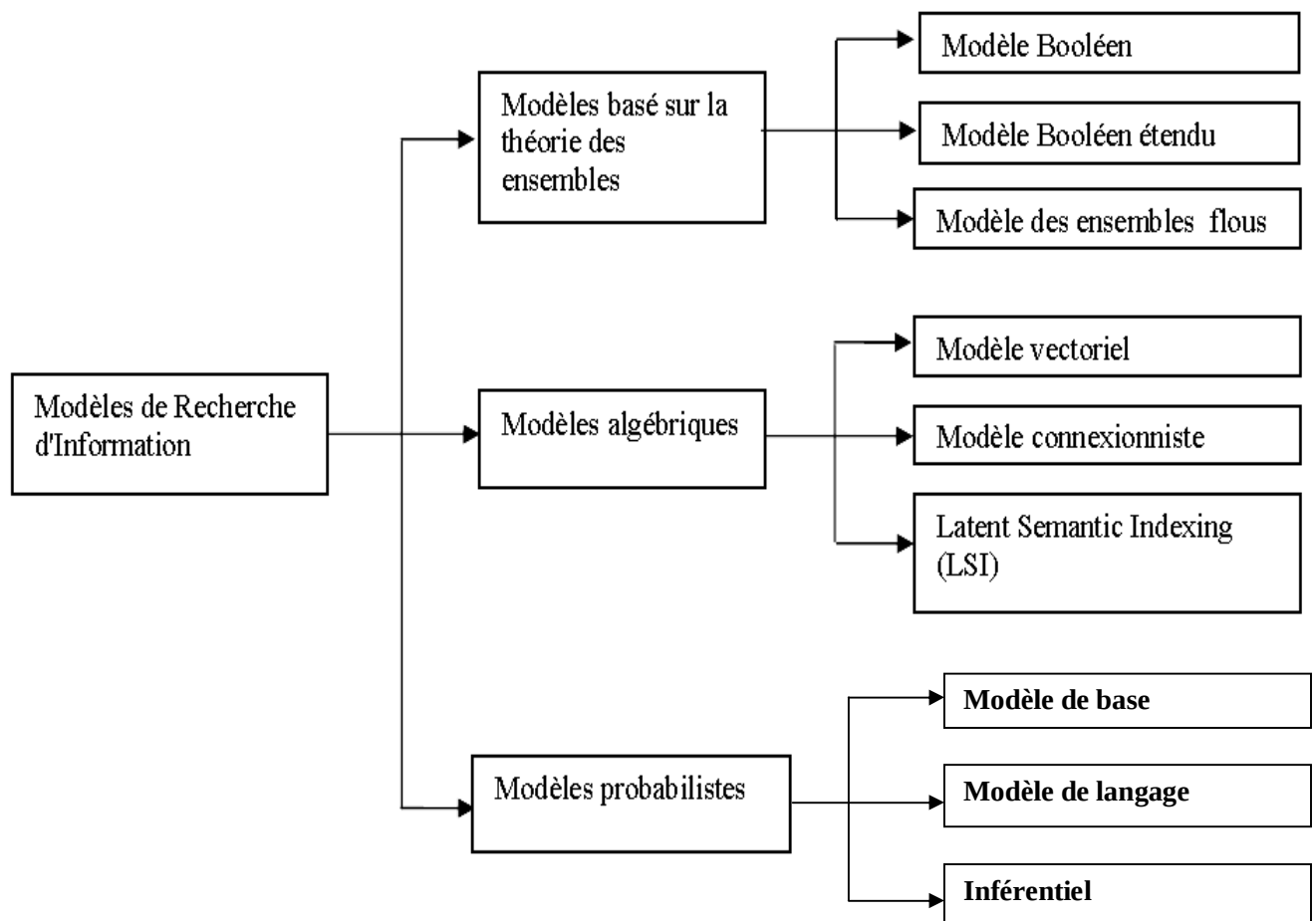


Figure 4: Taxonomie des Modèles de Recherche d'Information [Amrouche, 08].

1- les modèles booléens basés sur la théorie des ensembles :

Se décompose en modèle booléen (booléen model), modèle booléen étendu (extended boolean model) et modèle basé sur les ensembles flous (fuzzy set model)

Dans ces derniers, des opérateurs logiques (OR, AND, NOT) séparent les termes de la requête et permettent d'effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associés à chaque terme.

2- les modèles algébriques (vectoriels) :

Définie Le modèle vectoriel (vector model), le modèle vectoriel généralisé (generalized vector model) [Wong et al., 1985] , Latent Semantic model (LSI) [Deerwester & al., 1990], et le modèle connexionniste [Boughanem, 1992] [Mothe, 1994] [Kwok, 1989]. Dans ces modèles, la

pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel [Piwowarski,2003] .

3-les modèles probabilistes :

Introduit le modèle probabiliste général, le modèle de réseau de document ou d'inférence (Document Network), et le modèle de langage.

Ces modèles reposent sur la théorie des probabilités : pour ces modèles, la pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

Nous présentons dans la suite, les principaux modèles issus de chacun de ces trois groupes. et qui sont largement exploités en RI : le modèle booléen, le modèle vectoriel et le modèle probabiliste.

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

I.4.1.1 Le modèle booléen

Le plus simple des modèles de RI, le premier imposé dans le monde de la RI, reconnu pour sa force pour faire une recherche très restrictive et obtenir, pour un utilisateur expérimenté, une information exacte et spécifique

Il considère que les termes de l'indexe sont présents ou absents dans un document, en conséquence, les poids des termes dans l'indexe sont binaires c.à.d $w_{ij} = \{0,1\}$ ou w_{ij} : le poids du terme i dans un document j .

Dans ce modèle, un document d est représenté comme une conjonction logique des termes non pondérés. Exemple : $d = t_1 \wedge t_2 \wedge \dots \wedge t_n$.

Une requête q est composée de termes liés par 3 connecteurs (algèbre de Boole) :

ET ($\wedge$) : sert à indiquer la présence simultanée de plusieurs mots clés.

OU ($\vee$) : indique que l'on cherche des documents ayant pour mot clé l'un des mots donnés.

NON ($\neg$) : sert à éliminer des documents incluant un mot clé.

Exemple : $Q = (t_1 \wedge t_2) \vee (t_3 \neg t_4)$ ou Q : la requête, t_i : terme d'indexation.

La fonction de correspondance dans ce cas consiste à vérifier si le descripteur de chaque document d implique l'expression logique de la requête Q . Le résultat de cette fonction est donc binaire. Ainsi, la similarité document requête est calculée par :

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

Le modèle booléen présente le principal avantage de simplicité de mise en œuvre. Néanmoins, il présente quelques problèmes :

1. La correspondance entre un document et une requête est soit 1, soit 0. En conséquence, le système détermine un ensemble de documents non ordonnés comme réponse à une requête. Les usagers doivent encore fouiller dans cet ensemble pour trouver des documents qui les intéressent. Ceci est difficile dans le cas où beaucoup de documents répondent aux critères de la requête.
2. Tous les termes dans un document ou dans une requête étant pondérés de la même façon simple (0 ou 1), donc tous les termes ont la même importance dans le document. Ainsi, un document qui décrit en détail "informatique", mais mentionne un peu "commerce" se trouve être représenté par {informatique, commerce} dans laquelle les deux termes deviennent aussi importants l'un que l'autre. Cela ne correspond pas à ce qu'on souhaite avoir.
3. Il est difficile pour l'utilisateur d'exprimer son besoin en information avec des expressions booléennes et ces expressions formulées sont généralement très simples, ce qui ne permet pas d'utiliser au mieux les caractéristiques de ce modèle. Le modèle booléen standard n'est utilisé que dans très peu de systèmes de nos jours. Si on utilise un modèle booléen, c'est plutôt une extension de ce modèle qu'on utilise. Les extensions proposées essaient justement de corriger ces lacunes.

1.4.1.2 Le modèle Booléen étendu

Dans le modèle booléen de base, tous les documents qui satisfont une requête sont retrouvés (généralement classés dans un ordre chronologique). Ils ne sont pas classés selon leur pertinence. Cela est dû au fait qu'un document satisfait une requête ou ne la satisfait pas (1 ou 0).

Une approche venant compléter le modèle booléen de base défini dans [Fox, 1983] consiste à associer 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 [Salton, 1989].

Considérons un ensemble de termes $t_1, \dots, t_N$ et soit wd_{ij} le poids du terme t_i dans le document $D_j = (wd_{1j}, \dots, wd_{Nj})$, 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 :

$$RSV(D_j, Q_k) = \left(\frac{\sum_{i=1}^N \sum_{i=1}^N w q_{ik}^p w d_{ij}^p}{\sum_{i=1}^N w q_{ik}^p} \right)^{\frac{1}{p}}$$

$$RSV(D_j, Q_k) = \left(\frac{\sum_{i=1}^N \sum_{k=1}^N w q_{ik}^p (1 - w d_{ij}^p)}{\sum_{i=1}^N w q_{ik}^p} \right)^{\frac{1}{p}}$$

Où p est une constante telle que $0 \leq p \leq \infty$, et $w q_{ik}$ le poids du terme t_i dans la requête Q_k .

I.4.1.3 Le modèle booléen basé sur des ensembles flous

La logique floue a été proposée par Lotfi Zadeh dans le milieu des années 1960 [Zadeh, 1965]. A l'inverse de la logique booléenne, la logique floue permet à une condition d'être dans un autre état que vrai ou faux. Elle peut prendre une infinité de valeurs de vérité dans un intervalle $[0, 1]$. En recherche d'information, une extension du modèle booléen basée sur les ensembles flous a été proposée par Salton [Salton, 1989]. Cette extension vise également à tenir compte de la pondération des termes dans les documents. Un poids d'un terme exprime le degré d'appartenance de ce terme à un ensemble. Ainsi, un document peut être représenté par un ensemble de termes pondérés comme suit : $D_j = \{(td_{1j}, a_{1j}), \dots, (td_{nj}, a_{nj})\}$ où td_{ij} est le i ème terme du document D_j , et a_{ij} est le degré d'appartenance (une valeur comprise entre 0 et 1) du i ème terme au document D_j .

L'évaluation d'une requête Q_k par rapport à un document D_j , peut prendre plusieurs formes. Par exemple:

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

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

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

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

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

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

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

I.4.2 Les modèles algébriques

I.4.2.1 le modèle vectoriel

Le modèle vectoriel standard est un modèle de recherche d'information très connu. Il intègre dans un espace vectoriel une représentation qui symbolise les documents ou les requêtes en fonction

des termes d'indexation qui les composent. La forme d'implémentation la plus connue du modèle vectoriel est le système de recherche documentaire SMART (Salton's Magical Automatic Retriever of Text) [Salton, 1971] [Salton, 1983].

Ainsi dans ce modèle, un document ainsi qu'une requête sont représentés comme un vecteur de poids. Chaque poids dans le vecteur désigne l'importance d'un terme correspondant dans ce document ou dans la requête. Ces vecteurs sont définis dans un espace vectoriel composé de l'ensemble des termes de l'indexation. La pertinence d'un document vis-à-vis d'une requête est relative aux positions respectives des vecteurs documents et requête dans cet espace. Cette position est estimée par une distance ou une similarité définie sur cet espace. Dans ce modèle, les documents et les requêtes sont considérés de la même façon et représentés par des vecteurs de termes pondérés dans le même espace [Salton, 1971]. Chaque poids d'un terme dans un vecteur document (respectivement requête) désigne l'importance de ce terme dans ce document (respectivement requête). Ces termes pondérés sont utilisés pour calculer le degré de similarité entre chaque document et la requête de l'utilisateur. Les documents retrouvés sont présentés dans un ordre décroissant de leur degré de similarité correspondant.

Un document et une requête peuvent être définis dans un espace vectoriel de dimension N (N étant le nombre de termes d'indexation de la collection de documents) de la manière suivante :

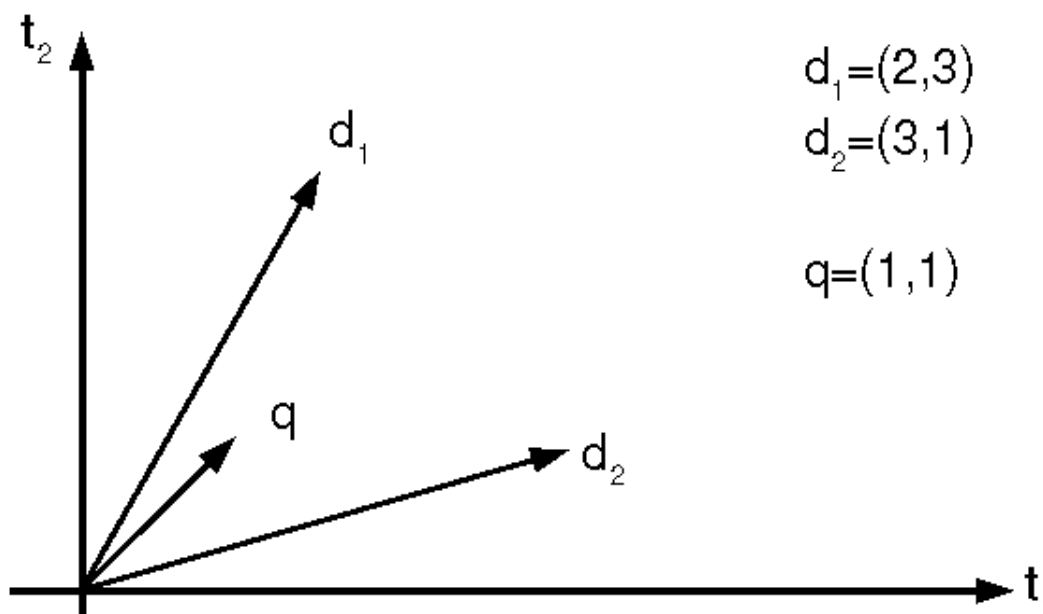


Figure 5 : Représentation des documents et requêtes dans un espace vectoriel.

Chaque document est représenté par un vecteur

$$D_j = (d_{1j}, d_{2j}, d_{3j}, \dots, d_{Nj}).$$

Chaque requête est représentée par un vecteur :

$$Q = (q_1, q_2, q_3, \dots, q_N).$$

Avec

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

Le mécanisme de recherche consiste à retrouver les vecteurs documents qui s'approchent le plus du vecteur requête. Les principales mesures de similarité utilisées sont :

Le produit scalaire :

$$RSV(Q, D_j) = \sum_{i=1}^N q_i \cdot d_{ij}$$

La formule de Jaccard :

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

La formule du cosinus :

$$RSV(Q, D_j) = \frac{\sum_{i=1}^N q_i \cdot d_{ij}}{(\sum q_i^2)^{1/2} \cdot (\sum d_{ij}^2)^{1/2}}$$

Le but final est d'arriver à retourner une liste ordonnée de documents selon ce degré.

L'avantage de ce modèle réside dans l'expression des besoins de l'utilisateur, contrairement au modèle booléen où les termes de la requête doivent être reliés par des connecteurs logiques, l'utilisateur peut ici exprimer son besoin en information en langage naturel ou sous forme d'une liste de mots-clés. La pondération des termes augmente les performances des systèmes, le modèle permet de renvoyer des documents qui répondent approximativement à la requête, et la fonction d'appariement permet de trier les documents selon leur degré de similarité avec la requête.

Théoriquement, le modèle vectoriel présente le principal inconvénient lié à l'indépendance mutuelle des termes d'indexation. Wong et al. [Wong & al, 1985] ont proposé un modèle vectoriel généralisé

(Generalized Vector Space Model) qui lève l'hypothèse d'indépendance des termes. Dans ce modèle chaque terme est représenté par un vecteur dans un espace vectoriel dont les axes sont orthogonaux par construction. Un document est représenté par la moyenne des vecteurs représentant les termes qu'il contient.

I.4.2.2 Le modèle connexionniste

Ce type de modèle se base sur les fondements des réseaux de neurones biologiques, tant pour la modélisation des documents et des informations descriptives associées (termes, auteurs, mots clés, etc.), que pour la mise en 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. Un réseau de neurone formel est construit à partir des représentations initiales des documents et de la requête. Le mécanisme de recherche d'information est fondé sur le principe de propagation de valeurs depuis les neurones descriptifs de la requête vers ceux des documents, à travers les connexions du réseau. Les résultats sont présentés à l'utilisateur selon le niveau d'activation des neurones documents. Le modèle connexionniste est connu pour sa capacité d'apprentissage, ce qui permet aux SRI de devenir adaptatifs.

Les premiers travaux utilisant les réseaux de neurones dans la RI ont été décrits dans [Belew, 1989].

Le fonctionnement du réseau se fait par propagation de signaux de la couche d'entrée vers la couche de sortie (voir schéma en Figure 6). Chaque neurone de la couche d'entrée reçoit une valeur d'activation, calcule une valeur de sortie et la transmet vers les neurones qui lui sont reliés dans la couche suivante. Ce processus se reproduit jusqu'à arriver à la couche de sortie, les valeurs de sortie dans la couche de sortie servant de critères de décision.

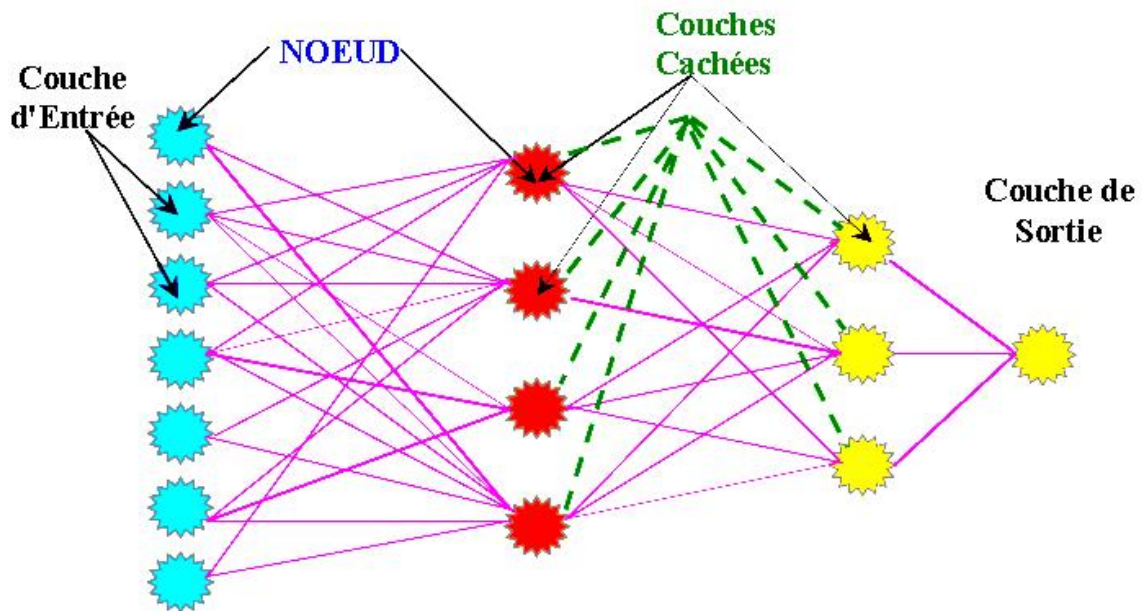


Figure 6: Un réseau de neurones à couches.

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

Le lien requête-terme indique l'apparition du terme dans la requête. Un poids est associé à ce lien.

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

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

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

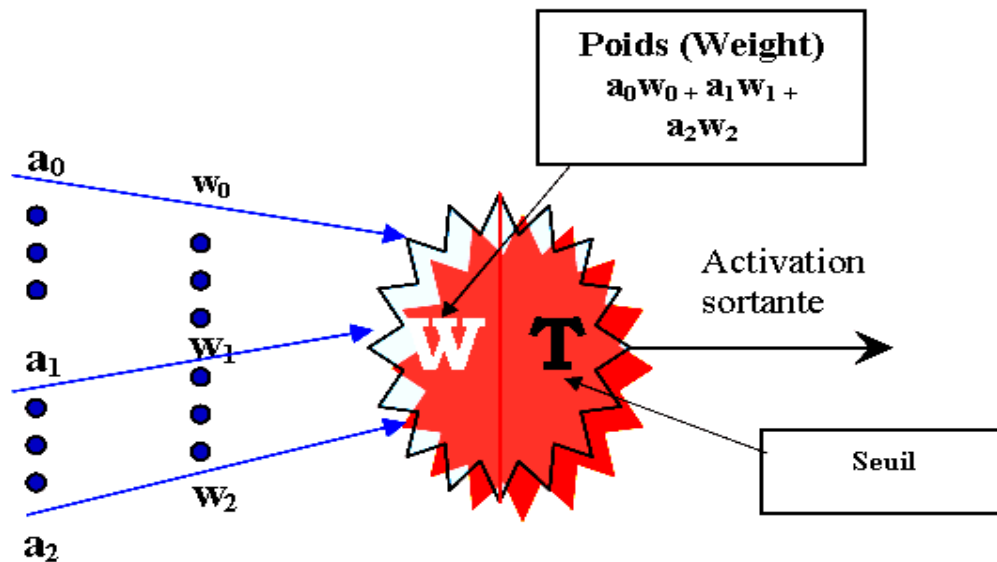


Figure 7 : Calcul d'une valeur à l'intérieur d'un nœud du réseau de neurone.

Actuellement, plusieurs modèles basés sur le principe des réseaux de neurones sont utilisés en recherche d'information [Boughanem, 1992] [Boughanem & Tamine, 2004] [Crestani, 1994] [Kwok, 1989]. Les modèles de RI basés sur les réseaux de neurones sont une solution pour combler les lacunes des modèles vectoriels.

La notion de réseau est convenable pour représenter les relations et associations qui existent entre les termes (exemples: synonymie, voisinage, etc.), entre les documents (exemples: similitude, référence, etc.), et enfin entre les termes et les documents (exemples: fréquence, poids, etc.).

Cependant, il n'existe pas de représentation unique d'un réseau de neurones pour la recherche d'information, c'est au constructeur du modèle de le définir, et ce en identifiant les éléments

Suivants :

- les différentes couches du réseau (couche d'entrée, de sortie, intermédiaires, etc.)
- les neurones de chaque couche,
- la fonction d'entrée de chaque neurone,
- la fonction de sortie de chaque neurone,
- les liens entre les neurones et leurs poids associés.

I.4.2.3 Latent Semantic Indexing (LSI)

Dans ce modèle, les documents sont représentés dans un espace de dimension réduite issu de l'espace initial des termes d'indexation [Deerwester & al, 1990]. Le but de ce modèle est d'aboutir à

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

Comparativement au modèle vectoriel, la technique LSI réduit la dimension de l'espace de représentation aux seuls vecteurs de représentation de l'information sémantique, et ce, en réduisant l'effet de variation d'utilisation des termes. Cette technique propose une représentation optimale à partir d'un espace de représentation termes-documents (voir la figure 8) [Dumais 1995] [Foltz 1990] [Furnas et al 1988]. Cette optimisation est basée sur une analyse en composantes principales de la matrice termes-documents.

Formellement, la transformation par LSI est présentée comme suit :

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

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

Avec :

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

D_0^t : une matrice orthogonale de dimension $m \times d$;

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

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

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

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

Avec :

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

D' : une matrice orthogonale réduite de dimension $k \times d$,

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

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

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

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

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

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

Plusieurs applications de ce modèle ont été proposées en recherche d'information, mais également pour le filtrage d'information et la recherche documentaire multilingue [Dumais & al, 1996] [Foltz & Dumais, 1992].

L'avantage de la méthode est qu'elle arrive à une représentation pseudo-conceptuelle des documents de la base, permettant de retrouver des documents même s'ils ne contiennent pas les mots des requêtes. Son inconvénient est qu'elle est sensible à la quantité et à la qualité des données traitées.

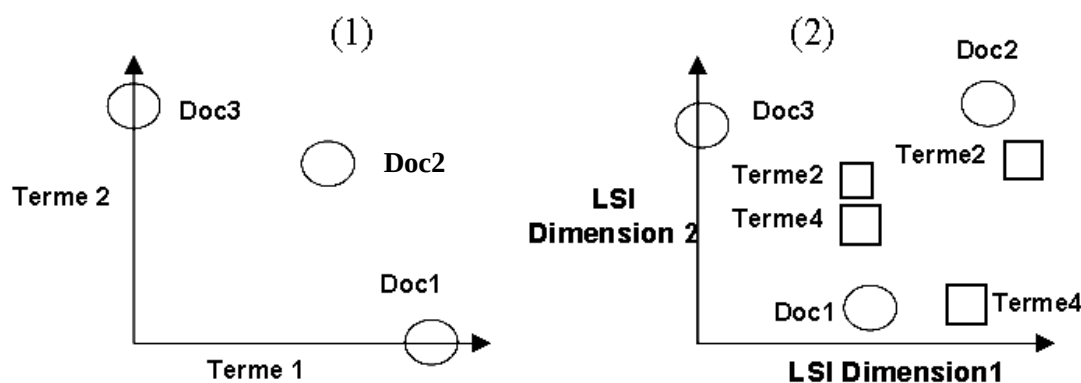


Figure 8: Représentation d'un espace vectoriel classique (1) et d'un espace réduit LSI (2).

I.4.3. Les modèles probabilistes :*I.4.3.1. Le modèle probabiliste :*

Le modèle probabiliste aborde le problème de la recherche d'information dans un cadre probabiliste. Le premier modèle probabiliste a été proposé par Maron et Kuhns [Maron & Kuhns, 1960] au début des années 1960. Ils proposent de modéliser le processus de sélection des documents dans un SRI en se basant sur la théorie des probabilités. Le principe de base du modèle probabiliste consiste à présenter les résultats de recherche d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête.

Robertson [Robertson, 1977] résume ce critère d'ordre par le "principe de classement probabiliste", désigné aussi par PRP (Probability Ranking Principle).

Le modèle probabiliste utilise un modèle mathématique fondé sur la théorie de la probabilité [Robertson & Sparck Jones, 1976], L'idée de base est de calculer deux probabilités conditionnelles $P(R/D)$ et $P(NR/D)$ pour une requête donnée avec R représentant la pertinence (ensemble des documents pertinents) et NR la non-pertinence (ensemble des documents non pertinents).

Les termes ne sont pas pondérés, mais prennent seulement la valeur 0 (terme absent) ou 1 (terme présent). Le principe du modèle consiste à calculer une fonction d'ordre ($O(D)$) qui permettrait de classer les documents répondant à une requête.

$$O(D) = P(R/D) / P(NR/D)$$

En utilisant le théorème de Bayes et sachant que la probabilité de pertinence et de non pertinence sont identiques (car la probabilité de tomber sur un document pertinent ou non pertinent dans un corpus est la même), on obtient la fonction suivante:

$$O(D) = P(D/R) / P(D/NR)$$

Avec

$P(D/R)$: probabilité que D fasse partie de l'ensemble des documents pertinents

$P(D/NR)$: probabilité que D fasse partie de l'ensemble des documents non pertinents.

Le document D peut être décomposé en plusieurs événements, chaque événement représente l'absence ou la présence d'un terme dans le document D , noté $(t_i=x_i)$ ou x_i égal 0 ou 1.

Ainsi : $P(D/R) = \prod_{(t_i=x_i) \in D} P(t_i=x_i/R)$

$$P(D/NR) = \prod_{(t_i=x_i) \in D} P(t_i=x_i/NR)$$

Dans cette formule, on ne prend en compte que la présence ou l'absence d'un terme dans un document.

D'autres auteurs intègrent la fréquence d'apparition des termes dans le document.

Ainsi, Croft et Harper [Croft & Harper, 1979] proposent de calculer la fonction de similarité requête - document en prenant en compte la probabilité d'apparition d'un terme t_i dans un document D_j notée $P(d_{ij})$. Cette fonction est la suivante :

$$RSV (D_j, Q_k) = C \sum_{i=1}^N w d_{ij} w q_{ik} + \sum_{i=1}^N P(d_{ij}) w d_{ij} w q_{ik} \log \left(\frac{N - n_i}{n_i} \right)$$

$P(d_{ij})$ peut simplement correspondre à la fréquence d'apparition du terme t_i dans le document D_j

$$P(d_{ij}) = \frac{t f_{ij}}{\text{Max} t f_j}$$

C : constante, N : nombre de documents, n_i : nombre de documents contenant le terme t_i

De même, Robertson et Walker [Robertson & Walker, 1994] ont développé un modèle probabiliste qui utilise la loi de poisson pour calculer la probabilité de pertinence en intégrant la fréquence du terme et la longueur du document.

La principale limite de ce modèle est qu'il ne prend pas en compte les liens de dépendance entre termes et engendre des calculs de probabilité complexes [Tamine, 2000].

Pour remédier à cette limite, le modèle de réseau inférentiel bayésien a été défini par Turtle [Turtle & Croft, 1991] qui se base sur le calcul de la probabilité conditionnelle de chaque terme par propagation des liens entre les termes.

1.4.3.2 Réseau de Document (Document Network)

Le réseau de document comprend les *noeuds de document* (un pour chaque document dans la collection), les *noeuds de représentation* de texte et les *noeuds de représentation de concepts*. Les noeuds de représentation de texte, comme le montre la Figure 9 ont à l'intersection des deux niveaux de représentation. Ils synthétisent l'information sur la manière dont le document est représenté, notamment lorsqu'il s'agit de documents non textuels, tels que le son et la vidéo. Un document peut avoir différentes représentations. Les noeuds représentant les concepts décrivent les différentes représentations identifiés dans le texte des documents et des requêtes.

De façon générale, le problème d'identification de concepts à partir des documents est très délicat. Il a fait l'objet de plusieurs travaux durant les trente dernières années. Une méthode de projection du document sur le réseau conceptuel de l'ontologie peut être envisagée, donc Si l'on suppose que l'on sait extraire les concepts à partir des documents, on peut les placer dans les noeuds de concepts du réseau. Une approximation brute des noeuds de concepts peut par exemple être les termes ou groupes nominaux (phrases) les plus fréquents dans le document.

Finalement, le réseau de document et le réseau de la requête sont combinés pour calculer une estimation de la "croyance" (belief) que le document soit pertinent pour la requête [Grossman et al.,

98]. La matrice des liens sert pour chaque noeud à identifier, pour un certain nombre d'entrées, la croyance en sortie. La matrice des liens permet d'avoir des entrées avec des poids différents. On peut trouver des matrices complexes mais aussi élémentaires qui se résument simplement à sommer les poids des entrées.

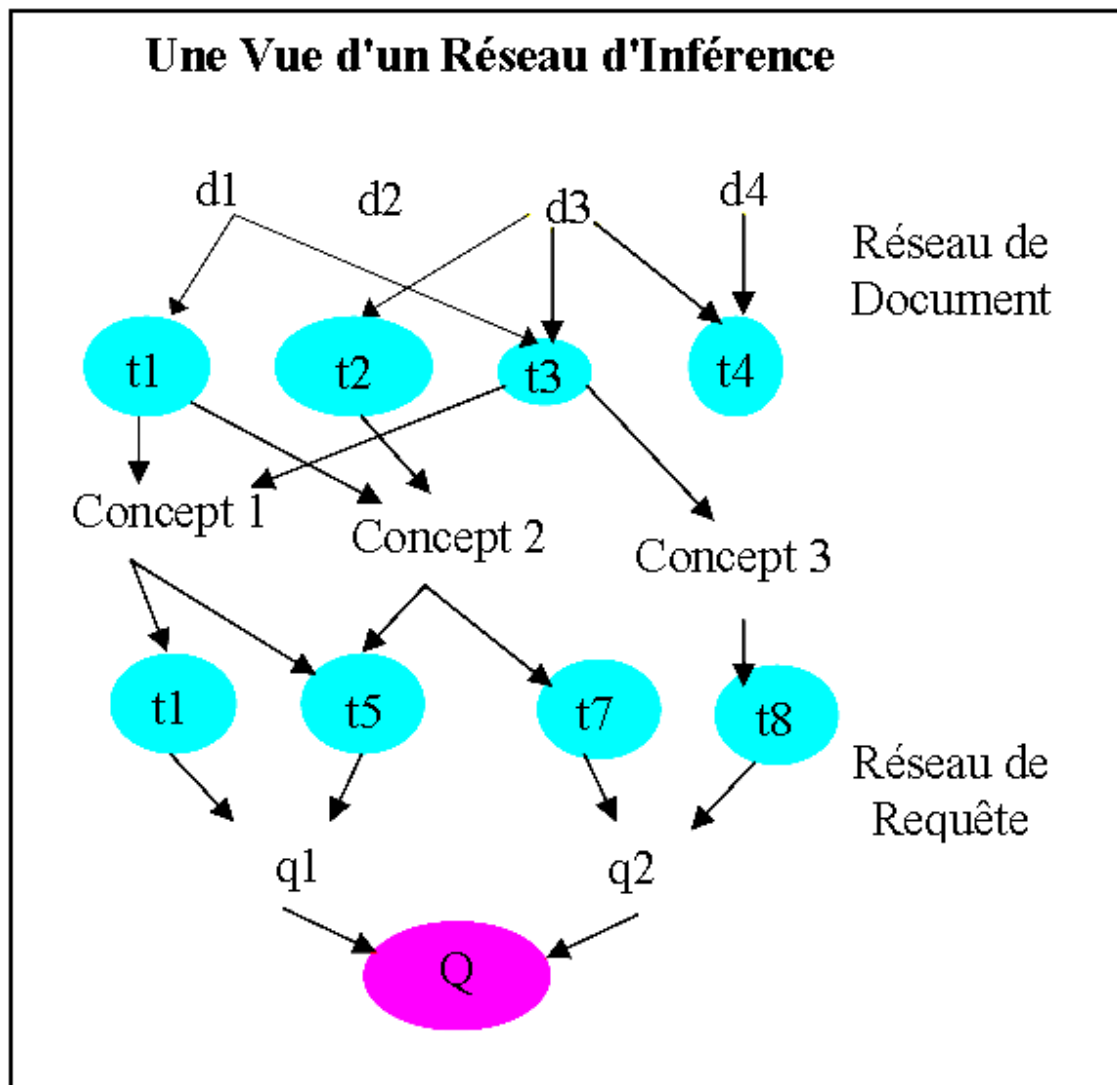


Figure 9: Représentation d'un réseau d'inférence.

L'avantage de ces modèles est qu'ils permettent d'avoir une vue claire des documents et de la requête, représentés dans un même réseau. Cependant, ils ne précisent pas comment les concepts sont extraits des documents et leurs applications en RI s'avère onéreuse.

I.4.3.3 Le modèle de langages

Dans les modèles classiques de recherche, on cherche à mesurer la similarité entre un document d_i et une requête q ou à estimer la probabilité que le document réponde à la requête ($P(Q_i/D_k)$). L'hypothèse de base dans ces modèles consiste à dire qu'un document n'est pertinent que s'il "ressemble" à la requête. Les modèles de langage, décrits par leurs initiateurs Ponte et Croft [Ponte et al., 98], ou encore Hiemstra [Hiemstra, 02] se sont basés sur des hypothèses différentes. Des hypothèses qu'on exposera en détail dans le chapitre 2 sections II.12.

I.5. Evaluation d'un SRI

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

En général, tout système de recherche d'information a deux objectifs principaux :

- Le premier est de retrouver tous les documents pertinents pour une requête utilisateur.
- Le deuxième est de rejeter les documents jugés non pertinents.

L'évaluation des systèmes peut être abordée selon deux angles : l'**efficience** et l'**efficacité**.

- L'efficience regroupe le temps et l'espace :

Un système est considéré meilleur lorsque le temps entre la formulation de la requête et la réponse du système est court et l'espace occupé par le système est faible.

- L'efficacité d'un système peut être mesurée par les critères suivants :

- L'effort, intellectuel ou physique, nécessaire aux utilisateurs pour formuler les requêtes, conduire leur recherche, voir les documents résultats
- La présentation du résultat (capacité de l'utilisateur à utiliser les documents retrouvés)
- La qualité du corpus vis à vis du besoin de l'utilisateur (dans quelle mesure tous les documents pertinents sont dans le corpus)
- La capacité du système à retrouver des documents intéressants et à éliminer les autres. Cette caractéristique semble être la plus importante.

I.5.1 Mesures d'évaluation

Plusieurs mesures d'évaluation sont proposées, on citera ci-dessous les plus importantes :

I.5.1.1 Rappel et précision

Précision :

La précision mesure la proportion de documents pertinents relativement à l'ensemble des documents restitués par le système. Elle mesure la capacité du système à rejeter tous les documents non pertinents à une requête donnée par le rapport entre l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés.

$$\text{Précision} = \frac{R_+}{M}$$

Rappel :

Le rappel mesure la proportion des documents pertinents restitués par le système relativement à l'ensemble des documents pertinents contenus dans la base documentaire. Elle mesure la capacité du système à retrouver tous les documents pertinents répondant à une requête.

$$\text{Rappel} = \frac{R_+}{R}$$

Où :

R : désigne le nombre de documents pertinents dans toute la collection.

M : est le nombre de documents sélectionnés par le système.

R_+ : est le nombre de documents pertinents sélectionnés par le système.

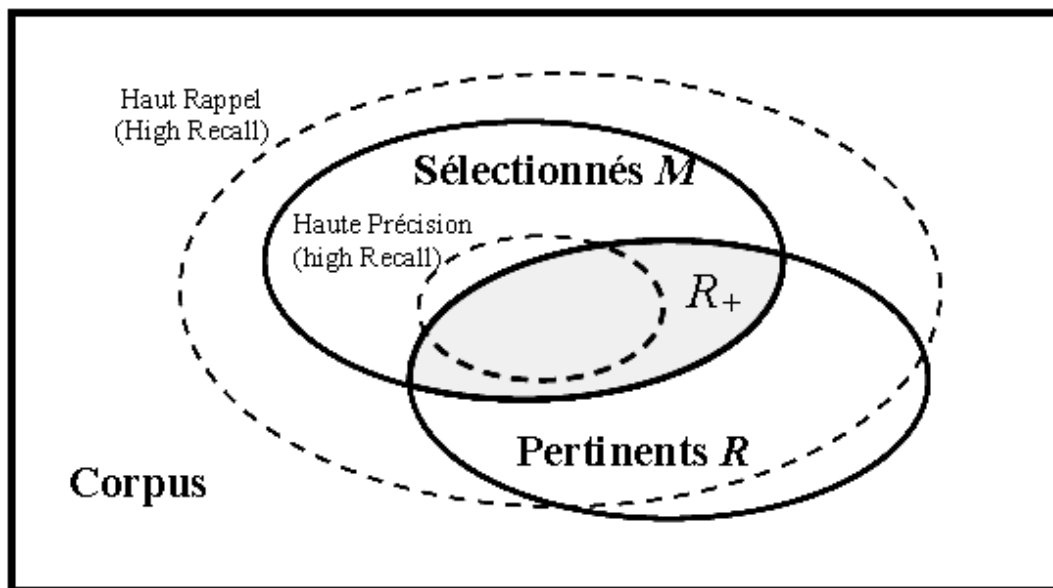


Figure 10: Représentation schématique des zones de précision et de rappel.

Cette figure, illustre la précision et le rappel du système suite à une requête quelconque. La zone de haute précision représente le cas où le système retourne peu de documents non pertinents, alors que la zone de haut rappel concerne le cas où seule une petite partie des documents pertinents, est omise par le système.

La courbe Rappel/Précision :

La précision mesurée indépendamment du rappel et inversement est peu significative. Pour pouvoir examiner les résultats efficacement, on calcule la paire des mesures (taux de rappel, taux de précision) à chaque document restitué.

Pour un système idéal, le taux de précision est égal à celui de rappel.

Le système parfait trouverait seulement les documents pertinents, avec une précision et un rappel de 100%. En pratique, les mesures de rappel et précision évoluent inversement, ce qui signifie que la courbe interpolée de précision en fonction du rappel est décroissante. Plus la courbe est élevée, plus le système est performant.

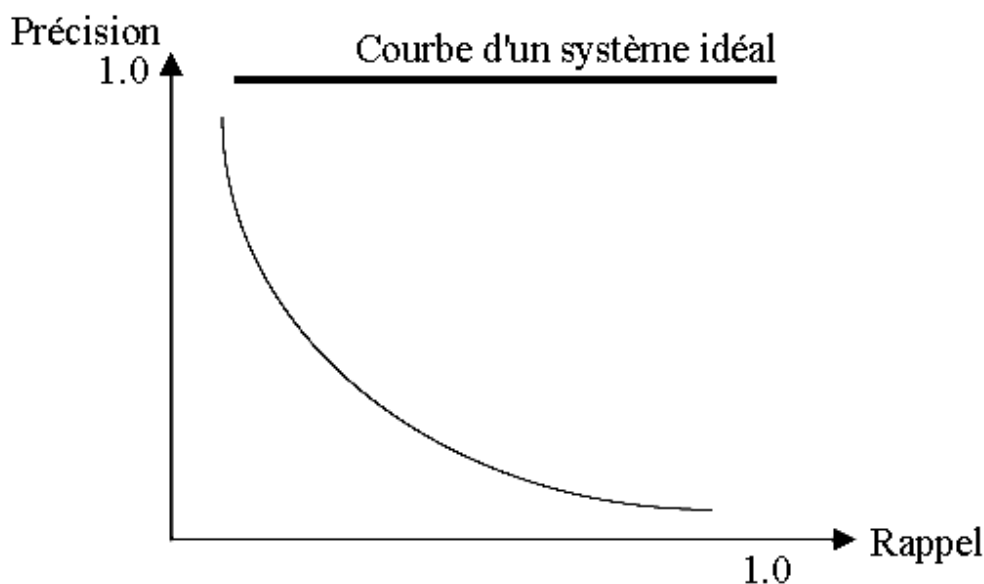


Figure 11 : Allure d'une courbe de rappel-précision.

Par exemple, considérons une requête pour laquelle cinq (5) documents sont pertinents dans la base. Soit la liste des réponses du système $\{d1, \dots, d15\}$. Les documents pertinents sont marqués par la lettre "P" comme indiqué par la troisième colonne du Tableau 3.

Document	Score	Pertinent	Précision	Rappel
d1	9,92	P	1,00	0,20
d2	9,77		0,50	0,20
d3	9,76	P	0,67	0,40
d4	9,59	P	0,75	0,60
d5	8,72		0,60	0,60
d6	6,85	P	0,67	0,80
d7	6,51	P	0,57	0,80
d8	4,32		0,63	1,00
d9	4,16		0,56	1,00
d10	3,47		0,50	1,00
d11	2,69		0,45	1,00
d12	2,04		0,42	1,00
d13	1,84		0,38	1,00
d14	1,67		0,36	1,00
d15	0,07		0,33	1,00

Tableau 3 : Exemple de valeur rappel-précision.

On considère d'abord le premier document d1 restitué par le système. A ce point, on a retrouvé un document pertinent parmi les 5 existants. Donc on a un taux de rappel de 0.2, La précision est 1/1. Le point de la courbe est donc (0.2, 1.0). On considère ensuite les deux (2) premiers documents restitués. Le taux de rappel est toujours de 0.2 et la précision est cette fois de 0.5 (un document sur deux est bon). Le point est donc (0.2, 0.5).

Ce processus est répété jusqu'à épuisement de la liste des réponses (qui peut être très longue en incluant tous les documents de la base). Les premiers points de la courbe sont alors représentés comme dans la Figure 12.

On peut remarquer que cette courbe qui représente une seule requête ne correspond pas à la forme générale de la Figure 11.

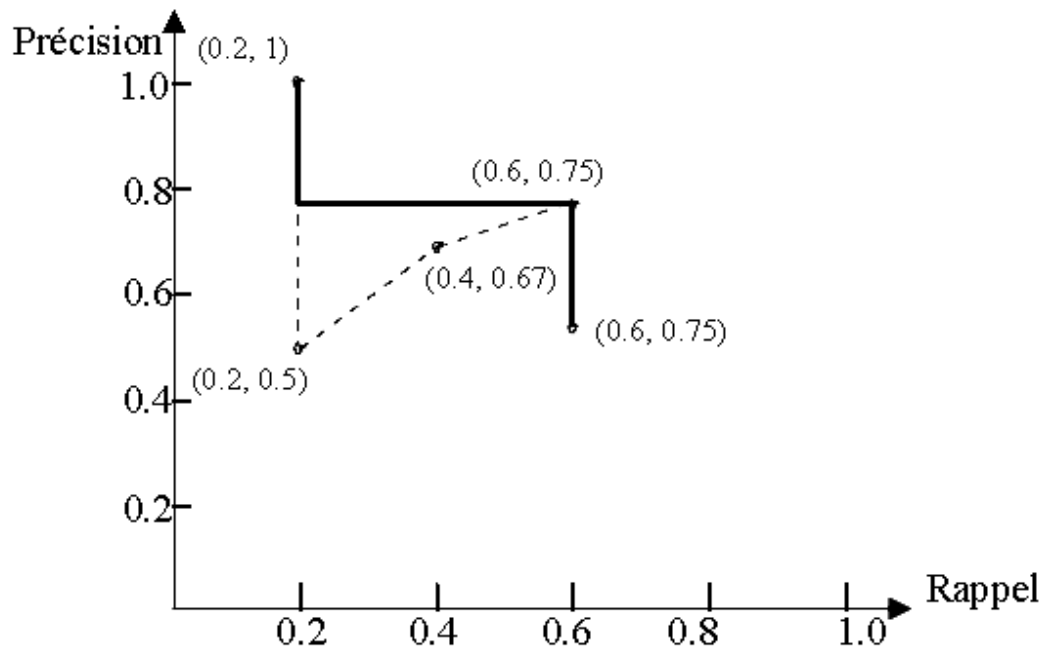


Figure 12 : Représentation des points de rappel-précision.

Il arrive souvent qu'on applique *l'interpolation* sur la courbe de chaque requête. L'interpolation vise à créer une courbe descendante ayant l'allure de la forme générale de la Figure 11.

1.5.1.2 la précision moyenne

La précision moyenne est la moyenne des valeurs de précision à chaque document pertinent de la liste ordonnée. Elle tient compte à la fois de la précision et du rappel. Elle représente la moyenne des précisions (non interpolées) calculées pour chaque document pertinent à trouver, au rang de ce document. Si un document pertinent est retourné à la 10^{ème} position, la précision pour ce document est "la précision à 10 documents". Si un document pertinent n'a pas été trouvé par le système, la précision pour ce document est nulle. De façon générale, étant donné un certain nombre de requêtes, la précision moyenne au taux de rappel, se calcule comme suit :

$$P(rp) = \sum_{i=1}^{N_q} \frac{P_i(rp)}{N_q} \text{ Où } P_i(rp) \text{ est la précision de la } i^{\text{ème}} \text{ requête au niveau de rappel } rp.$$

Où : N_q est le nombre de requêtes et $P_i(rp)$ représente la précision de la $i^{\text{ème}}$ requête au niveau de rappel rp .

Comme les niveaux de rappel ne sont pas unifiés pour l'ensemble des requêtes, 11 points de rappel standards : de 0.0 à 1.0 à pas de 0.1, sont retenus.

1.5.1.3 Les mesures combinées

Le rappel et la précision, en dépit de leur popularité, ne sont pas en elles-mêmes des mesures appropriées pour évaluer un système de recherche d'information. Des chercheurs ont été amenés à s'investir dans d'autres mesures. Les nouvelles mesures proposées se basent toujours sur les mesures de rappel et de précision, cependant elles sont combinées de façon à en sortir une valeur représentative des performances d'un système de RI.

Toutefois, ces mesures sont des fonctions ad-hoc et ne peuvent pas être justifiées de façon formelle [Rijsbergen, 81]. L'exemple le plus simple de ce type de mesure est la somme de la valeur de rappel et de précision :

$$S(j) = R(j) + P(j)$$

Où :

$R(j)$: représente la valeur de rappel au $j^{\text{ème}}$ document restitué,

$P(j)$: est la valeur de précision au $j^{\text{ème}}$ document restitué.

D'autres mesures plus compliquées [Rijsbergen, 81] sont les suivantes :

$$V(j) = 1 - \frac{1}{2 \cdot \frac{1}{P(j)} + 2 \cdot \left(\frac{1}{R(j)}\right) - 3}$$

$$Q(j) = \frac{R(j) - F(j)}{R(j) + F(j) - 2R(j) \cdot F(j)}$$

Avec

$$F = NP_r / NP$$

NP : le nombre de documents pertinents de la collection.

En restant dans cette optique de combinaison du rappel et de la précision, deux autres mesures ont été proposées : la mesure *Harmonique* et la mesure d'évaluation appelée *E*.

a. La mesure Harmonique

La mesure Harmonique H est une fonction de paires de valeurs de précision et de rappel :

$$H(j) = \frac{2}{\frac{1}{R(j)} + \frac{1}{P(j)}}$$

Elle tient ses valeurs dans un intervalle fermé $[0,1]$. Elle est égale à 0 lorsque aucun document pertinent n'est restitué et 1 lorsque tous les documents restitués sont pertinents. La valeur de H est élevée quand les valeurs de rappel et de précision sont élevées. Ainsi, la mesure Harmonique garantit le compromis entre les deux mesures de rappel et de précision.

b. La mesure E (E-measure)

Une autre mesure permettant de combiner le rappel et la précision appelée E est proposée par Van Rijsbergen [Rijsbergen, 1981]. Le but de cette mesure est de permettre à l'utilisateur de spécifier laquelle des valeurs de précision ou de rappel, est plus intéressante. La mesure d'évaluation E est définie comme suit :

$$E(j) = 1 - \frac{1 + b^2}{\frac{b^2}{R(j)} + \frac{1}{P(j)}}$$

Où est un paramètre de l'utilisateur lui permettant de spécifier l'importance du rappel ou de la précision. Si $b=1$, $E(j)$ serait alors le complément de la mesure Harmonique $H(j)$. Plus les valeurs de b sont supérieures à 1, plus l'importance donnée à la précision l'emporte sur le rappel et inversement.

I.6. Collections de test :

Pour pouvoir évaluer un modèle RI, la communauté en recherche d'information s'est dotée de collections de tests. Une collection de tests est composée lorsque l'on dispose, pour une collection de documents donnée, un ensemble de requêtes « questions-tests » (*topics*) avec leurs réponses idéales associées (*Qrels*).

Pour évaluer un SRI, il suffira alors de lui soumettre les questions-tests et de comparer les réponses qu'il fournit aux réponses-types. En mesurant l'écart entre la réponse du système et la réponse-type, on obtiendra une mesure de qualité sur les performances du système documentaire.

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

Exemple : Collection CACM, ISI, CLEF.

Le projet le plus ambitieux est sans aucun doute le projet d'évaluation TREC (Text Retrieval Conference) de la DARPA (Defense Advanced Research Projects Agency).

TREC est une campagne d'évaluation co-organisée par le NIST National (Institute of Standards and Technology) et la DARPA. Elle a été commencée en 1992 dans le but d'encourager la recherche dans la recherche documentaire basée sur de grandes collections de test ;

Pour chaque session TREC, le NIST fournit un ensemble de test documents et de questions. Les participants exploitent leurs propres systèmes de récupération sur les données, et renvoient au NIST une liste de document ordonnée, alors Le NIST met les différents résultats, juge en commun les documents recherchés pour l'exactitude, et évalue les résultats en prenant les k-documents les mieux classés des différents SRI participant à la campagne d'évaluation, ces documents forment l'ensemble des documents pertinents pour chaque requête.

Ces documents seront ensuite montrés à des juges qui décident finalement de la pertinence de chaque document.

I.7.Conclusion

Dans ce chapitre nous avons décrit les principes fondamentaux des systèmes de recherche d'information et les principaux concepts de base de la 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.

L'information mise à la disposition d'un large public est souvent hétérogène, et son volume est considérable en augmentation constante, la conception et la mise en œuvre d'outils efficaces, permettant notamment à l'utilisateur de n'avoir accès qu'à l'information qu'il juge pertinente, devient une nécessité absolue.

L'un des SRI soutenant ces caractéristiques et offrant de nombreux avantages à ce titre est l'Internet.

II.1. Introduction :

L'avènement du Web représente l'une des plus grandes innovations du siècle. Les progrès des technologies de l'information, et l'amélioration des capacités de stockage sont les rouages essentiels de cette innovation.

Ainsi, plusieurs travaux ont été menés afin de faciliter la recherche d'information, et la consultation de données disponibles dans ce gigantesque espace d'information qui est le web, et ceci de manière rapide et sans aucune contrainte, en utilisant des outils de recherche indépendant des spécificités du web et basés sur des modèles de recherche d'information (RI) développés pour des documents textuels atomiques.

Ce chapitre fera l'objet d'une étude détaillée consacrée à la recherche d'information sur le web, nous présenterons plusieurs outils web utilisés pour cet objectif. Puis nous exposerons des modèles de langages qui offrent un cadre probabilistique pour la description du processus de la RI, car ils permettent entre autre, l'incorporation de plusieurs sources d'informations concernant un document, parmi ces sources, le contenu du document et la structure de liens entre documents.

II .2. Les Concepts de base du web

Parmi les composants du web, on cite :

a. Le client (navigateur ou browser):

Logiciel client capable d'exploiter les ressources hypertextes et hypermédias du web, ainsi que les ressources d'internet dans son ensemble, qui permet donc l'accès à l'information. Il permet de représenter les contenus de pages web, d'envoyer des requêtes au serveur web et de gérer les résultats de ces requêtes. Les clients web les plus répandus sont : Netscape, Internet Explorer, Navigator, Google chrome....etc.

b. Le Serveur :

Un serveur web est un logiciel permettant à des clients d'accéder à des pages web, c'est-à-dire en réalité des fichiers au format HTML à partir d'un navigateur installé sur leur ordinateur distant. Un serveur web est donc un « simple » logiciel capable d'interpréter les requêtes http arrivant sur le port associé au protocole http, et de fournir une réponse avec ce même protocole. Les principaux serveurs web sur le marché sont : Apache, Microsoft IIS (internet information server) et Microsoft PWS (Personal Web Server).

c. Page web :

C'est un document informatique qui peut contenir du texte, des images, des formulaires à remplir et divers autres éléments multimédias et interactifs. Une page Web peut être téléchargée et consultée via son URL. Elle est constituée d'une ou plusieurs ressources distinctes. La principale ressource est généralement un document écrit en langage HTML (HyperText Markup Language), qui contient le texte et définit la disposition des autres ressources : images, animations, etc.

d. Protocole :

Ensemble des conventions nécessaires pour faire coopérer des entités distantes, en particulier pour établir et entretenir des échanges d'informations entre ces entités. Les protocoles les plus utilisés sont http (HyperText Transfert Protocol), FTP (File Transfert Protocol).

1. **Le protocole http : HyperText Transfert Protocol** est un protocole de la couche application, il peut fonctionner sur n'importe quelle connexion fiable. Les clients http les plus connus sont les navigateurs web permettant à un utilisateur d'accéder à un serveur contenant les données.
2. **Le protocole FTP : File Transfert Protocol** est un protocole de transfert de fichiers. Le protocole FTP définit la façon selon laquelle des données doivent être transférées sur un réseau TCP/IP. Le protocole FTP a pour objectifs de :
 - Ø Permettre un partage de fichiers entre machines distantes.
 - Ø Permettre une indépendance aux systèmes de fichiers des machines clientes et serveur.

e. World wide web :

Le World Wide Web, communément appelé le Web, parfois la Toile, littéralement la « toile (d'araignée) mondiale », est un système hypertexte public fonctionnant sur Internet et qui permet de consulter, avec un navigateur, des pages mises en ligne dans des sites. L'image de la toile vient des hyperliens qui lient les pages Web entre elles. Le Web n'est qu'une des applications d'Internet, avec le courrier électronique, la messagerie instantanée, Usenet, etc. Le Web a été inventé plusieurs années après Internet, mais c'est le Web qui a rendu les médias grand public attentifs à Internet. Depuis, le Web est fréquemment confondu avec Internet, en particulier, le mot Toile est souvent utilisé de manière très ambiguë.

On distingue deux types de structures associées au web qui sont :

- Ø **La structure interne aux pages** : qui est décrite en langage HTML.
 - Ø **La structure externe aux pages** : qui est décrite par le réseau de liens hypertextes, La structure externe se décompose de la manière suivante :
 1. **La structure hiérarchique** des sites Web, c'est-à-dire la structure arborescente interne à un site.
 2. **La structure hypertexte** des sites Web, c'est-à-dire la structure de graphe interne à un site.
 3. **La structure macroscopique** du Web, c'est-à-dire la structure de graphe externe aux sites.
- Le web est caractérisé par **Une nature dynamique**, en d'autres termes la nature du web permet la modification, l'ajout, et la suppression des différentes informations.

- Le web est une immense collection de documents hétérogène. Ces documents peuvent être de types « documents structurés » ou bien « hypertexte ».

II.3. Documents structurés :

Un document structuré est composé d'un ensemble de parties (contenu) organisées de manières hiérarchiques. Aussi, il a des attributs externes associés à chaque partie. Les principales composantes d'un document structuré sont : le contenu, la structure, les attributs externes et le sens de lecture.

II.3.1. Le Contenu :

Le contenu d'un document structuré désigne le contenu textuel ou multimédia, représenté sous forme d'un ensemble invisible et non structuré. Par exemple : Paragraphe, image,...etc. le contenu est alors l'atome de description des documents.

II.3.2. La Structure :

La structure d'un document consiste à décrire chacun des éléments textuels ou non textuels constituant un document. On peut distinguer deux types de structures, la structure physique et la structure logique.

- **La structure physique :** correspond à l'organisation d'affichage des données qui composent le document, exemple : page, en tête, .., Elle dépend aussi de l'environnement de présentation du document. Par exemple : format du papier ou l'écran de l'ordinateur.
- **Structure logique :** correspond à l'organisation hiérarchique des données du document. La structure logique est spécifiée par l'auteur, elle lui permet d'organiser le document pour mieux exprimer ces idées. Par exemple : un document se compose d'un titre, de section, elles-mêmes, composées de titre et de sous-sections.

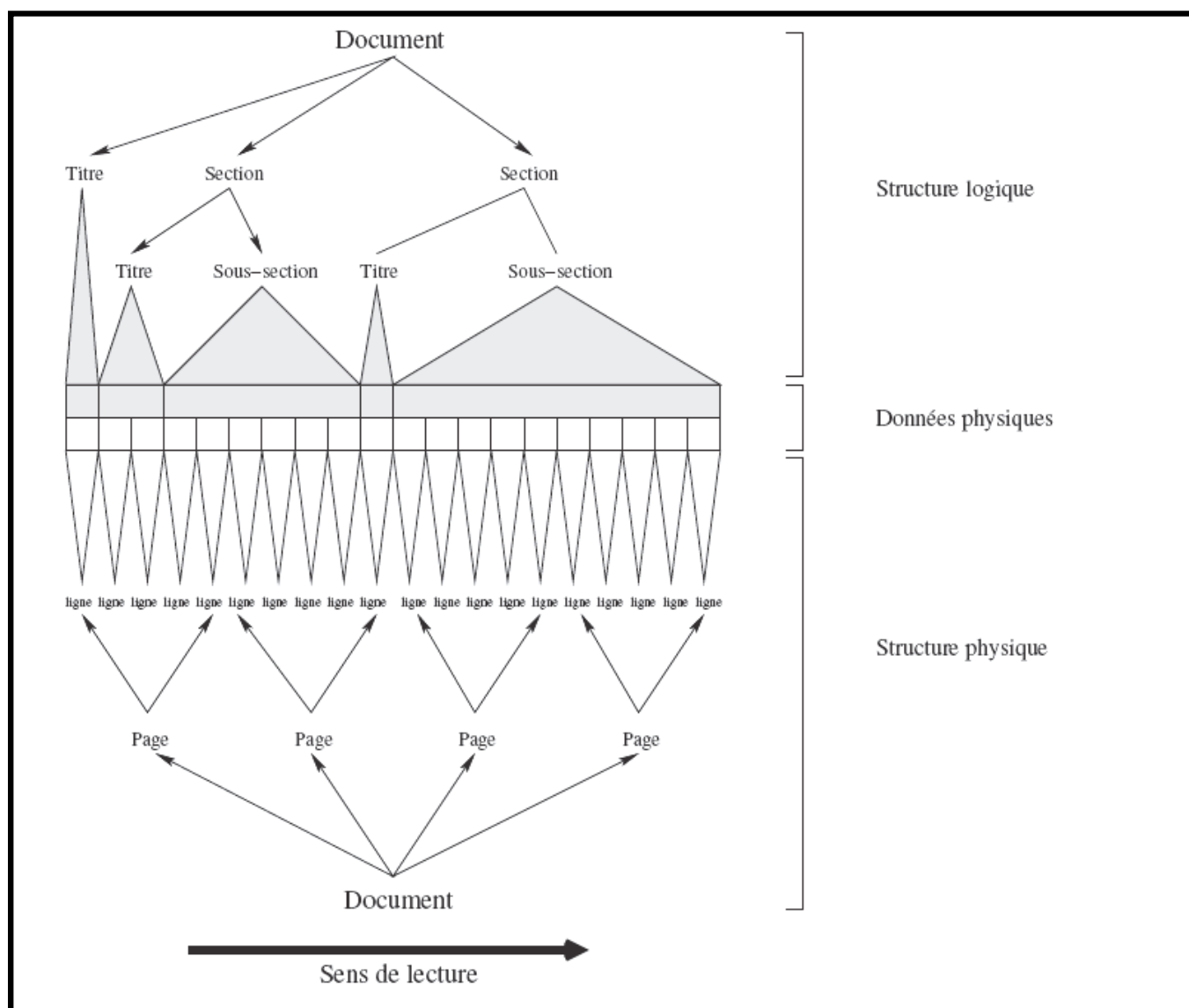


Figure 1 : Structure logique et structure physique d'un document.

La structure logique propose un « sens de lecture » implicite, correspondant à l'enchaînement des idées de l'auteur.

II.3.3. Sens de lecture :

La lecture d'un document structuré consiste à suivre un chemin de lecture linéaire et unique qui parcourt la totalité du document. Il est imposé par l'auteur, et le lecteur consulte les informations de manière séquentielle.

II.3.4. Les attributs externes :

Il s'agit de méta- informations décrivant les informations attachées aux éléments d'un document structuré. Par exemple : le titre, l'auteur, date de création du document. Les attributs externes apportent une information supplémentaire concernant le document et non son contenu.

II.4. Le web et les documents structurés :

Les documents qu'on retrouve sur le web ont les spécifications des documents structurés décrit par plusieurs langages comme le HTML. La structure hiérarchique est une structure qui organise le document de manière hiérarchique.

II.4.1. Structure hiérarchique interne des pages web :

Généralement les pages web sont décrites dans un langage qui est le HTML. Les pages web possèdent une structure interne dite « structure hiérarchique intra-page » grâce à l'utilisation de balise HTML.

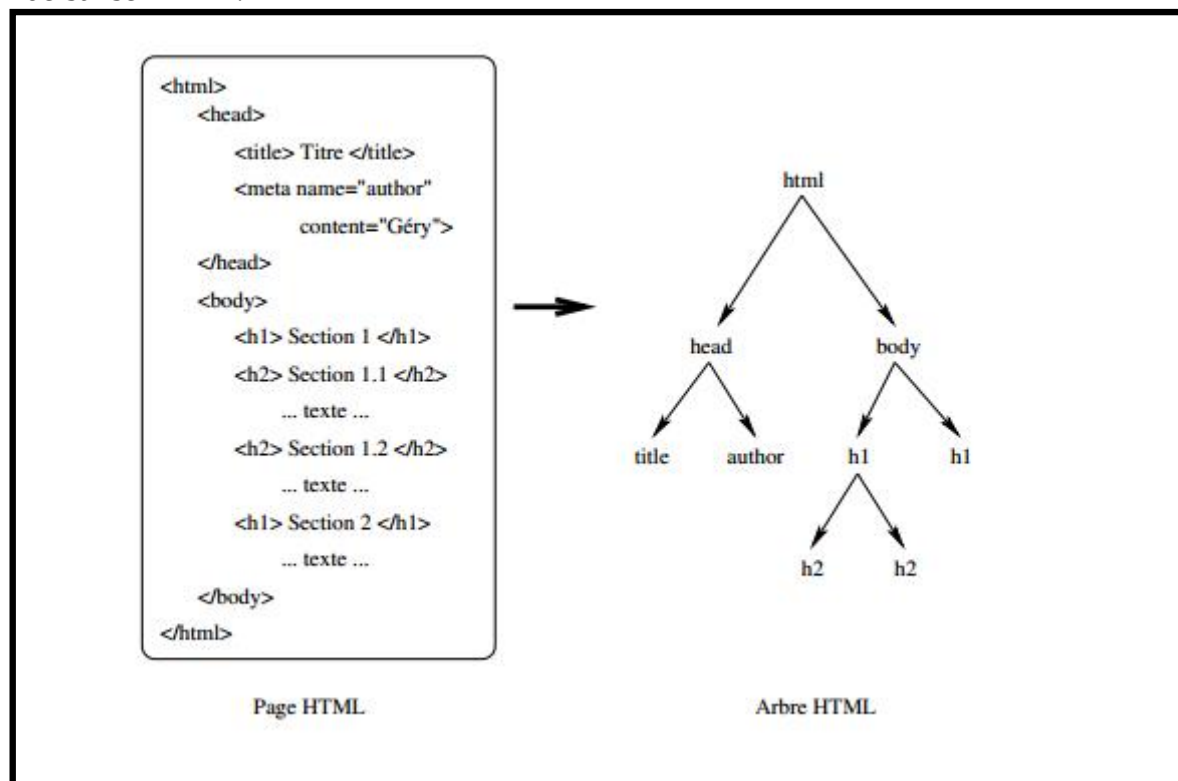


Figure 2. Structure hiérarchique d'une page Web

II.4.2. Structure hiérarchique interne des sites web :

Les liens hypertextes peuvent être utilisés pour décrire la structure interne d'un document. Ce type de structure se rencontre fréquemment sur le web, particulièrement en raison de l'utilisation de logiciel comme LATEX2HTML qui permet de transformer des documents structurés en un ensemble de page HTML reliées.

II.5. Hypertextes :

Le principe des hypertextes a été inventé par Vannevar Bush avec le système Memex, en s'inspirant du fonctionnement du cerveau humain par association d'idées [Bush45]. Un **hypertexte** est une représentation non-linéaire d'une information textuelle sous la forme d'un graphe de **noeuds** connectés par des **liens**. La consultation d'un hypertexte nécessite une phase interactive de **navigation**. Parmi les notions les plus basiques de l'hypertexte :

- **Noeud** : un **noeud** est une unité d'information textuelle, c'est-à-dire un fragment de texte (chapitre, section, paragraphe, etc.), ou un document entier. il peut s'agir d'un paragraphe, une carte (HyperCard), une page, ou même une image, ...etc.
- **Lien** : un **lien** définit une connexion entre deux noeuds de l'hypertexte, le noeud source et le noeud destination du lien. On distingue deux classes de liens :
 - **Liens explicites** : définis par l'auteur au moment de la création de l'hypertexte, modélisant une relation entre deux noeuds. sont décrits à l'aide d'URL
 - **Liens implicites** : ne sont pas définis au moment de la création de l'hypertexte, mais qui existent potentiellement. Ils peuvent être extraits de l'hypertexte.
 - **Les Hyperliens** :
Un hyperlien ou simplement lien est une référence dans un système hypertexte permettant de passer automatiquement d'un document consulté à un document lié. Les hyperliens sont notamment utilisés dans le world wide web pour permettre le passage d'une page web à une autre avec un simple clic.
 - **Les URLs** :
Une URL, « Uniform Ressource Locator » est une chaîne de caractères utilisée pour adresser une ressource du web. Elle est informellement appelée « adresse web ». Chaque hyperlien du web est construit avec l'URL de la ressource pointée, insérée avec une certaine syntaxe dans un document source qui le contient pour indiquer la localisation d'une autre ressource (un document) ou fragment de ressource.
- Ø une **ancree** matérialise la source ou la destination d'un lien dans un noeud.
- Ø la **navigation** consiste à “activer” un lien hypertexte pour se retrouver “transporté” sur l'objet référencé, à l'aide d'un logiciel de consultation du Web appelé “navigateur”. On distingue plusieurs types de navigation dans un hypertexte :
 - **Le tour guidé** (“guided tour”) est une navigation supervisée qui suit un parcours linéaire imposé par l'auteur.
 - **Le cheminement déambulatoire** (“browsing”) est une navigation non supervisée, le lecteur choisit lui-même les liens qu'il désire suivre, au fur et à mesure de sa “balade”.

II.6. Recherche d'information sur internet :

De nos jours, l'information doit être mise à la disposition d'un large public, cette information est souvent hétérogène, et son volume est considérable. Internet par exemple présente toutes ces caractéristiques et offre de nombreux avantages. Les informations de toute nature, offertes par ce dernier, connaissent aujourd'hui une croissance sans précédent. Celle-ci bouscule le monde de la documentation, qui se trouve ainsi confronté à des changements profonds et rapides, en termes d'élargissement des domaines, de volumes à traiter et d'évolution des outils et des données.

Dans ce cadre, les besoins en informations d'un utilisateur doivent être traduits sous forme de requêtes qui seront soumises au moteur de recherche (ou tout simplement au système de RI). Généralement, la formulation des besoins d'un utilisateur doit permettre d'associer un certain nombre de mots clés à la requête (c'est l'indexation). Ces mots clés résument la description de ces besoins. Une fois que la requête est exprimée, le système de RI a pour objectif de rechercher les informations pertinentes qui répondent à la requête.

II.7. Types de recherche d'information sur le web :

Les informations du web sont accessible directement par l'URL, et ceci en la pointant ou en la recherchant. Deux méthodes traitent le deuxième cas :

II.7.1. Recherche par Navigation :

La navigation (en anglais on dirait "browsing") convient bien à une recherche d'ordre général, cela implique d'aller sur un site annuaire, de rechercher par domaine ou par catégorie et ensuite aller d'un hyperlien à un autre jusqu'à trouver enfin le résultat qu'on cherche. Cette façon de faire présente le tps comme étant un avantage dans le cas où le site contenant l'information n'est pas indexé par le moteur de recherche, mais aussi un inconvénient majeur, vu que dans la plus part des cas cette démarche prend beaucoup de temps.

II.7.2. Recherche par interrogation :

Plutôt que d'avoir à naviguer à travers des sites et pages pour repérer l'information qui nous intéresse, on exploite directement le moteur de recherche en lui posant une requête directe qui modélise notre question puis il se charge de nous retourner la réponse. La recherche par interrogation se résume à poser des mots clés associés à notre sujet de requête puis on laisse le moteur de recherche faire le travail à notre place. Cette méthode est la meilleure façon d'agir à ce jour, vu que le temps de réponse est de plus en plus concluant, par exemple Google, affiche les résultats de recherche en moins d'une seconde.

II.8. Les outils de la recherche d'information :

Il existe de nombreux outils de recherche d'information sur le Web, ces outils qui se spécialisent en fonction des services utilisés et du type d'information qu'ils recensent. On qualifie d'ailleurs très souvent aujourd'hui tout interface de recherche et d'interrogation de moteur de recherche et ce quelle que soit la source interrogée et le système informatique utilisés [Largouet, 05]. Il convient en effet de distinguer différents types d'outils de recherche sur l'Internet.

Nous distinguons trois catégories d'outils pour la recherche d'information sur le web: les moteurs de recherche, les annuaires et les méta-moteurs. Cette distinction repose sur le mode d'indexation, et est essentiel, car elle induit des usages et des technologies très différentes.

II.8.1. les moteurs de recherche :

Un moteur de recherche est une application permettant de retrouver des ressources (pages web, 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 (Google Video par exemple est un moteur de recherche vidéo).

Ces outils de recherche sur le web sont constitué de « robots », encore 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 on peut citer par exemple Exalead Desktop, Google Desktop et Copernic Desktop Search, etc.

Des modules complémentaires sont souvent utilisés en association avec les trois briques de bases du moteur de recherche. Les plus connus sont les suivants :

1. **Le correcteur orthographique** : il permet de corriger les erreurs introduites dans les mots de la requête, et s'assurer que la pertinence d'un mot sera bien prise en compte sous sa forme canonique.
2. **Le lemmatiseur** : il permet de réduire les mots recherchés à leur lemme et ainsi d'étendre leur portée de recherche.
3. **L'anti dictionnaire** : utilisé pour supprimer à la fois dans l'index et dans les requêtes tous les mots "vides" (tels que "de", "le", "la") qui sont non discriminants et perturbent le score de recherche en introduisant du bruit.

En ce qui concerne les caractéristiques, les moteurs de recherche ont un fonctionnement commun, mais diffèrent par un certain nombre de critères [Bondu, 09]. Pour ce qui est de commun, rappelons simplement qu'ils procèdent tous des même étapes :

- D'abord l'exploration du web, durant laquelle ils vont collecter les informations sur chaque page rencontrée.
- Puis l'indexation, durant laquelle ils vont enregistrer dans une base de données les informations collectées.
- Enfin la recherche, durant laquelle ils vont rechercher les données collectées en fonction des mots clés. Si tous les moteurs passent par ces étapes communes, ils ont tous leurs différences. Voici quelques éléments sur lesquels ils se différencient :
 - D'abord la manière d'explorer le web.
 - Ensuite, le choix des informations qu'ils vont récupérer des sites visités. Certains moteurs vont conserver le titre de la page, la description qu'en a fait le créateur de la page (metatag), parfois une partie du contenu de la page, ...
 - La manière de construire l'index, et sa taille. C'est d'ailleurs un des critères de performance le plus souvent mis en avant. En effet, plus un moteur indexe de pages, plus il a de

chance de vous fournir un résultat correspondant à votre requête. Google passe pour avoir l'index le plus important. Les chiffres sont néanmoins difficiles à obtenir. En septembre 2005, Google indiquait avoir indexé 24 milliards de pages. Actuellement, si l'on demande à Google de présenter toutes les pages contenant simplement le chiffre '1' (c'est une astuce pour estimer le nombre de pages indexées), il propose 21 milliards de pages. Yahoo! en propose 40 milliards.

- La manière de rechercher dans l'index.

- La manière de présenter les résultats (on parle alors d'interface). Si Google présente sobrement le titre, un extrait, et quelques autres éléments, d'autres moteurs ont une interface beaucoup plus riche avec présentation d'images, de critères de pertinence, des graphiques, de mots clés, ...

Enfin la popularité des moteurs de recherche n'est pas absolue, généralement, nous ne considérons souvent que le moteur de recherche Google. Toutefois, son hégémonie n'est pas aussi importante dans le reste du monde. Nous parlons également souvent du trio Google, Yahoo, Bing. Auxquels nous rajoutons le moteur de recherche chinois 'Baidu' qui représente selon une étude du cabinet Comscore réalisée en janvier 2011 : 3,3 milliards de requêtes, soit 5,4 % du total, mais également 73 % du marché chinois (450 millions d'internautes). Baidu se classe dans le 'Top 3' des moteurs de recherche les plus utilisés dans le monde.

II.8.2. L'annuaire :

L'annuaire (ou directory en anglais) est une liste de liens subdivisés en catégories suivant une structure en arbre, accompagnée d'une brève description. Bien que ce procédé fût pionnier en la matière, il tend à disparaître. En effet, le fait de devoir sélectionner les catégories dans lesquelles on recherche suppose que l'on sache exactement où chercher. Et on peut se demander où se positionne le site qui appartient à plusieurs catégories. Mais à cette question, les moteurs utilisant ce procédé vous répliqueront qu'ils se trouvent dans toutes celles susceptibles de correspondre. Néanmoins, on doit reconnaître aux annuaires un gros avantage, celui de mettre en quelque sorte dans le contexte, ainsi les recherches dans la base de données sont diminuées, en plus d'obtenir des résultats plus pertinents.

Les annuaires sont donc des outils basés sur le recensement humain de l'information. Ils signalent des sites et des ressources de l'Internet comme un catalogue de bibliothèque signale des livres ou bien encore comme les pages jaunes signalent des entreprises. On distingue dans ce contexte deux catégories d'annuaires [Largouet, 05]

A) Les annuaires commerciaux (Tableaux)

Ils se financent grâce à la publicité. Ils ont en principe une couverture dit "générale" (ils couvrent toutes les disciplines). Ils peuvent concerner le monde ou une zone régionale, nous citons parmi eux :

- Annuaires généralistes internationaux : le plus connu est sans doute 'Yahoo Directory', mais il existe aussi 'DMIZ' de l'Open Directory Project et l'annuaire de 'Lycos'.
- Annuaires régionaux commerciaux : ce sont les annuaires qui recensent des sites en fonction de leur langue. Dans le cas de des annuaires francophones nous citons la version française de 'Yahoo Directory' ou encore l'annuaire 'Francité'.

- Les annuaires qui recensent d'autre pays ou parties du monde: comme l'annuaire 'Wohaa' pour l'Afrique et l'annuaire russe 'Yandex'.

B) Les annuaires non commerciaux

Sont des annuaires élaborés par des individus de façon bénévole ou bien par des institutions.

Ils sont soit généraux soit spécialisés. Leur préoccupation consiste toujours à identifier les ressources et les sites en tenant comptes de leur qualité :

- Annuaires à couverture (généraliste): comme le 'Vlib' (Virtual Library) et l'annuaire 'Resource Discovery network'.
- Annuaires à couverture thématique ou spécialisée : comme le répertoire en sciences humaines 'Voice of the Shuttle' et le répertoire de ressources juridiques 'Findlaw'.

Enfin, la distinction est importante entre les annuaires et les moteurs de recherche. Le tableau 1, présente une comparaison entre ces deux outils pour la RI.

Annuaires	Moteurs
Indexation de sites par des documentalistes	Indexation de mots par des robots
Recherche sur des sites et sur des catégories	Recherche en texte intégral sur des pages web
Avantages : choix des informations, classement raisonné par catégories et sous-catégories	Avantages : plus d'exhaustivité et mise à jour plus rapide
Inconvénients : moins d'exhaustivité et mise à jour moins rapide	Inconvénients : capture de pages web sans classement raisonné
À retenir : L'exploration des catégories s'avère souvent plus fructueuse que celle des sites.	À retenir : La recherche par mots-clés donne de meilleurs résultats sur les moteurs

Tableau 1– Comparaison entre annuaires et moteurs de recherche.

II.8.3. les méta- moteurs

Ils sont de création plus récente. Ils constituent en fait la première génération des agents dits "intelligents". Ils permettent d'interroger en une seule fois différents outils de recherche, qu'ils soient de type annuaire ou de type moteur, afin de fournir une réponse plus exhaustive.

Deux catégories de méta-moteurs: ceux en ligne et ceux consistant en un "logiciel client" à installer sur son ordinateur (le plus connu: COPENIC) [Largouet, 05]. Le principe de fonctionnement des méta-moteurs est différent, Certains indexent l'information contenue dans différents annuaires et moteurs, d'autres les interrogent simultanément de façon dynamique.

Certains de ces méta-moteurs retraitent plus au moins les réponses. Ils permettent ainsi de rechercher de façon plus large sur le Web. Toutefois, cela peut également générer du "bruit"

(réponses non pertinentes). La parade mise en oeuvre par certains méta-moteurs consiste à limiter le nombre de réponses de chaque outil interrogé (ce qui est indispensable et permet ainsi d'obtenir les réponses en principe les plus pertinentes). [Largouet, 05].

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

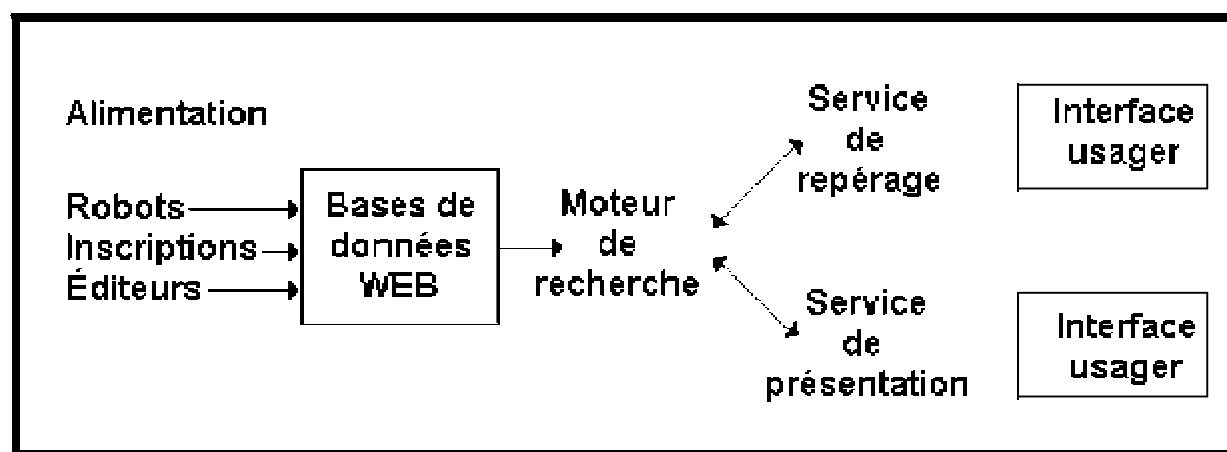


Figure 3 : Recension et repérage d'information sur le Web. [Carmel, 2000]

Les besoins en informations d'un utilisateur doivent être traduits sous forme de requêtes qui seront soumises au moteur de recherche (ou tout simplement au système de RI). Généralement, la formulation des besoins d'un utilisateur doit permettre d'associer un certain nombre de mots clés à la requête (c'est l'indexation). Ces mots clés résument la description de ces besoins. Une fois que la requête est exprimée, le système de RI a pour objectif de rechercher les informations pertinentes qui répondent à la requête.

II.9. Algorithme des moteurs de recherche d'information

Différentes études ont suggéré de tenir compte de la popularité des documents afin d'améliorer les performances de la recherche d'information. Le PageRank [Brin, 98] de Google et le HITS [Kleinberg, 99] de Kleinberg sont deux algorithmes fondamentaux qui utilisent les liens hypertextes pour classer les résultats d'une requête. Généralement, ces algorithmes fonctionnent en deux temps : Dans une première étape, un moteur de recherche retourne une liste de documents répondant à la requête posée, en fonction des termes de la requête et des termes d'indexation des documents. Dans une seconde étape, ces systèmes tiennent compte des liens hypertextes pour classer ces documents [Chibane, 08].

II.9.1. Algorithme Hyperlink-Induced Topic Search (HITS)

Kleinberg fut un des premiers à s'intéresser aux propriétés de connectivité du graphe représentatif d'Internet 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.

L'importance d'une page peut être extraite de la structure des liens du Web. Dans cette approche, deux types de page 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 et 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 pivots « 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 ou une publicité aura de nombreux liens vers elle. Cette situation est illustrée dans la figure 4.

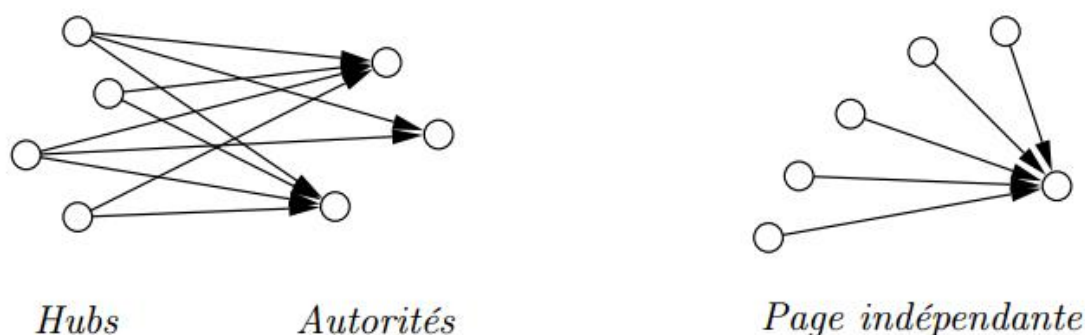


Figure 4. Répartitions des pages en bon Hubs, bonnes autorités et pages indépendantes.

Ce principe de renforcement mutuel de la notion de hubs et de pages autorités se traduit de la manière suivante :

$$\text{Auth}_p = \sum_{\forall q \rightarrow p} \text{Hub}_q$$

$$\text{Hub}_p = \sum_{\forall p \rightarrow q} \text{Auth}_q$$

Où : Auth_p est le score d'autorité de la page p.

Hub_p est le score (potentiel) de hub de la page p.

Avec une normalisation appropriée, par exemple $\frac{1}{c(p)}$, $C(p)$ représente le nombre de liens sortant de la page p . Il s'agit d'une propagation de pertinence avec une initialisation binaire : tous les $Auth_p$ et Hub_p sont initialisés à 1.

II.9.2. Algorithme PageRank

L'idée principale de cette méthode proposée par [Brin, 98] est de simuler le comportement d'un internaute naviguant de manière aléatoire sur Internet. La probabilité qu'il visite une page donnée est d'autant plus grande que cette page est 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 sur Internet. 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 (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 (1) montre la récursion permettant de calculer le PageRank pour tous les nœuds du graphe représentant Internet.

$$PR(p_j)_t = (1 - d) + d \sum_{p_i \rightarrow p_j}^n \frac{PR(p_i)_{t-1}}{c(p_i)} \quad \dots(1)$$

Ou :

$PR(p_j)_t$: la valeur du PageRank à l'itération t pour la page p_j .

$C(p_i)$: le nombre de liens sortants de la page p_i .

d : prend ses valeurs dans l'intervalle $[0-1]$ et est généralement placé à $d = 0.85$ d'après des études statistiques menées par Larry Page dans [Page et al., 1998].

« d » 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.

On constate dans la formule (1) que pour chaque itération toutes les pages distribuent leur PageRank en fonction de leurs liens sortants. Cette distribution s'effectue de manière plus ou moins importante en fonction du nombre de liens $C(p_i)$ de chaque page p_i (distribution uniforme). Cela peut aisément s'interpréter en considérant qu'une page possède un certain potentiel de pertinence et que l'auteur de la page donne une certaine crédibilité (PageRank) aux pages pour lesquelles il inscrit un lien.

L'équation (1) peut être reformulée de manière matricielle. Pour cela, on considère deux vecteurs colonne PR et E correspondant respectivement au rang de chaque page et à la loi de distribution de l'ensemble des nœuds du graphe, et une matrice A telle que l'entrée $A[p, q] = 0$ s'il n'existe pas de lien sortant de la page q vers la page p et $A[p, q] = 1/C(q)$ sinon.

A est une matrice sous-stochastique.

Cette équation prend ainsi la forme suivante :

$$PR = (1-d)E + d \cdot A^t \cdot PR$$

Où

A^t : la transposée de la matrice A.

II.9.3. Algorithme SALSA

L'algorithme SALSA, a été proposé par Lempel et Moran en 2000 [Lempel et Moran, 2000]. Son but est similaire à celui de l'algorithme HITS : 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 d'Internet 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 différent 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 d'Internet 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.

II.11. La Modélisation de Langue en Recherche d'Information

Les premiers travaux en RI qualifiés d'approche *classique*, sont apparus dans les années soixante, ils se sont focalisés à résoudre des problèmes principalement liés à la représentation de l'information, l'évaluation de requêtes ainsi que l'évaluation des performances de recherche.

Dans les modèles classiques 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 de pertinence d'un document face à une requête, soit $P(R|D,Q)$.

L'approche modélisation de langues permet de combiner deux composantes (indexation et Ranking) dans un seul modèle unifié. La plupart des modèles de langue développés pour la RI utilisent le principe de génération de la requête par un modèle de document. Se basant sur la représentation des documents et la fonction de Ranking, les approches de modélisation de langage pour la RI peuvent être classées en trois catégories :

o Génération de la requête par le modèle de document (Query Likelihood Models) :

Dans cette approche un modèle de langage est associé à chaque document. Les documents sont classés selon leurs probabilités de génération de la requête. Le score d'un document vis-à-vis d'une requête (la probabilité de générer une requête « Q » sachant le modèle de document « Md ») est donné par la formule suivante :

$$\text{Score}(D, Q) = P(D) \prod_{i=1}^n P(t_i / M_d)$$

Ou :

t_i : termes de la requête.

M_d : modèle de langage du document.

Cette catégorie de modèles de langage permet en particulier d'incorporer des informations sur la pertinence a priori du document, en utilisant le facteur $P(D)$ et en le conditionnant avec les caractéristiques de document.

o Génération de document à partir du modèle de la requête :

Cette approche procède dans le sens inverse. Ainsi, un modèle de langage de la requête est construit, ensuite les documents sont classés selon leurs probabilités que leur contenu soit généré par le modèle de la requête.

$$\text{Score}(Q, D) = \prod_{i=1}^m P(t_i | M_q)$$

Ou :

$t_i = t_1 t_2 \dots t_m$: termes du document.

M_q : modèle de langage de la requête.

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

o Similarité entre modèle de document et modèle de la requête :

Dans cette approche un modèle de langage est construit pour chaque document et un autre pour la requête. Les documents sont alors ordonnés selon la similarité de leurs modèles avec le modèle de la requête.

Ø Le lissage dans le modèle de langage

Plusieurs travaux de recherche se sont penchés sur le problème lié à l'utilisation des modèles de langage en RI et qui concerne la clair sémence de données (Data Sparseness) : dans les modèles de langage développés pour la RI, l'idée est de classer les documents selon la probabilité $P(Q|D)$ pour une requête « Q » donnée, généralement calculée par la multiplication des probabilités individuelles des termes de la requête $P(t_i|D)$. Cependant, si un document « D » contient tous les termes de la requête sauf un alors on lui attribue la probabilité nulle même si le document est pertinent. Pour y remédier la technique de lissage qui permet d'attribuer une probabilité non nulle aux termes non observés dans le document est utilisée. Par exemple «le lissage de Laplace, le lissage de Good-Turing, le lissage de backoff et le lissage par interpolation».

Zhai et Lafferty ont expérimenté plusieurs techniques de lissage en utilisant le modèle de langage uni-gramme et ils ont rapporté que la méthode de lissage par interpolation donne de meilleurs résultats.

La méthode de lissage par interpolation consiste à lisser le modèle de document « M_d » par le modèle de la collection « M_c ». La formule (1) devient alors :

$$\text{Score}(D, Q) = P(D) \prod_{i=1}^n \lambda P(t_i / M_d) + (1 - \lambda) P(t_i / M_c)$$

Avec :

λ : appartient à $[0,1]$.

II.12. RI comme génération de la requête par le document

II.12.1. Modèle de Ponte et Croft

Le premier modèle de langue est proposé par pont et croft [Ponte & Croft, 98], il est basé sur l'idée qu'une requête n'est pas créée de façon aléatoire, mais l'utilisateur a une idée (un modèle) sur le document idéal qui peuvent apparaître dans D et de son modèle M_D . À partir de M_D , l'utilisateur choisit (génère) les termes pour constituer sa requête. Ainsi, la requête devrait être générée à partir d'un document pertinent ou de son modèle. La probabilité de cette génération est considérée comme le score du document :

Nous apportons quelques remarques ici :

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

- Une requête peut être considérée comme une suite de mots : $Q = t_1, t_2, \dots, t_n$.
- Comme dans la plupart des modèles de langues utilisées en RI, la simplification suivante a été faite : on suppose que les mots dans une requête sont indépendants.
- Dans le modèle de Ponte et Croft [Ponte & Croft, 98], ils utilisent 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 d'autres modèles, on utilise plutôt le modèle multinomial, où on ne considère que les mots présents dans la requête.

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

$$P(Q | M_D) = P(t_1 t_2 t_3 \dots t_n | M_D) \times P(\neg t_{n+1} \neg t_{n+2} \neg t_{n+3} \dots \neg t_l | M_D) = \prod_{i=1}^n p(t_i | M_D) \times \prod_{i=n+1}^l p(\neg t_i | M_D) = \prod_{t_i \in Q} p(t_i | M_D) \times \prod_{t_i \notin Q} p(1 - P(t_i | M_D))$$

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 :

$$P_{PC}(t_i \setminus D) = \begin{cases} P_{ML}(t_i \setminus D)^{1-R(t_i, D)} \times P_{avg}(t_i)^{R(t_i, D)} & \text{Si } tf(t_i, D) > 0 \\ P_{ML}(t_i \setminus C) & \text{Sinon} \end{cases}$$

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

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

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, est une fonction de « risque » déterminée comme suit : $R(t_i, D)$.

$$R(t_i, D) = \frac{1}{1+f_t} \times \left(\frac{f_t}{1+f_t} \right)^{tf(t_i, D)}$$

Où,

f_t Est la fréquence moyenne du mot t dans les documents qui le contient. L'élément $P_{avg}(t_i)^{R(t_i, D)}$ est ajouté pour contrer le « risque » de l'estimation $P_{ML}(t_i \setminus D)^{1-R(t_i, D)}$ 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.

II.12.2. Modèle de Hiemstra et al. Et de Miller et al.

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

$$\text{Score}(D, Q) = P(D \setminus Q) = P(D \setminus t_1 t_2 \dots t_n) =$$

$$P(D) \frac{P(t_1 t_2 \dots t_n \setminus D)}{P(t_1 t_2 \dots t_n)}$$

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

$$\text{Score}(D, Q) = cP(D) \prod_{t \in Q} P(t_i \setminus D)$$

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

$$P(t_i \setminus D) = \alpha P_{ML}(t_i \setminus D) + (1 - \alpha)P_{ML}(t_i \setminus C) \quad (2)$$

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

$$\frac{tf(t_i, D)}{|D|}, \text{ Et } \frac{df(t_i)}{\sum_{t_j \in V} df(t_j)}$$

Une fois on prend le logarithme, on obtient :

$$\begin{aligned} \log(P(t_1 t_2 \dots t_n \setminus D)) &= \log \prod_{t \in Q} P(t_i \setminus D) \\ &= \sum_{i=1}^n \log(\alpha P_{ML}(t_i \setminus D) + (1 - \alpha)P_{ML}(t_i \setminus C)) \\ &= \sum_{i=1}^n \log\left(1 + \frac{\alpha P_{ML}(t_i \setminus D)}{(1 - \alpha)P_{ML}(t_i \setminus C)}\right) + \sum_{i=1}^n \log((1 - \alpha)P_{ML}(t_i \setminus C)) \end{aligned}$$

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

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

La probabilité a priori d'un document D est estimée comme suit :

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

En somme, on a : $\log \text{Score}(Q, D) = \sum_{t \in Q} \log\left(1 + \frac{\alpha * tf(t, D) \sum_{t_i \in V} df(t_i)}{(1 - \alpha)|D|df(t)}\right) + \log \frac{|D|}{|C|}$

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

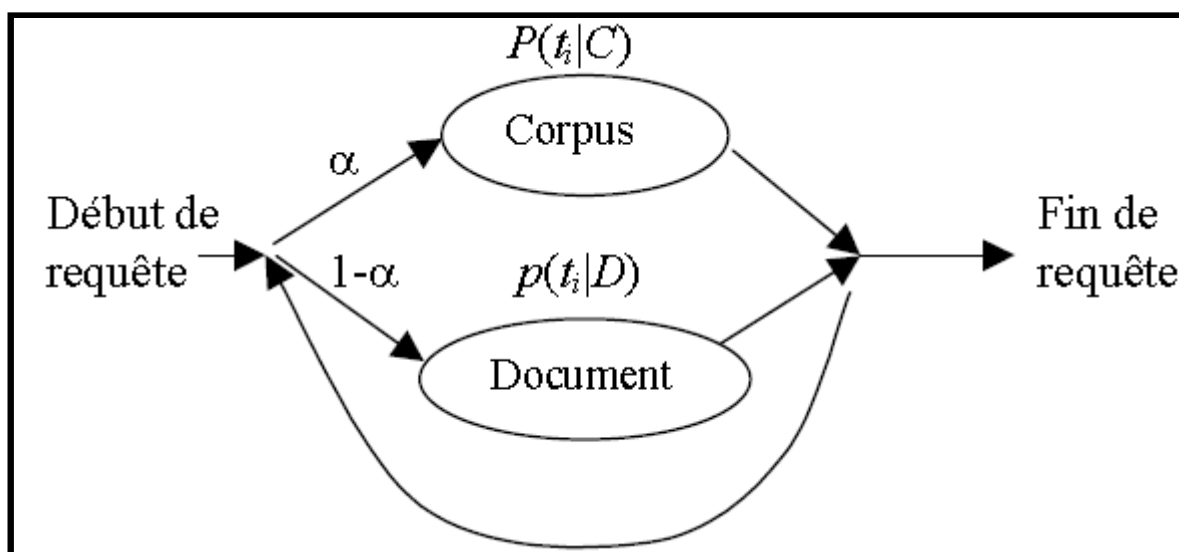


Figure 5. Modèle de Markov caché à deux états

Ce modèle correspond à la formule suivante :

$$P(Q|D \text{ est pertinent}) = \prod_{t_i \in Q} \alpha P_{ML}(t_i | D) + (1 - \alpha) P_{ML}(t_i | C)$$

Ce qui est similaire au modèle de Hiemstra .

Bien que Hiemstra et Miller aient tous utilisé le lissage par interpolation, leur intention initiale n'était pas pour traiter la clairescence de données, mais pour mieux modéliser la requête, dans laquelle certains mots peuvent être non-pertinents. Ainsi, dans la formulation de Hiemstra, un autre élément $tf(t, Q)$ – la fréquence du terme t dans la requête Q – est inséré pour tenir compte de ce fait. Dans notre présentation, nous n'en avons pas décrit les détails.

II.13. Pertinence a priori d'un document :

Selon l'approche adoptée les propriétés (taille de document, nombre de liens entrants, etc.) indépendantes des requêtes peuvent être utilisées pour conditionner la probabilité a priori de pertinence d'un document. Si la probabilité a 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 a priori de pertinence de document peut être ignorée lors du classement des documents.

Par contre, si la probabilité a priori est conditionnée par l'une de ces caractéristiques alors les documents de la collection n'ont pas la même probabilité a priori, et les documents ne sont pas équiprobables. Par exemple, si la caractéristique utilisée est le score de popularité de 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 a priori d'un document comme : la longueur du document [Miller, 99], la structures des liens [Hauff & Azzopardi, 05][Kraaij & Hiemstra, 02], le facteur temps[Diaz & Jones, 04],[Lavrenko, 01], le rapport

Information/Bruit[Zhu & Gauch, 00]. Elles expriment qu'un document est plus probable parce que : il est plus long, il est plus populaire, il est plus récent, ou contient plus d'informations que de bruit.

Les caractéristiques les plus souvent utilisées sont :

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

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

Avec $|D|$ 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 [Miller, 99], [Kraaij, 02].

- Ø **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. :

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

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

- Ø **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, 00]:

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

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.

- Ø **Type d'URL du document** : Kraaij et al [Kraaij, 02] ont utilisé le type d'URL pour estimer la probabilité qu'une page soit une page d'entrée.

$$P(D) = P(PE \setminus URLtype(D) = t_i) = \frac{c(PE, t_i)}{c(t_i)}$$

Où URL_{type} 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` Sur la base de ces quatre types d'URL ont mené des expérimentations sur la collection WT10g (collection utilisée dans TREC 2001) pour estimer la probabilité qu'une page soit une page d'entrée sachant son type d'URL. Il a été constaté que cette information est un bon indicateur pour prévoir la pertinence d'une page d'être une page d'entrée.

II.14. Conclusion :

Dans ce chapitre nous avons présenté les principales notions, concepts, et outils de recherche sur le web.

A travers les différentes sections que nous avons exposées, nous concluons que la recherche d'information, s'attache à définir des modèles et des systèmes afin de faciliter l'accès à un ensemble de documents se trouvant dans des bases documentaires ou encore sur le web. Le but est de permettre aux utilisateurs de retrouver les documents dont le contenu répond à leur besoin en information, il s'agit donc de retourner l'ensemble des documents pertinents.

Jusqu'ici la pertinence d'un document compte les caractéristiques du document lui même, nous essayerons, dans le chapitre suivant, d'introduire et d'implémenter d'autre caractéristiques, afin de calculer cette pertinence.

III.1. Introduction :

Dans ce chapitre, nous allons présenter en premier lieu, l'approche a implémenté, ensuite, le support matériel et logiciel qui nous a servi d'appui pour le développement de notre travail. Puis, Nous exposerons les expérimentations effectuées pour l'évaluation de notre approche en utilisant la collection TREC .GOV. L'implémentation de notre approche est faite sous la plate forme Terrier (3.0). On terminera par une conclusion.

III.2. Approche à implémenté :

La recherche d'information sur le web privilégie la minimisation du temps de réponses par rapport à la qualité des documents retournés à l'utilisateur. Ceci vient du faite que ce n'est pas toutes les dimensions d'un document web dans le processus d'indexation et de recherche qui sont prie en compte.

Pour estimer la pertinence d'un document D par apport à une requête Q, le modèle de langue procède ainsi :

$$\text{Score (D, Q)} = \underbrace{P(D)}_{(A)} \underbrace{\prod_{i=1}^n \lambda P(t_i / M_d) + (1 - \lambda) P(t_i / M_c)}_{(B)}$$

Cette formule combine deux sources d'informations : (A) probabilité a priori de pertinence et (B) le contenu du document.

Pour estimer la probabilité à priori de pertinence $P(D)$; plusieurs caractéristiques sont utilisées (citées en section 13 chapitre II). Néanmoins, Les caractéristiques utilisées jusqu'ici dépendent du document web seulement. Or, une page web fait partie en général d'un site lequel fait partie du web. L'idée que l'on veut explorer est l'utilisation des caractéristiques du site web dans le but de 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 ».

a. Utilisation du facteur « PageRank »

Dans l'intérêt d'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 nous introduisons le facteur importance du site web(page principale) dans l'évaluation de l'importance de la page. Les formules proposée comme suite, exprime la probabilité à priori de pertinence d'une page, qui intègre l'importance du site qui la contient, la différence entre les deux, est que la formule (1) se base sur les propriétés de la page elle même, sans faire appel au site web qui la contient, par contre la formule (2) utilise le paramètre PageRannk du site :

$$\text{Prior}(\pi) = \log (\text{PageRank_p} (\pi) + 1) \dots \dots \dots (1)$$

$$\text{Prior}(\pi) = \log [\lambda (\text{PageRank_p}(\pi)) + (1 - \lambda)(\text{PageRank_s})] \dots \dots \dots (2)$$

Tels que :

λ : paramètre variable, sa valeur est comprise entre 0.1 et 0.9, il est utilisé dans la formule de la pertinence a priori.

PageRank_p (pi) : PageRank de la page pi.

PageRank_s : PageRank du site contenant la page pi.

b. Intégration du nombre de page par site :

En d'autres termes, on associe à chaque site le nombre de page web qu'il contient, Dans le but d'assigner plus d'importance au site qui contient le moins de pages.

La formule proposée, définit la probabilité à priori de pertinence d'une page qui utilise le paramètre « nombre de pages par site » pour exprimer l'importance du site qui la contient :

$$\text{Prior (pi)} = \log (1 + (\lambda * \text{PageRank(pi)}) + ((1 - \lambda) * (\text{PageRank}_s / \text{nbr_p_s})) \dots (3)$$

Nbr_p_s : nombre de page par site (site contenant la page pi)

c. Intégration du niveau de la page

On s'est basé sur l'hypothèse suivante : « on doit assigner plus de confiance aux pages qui sont proche de la page principale (site). De ce fait, plus le niveau est grand plus les pages s'éloignent du site, et donc moins importante elles sont », le niveau d'une page est définis par le nombre de « / ». Concernant la page principale, son niveau est : 0.

$$\text{Prior (pi)} = \log (1 + (\lambda * \text{PageRank_p(pi)}) + ((1 - \lambda) * (\text{PageRank}_s / (\text{niveau} + 1))) \dots (4)$$

Niveau : niveau de la page (pi).

Exemple :

www.fas.usda.gov/oilseeds/circular/2001/01-09/table9.pdf est l'URL d'une page web, composé de la page principale (le site) : www.fas.usda.gov, le niveau de page est de cinq(5).

d. Intégration de la similarité de la page avec son site

En partant de l'hypothèse qu'une page web qui est similaire avec le site qui la contient doit avoir une grande importance dans son site ; pour cela une telle page recevra une partie importante du score de son site (PageRank)

$$\text{Prior (pi)} = (\text{Sim} * \text{PageRank_p (pi)}) + (\text{Sim} * \text{PageRank}_s) \dots (5)$$

Tel que:

Sim: similarité de la page avec son site.

III.3. Architecture Logicielle :

Le schéma suivant illustre la structure des différents programmes et données de l'implémentation de l'approche décrite.

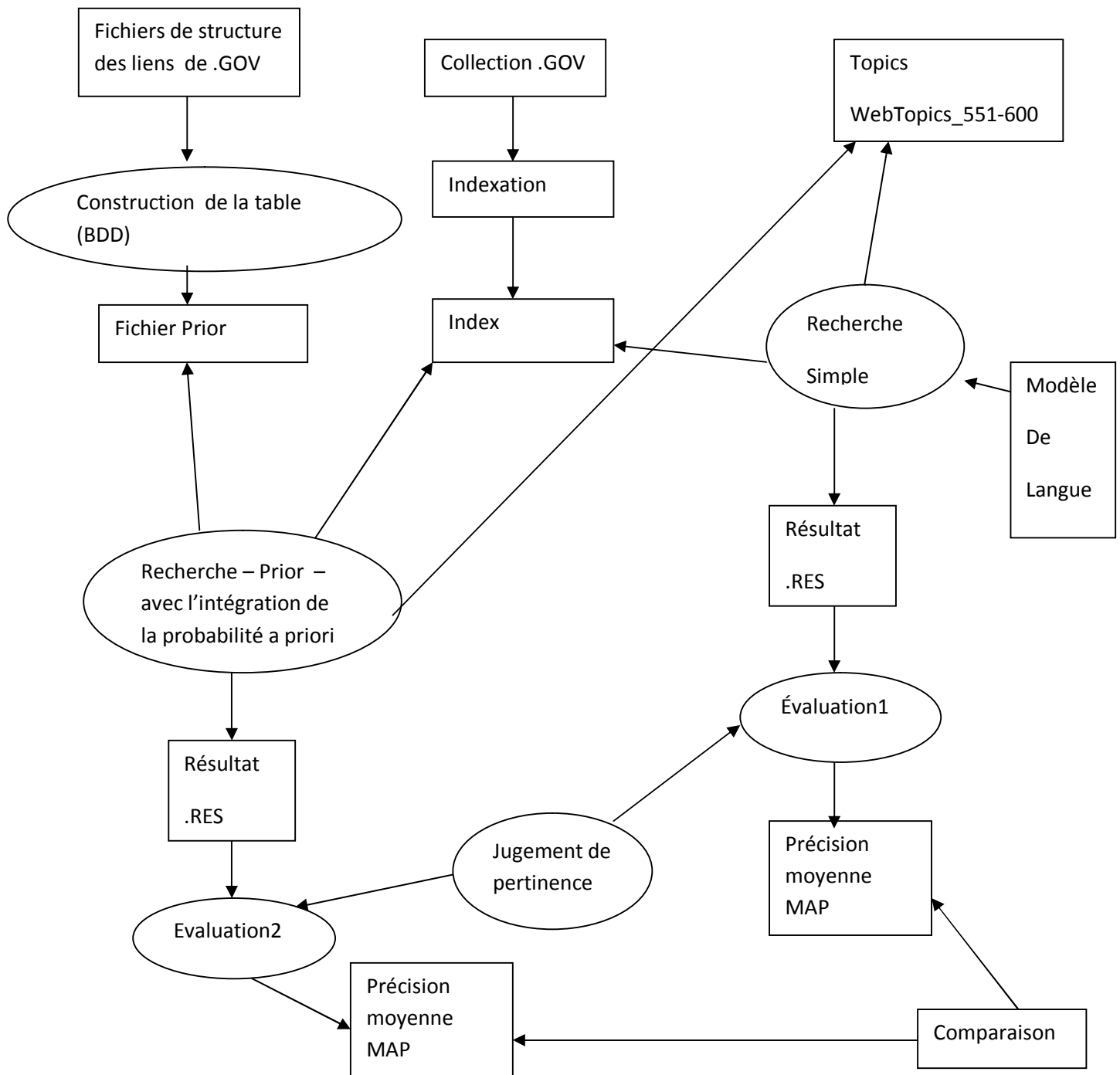


Figure 1 : Schéma l'Architecture Logicielle

III.3.1. Description des données utilisées

1) La collection .GOV : Après TREC 2001, les développeurs Web se sont penchés sur l'idée de construire une nouvelle collection web pour les tests. Ces derniers ont pensé aux pages web du gouvernement des États-Unis, qui sont une vaste et très diversifiée collection d'informations et de services relativement libres de droits d'auteur et de distribution, idéal pour un premier essai.

Après beaucoup de discussions sur ce que cette nouvelle collection doit contenir, Charlie Clarke, de l'Université de Waterloo rassemble 95GB, environ 1 million de pages .Gov en Janvier 2002 à l'aide des systèmes hébergés en Virginie tech. L'ensemble recueilli comprend des pages web HTML, des images associés à Microsoft Word, Postscript, ainsi que des fichiers PDF.

Nick Craswell du CSIRO a traité l'ensemble des données rassemblées pour obtenir ce que l'on appellera finalement la collection .GOV, et donc les documents ont été tronquées à 100KB, assemblés en grappes, et se sont vu attribué des identifiants. Les Textes ASCII ont été extrait à partir du Word, Postscript, et d'autres fichiers PDF en utilisant des outils libres et gratuits.... La collection résultante contient 1,2 million de documents textuels et sa taille est de 18.1GB.

Documents:	1247753 (1,25 millions)
texte / html	1053372 (types MIME rapportés par le serveur)
application / pdf	131333
application / msword	13842
application / postscript	5673
Autre chose qui s'est avéré être du texte	44
Taille totale:	19455030550 = 18.1G
Longueur de troncature Doc:	100kb
Docs sans paroles	55

Tableau 1 : les composants de la collection .GOV

2) Les fichiers décrivant la structure des liens de la collection .GOV :

- a) **Fichier links_id** : Contiens la description de la structure de lien entre les documents de la collection. La structure de ce fichier est une structure en ligne, chaque ligne indique : l'identificateur de la page pointant, et l'identificateur de la page pointée, Ces deux identificateurs sont séparés par un espace (blanc). Comme illustré dans la figure 2.

```
G00-00-0000000 G00-33-2790351
G00-00-0000000 G01-85-0285437
G00-00-0000000 G01-59-3325977
G00-00-0000000 G01-01-1404866
G00-00-0000000 G00-31-1234802
G00-00-0000000 G01-06-1330028
G00-00-0000000 G00-34-3481979
G00-00-0000000 G00-17-3380592
G00-00-0000000 G00-33-1791477
G00-00-0000000 G01-20-0000000
G00-00-0000000 G01-94-0011373
G00-00-0000000 G00-14-3440992
G00-00-0000000 G01-26-0321033
G00-00-0000000 G01-33-2181585
G00-00-0000000 G01-22-1883781
G00-00-0000000 G00-41-2384290
G00-00-0000000 G01-49-3598750
G00-00-0000000 G01-71-3558332
G00-00-0000000 G01-85-1154012
G00-00-0000000 G00-82-1537742
G00-00-0000000 G00-30-4085429
G00-00-0000000 G01-05-2607363
G00-00-0000000 G00-69-1021503
G00-00-0000000 G01-87-3919155
G00-00-0000000 G01-34-2253096
G00-00-0000000 G00-45-0466544
```

Figure 2 : extrait du Fichier Links_id

Par exemple, la ligne encadrée en rouge présente deux parties :

G00-00-0000000 : c'est l'identificateur de la page web.

G01-01-1404866 : c'est l'identificateur de la page pointée par la première page
(**G00-00-0000000**).

- b) Fichier Url2id** : chaque ligne de ce fichier contient l'URL de la page web ainsi que son identificateur, ces dernières sont séparées par un espace.

```

www.epa.gov/pesticides/cumulative/prac-op/iii_q_2-f.pdf G46-00-
0000000
www.fas.usda.gov/oilseeds/circular/2001/01-09/table9.pdf G46-
00-0004743
www.fakr.noaa.gov/npfmc/Newsletters/1000news.pdf G46-00-
0016738
www.fakr.noaa.gov/1997/news.htm G46-00-0109349
www.opm.gov/fedclass/fwsleadr.pdf G46-00-0125616
www.state.gov/www/regions/wha/colombia/001120
mccaffery_colombia.html G46-00-0168766
www.gpo.gov/contractappeals/decisions/01-79.htm G46-00-0190294
www-didc.lbi.gov/NetLogger/API/fortran//write-gt.html G46-00-
0206341
nssdc.gsfc.nasa.gov/planetary/text/asteroid_pr_980109.txt G46-
00-0208853
www.mied.uscourts.gov/_jurorinfo/basics.htm G46-00-0214635
www.bisnis.doc.gov/nis/docs/ngolist.htm G46-00-0232316
www-ne.cr.usgs.gov/highplains/hp98_web_report/PDF/fig6.pdf
G46-00-0262635
www.ecd.bnl.gov/pubs/BNL64934.pdf G46-00-0263345
naca.larc.nasa.gov/reports/1945/naca-tn-951/index.cgi?
thumbnail160 G46-00-0331800
www.notes.dol.gov/said.nsf/f6611e5257258aa8852564c200718c5e?
OpenView G46-00-0334058
nccam.nih.gov/ne/press-releases/archive.html G46-00-0342770
www.moept.uscourts.gov/moept/documents.nsf/519a025470af2daf882
56406008016b7?OpenView&Start=1 G46-00-0366859
www.rdc.noaa.gov/~facmd/ssmc_jan2002_update.htm G46-00-0371397

```

Figure 3 : Extrait du fichier Url2id

Par exemple la ligne encadrée en rouge montre :

www.gpo.gov/contractappeals/decisions/01-79.htm : qui est L'Url de la page web, son niveau est égal à 3. Le site web contenant la page est www.gpo.gov.

G46-00-0190294 : l'identificateur de la page.

III.4. L'environnement technique de développement :

III.4.1. Présentation du matériel :

Pour réaliser notre application, nous avons utilisé une machine ayant les caractéristiques suivantes :

- Processeur : Intel(R) Core(TM) Duo CPU.
- Fréquence d'horloge 2GHz
- RAM de 2GO

III.4.2. L'environnement Logiciel :

Pour réaliser notre application, nous avons travaillé sous le système d'exploitation Microsoft windows 7 version integrale. Et nous avons utilisé les outils suivant : Terrier comme plateforme de RI, Netbeans comme environnement de programmation, Java comme langage de programmation, PhpMyadmin comme outils de gestion de BDD.

1) Présentation de la plateforme Terrier 3.0 :

Terrier (TerabyteRetrieval) est un logiciel libre et open source, classé dans la catégorie des SRI, Sa création fut par le département d'informatique de l'université de Glasgow au Royaume-Uni.

Son fonctionnement repose sur différentes méthodes d'indexation, plusieurs modèles de pondération et il permet la reformulation des requêtes.

Deux principales étapes caractérisent terrier :

- L'Indexation
- La recherche
- Evaluation

1.1) Processus d'indexation :

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

1.2) Construction de l'index :

Le but de cette étape est d'effectuer plusieurs traitements afin de parvenir à la création de la structure d'index. La base de ces fichiers est le lexicon qui est composé de terme et de sa fréquence d'occurrence. Le stockage des différentes statistiques se fait sur les quatre principaux types de fichiers :

- **Document index** : contient des informations à propos des documents tels que leurs identificateurs, leurs tailles, etc.
- **Lexicon et Lexicon index** : sauvegarde le vocabulaire.
- **Fichier inverse** : index inverse
- **Fichier directe** : index directe.

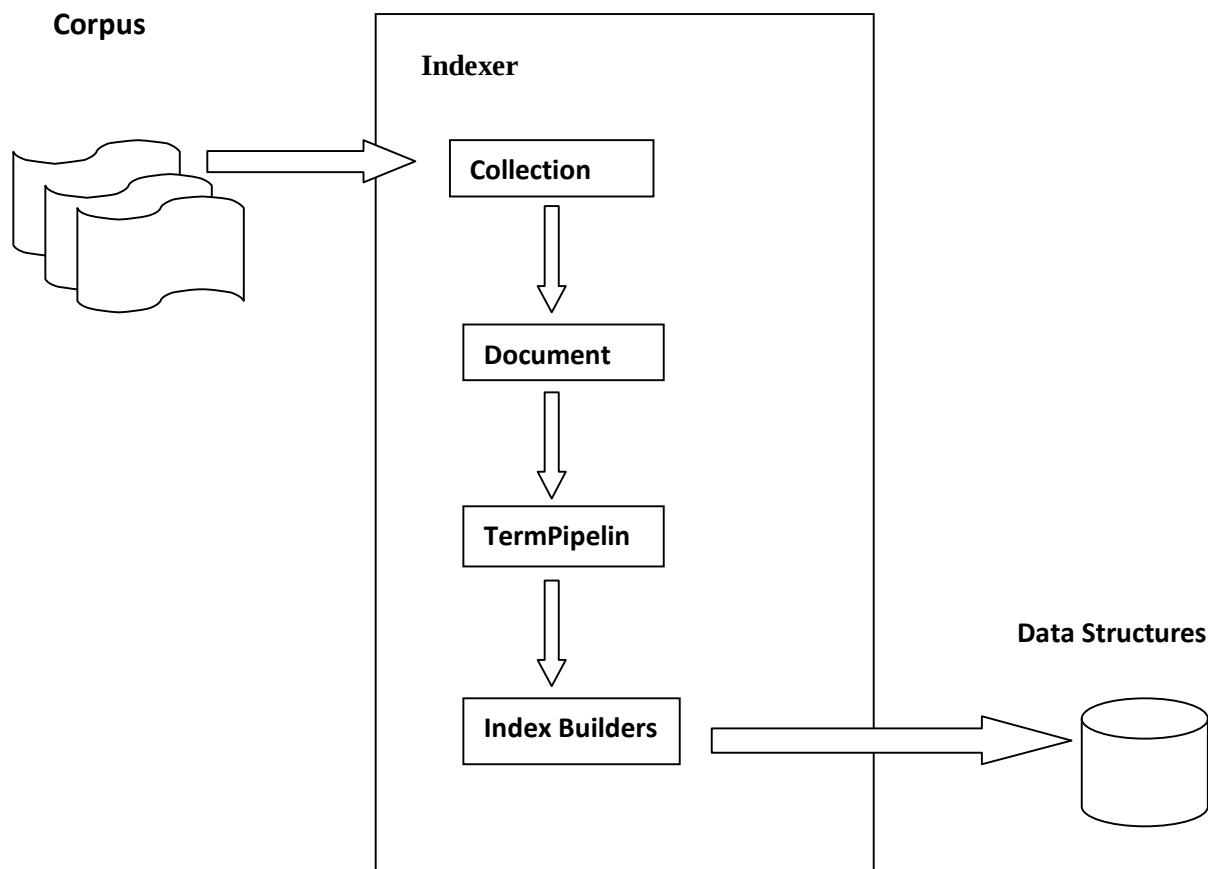


Figure 4. Présentation du processus d'indexation de Terrier.

Les Modules Collection, Document, TermePipeline, Index Builder font partie de l'API d'indexation de Terrier, on les présentera en détails au niveau de l'annexe (1).

1.3) La recherche sur terrier :

La recherche s'effectue sur les index créés après analyse de la requête par le système, cela dans le but de retourner des documents triés selon l'ordre décroissant de leurs score.

Pour effectuer cette tâche terrier suit les étapes suivantes :

Analyse : reconnaissance des mots de la requête. Elle se fait de la même manière que dans l'étape d'indexation.

- o **Prétraitement** : contient des processus d'élimination des mots vide, lemmatisation, normalisation des termes.
- o **Appariement et attribution de score** : assignation des scores aux documents visant à décrire la pertinence de ces derniers vis-à-vis de la requête en utilisant différents modèles de pondération. Parmi ces modèles nous pouvons citer :
 - TF-IDF : la formule TF est donnée par Robertson et IDF par Sparck Jones.
 - Lemur TF-IDF : le modèle proposé par Lemure
 - BM25 : le modèle probabilistique BM25.

- Hiemstra _LM : Modèle de langage proposé par Hiemstra.
- o **Post-traitement** : permet de modifier les résultats de plusieurs manières, tel que : la reformulation de la requête.
- o **Filtrage** : permet soit l'inclusion ou l'exclusion d'un document. Elle est surtout utilisée pour les besoin de restriction de l'utilisateur pour un domaine particulier.
- o **Résultat** : une fois les étapes précédentes effectué, terrier s'occupe de retourner un ensemble de documents triés selon l'ordre décroissant de leurs scores.

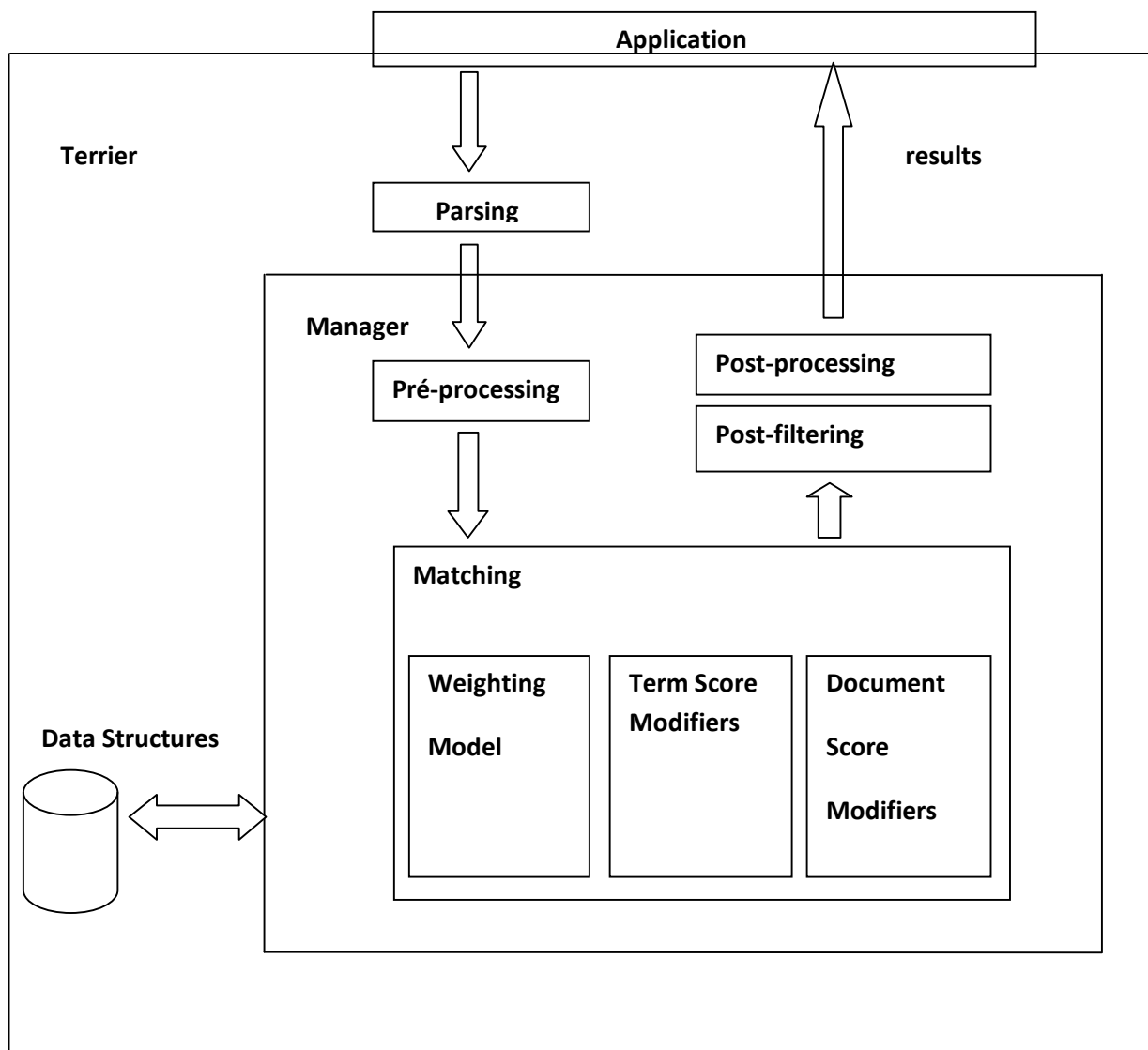


Figure 5 : Présentation du processus de recherche de Terrier

Terrier compte trois composantes principales pour la recherche : *Query*, *Manager*, *Matching*, seront Vue en détails dans l'annexe (1).

2) Le langage de programmation utilisé (java) :

Le choix de JAVA comme langage de programmation s'est imposé vu que l'Open Source de Terrier est entièrement écrit en JAVA. De plus Java est un langage de programmation récent (les dernières versions datent de 1995) développé par Sun Microsystems. Il est fortement inspiré des langages C et C++.

Le langage Java trouve son origine dans les années 1990. A cette époque, James Gosling (Sun Microsystems) développe un premier langage, Oak, permettant de programmer dans un environnement indépendant du Hardware. En 1992, Bill Joy (cofondateur de Sun Microsystems) face à une puissante montée d'internet, insiste sur l'élaboration d'un langage indépendant des plates formes et des environnements (et ceci vient des problèmes d'hétérogénéité Matériel et Logiciel utilisé par Internet), des lors, Oak est renommé JAVA en 1995. Une machine virtuelle, un compilateur ainsi que de nombreuses spécifications sont données gratuitement et JAVA entame une conquête fulgurante.

Aujourd'hui, après de nombreuses améliorations n'est plus uniquement une solution liée à internet. De plus en plus de sociétés font appel à ce langage de programmation pour l'ensemble de leurs développements (applications classiques, interface homme-machine, application réseaux, ...etc).

3) Les principales raisons du succès de java :

Java a rapidement intéressé les développeurs pour quatre raisons principales :

- Ø C'est un langage orienté objet dérivé du C, mais plus simple à utiliser et plus pur que le C++. On entend par pur le fait qu'en java on ne peut faire que de la programmation orienté objet contrairement au C++ qui reste un langage hybride, c'est-à-dire autorisant plusieurs styles de programmation.
- Ø Il s'est doté en standard de bibliothèques de classe très riches comprenant la gestion des interfaces graphiques (fenêtres, boîte de dialogue, contrôles, des exceptions, les accès aux fichiers et aux réseaux...
L'utilisation de ces bibliothèques facilite grandement la tâche du programmeur lors de la construction d'applications complexes.
- Ø Il est doté en standard d'un mécanisme de gestion d'erreurs (les exceptions) très utile et très performant. Ce mécanisme, inexistant en C, existe en C++ sous la forme d'une extension du langage beaucoup moins simple à utiliser qu'en java.
- Ø Il est multi plates-formes : les programmes tournent sans modification sur tous les environnements où java existe (Windows, Unix et Mac).

4) Présentation de NetBeans :

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution Licence). En plus de java, NetBeans permet également, comme python, C, C++, JavaScript,

XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un EDI moderne (éditeur en couleur, projet multi-langage, refactoring, éditeur graphique d'interfaces et de page Web).

Conçu en java, NetBeans est disponible sous Windows, linux, Solaris (sur X86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle java JVM). Un environnement Java Development Kit JDK est requis pour les développements en Java. NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing(Java)). L'IDE NetBeans s'appuie sur cette plateforme.

Dans notre travail on a utilisé la version NetBeans 7.0.1, la figure suivante illustre son interface Principale.

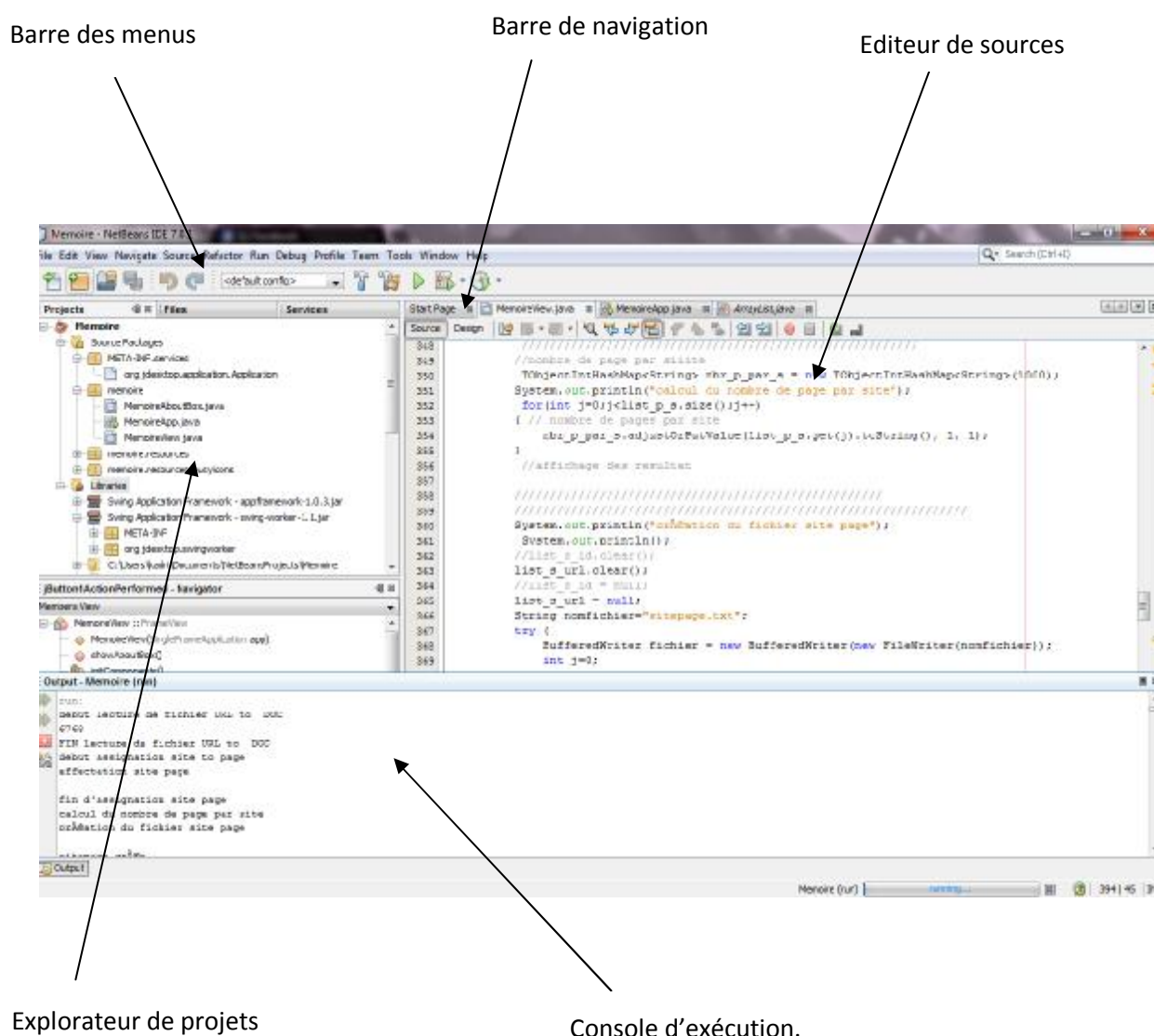


Figure 6 : environnement de développement NetBeans.

5) Présentation PhpMyAdmin

PhpMyAdmin est l'une des plus célèbres interfaces pour gérer une base de données MySQL sur un serveur PHP. De nombreux hébergeurs, qu'ils soient gratuits ou payants, le proposent ce qui permet à l'utilisateur de ne pas avoir à l'installer.

Cette interface pratique permet d'exécuter, très facilement et sans grandes connaissances dans le domaine des bases de données, de nombreuses requêtes comme les créations de table de données, les insertions, les mises à jour, les suppressions, les modifications de structure de la base de données. Ce système est très pratique pour sauvegarder une base de données sous forme de fichier .SQL et ainsi transférer facilement ses données. De plus celui-ci accepte la formulation de requêtes SQL directement en langage SQL, cela permet de tester ses requêtes par exemple lors de la création d'un site et ainsi de gagner un temps précieux.

Création de la base de données

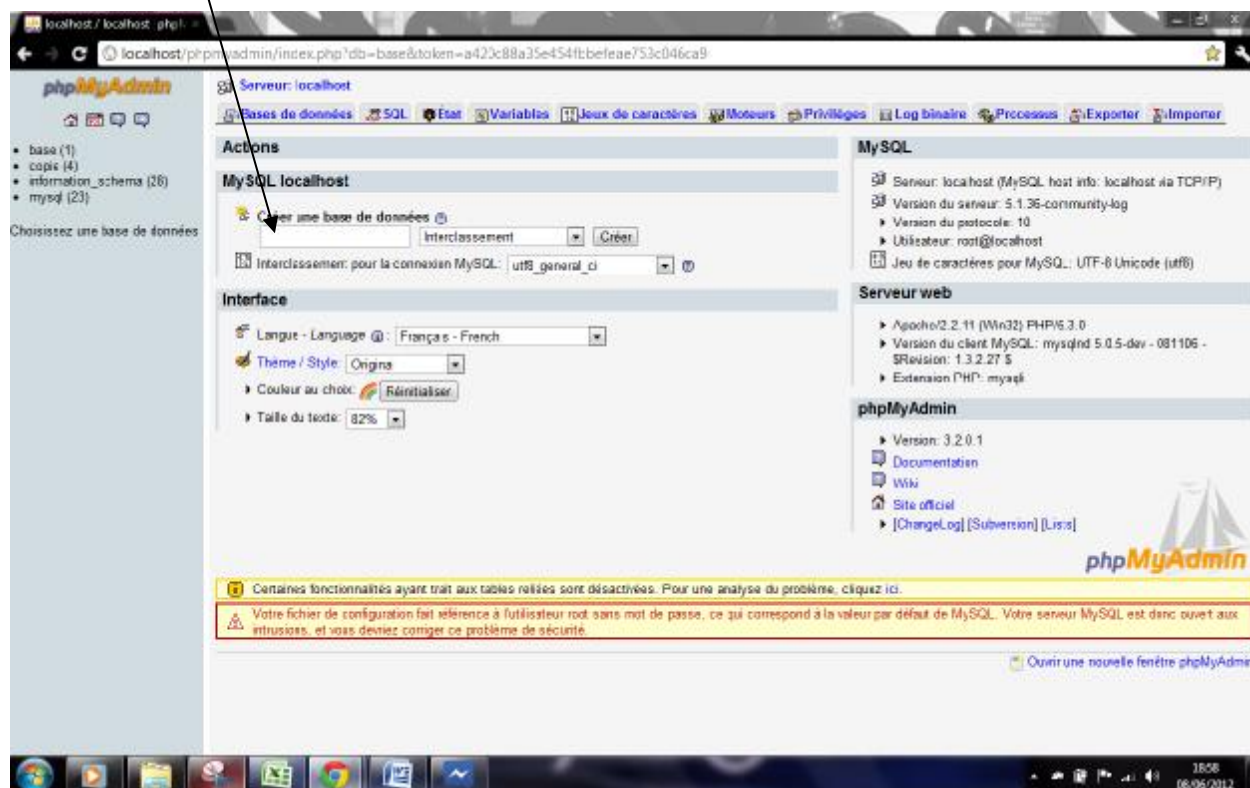


Figure 7. Accueil de PhpMyadmin

- **Création d'une table sous PhpMyAdmin :** L'aspect "relationnel" des SGBDR est caractérisé en partie par la possibilité de regrouper dans un même fichier des données hétérogènes. L'idée des tables est donc d'abstraire la notion de fichiers, et de définir tout simplement ce que contiendra chaque *enregistrement* de la table.

Figure 8 : créer une table sous PhpMyadmin

Figure 9 : remplissage des champs de la table

III.5. Etapes d'implémentation de notre approche:

Pour implémenter notre approche, nous utiliserons les fichiers de structures des liens de notre collection TREC .GOV présentées ci-dessus, comme données afin d'exécuter le programme 1 correspondant au code source qui va nous permettre de calculer les caractéristiques auquel on fera appel par la suite, pour conditionner la probabilité a priori de pertinence d'une page web par apport au site qui la contient, les résultats de l'exécution sont présentés par Résultat 1, ils sont notés dans une table BDD.

Le programme 2, comprend l'évaluation des formules qu'on utilisera pour calculer la pertinence de la page web tout en intégrant les caractéristiques du site qui la contient (resultat 1). Enfin, Résultat 2, représente les résultats retournés par programme 2, construisant ainsi des fichiers dit Priors.

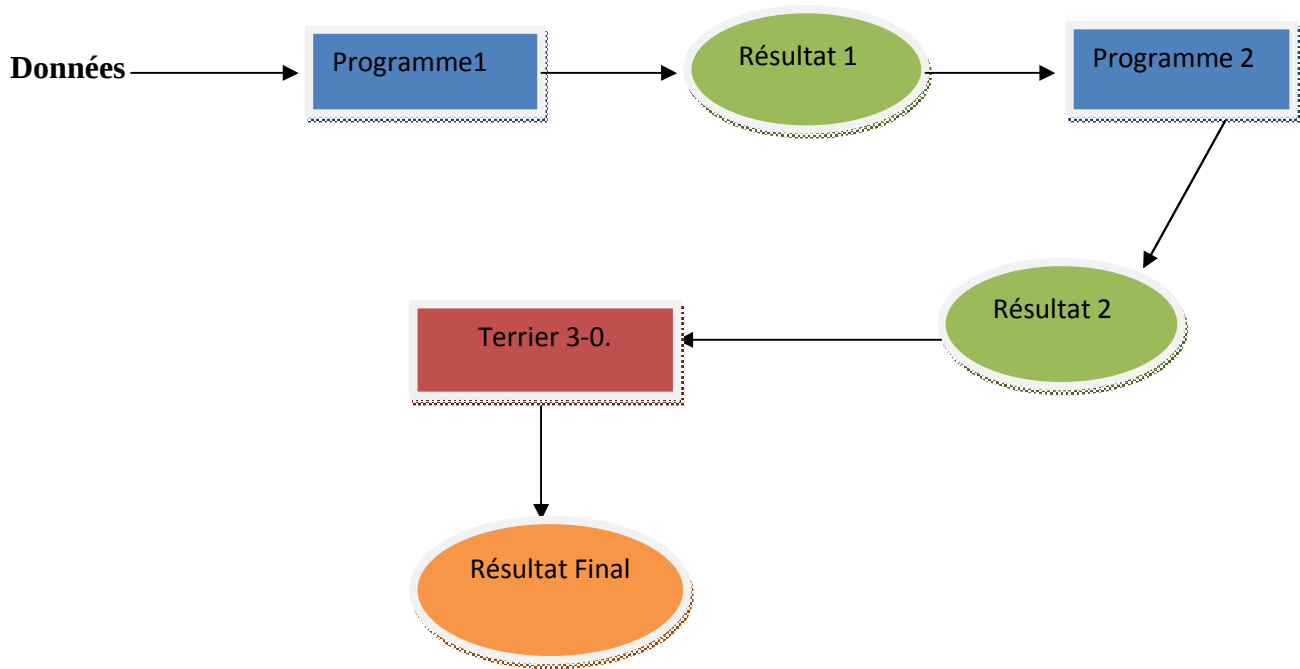


Figure 10 : Schéma de L'implémentation

III.5.1. implémentation et évaluation des formules (1) et (2)

- **Programme 1** : les données seront utilisées dans le but de calculer le nombre de liens sortant de chaque page, le PageRank de chaque page, et le PageRank de chaque site.

Algorithme PageRank

Var Nb_p : entier \\\ nombre de pages contenues dans le site.

Nb_L_E_i : entier \\\ nombre de liens entrant vers la page i.

Nb_L_S_j : entier \\\ nombre de liens sortant de la page j.

Somme :entier

PRi : réel \\\ PageRank de la page i.

i,j : entiers \\\ indice indiquant les pages.

m : entier

Const d : réel \\\ facteur de probabilité compris entre 0 et 1.

Debut

Pour m allant de 0 jusqu'à 50 faire

debut

Pour i allant de 0 jusqu'à Nb_p faire

Debut

Somme =0 ;

Pour j allant de 0 jusqu'à NB_p faire

Debut

Si (j pointe sur i = Vraie) **alors**

Debut

Nb_L_E_i= Nb_L_E_i + 1;

Calculer (PR j);

Calculer (Nb_L_S_ j);

Fin

Somme= Somme + [(PR j) \ (Nb_L_S_ j)] ;

Fin

PRj = (1-d) + d* Somme ;

Fin

Fin

Fin

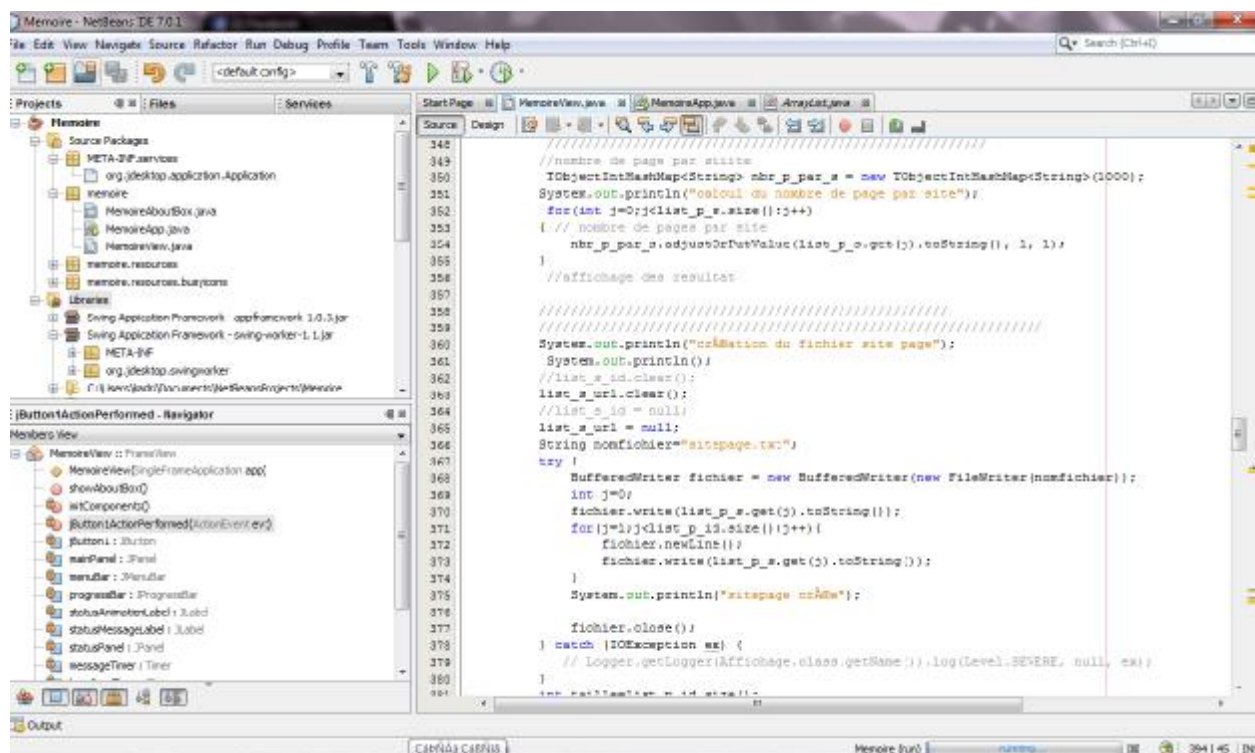


Figure 11 : Programme 1 d'exécution Java

- **Résultat 1 :** les résultats obtenus sont mis dans une table préalablement construite à quatre (4) champs. Son schéma est le suivant :

Id_page : identificateur de chaque page.

Nbrlien_sp : nombre liens sortant de chaque page.

Prank_p : PageRank de chaque page.

Prank_s : PageRank de chaque site.

		id_page ▲	nbrlien_sp	prank_p	prank_s
		G02-62-3400627	31	23	1
		G02-62-3421639	3	1	3
		G02-62-3423824	42	1	5
		G02-62-3450602	11	1	3
		G02-62-3457623	0	1	2
		G02-62-3479680	0	2	2
		G02-62-3480282	0	1	16
		G02-62-3489574	3	1	1413
		G02-62-3535149	4	1	115
		G02-62-3537375	0	2	5
		G02-62-3583583	3	2	15
		G02-62-3588881	1	1	14
		G02-62-3594331	15	2	3
		G02-62-3602926	0	1	230
		G02-62-3611822	22	1	5
		G02-62-3714230	10	2	16
		G02-62-3730846	2	1	27
		G02-62-3733335	45	2	2
		G02-62-3766154	0	2	413
		G02-62-3772304	0	3	54
		G02-62-3773044	0	1	7
		G02-62-3875452	13	0	3
		G02-62-3921276	1	1	229
		G02-62-3926525	6	1	38
		G02-62-3932879	31	2	464
		G02-62-3953584	1	0	63

Figure 12 : Partie de la table de la base de données.

Exemple :

La ligne encadrée en rouge illustre la structure de notre base de données faite comme ceci : La première case nous permet de voir l'identificateur de la page web (G02-62-3423824), la seconde montre le nombre de lien sortant de chaque page (42), la troisième case contient le PageRank de la page (1) et la dernière contient le PageRank du site contenant la page (5).

- **Programme 2 :** on prendra les résultats de la table retournée par le programme 1 dans l'intérêt de calculer la pertinence à priori de chaque page $P(D)$. Cela en utilisant les formules (1) et (2)
- **Résultat 2 :** En faisant varier la valeur de λ , on obtient à chaque fois un fichier texte dit « Prior λ ». Au total, on aura neuf fichiers priors pour la formule (2), et un seul concernant la formule (1). Chaque ligne de ce fichier Prior contient l'identificateur de la page web suivi de sa pertinence a priori. Ces derniers sont séparés par un espace. La ligne encadrée dans la figure qui suit montre cela.

La page dont l'identificateur est : G00-00-0028863, possède une pertinence à priori d'odore : 0,176091259.

Prior1 - Bloc-notes	
Fichier	Edition
G00-00-0000000	0.079181246
G00-00-0008511	0.51851394
G00-00-0020417	0
G00-00-0021948	0.301029996
G00-00-0025946	0.041392685
G00-00-0027876	0
G00-00-0028863	0.176091259
G00-00-0029827	0
G00-00-0050405	0.079181246
G00-00-0066311	0.740362689
G00-00-0079109	0
G00-00-0080919	0.079181246
G00-00-0087702	0.079181246
G00-00-0088569	0.380211242
G00-00-0101365	0.544068044
G00-00-0113238	0.301029996
G00-00-0114013	0.113943352
G00-00-0118653	0
G00-00-0120807	0.079181246
G00-00-0124389	0.431363764
G00-00-0157119	0.079181246
G00-00-0158061	0.176091259
G00-00-0165832	0.079181246
G00-00-0173343	0.041392685
G00-00-0267068	1.611723308
G00-00-0297437	0.963787827
G00-00-0300859	0.079181246
G00-00-0307496	0.041392685
G00-00-0314076	0.73239376
G00-00-0316870	0.079181246
G00-00-0318103	0.740362689
G00-00-0337466	0.62324929
G00-00-0344704	0.041392685
G00-00-0350612	0.204119983
G00-00-0381817	0.176091259
G00-00-0386885	0.204119983
G00-00-0387914	0.477121255
G00-00-0409603	1.662757832
G00-00-0437124	0.204119983
G00-00-0451000	0.748188027
G00-00-0458107	0.397940009
G00-00-0465038	0.892094603
G00-00-0465454	0.51851394
G00-00-0467771	0.079181246
G00-00-0470256	0.977723605
G00-00-0478398	0.431363764
G00-00-0485739	0.792391689
G00-00-0495526	0
G00-00-0497574	0.079181246
G00-00-0502670	1.376576957
G00-00-0503555	1.768638101

Figure 13 : Exemple de fichier Prior.

- Résultats expérimentaux relatifs aux formules (1) et (2)

Nous allons exposer les résultats auxquels nous avons abouti sous forme de graphiques pour mieux illustrer les différentes variations de précision pour les différentes valeurs de paramètres utilisés. Les tableaux représentant en détails les résultats obtenus sont notés dans l'annexe 2.

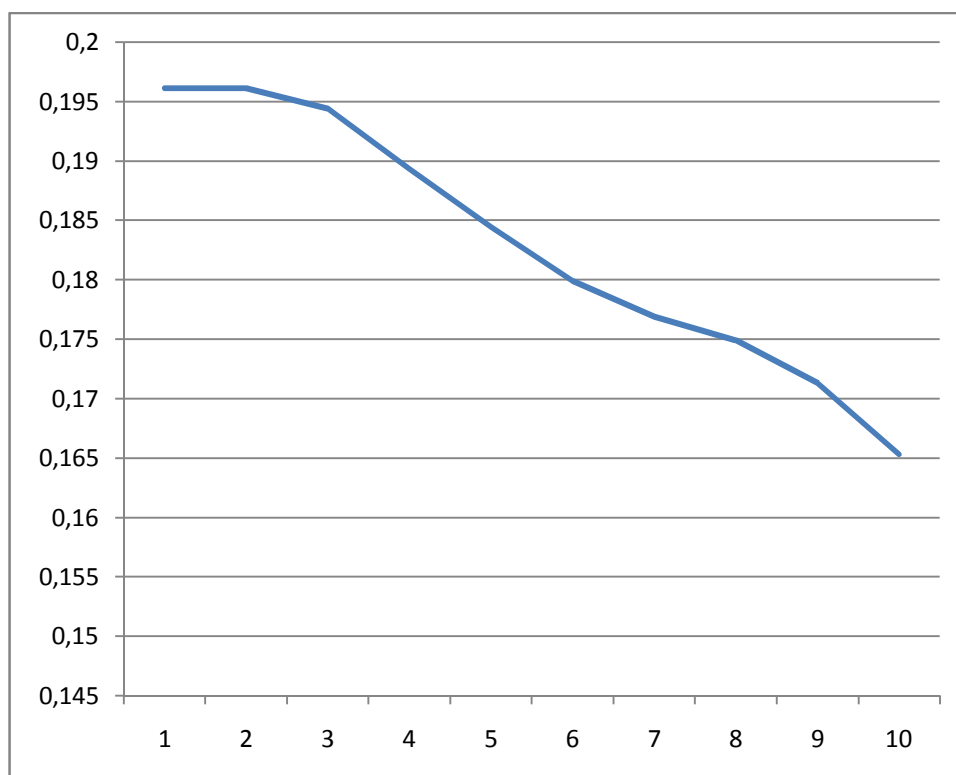


Figure 14 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule(1)

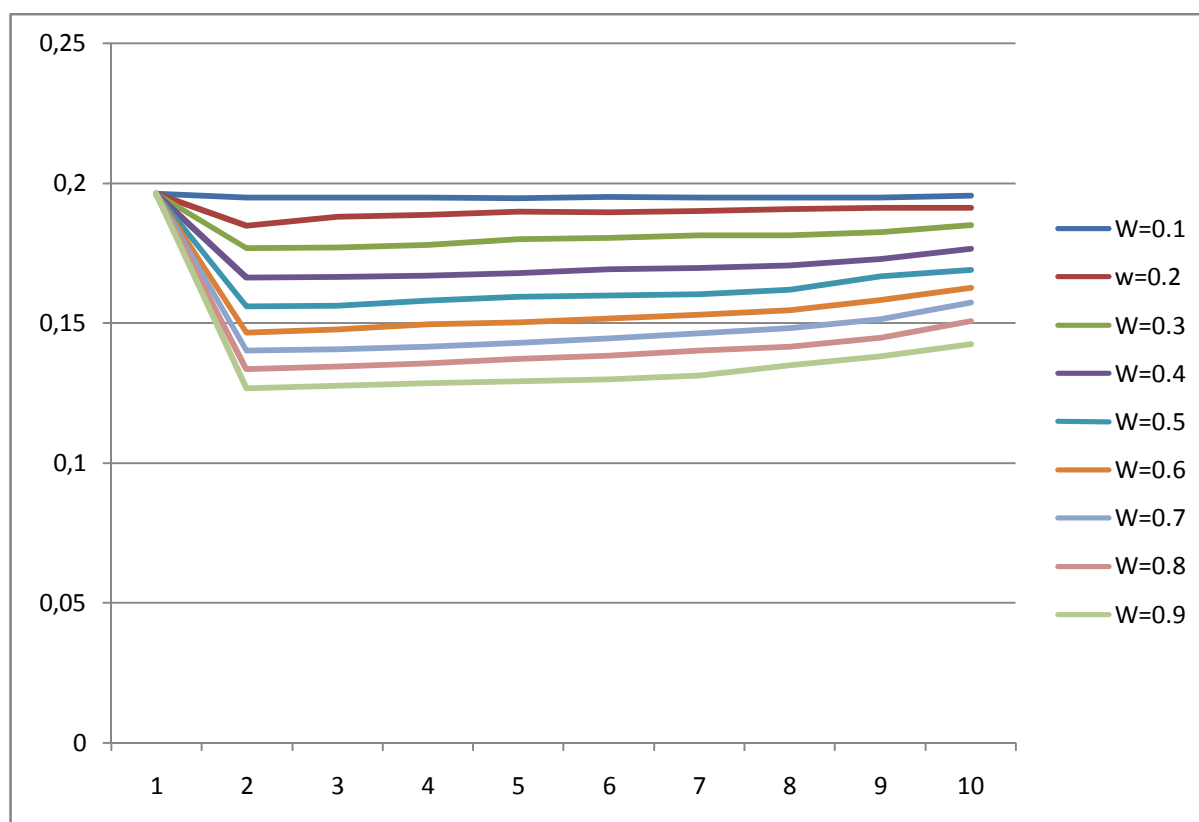


Figure 15 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (2)

Interprétation :

Nous constatons qu'il n'y a pas d'amélioration au niveau des résultats associés aux fichiers Priors par apport à la première valeur prise comme témoin. Les valeurs obtenues sont très rapprochées, néanmoins la valeur maximum est atteinte pour la valeur $W=0.1$ et de $\lambda=0.9$, la MAP =0.1954 en ce qui concerne la formule (2). Même constat pour le fichier Prior construit à partir de la formule (1), la valeur maximum obtenue est égale à la valeur témoin =0.1961, pour $w=0.1$.

D'après ces résultats, on peut dire que la formule (1) est plus performante que la formule (2), parce qu'elle permet d'offrir une pertinence a priori meilleurs sans dépasser la valeur témoin.

Les détails des valeurs obtenus sont notés dans l'annexe2 (Section 2.2).

III.5.2. implémentation et évaluation de la formule (3)

- **Programme 1** : on va utiliser les données pour calculer une autre caractéristique d'un document web, qui est le nombre de page par site.
- **Résultat 1** : les résultats de l'exécution du programme 1, seront présentés au niveau de la même table de la base de données utilisée auparavant, et donc cette fois ci elle aura cinq (5) champs. Comme suit :

Id_page : identificateur de chaque page.

Nbrlien_sp : nombre liens sortant de chaque page.

Prank_p : PageRank de chaque page.

Prank_s : PageRank de chaque site.

Nbr_p_s : nombre de page par site.

		id_page ▲	nbrlien_sp	prank_p	prank_s	nbr_p_s
		G01-86-1087166	46	28	8	1935
		G01-86-1107938	12	45	1217	228
		G01-86-1114829	91	22	11	6638
		G01-86-1136122	16	5	29	154
		G01-86-1145542	6	3	22	27
		G01-86-1155182	32	10	3	389
		G01-86-1174479	0	1	128	86
		G01-86-1177213	13	1	83	76
		G01-86-1183625	6	1	23	16745
		G01-86-1199283	2	2	275	173
		G01-86-1205170	9	46	30	88
		G01-86-1209733	11	2	25	258
		G01-86-1219560	0	2	2	282
		G01-86-1321968	16	2	64	133
		G01-86-1335211	2	2	7	119
		G01-86-1342876	22	97	16	282
		G01-86-1363900	14	13	2	39
		G01-86-1372095	4	1	63	641
		G01-86-1374415	14	0	64	347
		G01-86-1380604	1	2	50	76
		G01-86-1382054	38	1	139	1625
		G01-86-1399011	16	0	222	63
		G01-86-1402720	232	175	14	570
		G01-86-1443753	16	2	2	20
		G01-86-1469646	16	13	2	1192
		G01-86-1478416	1	1	63	214
		G01-86-1501995	28	6	162	52
		G01-86-1516458	1	1	270	64
		G01-86-1566031	11	1	77	54

Figure 16 : Extrait de la table de la base de données.

- **Programme 2 :** Les résultats retournés par la table de la base de données vont être utilisés pour calculer la pertinence à priori d'un document web, en utilisant la formule (3).
- **Résultat 2 :** de la même manière que dans l'étape d'évaluation des formules précédente(1) et (2), en faisant varier (λ) entre 0.1 et 0.9, on obtient neuf (9) fichiers Priors.

- **Résultats expérimentaux relatifs à la formule (3) :**

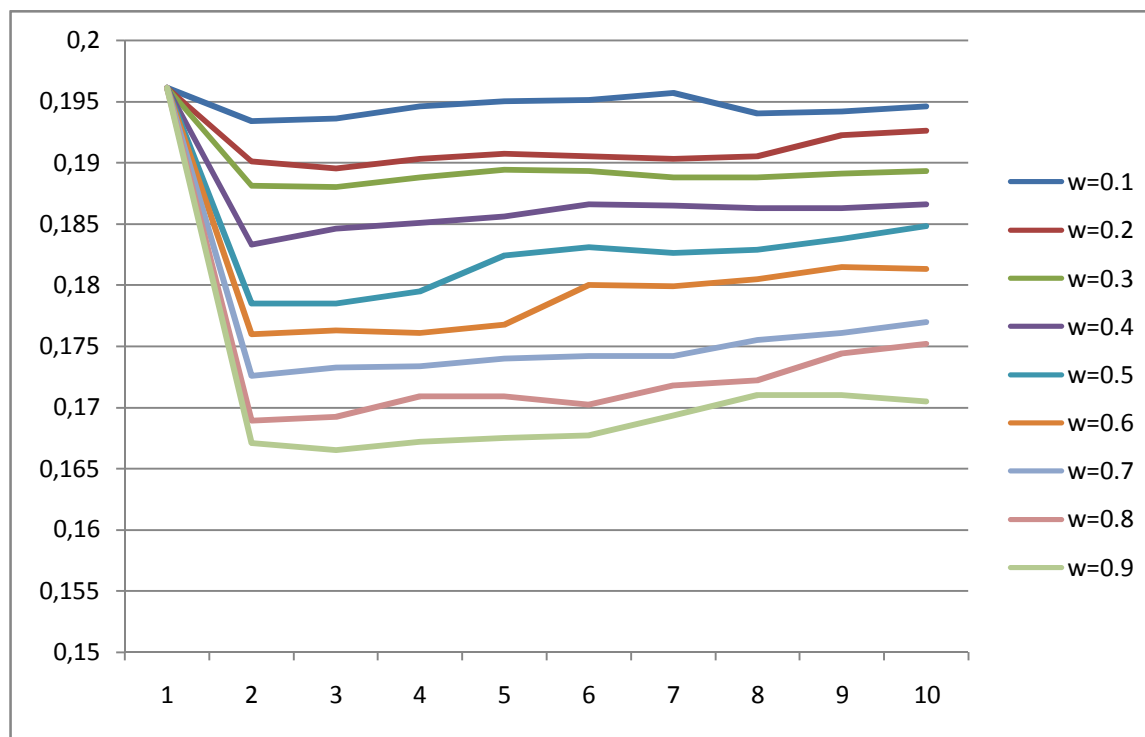


Figure 17 : Graphe de l'influence de l'ajout de pertinence a priori concernant la formule (3)

Pas d'améliorations constatées par apport à la valeur témoin qui est de MAP= 0.1961. d'après les résultats obtenus, La valeur maximum enregistré est MAP=0.1957, pour $\lambda=0.6$ et $w=0.1$. Les détails des expérimentations évaluées à partir des fichiers Priors, en utilisant la formule (3), sont notés dans l'annexe 2(section 2.3).

La caractéristique utilisé pour le calcul de la probabilité a priori, c.à.d. le nombre de page par site, ne montre aucune amélioration concernant la valeur témoin.

III.5.3. implémentation et évaluation concernant la formule (4) :

- **Programme 1** : les données seront utilisés pour calculer le niveau de la page web, dans son site.
- **Résultat 1** : les résultats du programme 1, seront mis dan la même table, après avoir créer un champ supplémentaire qui recevra le niveau de chaque page dans son site, le nombre total des champs que contient la table sera six (6) champs. Comme suit :

Id_page : identificateur de chaque page.

Nbrlien_sp : nombre liens sortant de chaque page.

Prank_p : PageRank de chaque page.

Prank_s : PageRank de chaque site.

Nbr_p_s : nombre de page par site.

Niveau : niveau de la page dans son site.

		id_page ▲	nbrlien_sp	prank_p	prank_s	nbr_p_s	Niveau
		G01-86-1087166	46	28	8	1935	2
		G01-86-1107938	12	45	1217	228	1
		G01-86-1114829	91	22	11	6638	1
		G01-86-1136122	16	5	29	154	1
		G01-86-1145542	6	3	22	27	1
		G01-86-1155182	32	10	3	389	1
		G01-86-1174479	0	1	128	86	1
		G01-86-1177213	13	1	83	76	2
		G01-86-1183625	6	1	23	16745	2
		G01-86-1199283	2	2	275	173	2
		G01-86-1205170	9	46	30	88	1
		G01-86-1209733	11	2	25	258	1
		G01-86-1219560	0	2	2	282	3
		G01-86-1321968	16	2	64	133	1
		G01-86-1335211	2	2	7	119	2
		G01-86-1342876	22	97	16	282	1
		G01-86-1363900	14	13	2	39	1
		G01-86-1372095	4	1	63	641	3
		G01-86-1374415	14	0	64	347	3
		G01-86-1380604	1	2	50	76	3
		G01-86-1382054	38	1	139	1625	1
		G01-86-1399011	16	0	222	63	2
		G01-86-1402720	232	175	14	570	3
		G01-86-1443753	16	2	2	20	1
		G01-86-1469646	16	13	2	1192	1
		G01-86-1478416	1	1	63	214	2
		G01-86-1501995	28	6	162	52	3
		G01-86-1516458	1	1	270	64	4
		G01-86-1566031	11	1	77	54	1

Figure 18 : partie de la table de la base de données.

- **Programme 2** : Les résultats retournés seront évalué en utilisant la formule (4).
- **Résultat 2** : en variant (λ) entre 0.1 et 0.9, neuf (9) fichiers Priors vont être construits.

- **Résultats expérimentaux relatif a la formule (4) :**

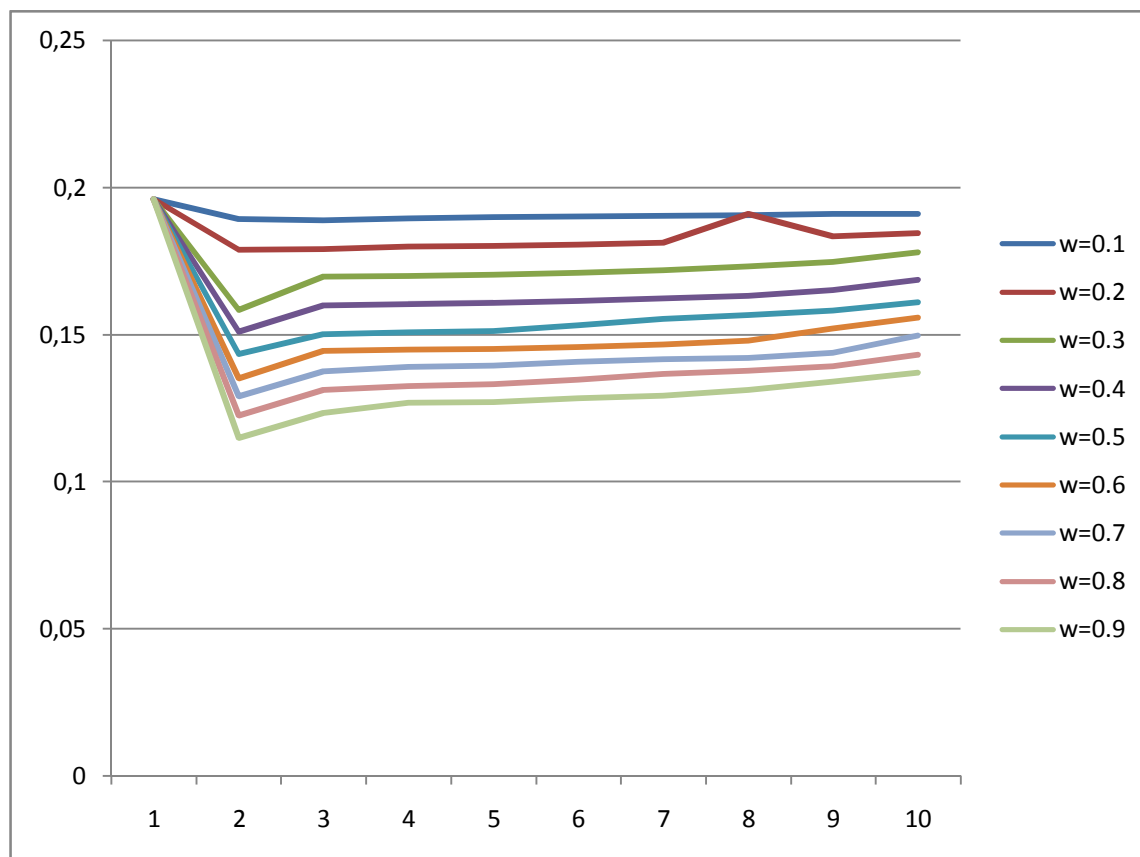


Figure 19 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (4)

D'après les évaluations effectués avec les fichiers Priors, Aucune Amélioration n'est constaté par apport à la valeur témoin $MAP=0.1961$, ceci dit, la valeur maximum la plus proche de la valeur témoins concerne le fichier Prior 1 ($W=0.1$) pour une $MAP=0.1911$ et λ égal à **0.8**. Le tableau des résultats des évaluations requête par requête sont noté en annexe 2 (section 2.5).

On peut déduire, d'après ces résultats que le paramètre (niveau de la page) n'apporte pas d'amélioration dans l'estimation de l'importance de la page Web en considérant son site.

III.5.4. implémentation et évaluation concernant la formule (5) :

- **Programme 1** : les données, seront utilisés pour calculer la similarité entre la page web et son site.
- **Résultat 1** : les résultats vont être sauvegardés dans la même table, auquel on ajoute un champ « similarite »

		id_page ▲	nbrlien_sp	prank_p	prank_s	nbr_p_s	Niveau	Similarite
	✗	G01-86-1087166	46	28	8	1935	2	0.172920311633439
	✗	G01-86-1107938	12	45	1217	228	1	0.509670485259457
	✗	G01-86-1114829	91	22	11	6638	1	0.107064290916402
	✗	G01-86-1136122	16	5	29	154	1	0.463194005454651
	✗	G01-86-1145542	6	3	22	27	1	0.558946741725606
	✗	G01-86-1155182	32	10	3	389	1	0.69858051287236
	✗	G01-86-1174479	0	1	128	86	1	0.617209604543204
	✗	G01-86-1177213	13	1	83	76	2	0.60831778802342
	✗	G01-86-1183625	6	1	23	16745	2	0.304504192391529
	✗	G01-86-1199283	2	2	275	173	2	0.461397580394808
	✗	G01-86-1205170	9	46	30	88	1	0.581672983888268
	✗	G01-86-1209733	11	2	25	258	1	0.509310613904678
	✗	G01-86-1219560	0	2	2	282	3	0.457523944683421
	✗	G01-86-1321968	16	2	64	133	1	0.469319919000995
	✗	G01-86-1335211	2	2	7	119	2	0.227802844500757
	✗	G01-86-1342876	22	97	16	282	1	0.399637626487552
	✗	G01-86-1363900	14	13	2	39	1	0.701639969815595
	✗	G01-86-1372095	4	1	63	641	3	0.592694229261309
	✗	G01-86-1374415	14	0	64	347	3	0.165965102867391
	✗	G01-86-1380604	1	2	50	76	3	0.297408958257966
	✗	G01-86-1382054	38	1	139	1625	1	0.551509377945774
	✗	G01-86-1399011	16	0	222	63	2	0
	✗	G01-86-1402720	232	175	14	570	3	0.388821986201195
	✗	G01-86-1443753	16	2	2	20	1	0.0948351441877036
	✗	G01-86-1469646	16	13	2	1192	1	0.424868762246103
	✗	G01-86-1478416	1	1	63	214	2	0.380870889628561
	✗	G01-86-1501995	28	6	162	52	3	0.65340655881295
	✗	G01-86-1516458	1	1	270	64	4	0.348860441302162

Figure 20:extrait de la table

- **Programme 2** : la formule que l'on va utiliser est (5).
- **Résultat 2** : avec cette formule, il n'y'aura qu'un seul fichier Prior, qui va être créé puis évalué.

- **Résultats expérimentaux associé a la formule (5):** (1 seul fichier prior)

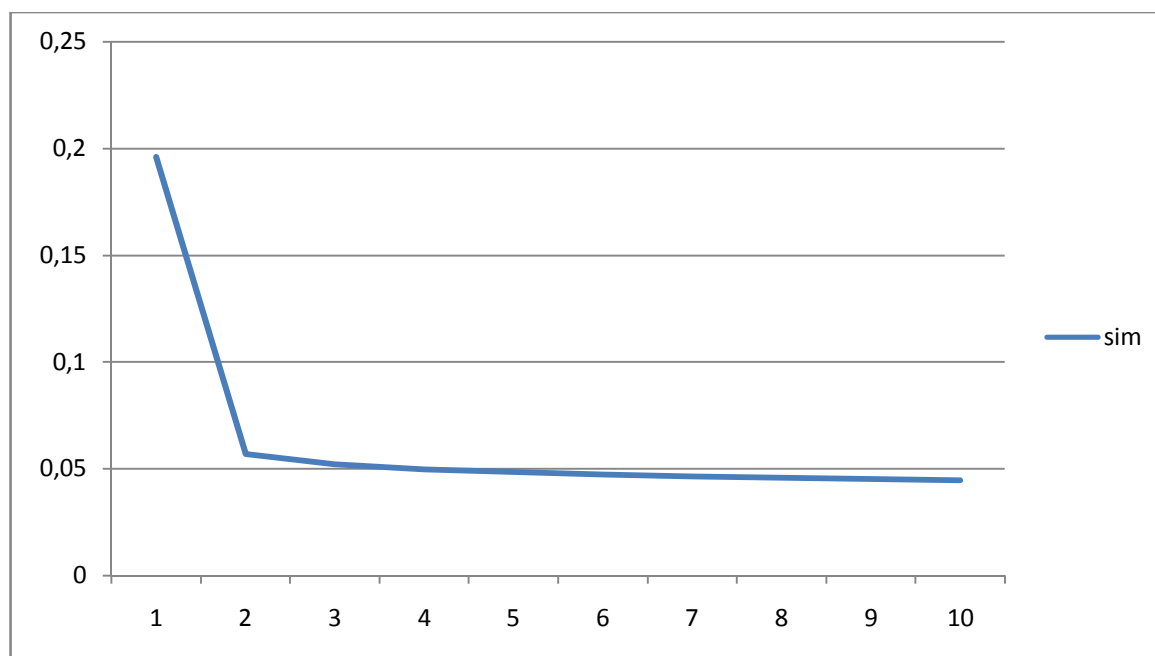


Figure 21 : Graphe de l'influence de l'ajout de pertinence à priori concernant la formule (5)

D'après les résultats obtenus, à partir des évaluations effectuées sur le fichier Prior, non seulement y'a pas d'amélioration, mais aussi les valeurs notées sont les plus faibles obtenus comparativement aux autres formules, le maximum qu'on peut définir est $MAP=0.0564$, pour $w=0.1$, qui est très éloignée de la valeur témoin $MAP=0.1961$. Les résultats de toutes les valeurs obtenus requête par requête sont notés dans l'annexe 2.

On peut conclure que le paramètre, similarité entre la page web et son site, n'apporte aucune amélioration à la pertinence de la RI.

III.6. Conclusion

Dans ce chapitre, nous avons abordé les expérimentations qu'on a réalisées, et cela en présentant en premier lieu notre environnement de travail, ensuite nous avons détaillé l'architecture générale de notre application.

Enfin, nous avons illustré les résultats obtenus en utilisant des tables (présentés en Annexe 2) et des graphiques.

Conclusion générale :

De nouveaux modèles de production, de traitement et réception de l'information émergent. Si l'arrivée du web a eu un impacte important dans le champ scientifique, économique ou social, (les réseaux sociaux) ce qui a élargit cet impact à notre espace privé. Dans les deux cas, les sources d'informations s'accroissent rendant le besoin de disposer de systèmes permettant un accès intelligent à l'information de plus en plus indispensable.

Le travail présenté dans ce mémoire a pour but de formaliser la notion de pertinence qui est un élément fondamental de tout SRI, en d'autres termes, on implémentera une approche qui intègre l'importance du site dans le calcul de la probabilité a priori de pertinence d'une page web et l'une des caractéristiques utilisée est le PageRank.

La première partie du mémoire est consacrée à la présentation des principaux modèles de recherche et à la description des processus d'indexation qui ont permis ainsi de dégager les choix réalisés par les concepteurs mais aussi leur limites. Puis nous nous sommes orienté, au niveau du deuxième chapitre, vers la recherche d'information sur le web, dans le but d'étendre nos connaissances sur les principes, outils et concepts nous permettant de réaliser cela. Puis, dans la dernière partie de notre mémoire, nous avons noté Les résultats expérimentaux graphique, qui sont des résultats ne dépassant jamais le seuil de la valeur témoin, et ce la avec l'ensemble des formules qu'on a proposé.

Nos perspectives futures, seront de tester les mêmes formules, sur d'autres collections. Et c'est à partir des résultats expérimentaux obtenus que l'on pourra juger de l'efficacité de ces formules.

Concernant nos acquis, durant la période de développement de notre travail, nous avons approfondis nos connaissances dans le domaine de recherche d'information classique et la recherche d'information sur le web, nous avons amélioré nos compétences de programmation JAVA dans les environnements de développement NETBEANS et PHPMYADMIN, et aussi nous nous sommes familiarisé avec la plateforme Terrier, logiciel libre et open source, l'un des éléments les plus importants durant la concrétisation de notre projet.

Annexe 1 :

1.1. Présentation de Terrier :

Terrier², TeRabyte RetriEveR a été développé par le département informatique de l'université Glasgow au Royaume-Uni. C'est un logiciel libre et 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 centralisés et distribués. C'est une plateforme qui est destinée à l'indexation de volume important de documents en plein-texte : jusqu'à 25 millions de documents. Selon l'étude comparative sur les SRI en open sources effectuée par Ricardo Baeza-yates, terrier fait partie des trois meilleurs systèmes dans un environnement Java. Les deux autres sont MG4J et Lucene.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requêtes basé sur le Framework DFR (divergence From Randomness)³. Comme tous les moteurs de recherche Terrier possède les principales facettes suivantes :

- **Indexation** : Permet l'extraction des termes des différents documents du corpus (basic indexed unit).
- **Recherche** : Permet de générer des résultats aux requêtes formulées par les utilisateurs.
- **Evaluation** : Permet d'évaluer les résultats obtenus en utilisant différentes mesures tel que la précision moyenne.

1.2. Installation de Terrier :

1.2.1. Préalablement :

Pour pouvoir utiliser Terrier il est nécessaire d'installer une JRE (1.5.0 ou plus). La JRE ou la JDK peuvent être téléchargés sur le site Java⁴.

1.2.2. Installation :

Après avoir téléchargé une copie de Terrier sur la page d'accueil du projet Terrier⁵, il faudra créer un nouveau répertoire et dézipper Terrier dans ce dernier.

1.3. La structure des répertoires de Terrier

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

² <http://www.terrier.org>

³ http://terrier.org/docs/v.2.2.1/dfp_description.html

⁴ <http://www.oracle.com/technetwork/java/downloads/index.html>

⁵ <http://www.terrier.org>

- ▼ 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 des différentes librairies externes utilisées par terrier.
- ▼ Share\ : contient la liste des mots vides (stop worst list)
- ▼ Src\ : Contient le code source de Terrier.
- ▼ Var/index : Contient les structures de données : fichier inverse, fichier directe, lexicon et lexicon index, document index.
- ▼ Var/ results : Contient les résultats de la recherche.
- ▼ Licences\ : contient des informations sur la licence des différents composants inclus dans Terrier.

1.4. Les applications de Terrier :

Terriers offre trois applications :

- 1.4.1. Batch (TREC) Terrier** : Permet l'indexation, la recherche et l'évaluation des résultats d'une collection TREC⁶.
- 1.4.2. Interactive Terrier** : permet une recherche interactive en exécutants le script `interactive_terrier`⁷. C'est un moyen facile de tester Terrier.
- 1.4.3. Desktop Terrier** : C'est une interface graphique pour la plateforme de RI Terrier.

Autres que l'interface Desktop Terrier, Terrier propose une interface Web (Web-based interface) pour réaliser l'interactivité.

⁶ <http://trec.nist.gov/>

⁷ <http://terrier.org/docs/v2.2.1/javadoc/uk/ac/gla/terrier/applications/interactiveQuerying.html>

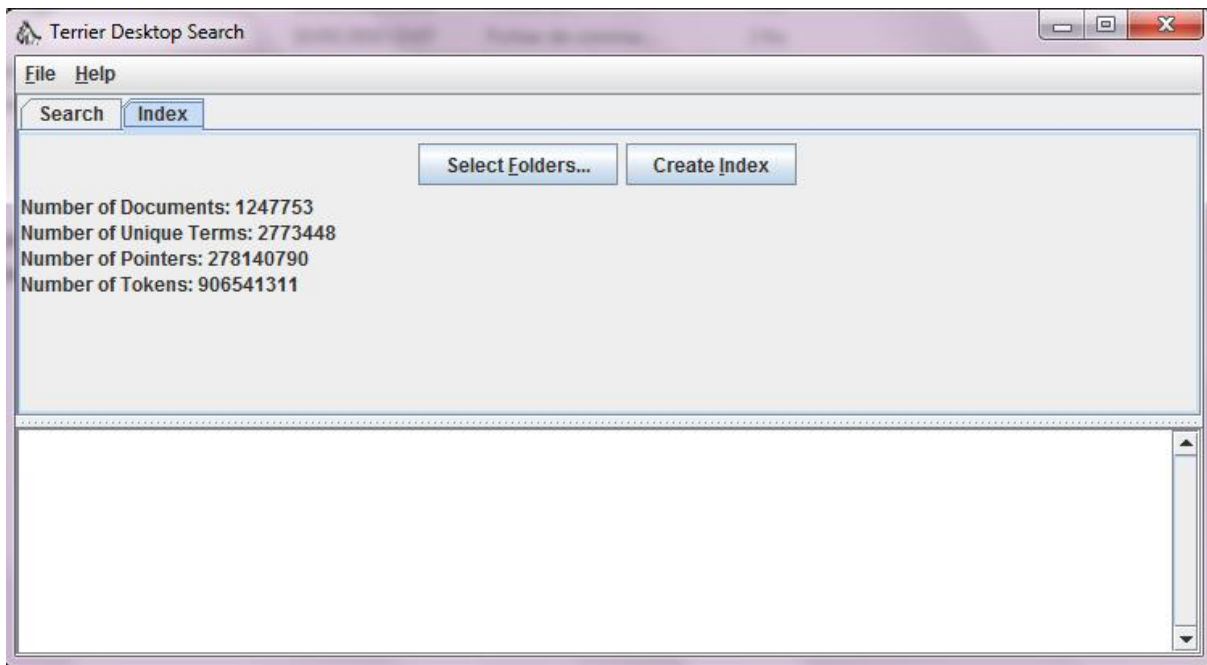


Figure 1. Présentation de l'interface Desktop Terrier.

Autre que l'interface Desktop Terrier, Terrier propose une interface Web (Web-Based interface) pour réaliser l'interactivité.



Figure 2. Présentation de l'interface Web de Terrier.

1.5. Le processus d'indexation de Terrier :

Pour l'indexation d'une collection de documents, Terrier se base sur l'indexation classique à 4 étapes :

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

- ▼ **Construction d'index** : Le but de cette étape est d'effectuer plusieurs traitements afin de parvenir à la création de la structure d'index.
- Ø **Direct Index** : l'index direct enregistre pour un document les termes qui apparaissent dans ce dernier. Il est souvent utilisé pour la reformulation de la requête, la classification et la comparaison de document.
- Ø **Inverted Index** : L'index inversé contrairement à l'index direct enregistre pour le terme les documents dans lesquels il apparaît, il garde aussi la position de chaque terme et sa fréquence dans le document.

Pour que Terrier puisse créer ces fichiers, il utilise deux méthodes que voici :

▼ **Méthode de tri en fonction inverse** :

Cette méthode commence par l'enregistrement des identificateurs pour chaque terme du document (écrire l'index direct pour chaque document). Puis construire les posting-list pour chaque terme du document dans la mémoire.

Une fois qu'on sait à quels documents appartient chaque terme, on inverse les listes construites avant. Cette méthode est coûteuse en mémoire.

▼ **Méthode à une seule phase** :

Premièrement, analyser la collection, construire l'index inversé dans la mémoire. Ensuite, fusionner les fichiers inversés créés.

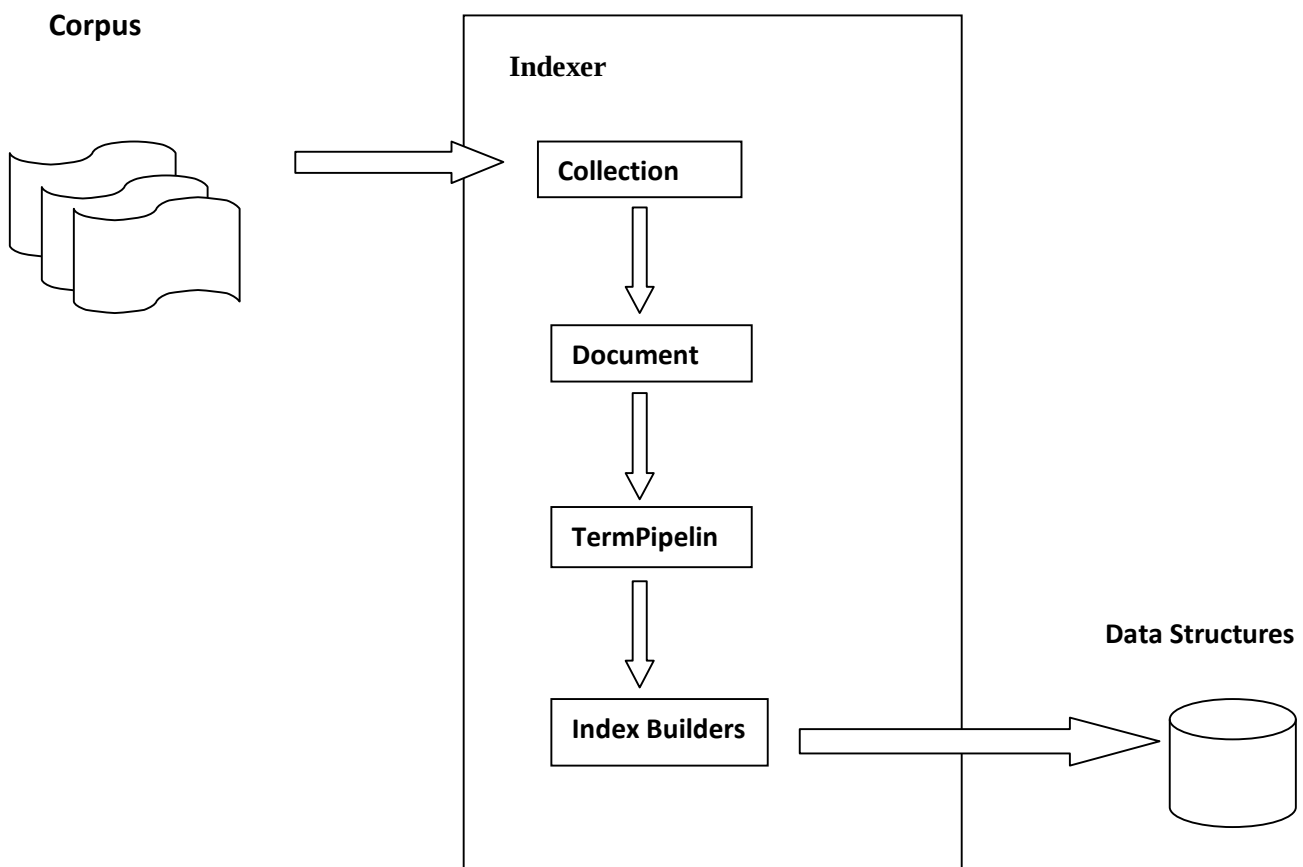


Figure 3. Présentation du processus d'indexation de Terrier.

Le résultat de l'indexation consiste en un index constitué des structures de données suivantes : Inverted File, lexicon File, Direct File, Document Index.

Les Modules Collection, Document, TermePipeline, Index Builder font partie de l'API d'indexation de Terrier.

1.6. L'API d'indexation de Terrier :

Comme présenté préalablement, L'indexation dans Terrier est divisée en quatre procédures (étages) et à chaque étage des plug-ins (des classes java) peuvent être ajouté pour la personnalisation du Système. Les Quatre étages sont :

1. Extraction de l'objet Document de la collection qui est généré par l'ensemble des Corpus reçus en entrée par Terrier.
2. Parcourir chaque document de la collection pour en extraire les termes ainsi que les informations relatives.
3. Traitement des termes extrait en utilisant TermPipelines.
4. La construction de l'index.

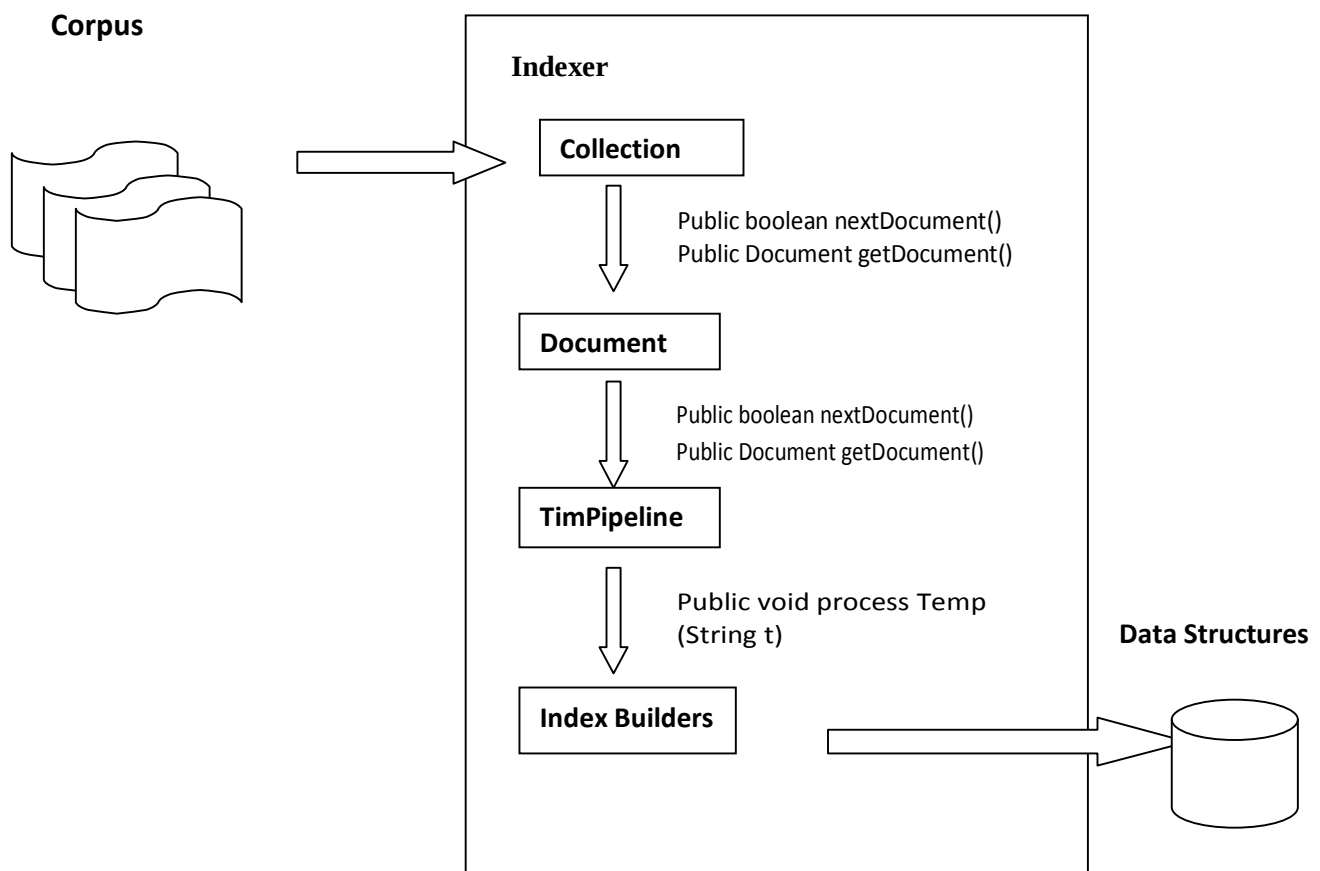


Figure 4. Présentation du processus d'indexation dans Terrier [Ounis, 10].

Les composants principaux du processus d'indexation sont présentés dans les paragraphes suivant :

1. Collection :

Cette composante englobe le concept le plus fondamental de l'indexation avec Terrier qui est la « collection ». C'est un objet qui représente le corpus, c'est-à-dire un ensemble de documents. Collection⁸ est une interface qui se trouve dans le package `org.terrier.indexing`. Utilisée généralement par Terrier pour intégrer de nouvelles sources de données (nouveaux corpus) et cela en ajoutant une nouvelle classe java qui implémente cette interface. Elle permet de parcourir un ensemble de document et de renvoyer un objet document en faisant appel à sa méthode `nextDocument ()`.

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

2. Document :

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

3. Term Pipeline :

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

4. Indexer :

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

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

5. Les structures Index :

L'index de Terrier consiste en quatre structures de données, en plus de quelques fichiers auxiliaires :

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

⁸ Pour plus d'Information voir la javadoc de terrier.

Le **tableau.1** ci-dessous permet de présenter le contenu de ces différentes structures index.

Index Structure	contents
Lexicon	Term Term id Term Frequency Document Frequency Byte offset in inverted file Bit offset in inverted file
Inverted Index	Document id Term Frequency Fields (# of fields bits) Block Frequency [Block id]
Document Index	Document id Document length Document Number Byte offset in direct file Bit offset in direct file
Direct Index	Term id Term Frequency Fields (# of fields bits) Block frequency [Block id]

Tableau.1: présentation des différentes structures Index.

- **Lexicon:** comprend le terme et son id (un unique numéro attribué à chaque terme), ainsi que les statistiques globales du terme (la fréquence du terme globale et le document fréquence associé au terme), les décalages de la liste des affectations contenus dans l'index inversé.
- **Inverted index:** inclut pour chaque terme, l'identifiant du document correspondant, et la fréquence du terme en question dans ce document. Les champs dans lesquels le terme apparaît sont codés en utilisant un bit.
- **Document index :** définit le numéro du document (un externe identifiant unique du document), l'id du document (identifiant unique interne du document); la longueur du document en termes de jetons, et le décalage du document de l'index directe.
- **Direct Index:** se compose des termes et de la fréquence des termes présents dans chaque document. L'objectif de l'index direct est de faciliter de manière efficace l'expansion de la requête. les Ids des termes sont écrit en utilisant le codage Gamma.

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

1.7. Processus de recherche de Terrier :

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

1.8. L'API de recherche de Terrier :

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

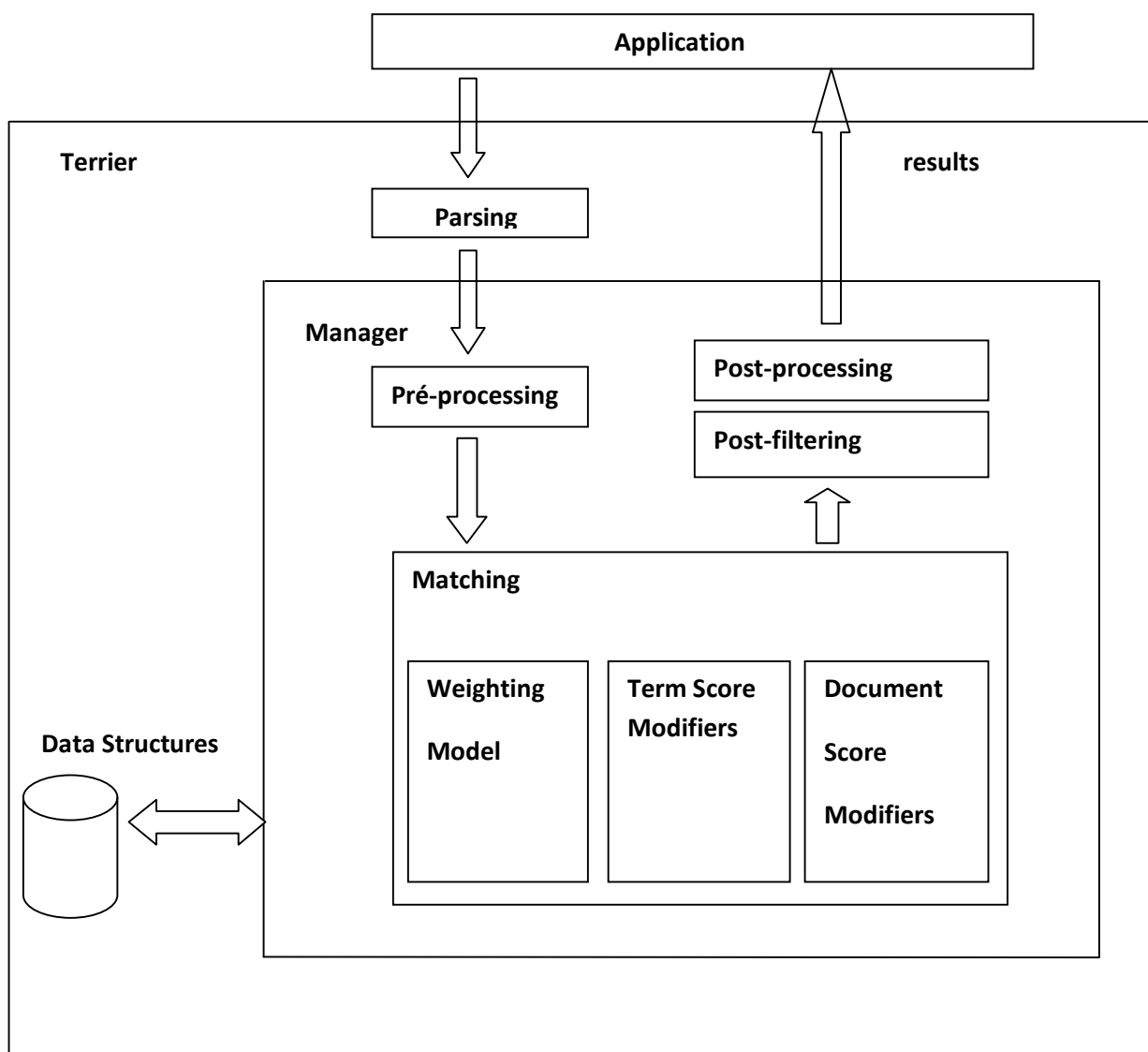


Figure 5 : Présentation du processus de recherche de Terrier [Ounis, 10].

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 *org.terrier.querying.parser* qui modélise les requêtes. Elle consiste soit en un *sub-queries* ou bien un *query terms*. Un objet *Query* est créé pour chaque requête.

Terrier offre un support pour différents types de requête :

- ▼ *SingleTermQuery* : c'est le modèle de requête avec un seul terme.
- ▼ *MultiTermQuery* : consiste en un modèle de requêtes qui contiennent plus d'un terme.
- ▼ *FieldQuery* : c'est le modèle de requête qualifié par un champ (field).

2. *Manager* :

C'est le module qui est chargé de la gestion de la recherche. Dans un premier niveau il lit chaque requête en utilisant le parseur approprié. Puis il crée un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaire et en spécifiant un modèle de pondération. Puis il crée un fichier résultat (result file) ou il associe à chaque requête les documents qui lui sont pertinents. Le résultat de chaque requête est donné par le second niveau de gestion.

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

3. *Matching*:

Le composant *Matching* est responsable de déterminer qui des documents correspondent à la requête spécifié et donne un score au document qui vérifie la requête.

Il utilise le *Weighting Models* pour assigner un score à chaque mot de la requête dans le document.

Terrier contient un nombre important de modèle de pondération qui offre des performances robustes.

- ▼ *Document Score Modifiers* : Modifie le score des documents en fonction de la requête.
- ▼ *Term Score Modifiers* : Modifie le score des documents en fonction de la position des termes.

1.8. Modification de Terrier :

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

1.9. Utilisation de Batch (TREC) Terrier :

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

1.9.1. Indexation :

1. Allez au répertoire où Terrier est installé en utilisant la commande `cd` :
`>> cd terrier-3.0`
2. Initialisation de Terrier pour l'indexation d'une nouvelle collection TREC
`>>.bin\trec\trec_setup <le chemin absolu du répertoire contenant les documents à indexer>`
3. Maintenant vous pouvez indexer la collection TREC.
`>>.bin\ trec_terrier -i`

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

1.9.2. Recherche et Evaluation :

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

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

```
#add the qrels files to use for evaluation
D:\Master\terrier-3.0\qrels.distillation.txt
```

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

```
#add the topic files to use for querying
D:\Master\terrier-3.0\webtopics_551-600.txt
```

3. Spécification du modèle de pondération (ex. BM25) à utiliser dans le fichier **etc\trec.models**.

Après que ces étapes soient faites la recherche et l'évaluation peuvent être lancés sur Terrier.

4. Pour lancer la recherche dans Terrier il faut exécuter la commande :

.bin\trec_terrier -r.

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

.bin\trec_terrier -e.

1.10. Comparaison des plateformes de RI :

Dans le tableau qui suit, nous présentons une comparaison entre les plateformes de RI : Lemure, Lucene, Terrier.

	<i>Lemure</i>	<i>Lucene</i>	<i>Terrier</i>
<i>Indexation</i>	- capacité d'indexé des Terabytes de données et plus encore. - indexation progressive	-peut indexer jusqu'à 20MB/minute sur la machine. -requière une petite RAM – seulement 1MB -taille d'indexation 20% -30% la taille du texte indexé (400GB->80GB) -Nutch supporte l'indexation distribué. -indexation progressive	- Quelques nombres: la taille des fichiers à indexer : 400GB - Taille résultante des fichiers indexe : 17GB=>4% du texte courant. Temps de construction: 3jours (2processeurs) Temps de récupération : 4sec/requête (8 processeurs) -supporte l'indexation distribuée. -ne supporte pas l'indexation progressive.
<i>Retrieval Models</i>	-KL -divergence -espace vectoriel -Okapi BM25 -Modèle de langue TF-IDF	-VSM - retrieval model booléen	-Modèle 126 Divergence From Randomness (DFR) -Okapi BM25 -modèle de langue TF-IDF
<i>Lang. De Prog.</i>	C++	JAVA	JAVA

Tableau.2 : Comparaison des plates formes de RI : Lemure, Lucene, Terrier.

Annexe 2 :

2.1. Détails des résultats d'expérimentations obtenus

On considère les paramètres ci après :

- **Témoin** : Correspond à la première partie de la formule (3), permet l'évaluation en comparaison avec cette valeur obtenus dans l'exécution de terrier.

La formule utilisée pour le calcul du score dans Terrier, est le modèle de langage :

$$\text{Score (pi)} = \prod_{i=1}^n \lambda P(t_i / M_d) + (1 - \lambda) P(t_i / M_c) \dots (1)$$

Ou

- t_i : termes de la requête.
- M_d : modèle de langue du document.
- M_c : modèle de la collection
- λ : C'est le paramètre utilisé dans la formule de la pertinence a priori. Sa Valeur est comprise entre 0.1 et 0.9. Le $(1 - \lambda)$ régit l'apport de l'importance du site dans les pages.
- W : est un paramètre utilisé lors de la modification du score de document, il permet de fournir un pourcentage de la pertinence à priori d'un document. W varie entre 0 et 1, La formule dans laquelle il apparait est la suivante :

$$\text{Nouveau score (Pi)} = \text{score (Pi)} + W * \text{Prior (Pi)} \dots (2)$$

Ou :

P_i : C'est la pertinence a priori de la page i.

Score (P_i) : est le score de la page i calculer par Terrier.

- **MAP** : c'est la moyenne des précisions dans le cas de l'utilisation de plusieurs requêtes pour l'évaluation d'une même collection de documents.

2.2. Résultats d'évaluation des formules de calcul de probabilité a priori de pertinence:

- ▼ Concernant les deux formules suivantes (3) et (4), contenant les paramètres PageRank de site et PageRank de page, les résultats de leur évaluation sont notés dans les tableaux (3) et (4), respectivement.

$$\text{Prior (pi)} = \log (\text{PageRank_p (pi)} + 1) \dots (3)$$

$$\text{Prior (pi)} = \log [\lambda * (\text{PageRank_p (pi)}) + (1 - \lambda) * (\text{PageRank_s})] \dots (4)$$

Tableau 1 : Résultats d'évaluation de la formule (3)

W	MAP	ETAT
	0.1961	Témoin
0.1	0.1961	MAX
0.2	0.1944	
0.3	0.1893	
0.4	0.1844	
0.5	0.1799	
0.6	0.1769	
0.7	0.1749	
0.8	0.1713	
0.9	0.1653	

Tableau 2 : résultats d'évaluation de la formule (4)

λ	$1-\lambda$	w	MAP	ETAT
			0.1961	Témoin
0,1	0,9	0,1	0.1949	
0,2	0,8	0,1	0.1950	
0,3	0,7	0,1	0.1950	
0,4	0,6	0,1	0.1947	
0,5	0,5	0,1	0.1951	
0,6	0,4	0,1	0.1948	
0,7	0,3	0,1	0.1950	
0,8	0,2	0,1	0.1949	
0,9	0,1	0,1	0.1954	MAX
0,1	0,9	0,2	0.1847	
0,2	0,8	0,2	0.1878	
0,3	0,7	0,2	0.1886	
0,4	0,6	0,2	0.1897	
0,5	0,5	0,2	0.1896	
0,6	0,4	0,2	0.1899	
0,7	0,3	0,2	0.1907	MAX
0,8	0,2	0,2	0.1911	
0,9	0,1	0,2	0.1910	
0,1	0,9	0,3	0.1767	
0,2	0,8	0,3	0.1770	
0,3	0,7	0,3	0.1778	
0,4	0,6	0,3	0.1798	

0,5	0,5	0,3	0.1803	
0,6	0,4	0,3	0.1812	
0,7	0,3	0,3	0.1813	
0,8	0,2	0,3	0.1825	
0,9	0,1	0,3	0.1850	MAX
0,1	0,9	0,4	0.1662	
0,2	0,8	0,4	0.1664	
0,3	0,7	0,4	0.1669	
0,4	0,6	0,4	0.1679	
0,5	0,5	0,4	0.1692	
0,6	0,4	0,4	0.1696	
0,7	0,3	0,4	0.1705	
0,8	0,2	0,4	0.1728	
0,9	0,1	0,4	0.1764	MAX
0,1	0,9	0,5	0.1560	
0,2	0,8	0,5	0.1563	
0,3	0,7	0,5	0.1580	
0,4	0,6	0,5	0.1595	
0,5	0,5	0,5	0.1599	
0,6	0,4	0,5	0.1604	
0,7	0,3	0,5	0.1620	
0,8	0,2	0,5	0.1666	
0,9	0,1	0,5	0.1690	MAX
0,1	0,9	0,6	0.1465	
0,2	0,8	0,6	0.1477	
0,3	0,7	0,6	0.1495	
0,4	0,6	0,6	0.1501	
0,5	0,5	0,6	0.1516	
0,6	0,4	0,6	0.1528	
0,7	0,3	0,6	0.1546	
0,8	0,2	0,6	0.1582	
0,9	0,1	0,6	0.1626	MAX
0,1	0,9	0,7	0.1402	
0,2	0,8	0,7	0.1407	
0,3	0,7	0,7	0.1415	
0,4	0,6	0,7	0.1429	
0,5	0,5	0,7	0.1445	

0,6	0,4	0,7	0.1464	
0,7	0,3	0,7	0.1482	
0,8	0,2	0,7	0.1513	
0,9	0,1	0,7	0.1572	MAX
0,1	0,9	0,8	0.1335	
0,2	0,8	0,8	0.1343	
0,3	0,7	0,8	0.1356	
0,4	0,6	0,8	0.1372	
0,5	0,5	0,8	0.1383	
0,6	0,4	0,8	0.1401	
0,7	0,3	0,8	0.1414	
0,8	0,2	0,8	0.1447	
0,9	0,1	0,8	0.1506	MAX
0,1	0,9	0,9	0.1268	
0,2	0,8	0,9	0.1278	
0,3	0,7	0,9	0.1286	
0,4	0,6	0,9	0.1293	
0,5	0,5	0,9	0.1299	
0,6	0,4	0,9	0.1314	
0,7	0,3	0,9	0.1350	
0,8	0,2	0,9	0.1382	
0,9	0,1	0,9	0.1426	MAX

- ▼ Evaluation de la formule (5), noté comme suite, en insérant le paramètre nombre de page pour chaque site.

$$\text{Prior (pi)} = \log(1 + (\lambda * \text{PageRank(pi)}) + ((1 - \lambda) * (\text{PageRank}_s / \text{nbr_p_s}))) \dots (5)$$

Tableau 3 : résultats d'évaluation de la formule (5)

λ	$1 - \lambda$	w	Map	ETAT
			0.1961	Témoin
0,1	0,9	0,1	0.1934	
0,2	0,8	0,1	0.1936	
0,3	0,7	0,1	0.1946	
0,4	0,6	0,1	0.1950	
0,5	0,5	0,1	0.1951	
0,6	0,4	0,1	0.1957	MAX
0,7	0,3	0,1	0.1940	
0,8	0,2	0,1	0.1942	

0,9	0,1	0,1	0.1946	
0,1	0,9	0,2	0.1901	
0,2	0,8	0,2	0.1895	
0,3	0,7	0,2	0.1903	
0,4	0,6	0,2	0.1907	
0,5	0,5	0,2	0.1905	
0,6	0,4	0,2	0.1903	
0,7	0,3	0,2	0.1905	
0,8	0,2	0,2	0.1922	
0,9	0,1	0,2	0.1926	MAX
0,1	0,9	0,3	0.1881	
0,2	0,8	0,3	0.1880	
0,3	0,7	0,3	0.1888	
0,4	0,6	0,3	0.1894	MAX
0,5	0,5	0,3	0.1893	
0,6	0,4	0,3	0.1888	
0,7	0,3	0,3	0.1888	
0,8	0,2	0,3	0.1891	
0,9	0,1	0,3	0.1893	
0,1	0,9	0,4	0.1833	
0,2	0,8	0,4	0.1846	
0,3	0,7	0,4	0.1851	
0,4	0,6	0,4	0.1856	
0,5	0,5	0,4	0.1866	MAX
0,6	0,4	0,4	0.1865	
0,7	0,3	0,4	0.1863	
0,8	0,2	0,4	0.1863	
0,9	0,1	0,4	0.1866	MAX
0,1	0,9	0,5	0.1785	
0,2	0,8	0,5	0.1785	
0,3	0,7	0,5	0.1795	
0,4	0,6	0,5	0.1824	
0,5	0,5	0,5	0.1831	
0,6	0,4	0,5	0.1826	
0,7	0,3	0,5	0.1829	
0,8	0,2	0,5	0.1838	
0,9	0,1	0,5	0.1848	MAX

0,1	0,9	0,6	0.1760	
0,2	0,8	0,6	0.1763	
0,3	0,7	0,6	0.1761	
0,4	0,6	0,6	0.1768	
0,5	0,5	0,6	0.1800	
0,6	0,4	0,6	0.1799	
0,7	0,3	0,6	0.1805	
0,8	0,2	0,6	0.1815	MAX
0,9	0,1	0,6	0.1813	
0,1	0,9	0,7	0.1726	
0,2	0,8	0,7	0.1733	
0,3	0,7	0,7	0.1734	
0,4	0,6	0,7	0.1740	
0,5	0,5	0,7	0.1742	
0,6	0,4	0,7	0.1742	
0,7	0,3	0,7	0.1755	
0,8	0,2	0,7	0.1761	
0,9	0,1	0,7	0.1770	MAX
0,1	0,9	0,8	0.1689	
0,2	0,8	0,8	0.1692	
0,3	0,7	0,8	0.1709	
0,4	0,6	0,8	0.1709	
0,5	0,5	0,8	0.1702	
0,6	0,4	0,8	0.1718	
0,7	0,3	0,8	0.1722	
0,8	0,2	0,8	0.1744	
0,9	0,1	0,8	0.1752	MAX
0,1	0,9	0,9	0.1671	
0,2	0,8	0,9	0.1665	
0,3	0,7	0,9	0.1672	
0,4	0,6	0,9	0.1675	
0,5	0,5	0,9	0.1677	
0,6	0,4	0,9	0.1693	
0,7	0,3	0,9	0.1710	MAX
0,8	0,2	0,9	0.1710	MAX
0,9	0,1	0,9	0.1705	

- ▼ Évaluation de la formule (6), utilisant le paramètre niveau de la page dans son site.

$$\text{Prior (pi)} = \log(1 + (\lambda * \text{PageRank_p(pi)}) + ((1 - \lambda) * (\text{PageRank_s/ (niveau+1)})) \dots (6)$$

Tableau 4 : résultats d'évaluation de la formule (6).

□	1- □	w		ETAT
			0.1961	Temoin
0,1	0,9	0,1	0.1893	
0,2	0,8	0,1	0.1888	
0,3	0,7	0,1	0.1894	
0,4	0,6	0,1	0.1900	
0,5	0,5	0,1	0.1902	
0,6	0,4	0,1	0.1903	
0,7	0,3	0,1	0.1906	
0,8	0,2	0,1	0.1911	MAX
0,9	0,1	0,1	0.1910	
0,1	0,9	0,2	0.1788	
0,2	0,8	0,2	0.1790	
0,3	0,7	0,2	0.1799	
0,4	0,6	0,2	0.1801	
0,5	0,5	0,2	0.1805	
0,6	0,4	0,2	0.1812	
0,7	0,3	0,2	0.1910	MAX
0,8	0,2	0,2	0.1834	
0,9	0,1	0,2	0.1846	
0,1	0,9	0,3	0.1583	
0,2	0,8	0,3	0.1696	
0,3	0,7	0,3	0.1699	
0,4	0,6	0,3	0.1704	
0,5	0,5	0,3	0.1709	
0,6	0,4	0,3	0.1718	
0,7	0,3	0,3	0.1731	
0,8	0,2	0,3	0.1747	
0,9	0,1	0,3	0.1780	MAX
0,1	0,9	0,4	0.1511	
0,2	0,8	0,4	0.1600	
0,3	0,7	0,4	0.1605	

0,4	0,6	0,4	0.1609	
0,5	0,5	0,4	0.1616	
0,6	0,4	0,4	0.1624	
0,7	0,3	0,4	0.1633	
0,8	0,2	0,4	0.1652	
0,9	0,1	0,4	0.1687	MAX
0,1	0,9	0,5	0.1434	
0,2	0,8	0,5	0.1501	
0,3	0,7	0,5	0.1507	
0,4	0,6	0,5	0.1512	
0,5	0,5	0,5	0.1533	
0,6	0,4	0,5	0.1554	
0,7	0,3	0,5	0.1566	
0,8	0,2	0,5	0.1582	
0,9	0,1	0,5	0.1611	MAX
0,1	0,9	0,6	0.1351	
0,2	0,8	0,6	0.1444	
0,3	0,7	0,6	0.1448	
0,4	0,6	0,6	0.1452	
0,5	0,5	0,6	0.1457	
0,6	0,4	0,6	0.1466	
0,7	0,3	0,6	0.1479	
0,8	0,2	0,6	0.1521	
0,9	0,1	0,6	0.1557	MAX
0,1	0,9	0,7	0.1291	
0,2	0,8	0,7	0.1376	
0,3	0,7	0,7	0.1392	
0,4	0,6	0,7	0.1395	
0,5	0,5	0,7	0.1408	
0,6	0,4	0,7	0.1417	
0,7	0,3	0,7	0.1422	
0,8	0,2	0,7	0.1438	
0,9	0,1	0,7	0.1497	MAX
0,1	0,9	0,8	0.1225	
0,2	0,8	0,8	0.1313	
0,3	0,7	0,8	0.1326	
0,4	0,6	0,8	0.1332	

0,5	0,5	0,8	0.1347	
0,6	0,4	0,8	0.1366	
0,7	0,3	0,8	0.1377	
0,8	0,2	0,8	0.1392	
0,9	0,1	0,8	0.1431	MAX
0,1	0,9	0,9	0.1149	
0,2	0,8	0,9	0.1234	
0,3	0,7	0,9	0.1269	
0,4	0,6	0,9	0.1271	
0,5	0,5	0,9	0.1284	
0,6	0,4	0,9	0.1293	
0,7	0,3	0,9	0.1313	
0,8	0,2	0,9	0.1341	
0,9	0,1	0,9	0.1371	MAX

- ▼ Évaluation de la formule (7), introduisant le paramètre similarité existante entre la page web et le site qui la contient.

$$\text{Prior (pi)} = (\text{Sim*PageRank}_p(\text{pi})) + (\text{Sim*PageRank}_s) \dots (7)$$

Tableau 5 : notation des résultats d'évaluation de la formule (7).

W	MAP	ETAT
	0.1961	Temoin
0.1	0.0571	Max
0.2	0.0522	
0.3	0.0499	
0.4	0.0485	
0.5	0.0474	
0.6	0.0465	
0.7	0.0458	
0.8	0.0452	
0.9	0.0447	

Références bibliographiques

- [Amrouche, 08] Karima AMROUCHE. Passage à l'échelle en Recherche d'Information : Méthode d'élagage pour la réduction de l'espace de recherche Thèse de Doctorat en Informatique, Institut National de formation en Informatique (I.N.I) Oued-Smar Alger. 2007/2008.
- [Belew, 1989] R. Belew, "Adaptative Information Retrieval", In processing of the twelfth annual international ACM SIGIR conference on research and development in information retrieval, pp 11-20, 1989.
- [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 & Tamine, 04] Boughanem M. & Tamine L., "Connexionnisme et génétique pour la recherche d'information", dans : Les systèmes de recherche d'informations, (Eds.), Hermes-Lavoisier, Lavoisier, 2004.
- [Belkin, 92] Nicholas J. Belkin, Peter Ingwersen, Annelise Mark Pejtersen: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992 ACM 1992.
- [Bush45] Bush (Vannevar). – As We May Think. *The Atlantic Monthly*, vol. 176, Juillet 1945, pp. 101–108.
- [Bondu, 09] Jérôme Bondu. "Panorama d'outils de recherche d'informations gratuits et en ligne". Inter-Ligere Sarl. <http://www.inter-ligere.com/article-30587376.html>
- [Brin, 98] S. Brin, L. Page. "The anatomy of a large-scale hypertextual Web search engine". *Computer Networks and ISDN Systems*, 30(1–7) :107–117, 1998.
- [Carmel, 2000] [Carmel, 2000] Carmel L., " Recherche d'information", Janvier 2000, [Http://www.esi.umontreal.ca/~carmellu/BLT6057/notes3.htm](http://www.esi.umontreal.ca/~carmellu/BLT6057/notes3.htm)
- [Croft & Harper, 1979] Croft W.B & Harper D.J, "Using probabilistic models of document retrieval without relevance information", *Journal of Documentation*, 35:285-295, 1979.

- [Crestani, 1994] Crestani F., "Comparing neural and probabilistic relevance feedback in an interactive information retrieval system", In Proceedings of the 1994 IEEE International Conference on Neural Networks, pages 3426–3430, Orlando, Florida, USA, 1994.
- [Chibane, 08] I. Chibane. "Impact des liens hypertextes sur la précision en recherche d'information. Conception d'un système de recherche d'information adapté au Web", 2008.
- [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.
- [Deerwester & al, 1990] Deerwester S., Dumais S., Furnas G., Landauer T., & Harshman R., "Indexing by latex semantic analysis" Journal of the American Society for Information Science, 41(6):391–407, 1990.
- [Dalbin, 2002] Dalbin S., "Instruments de recherche sur le Web", Dans la Recherche d'Information sur les réseaux, cours INRIA, 2002.
- [Dumais & al, 1996] Dumais S., Landauer t. & Littman M., "Automatic cross-linguistic information retrieval using latent semantic indexing", In SIGIR'96 – Workshop on Cross-Linguistic Information Retrieval, pages 16–23, 1996.
- [Dumais 1995] S. Dumais *Latent Semantic Indexing (LSI)*, TREC-3 Report In proceedings of TREC-3, pages : 319-230, 1995.
- [Diaz & jones, 04] Diaz F., Jones R. : Using temporal profiles of queries for precision prediction. The 27th annual international conference on Research and development in information retrieval, ACM Press, (2004) 18–24
- [Deerwester & al., 1990] S.C. Deerwester, S. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman. Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science, 41(6), p. 391-407, 1990.
- [Declaris et al., 94] DeClaris N, Harman D, Faloutsos C, Dumais S and Oard D. (1994) Information filtering and retrieval: overview, issues and directions, Proceedings of the 16th Annual International Conference of the IEEE Engineering in Medicine and Biological Society, pp. 42-49, 1994.
- [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.

- [Foltz & Dumais, 1992] Foltz P. & Dumais S., "Personalized information delivery: An analysis of information filtering methods", *Communications of the ACM*, 35(12): 51–60, 1992.
- [Foltz 1990] P. W. Foltz, *Using Latent Semantic Indexing for information filtering*. CACM, pages : 40-47, 1990.
- [Furnas et al 1988] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, K. E. Lochbaum, *Information retrieval using a singular value decomposition model of latent semantic structure*. Proceedings of ACM SIGIR 88, pages : 465-480, 1988.
- [Gaussier et al. , 97] E. Gaussier, G. Grefenstette, et M. Schulze. Traitement du langage naturel et recherché d'informations: quelques expériences sur le français. Premières journées Scientifiques et technique des réseaux Francophones de l'ingénierie de la langue de l'AUPELF-UREF, 1997.
- [Grossman et al., 98] Grossman D.A. & Frieder O., "Information retrieval: Algorithms and heuristics", Kluwer Academic Publishers, 1998.
- [Gaussier et al., 00] E. Gaussier, G. Grefenstette, D. Hull, et C. Roux. Recherche d'information en français et traitement automatique des langues. *Revue traitement Automatique des langue (TAL)*, 41(2): 473-494, 2000.
- [Hiemstra98] HIEMSTRA D., KRAAIJ W., « Twenty-One at TREC-7 : Ad hoc and Cross-language track », dans [VOO 99], p. 227-238, 1998.
- [Hiemstra, 02] Djoerd Hiemstra: Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term. *SIGIR 2002*: 35-41.
- [Hauff & Azzopardi, 05] Hauff C., Azzopardi L. : Age dependent document priors in link structure analysis. *The 27th European Conference in Information Retrieval* ». Springer, (2005) 552–554.
- [Kraaij, 04] Wessel Kraaij. "Variations on Language Modeling for Information Retrieval" PHD thesis. Centre for Telematics and Information Technology. Netherland. 2004.
- [Kwok, 1989] Kwok K., "A neural network for probabilistic information retrieval", In *Proceedings of ACM SIGIR*, pages 21–30, 1989.

- [Kleinberg, 99] J. M. Kleinberg. "Authoritative sources in a hyperlinked environment". J. ACM, 46(5):604–632, 1999.
- [Kraaij & Hiemstra, 02] Kraaij W., Westerveld T., Hiemstra D. : 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
- [Ieféve, 00] *la recherche d'information*, du texte intégrale au thesaurus, Paris, hermès, 253 p .
- [Luhn, 58] H. P. Luhn. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development 2(2), 159–165. 1958.
- [Largouet, 05] Anita Largouet. "La recherche d'informations sur Internet. Rapport de recherche", Service Commun de Documentation - Université Michel de Montaigne - Bordeaux3. 2005.
- [Lempel et Moran, 2000] R. Lempel et S. Moran. The stochastic approach for link structure analysis (SALSA) and the TKC effect. j-COMP-NET-AMSTERDAM, 33(1–6) :387–401, June 2000.
- [Lavrenko, 01] Lavrenko V., Croft, W. B. : Relevance-based language models. The 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana, Croft W.B., Harper D.J., Kraft D.H., Zobel J. (Eds.), New York: ACM, (2001) 120-127
- [Mooers, 48] C.N. Mooers. Application of Random Codes to the Gathering of Statistical Information, MIT Master's Thesis, 1948.
- [MONTEIL, 95] MONTEIL Marie Gaëlle, indexation manuelle et automatique : comparaison et perspectives, IDT 95, 12^e Congrès , Paris , 12-15 juin 1995, pp. 214-215.
- [Miller98, Miller99] MILLER D. R. H., LEEK T., SCHWARTZ R. M., « BBN at TREC-7: using Hidden Markov Models for Information Retrieval », dans [VOO 99], p. 133-142, 1998.
- [maniez, 02] Maniez (J.). *Actualité des langages documentaires, fondements théoriques de la recherche d'information*, ABDS, Paris, 2002.
- [Mothe, 1994] J. Mothe : "Modèle Connexionniste pour la Recherche d'Information, Expansion dirigée de requêtes et apprentissage", Thèse de Doctorat de l'Université Paul Sabatier, Toulouse (France), 1994.

- [Maron & Kuhns, 1960] M.E. Maron and J.L. Kuhns, On relevance, probabilistic indexing and information retrieval, *Journal of the Association for Computing Machinery*, 7: 216-244, 1960.
- [Miller & leek, 99] Miller D. R. H., Leek T., Schwartz R. M. : A hidden markov model information retrievalsystm. Hearst *et al.* (Eds.), (1999) 214–221
- [Nie et al. , 99] F. Ren, L. Fan, and J.-Y. Nie. Approach: How to Acquire Knowledge in an Actual Application System, IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, 1999, pp.136-140.
- [ounis 07] "combining fields for query expansion ans adaptive query expansion", Ben He- ladh Ounis, Departement of Computing Science, University of glasgow , Glasgow G12 8QQ , United kingdom, available online 26 january 2007
- [Piwowarski, 2003] Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information,*Benjamin Piwowarski*, Université Pierre et Marie Curie (UPMC), 2003
- [Ponte et al., 98] Jay M. Ponte, W. Bruce Croft: A Language Modeling Approach to Information Retrieval. SIGIR 1998: 275-281 .
- [Ponte & Croft, 98] Jay M. Ponte and W. Bruce Croft. "A Language Modeling Approach to Information Retrieval". Research and Development in Information Retrieval, Proc. ACM-SIGIR, pp. 275-281, 1998
- [Ponte98] PONTE J. M., CROFT W. B., « A Language Modeling Approach to Information Retrieval », dans [CRO 98], p. 275-281, 1998.
- [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.
- [Picarougne, 04] F. Picarougne. "Recherche d'information sur Internet par algorithmes évolutionnaires". Thèse de doctorat en informatique, Université François Rabelais Tours, page 29, Novembre 2004.
- [Robertson & Walker, 94] ROBERTSON S. E., WALKER S., « Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval », *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, p. 232-241, 1994.

- [Rijsbergen, 1979] Rijsbergen C. V., "Information retrieval", In Information retrieval experiments. Butterworths, London, 2nd edition, 1979.
- [Robertson, 1977] Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, 33 (4), 294-304.
- [Robertson & Sparck Jones, 76] Robertson S., & Sparck-Jones K., "Relevance weighting of search terms", *JASIS*, 27(3):129–146, 1976.
- [Rijsbergen, 81] C J van Rijsbergen, D. Harper and M. Porter. The selection of good search terms. *Information Processing and Management*. 17. pp 77-91. 1981.
- [Robertson et al. , 97] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conférence on Research and development in information retrieval*, pages 16–24. ACM Press, 1997.
- [Salton et al., 83] Salton, G., E.A. Fox, H. Wu. "Extended Boolean information retrieval system". *CACM* 26(11), pp. 1022-1036, 1983.
- [Salton, 1971] Salton G., "The SMART Retrieval System - Experiment in Automatic Document Processing", Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [Salton, 1983] Salton G., "Introduction to modern information retrieval", New York McGraw-Hill, 1983.
- [Salton, 1989] Salton, G., "Automatic text processing: The transformation, analysis and retrieval of information by computer", Addison-Wesley publishing, 1989.
- [Song et al., 99] Song, F. and Croft, B. 1999. "A general language model for information retrieval" . In: *CIKM'99*, pp. 316–321.
- [Salton, 88] Salton, G. Syntactic approaches to automatic book indexing. In *Proc. Of the annual meeting on Association for Computational Linguistics (ACL) (1988)*, Departement of computer Science, Cornell University, Ithaca, New York, pp.204-210.
- [Salton et al 88] G. Salton, and C. Buckley. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal* 24 (5), 513-523.
- [Singhal et al. ,97] Singhal, A., Mitra, M., and Buckley, C. 1997. Learning routing queries in a query zone. In *Proceedings of the 20th Annual*

- international ACM SIGIR Conference on Research and Development in information Retrieval (Philadelphia, Pennsylvania, United States, July 27 - 31, 1997). N. J. Belkin, A. D. Narasimhalu, P. Willett, and W. Hersh, Eds. SIGIR '97. ACM Press, New York, NY, 25-32.
- [Singhal et al., 95] A. K. Singhal, G. Salton, M. Mitra, C. Buckley, *Document length normalization*, Information Processing and Management, 32(5), pages : 619-633, 1995
- [SparkJone, 71] K. Sparck Jones. Automatic keywords classification for information retrieval. 1971.
- [SparkJones, 79] K. Sparck Jones. Experiments in relevance weighting of search terms. Inf. Process. Manage. 15(3): 133-144, 1979.
- [Salton, 1989] G. Salton, *Automatic Text Processing*, The Transformation Analysis and Retrieval of Information by Computer. 1989. Addison Wesley.
- [Salton, 1971] G. Salton. The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.
- [Tamine, 2000] Tamine L., "Optimisation de requêtes dans un système de recherche d'information : Approche basée sur l'exploitation de techniques avancées de l'algorithmique génétiques", thèse de Doctorat en informatique, Université Paul Sabatier, Toulouse, 2000.
- [Turtle & Croft, 1991] Turtle H.R, & Croft W.B, "Evaluation of an Inference Network-Based Retrieval Model", ACM Transactions On Information Systems, 9(3):187-222, 1991.
- [Wong & 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.
- [Wong et al., 1985] S.K.M. Wong, W. Ziarko and P.C. N. Wong : Generalized Vector Space Model in Information Retrieval. In Proc of the 8th ACM SIGIR Conference on Research and Development, New-York USA, 1985.
- [Zadeh, 1965] L.A. Zadeh, "Fuzzy sets", Information and control , 1965.
- [Zipf, 1949] Zipf G., "Human Behaviour and the Principle of Least Effort", 1st edition, Addison Wesley, 1949.

- [Zhu & Gauch, 00] Zhu X. L., Gauch S. : 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