Table des matières

Chapitre III: Contribution à la sélection de collections

Chapitre IV: Expérimentations et Evaluation

Tables des figures

Table des tableaux

Introduction générale

Les systèmes de recherche d'information ont évolué depuis que l'homme a commencé à rassembler ses connaissances dans des manuscrits. Il parait que les égyptiens étaient les premiers à avoir commencé l'indexation de la grande bibliothèque d'Alexandrie.

De nos jours, grâce à la numérisation, nous disposons de librairies numériques (digital libraries, en anglo-saxon) voluminisées par l'active participation des internautes et par la diversité des sources d'information. Il parait que, pour la recherche scientifique à elle-seule, plus de 1.4 millions d'articles sont publiés chaque jour, d'autant plus que le nombre des éditeurs ne cesse d'augmenter.

Les données enregistrées, de point de vue moteur de recherche, « car nous ne traitons pas les données structurées dans cette thèse », sont de deux types. Le premier type est composé de l'ensemble de documents enregistrés les uns indépendamment des autres. Autrement dit, chaque document possède ses propres attributs caractéristiques comme l'adresse URL, qui est unique et permet le rend accessible. Ce type de documents peut être indexé par un moteur général comme Google ou Ask ou Yahoo ou tout autre moteur de recherche commercial ou académique. Le second type de documents est composé par des collections très larges provenant de la même source et traitant le même sujet. Ces collections sont formées à partir d'un ensemble de documents portant chacun un identifiant dans la collection. Il s'agit ici de rapports boursiers ou de stories de reportages ou rapport de médecins ou tout autre domaine. Ces collections ne peuvent pas être indexées par des moteurs de recherche généraux. Il faut disposer de mécanismes bien spécifiques qui permettent de localiser l'information qu'elles contiennent.

Dans le but d'extraire l'information contenue dans ces collections et afin de satisfaire ses usagers, tout système d'information doit réunir un arsenal important d'outils, qui permettent de crawler, indexer, clustériser, compresser, de façon quasi continue. Cette activité perpétuelle est indispensable pour ne pas perdre la "clientèle" car, en réalité, les utilisateurs sont exigeants en matière de qualité et temps de réponse.

Dans cette politique de recherche, plusieurs modèles ont été établis. Le but est de se rapprocher au mieux de la réponse théorique qui pourrait intéresser un utilisateur. Parmi ces modèles, on rencontre le modèle booléen où il s'agit de requête sur des termes raccordés par les opérateurs booléens "ET" et "OU". Ce modèle est simple à implémenter mais les résultats qui peuvent être retournés ne sont pas d'une grande pertinence. Ce modèle a été étendu et même de nos jours, on rencontre des travaux qui l'implémentent. Le modèle vectoriel est une

autre spécification qui représente le document sous forme de vecteur. Chaque composante est une pondération d'un terme représentant ainsi son importance. Ce modèle est maniable et offre des possibilités énormes pour répondre correctement aux besoins des utilisateurs. Il reste toujours à trouver la meilleure pondération afin de ne pas pénaliser certains documents et favoriser d'autres. Une énorme importance est donnée au modèle probabiliste. L'idée derrière cette spécification est d'exploiter la formulation bayésienne pour répondre aux requêtes. Malgré les grands résultats obtenus, il reste toujours difficile de l'accepter car dans ce type de formulation beaucoup de paramètres peuvent être utilisés. Ces paramètres ne sont pas stables et doivent être ajustés à chaque utilisation sur les différents jeux de données.

L'indexation est une tâche indispensable dans le processus d'évaluation des requêtes. Elle-même est composée d'un nombre de phases nécessaires selon le cas d'études, comme il sera expliqué dans ce manuscrit. La finalité de cette étape génère un index sous forme de listes inversées, dans le cas général. Ces listes déterminent les emplacements des termes, avec certaines caractéristiques, dans les documents. Ces listes sont, logiquement, trop longues. Il est évident qu'un bon mécanisme peut tirer l'information la plus pertinente qui puisse satisfaire les utilisateurs en matière de qualité de réponse. Néanmoins, l'énormité des listes rend difficile, voire inapproprié, leur balayage car, d'une part, c'est une opération coûteuse en termes de temps et d'effort, et, d'autre part, plus le temps passe plus les informations deviennent obsolètes. Donc, la manipulation des anciennes versions des données se feras uniquement sous l'insistance de l'utilisateur.

De ce constat, naquis les algorithmes qui s'arrêtent au plutôt (early termination) comme les algorithmes WAND, TA, TPUT, … Ce type d'algorithmes localise les meilleures (Top-k) réponses selon un ou plusieurs critères. Ces réponses sont aussitôt retournées à l'utilisateur. La logique derrière cette stratégie est que le système d'information propose le meilleur de ce qu'il détient comme information. Cette manière de détection permet d'évite le scanne total des listes inversées.

Nous avons aussi assisté à la naissance d'un autre type d'algorithmes dits probabilistes, comme nous l'avons déjà mentionné. Ceux-ci estiment qu'il y a une forte probabilité que les réponses, localisées dans le système, satisferont l'utilisateur.

Ces algorithmes implémentent des probabilités conditionnelles ainsi que des mesures d'efficacité. Dans cette catégorie d'algorithmes, on rencontre des algorithmes comme CORI, BM25, CS, … Cependant, le cheval de bataille réside dans la définition de ce qu'on appelle "Pertinence". Ce terme exprime le degré de certitude qu'un document soit la meilleure réponse pour une question donnée. Ces algorithmes définissent dans leurs fonctions de calcul des scores des paramètres heuristiques. Ces paramètres peuvent être un handicap. Dans la littérature, nous avons trouvé que la même méthode peut subir plusieurs formulations et plusieurs versions suite au problème de la détermination des valeurs de ces paramètres. De

façon simple, nous disons que si les conditions du choix des valeurs des paramètres ne sont pas réunies, la méthode de localisation devient instable.

Le sujet de cette thèse rentre dans la problématique de la recherche d'information et se consacre principalement à la sélection de collections. Nous nous proposons de définir une fonction de sélection qui n'utilise aucun paramètre extra-collection issus d'heuristique. On suggère l'exploitation de l'anatomie de la collection sans aucune intrusion qui puisse avantager des collections et pénaliser d'autres. Nous avons appelé notre méthode de sélection : Sélection basée sur le degré de pertinence ou CSRD (pour Collection Selection Based on the Relevance Degree, en anglo-saxon). Dans sa logique, CSRD retourne les meilleures collections en se basant sur la similarité entre les termes des requêtes et les documents. Une collection est jugée pertinente si les degrés de pertinence de ses documents sont élevés. Dans un objectif d'optimisation, nous implémentons un broker qui contrôle la sélection dans la mesure où il n'interroge que les collections qui ont effectivement un lien avec la requête.

Pour démontrer l'efficacité de CSRD, nous nous proposons de la comparer avec les méthodes populaires CORI et CS. Celles-ci utilisent des procédés probabilistes pour le classement des documents. Ces méthodes définissent dans leurs formules des paramètres heuristiques, comme toutes les méthodes de ce domaine. Ces paramètres ont un mauvais impact sur la stabilité des méthodes en question. Un calibrage donc devient obligatoire. Les expérimentations que nous avons menées vont montrer le degré de perte de ces méthodes et que CSRD est très compétitive.

Le reste de ce manuscrit se présente comme suit :

Le premier chapitre est une introduction aux systèmes d'information. Nous présentons alors la définition, l'architecture et le processus de recherche. Ce chapitre définit aussi d'une façon générale la notion de pertinence, un acteur clé dans le processus de la sélection. Du coup nous discuterons plusieurs définitions dans ce sens.

Le deuxième chapitre est une présentation des modèles de la recherche d'information. Ces modèles, Booléen, Vectoriels et Probabilistes, sont donnés en détails. Ce chapitre ne peut se terminer sans la présentation du système Okapi et sa méthode BM25. Celle-ci est une référence et dresse un état de l'art de la recherche de document. BM25 a subi plusieurs formulations depuis sa première définition en 1997.

Le troisième chapitre traite le sujet de cette thèse. Il définit la notion de la pertinence, le degré de la pertinence des documents et des collections. La notion de degré de pertinence et sa nouvelle définition. Ce chapitre inclut la présentation de plusieurs méthodes de sélection des collections dans des systèmes centralisés et des systèmes décentralisés. Nous présenterons en détails les méthodes CORI [27],CS [2]. Nous enchaînerons par la présentation de la méthode publiée dans [28] ; celle-ci est prometteuse. L'objectif est de donner un maximum d'informations aux lecteurs. On continue cette partie par la présentation détaillée de la méthode CSRD.

Le quatrième chapitre aborde tous les aspects de l'implémentation. Les expérimentations de comparaison menées sont présentées et commentées en détails.

Nous clôturons ce manuscrit par une conclusion générale et des travaux futurs ;

CHAPITRE I

Recherche d'information

Sommaire

1. Introduction

De nos jours, un très grand nombre d'utilisateurs utilisent des systèmes de recherche d'information pour trouver l'information dont ils ont besoin. Le processus qui consiste à rechercher dans une collection de documents l'information répondant à un besoin est appelé Recherche d'Information (RI) (ou Information Retrieval IR en anglo-saxon). La recherche d'information n'est pas un domaine récent. Elle date des années 1940, dès la naissance des ordinateurs. La recherche d'information est le domaine qui étudie la manière de retrouver des informations dans un corpus suite à une requête d'utilisateur donnée, et, par conséquent, ce dernier manipule différent concepts : la requête (besoin en information), la collection, les documents, la pertinence, etc. L'objectif de ce chapitre est de présenter les concepts de base de la RI.

2.Définition de la Recherche d'Information:

D'une façon générale [15][19], la recherche d'information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer un ensemble de documents à un utilisateur en fonction de son besoin en information. Le défi est de pouvoir, parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur (la bonne sélection). L'opérationnalisation de la recherche d'information est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI). D'une façon plus précise, on peut définir la recherche d'information comme étant un domaine qui traite la représentation (l'indexation), le stockage, l'organisation et la sélection d'information répondant aux besoins des utilisateurs.

3. Concepts de base de la RI :

La recherche d'information est traditionnellement considérée comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur. La gestion de ces informations implique le stockage, la recherche et l'exploration des documents pertinents. De cette constatation plusieurs concepts clés peuvent être définis. Nous avons donc trouvé utile de les clarifier. A cet effet, une synthèse des travaux de [9] [20] [49] [19] nous a permis de dégager les concepts suivants :

3.1. Collection de documents

La collection de documents (ou fond documentaire) constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents. Dans le cas général et pour un souci d'optimalité, la base contient des représentations simplifiées mais suffisantes pour ces documents. Ces représentations sont étudiées de telle sorte que la gestion (ajout et suppression d'un document) et l'interrogation (recherche) de la base se fassent dans les meilleures conditions de coût.

3.2. Document

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

3.3.Besoin d'information

La notion de besoin en information en recherche d'information est souvent assimilée au besoin de l'utilisateur. Trois types de besoin utilisateur ont été définis par Ingwersen [43] :

- **Besoin vérificatif:** l'utilisateur cherche à vérifier le texte avec les données connues qu'il possède déjà. Il recherche donc une donnée particulière, et sait même souvent comment y accéder. La recherche d'un article sur Internet à partir d'une adresse connue serait un exemple d'un tel besoin. Un autre exemple serait de chercher la date de publication d'un ouvrage dont la référence est connue. Un besoin de type vérificatif est dit stable, c'est-à-dire qu'il ne change pas au cours de la recherche.
- **Besoin thématique connu:** l'utilisateur cherche à clarifier, à revoir ou à trouver de nouvelles informations dans un sujet et un domaine connus. Un besoin de ce type peut être stable ou variable ; il est très possible en effet que le besoin de l'utilisateur s'affine au cours de la recherche. Le besoin peut aussi s'exprimer de façon incomplète, c'est-à-dire que l'utilisateur n'énonce pas nécessairement tout ce qu'il sait dans sa requête mais seulement un sous-ensemble. C'est ce qu'on appelle dans la littérature le label.
- **Besoin thématique inconnu :** cette fois, l'utilisateur cherche de nouveaux concepts ou de nouvelles relations en dehors des sujets ou des domaines qui lui sont familiers. Le besoin est intrinsèquement variable et il est toujours exprimé de façon incomplète.

3.4. Requête

La requête constitue l'expression du besoin en information de l'utilisateur. Elle représente l'interface entre le SRI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique.

3.5. Modèle de représentation

Un modèle de représentation est un processus permettant 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 ou groupes de termes (concepts), significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids, pour différencier leurs degrés de représentativité du contenu sémantique de l'unité en question. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelée dictionnaire constituant le langage d'indexation. Ce type de langage garantit le rappel de documents lorsque la requête utilise dans une large mesure les termes du dictionnaire. En revanche, il y a un risque important de perte d'informations lorsque la requête s'éloigne de ce vocabulaire.

3.6. Modèle de recherche

Il représente le modèle 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 il est étroitement lié au modèle de représentation des documents et des requêtes.

3.7. La pertinence

La pertinence est une notion fondamentale et cruciale dans le domaine de la RI. La pertinence d'un point de vue système est différente de celle d'un point de vue utilisateur. Les modèles de RI définis dans la littérature mesurent cette pertinence comme un score, cherchant à évaluer la pertinence des documents vis-à-vis d'une requête. Cette pertinence est mesurée par une similarité de représentation document-requête (modèle vectoriel) et une probabilité de pertinence des documents étant donnée une requête (modèle probabiliste).

Dans la section 5.3.3, nous verrons des définitions générales de la notion de pertinence. Ces définitions sont principalement liées à la manière avec laquelle on veut évaluer des requêtes. On doit aussi remarquer que cette notion dépend du contexte du travail. Par exemple, dans la conférence TREC[1] [84] [87], elle consiste à réaliser un matching entre les requêtes, les documents et les jugements des experts. Alors pour les collections ouvertes, c'est-à-dire le domaine des moteurs de recherche généralisés, tels que PageRank de Google [71] ou HITS de Ask [48], la réponse aux requêtes consiste à trouver les documents contenant un maximum de termes en intersection avec la requêtes, sans se soucier pour autant de la satisfaction de

[1] http://trec.nist.gov

l'utilisateur, puisque la fonction de tri ne dépend d'aucun critère reflétant l'envi de cet utilisateur.

Dans notre travail, nous avons introduit d'une manière plus précise cette notion, et nous avons offert des nouvelles mesures de la pertinence qui permettent de refléter un ordre naturel entre les collections. Les détails sont donnés dans le chapitre III section 5.1 page (85).

4. Evolution de la recherche d'information

Comme il est cité dans la définition, la recherche d'information s'intéresse à la représentation, le stockage, l'organisation et l'accès aux informations. La représentation et l'organisation des informations doivent permettre à l'utilisateur un accès facile et rapide aux informations dont il a besoin.

Le but de la recherche d'information était au début, l'indexation d'un texte (extraire de ce texte des mots clés) et la recherche de documents dans un corpus à partir de ces mots clés.

Aujourd'hui, la recherche d'information s'intéresse aussi à la modélisation, la classification de documents et leur catégorisation, les interfaces, la visualisation de la réponse à la requête, le filtrage, les langages, etc.

A partir de 1940 [4], date de naissance de la recherche d'information, le problème du stockage et de la recherche d'information était déjà posé et le constat était que les volumes d'informations augmentaient, et par conséquent, les accès rapides étaient de plus en plus difficiles.

Dans les années 1950, on commençait de petites expérimentations en utilisant des petites collections de documents (références bibliographiques). Le modèle utilisé était le modèle booléen.

Dans les années 1960 et 1970, des expérimentations plus larges ont été menées. Les chercheurs ont développé une méthodologie d'évaluation du système qui est actuellement utilisée dans d'autres domaines aussi. Des collections de test ont été conçues pour évaluer des systèmes différents. Ces collections de test ont beaucoup contribué à l'avancement de la RI. Car on pouvait les utiliser pour comparer différentes techniques, et mesurer leurs impacts en pratique.

Durant les années 1980, la recherche d'information a été influencée par le développement de l'intelligence artificielle.

Enfin, une vision nouvelle a fait son apparition ces dernières années qui consiste à définir des outils de la RI dans le domaine des applications multimédia et hypertexte.

Cette vision a connu un essor considérable au début des années 90 avec l'introduction du Word Wide Web. La venue de l'internet a aussi modifié la recherche d'information. La problématique est élargie. Par exemple, la RI traite maintenant plus souvent des documents multimédias qu'avant. Du fait du nombre important d'utilisateurs, de l'expansion des réseaux de communication et de l'explosion des sources d'informations, de nombreux travaux ont été réalisés dans l'optique de définir des modèles et techniques efficaces de la RI. Pour de tels systèmes des outils efficaces d'organisation et de recherche sont nécessaires.

5. Définition d'un système de recherche d'information

Plusieurs définitions d'un système de recherche d'information ont vu le jour. Nous citons dans ce contexte les deux définitions suivantes :

Définition 1 : Un système de recherche d'information (SRI) est un ensemble de logiciels assurant l'ensemble des fonctions nécessaires à la recherche d'information. Le moteur de recherche est au cœur d'un tel système mais il n'en est qu'un composant[2].

Définition 2 : Un système de recherche d'information (SRI) intègre un ensemble de modèles et de processus permettant de sélectionner des informations pertinentes en réponse au besoin en information d'un utilisateur représenté à l'aide d'une requête. Dans un contexte documentaire, un SRI permet de gérer une collection de documents stockés sous forme d'une représentation intermédiaire permettant de refléter aussi fidèlement que possible leur contenu sémantique. Un SRI peut être défini comme l'ensemble des procédures et des opérations permettant la gestion, la représentation, l'interrogation, la recherche, le stockage et la sélection des informations répondant aux besoins d'un utilisateur [20].

Le but principal d'un SRI est de retrouver les documents pertinents en réponse à une requête utilisateur. Ainsi, le rôle de SRI est de mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents (fiche ou enregistrement) au moyen d'une fonction de comparaison (ou de correspondance).

L'essor du web a remis la recherche d'information face à de nouveaux défis d'accès à l'information surtout le problème du web profond (Deep Web) et le problème de la bonne

[2]https://fr.wikipedia.org/wiki/SystC3A8me_de_recherche_d27information

sélection de collections. Il s'agit Donc de retrouver une information pertinente dans un espace diversifié et de taille considérable.

La figure 1.1 suivante représente un système de recherche d'information :

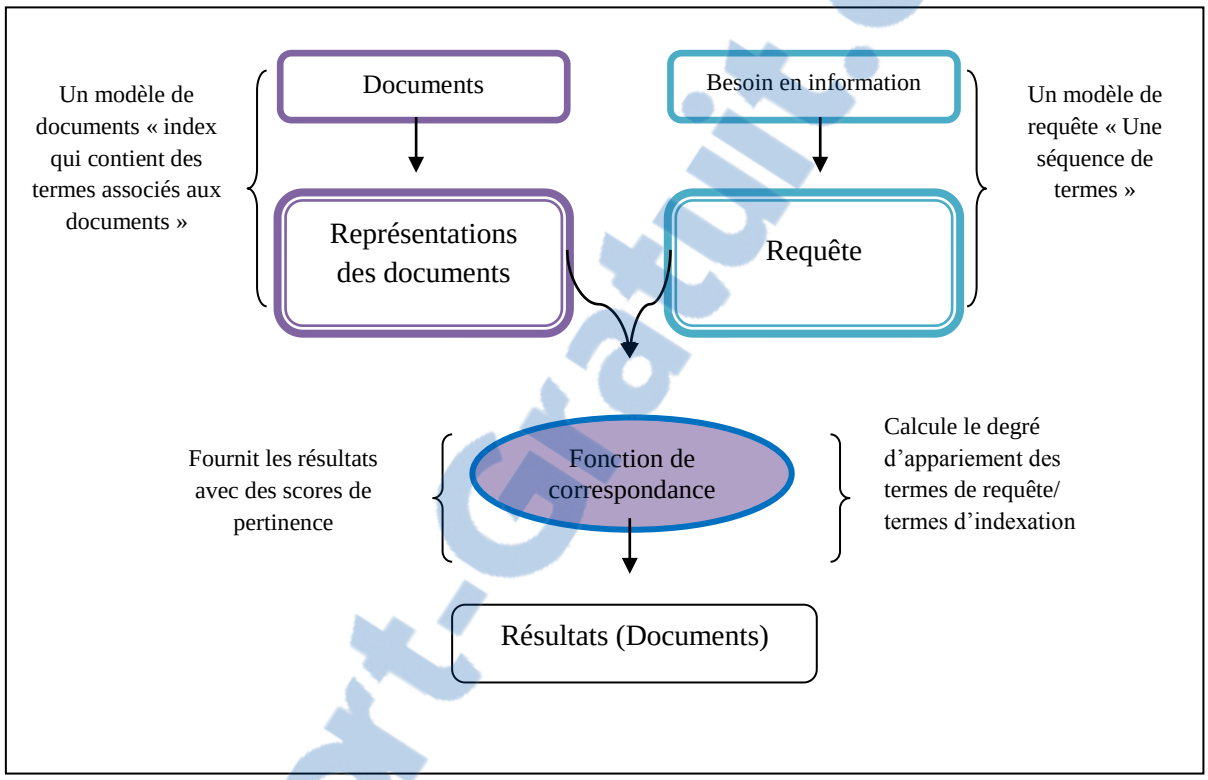


Figure 1.1Système de Recherche d'information

6. Le processus de Recherche d'Information :

Le processus de recherche d'information a pour but la mise en relation des informations disponibles, d'une part, et les besoins de l'utilisateur d'autre part. Ces besoins sont traduits de façon structurée par l'utilisateur sous forme de requête. Cette dernière est formulée, par l'utilisateur, dans un langage de requête qui peut être le langage naturel, un langage à base de mots clés ou le langage booléen. La mise en relation des besoins utilisateurs et des informations est effectuée grâce à un système de Recherche d'Information (SRI), dont le but est de retourner à l'utilisateur le maximum de documents pertinents par rapport à son besoin et le minimum de documents non pertinents.

Un système de recherche d'information manipule un corpus de documents qu'il transpose à l'aide d'une fonction d'indexation en un corpus indexé (le système crée des représentations internes pour la requête et les documents : ce processus de représentation est appelé

indexation). Ce corpus permet de résoudre des requêtes traduites à partir de besoins utilisateur. Un tel système repose sur la définition d'un modèle de recherche d'information qui effectue ces deux transpositions et qui fait correspondre les documents aux requêtes. La transposition d'un document en un document indexé repose sur un modèle de documents. De même, la transformation du besoin utilisateur en requête repose sur un modèle de requêtes. En fin, la correspondance entre une requête et des documents s'établit par une relation de pertinence [19][57]. La figure 1.2 suivante représente les différentes étapes d'un processus de recherche d'information.

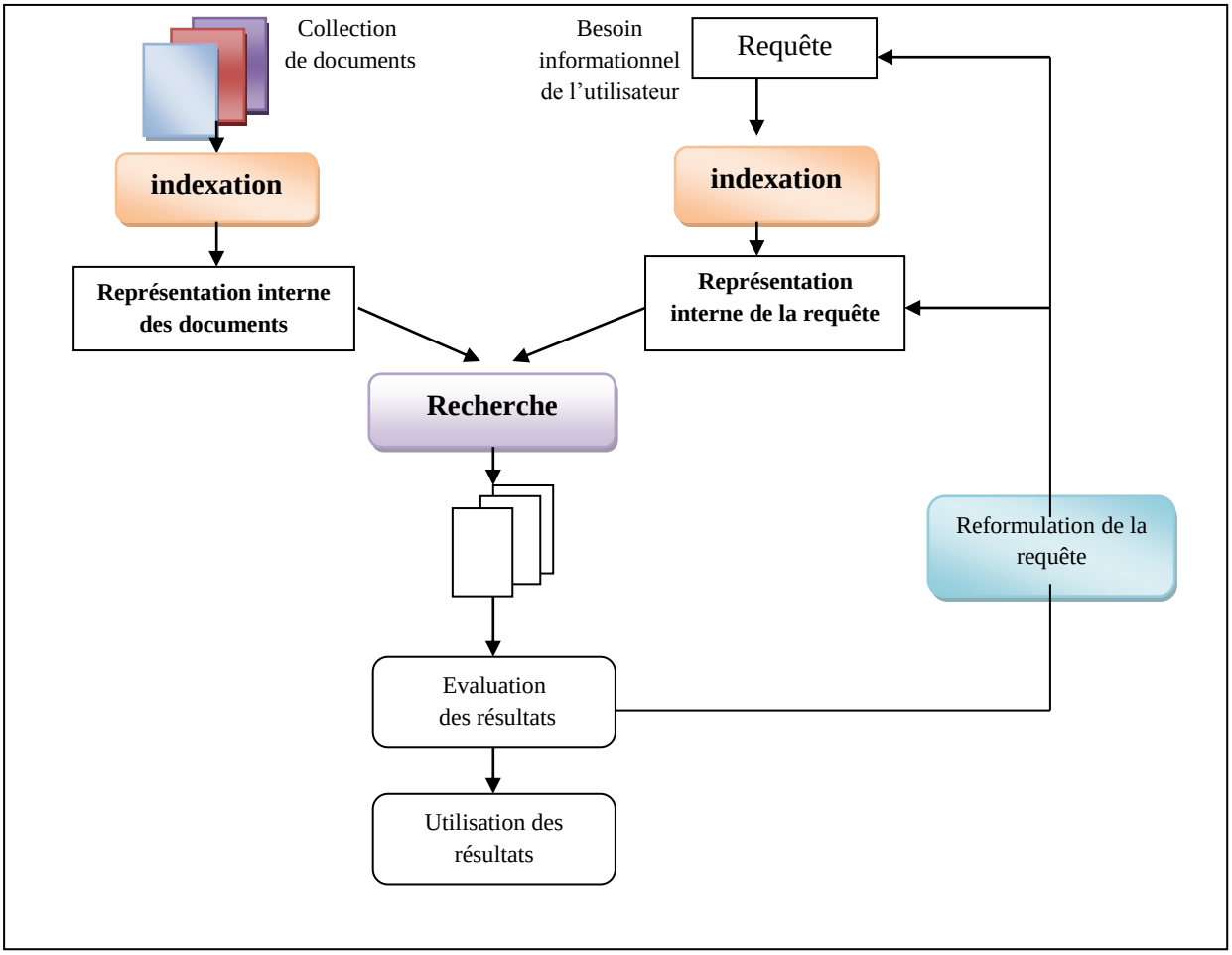


Figure 1.2 Processus en U de la RI [1]

6.1. Le processus d'indexation

Le processus d'indexation est mis en œuvre afin d'extraire préalablement une représentation homogène du contenu sémantique, sous forme de termes d'indexation qui sont des éléments d'un langage d'indexation. Dans un SRI, un document est considéré comme un support qui véhicule de l'information. La phase d'indexation permet donc, de capturer cette information et de la représenter selon un modèle : le modèle de documents. L'information capturée

correspond au contenu sémantique du document et la représentation de ce contenu sémantique est appelée un index de document. Dans les systèmes classiques de recherche d'informations, l'indexation est organisée en trois étapes, comme l'illustre la figure 1.3 :

- L'extraction des termes du document ;
- La sélection des termes discriminatifs pour un document ;
- La pondération des termes.

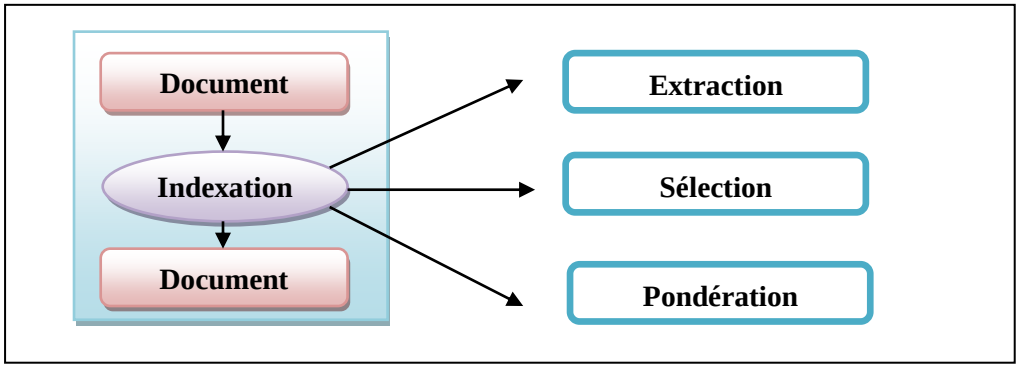


Figure 1.3 Indexation d'un document [105]

6.1.1. Les différentes formes d'un index

Les index peuvent prendre différentes formes allant de mots simples à des structures sémantiques plus complexes impliquant plusieurs concepts et relations [74][7][112]. Les descripteurs représentent l'information atomique d'un index. Ils sont censés indiquer de quoi parle le document. On parle aussi d'unités élémentaires (en anglais "Tokens")[78]. Le but étant de les choisir de manière à ce que l'index (qui réduit la représentation) perd le moins d'informations sémantiques possible. Habituellement les descripteurs sont des mots du document, des N-grammes ou des concepts.

- **Les mots du document** : toute chaîne de caractères compris entre deux séparateurs (espace, virgule…). Au niveau de l'indexation, on peut extraire les mots tels qu'ils sont présentés dans le document comme on peut effectuer certaines transformations sur ce mot en vue d'une normalisation. On peut ainsi lemmatiser ou raciniser ("Stemming") les mots, qui sont deux processus linguistiques qui consistent, pour le premier à présenter les mots sous des formes non fléchies, et pour le second à retrouver les éléments de base porteurs du sens du mot, obtenu par l'élimination des affixes et des désinences. Au lieu d'indexer par des mots on indexe alors par le lemme correspondant.

- **Les concepts :** termes ou mots-clés: il s'agit d'expressions pouvant contenir un ou plusieurs mots. Ces concepts sont le plus souvent entrés manuellement (cas de l'indexation manuelle, ou semi-automatique) et peuvent être écrits de manière libre par un utilisateur, ou, ce qui est souvent le cas, doivent être choisis parmi une liste de concepts (on parle alors de vocabulaire contrôlé). Cette liste de concepts sera le plus souvent décrite dans un thésaurus (dans le cas des termes, on parlera de terminologie).
- **Les N-grammes :** Il s'agit d'une représentation originale d'un texte en séquences de N caractères consécutifs. On trouve des utilisations de bi-grammes et trigrammes dans la recherche documentaire. Ils permettent de reconnaître des mots de manière approximative et ainsi de corriger des flexions de mots ou même des fautes de frappe ou d'orthographe. Le tableau ci-dessous représente un exemple de ces différentes formes d'index :

Texte		Modèles de la recherche d'information structurée
Mot	**Origine**	Modèle, de, la, recherche, d, information, structurée
	Lemme	Modèle, de, la, recherche, information, structure
	Racine	Modèl, d, l, recherch, inform, structur
Concept		R.I
Bi-gramme		Mo, od, de, el, le, es, s_, _d, de, e_, _l, la,…, ée

Tableau 1. 1 Différentes formes d'index

Dans la pratique, la forme la plus utilisée est la représentation par mots-clés. L'extraction automatique des concepts d'une collection de documents est souvent une entreprise très délicate et nécessite l'utilisation des techniques du traitement automatique du langage naturel, vu que ces derniers sont directement liés à la langue utilisée. [40] donne plus de détails sur l'utilisation des syntagmes nominaux en recherche d'information. L'indexation à base de concepts est souvent manuelle ou semi-automatique, donc inadaptée aux larges collections de documents. Les N-grammes, quant à eux, sont indépendants de la langue utilisée. Mais nécessitent, par contre, un espace mémoire assez important et plusieurs traitements doivent être effectués sur la requête dans un processus de recherche d'information. Ils sont plus utilisés pour la classification des documents que pour la recherche d'information [44]. Dans la suite de ce manuscrit, nous considérons uniquement les descripteurs sous forme de termes (mots-clés).

Le processus d'indexation effectue le transfert de l'information contenue dans le texte d'un document vers un autre espace de représentation traitable par un système informatique [78]. A partir d'une collection de documents, le processus d'indexation nous renvoie une liste d'index structurée (figure 1.4). On utilise ce résultat, le plus souvent, pour effectuer des recherches

d'informations. Mais, il peut également servir à comparer et classifier des documents, proposer des mots-clés, faire une synthèse automatique de documents, calculer les co-occurrences de termes…

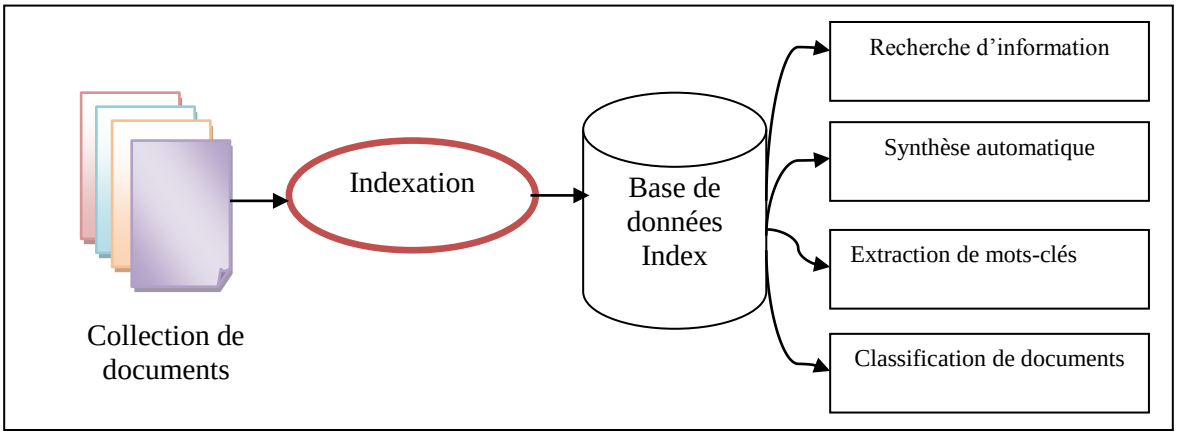


Figure 1.4 Processus d'indexation [78]

En réalité, l'index suscite jusqu'alors des travaux intenses afin d'accéder le plus rapidement et efficacement possible à l'information. [64] est un travail qui a pour but de montrer le caractère crucial pour réaliser un index des collections larges du Web. Cette technique se fait sous forme d'un pipelining entre processeurs pour réduire substantiellement le temps de montage de l'index. [8] présente un nouvel index composé de trois parties. Cette composition permet des mises plus rapides que les index traditionnels. L'objet de [56] est de fournir une manière efficace pour le maintien de l'index inversé.

Vu la taille immense que peut avoir [109] présente une méthode statique pour la réduction de cette taille. La méthode proposée retourne le document entièrement selon sa pertinence et son importance.

Dans ce même but de réduction de l'éventuelle taille des index, [37] présente une méthode de compression des index en utilisant le code Golomb. Cette méthode étend ce code pour la compression des nombres entiers.

Pour des besoins de privacité, [110] présente un schéma pour la sécurisation de l'index inversé. Ce schéma est basé sur la combinaison d'un ensemble de clés publiques et supporte la recherche à base de plusieurs mots clés.

6.1.2.Les étapes du processus d'indexation

Le processus d'indexation se compose de plusieurs étapes que nous avons schématisées dans la figure 1.5 ci-dessous.

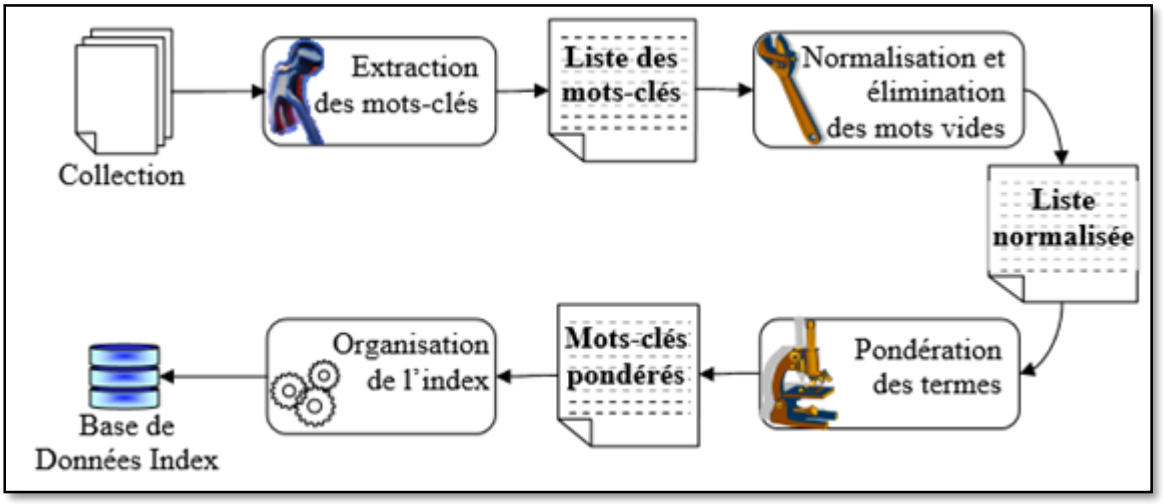


Figure 1.5 Etapes du processus d'indexation

En général, L'indexation consiste d'abord à extraire les termes du document n'apparaissant pas dans un anti-dictionnaire (stop-listes) et dans une certaine limite de fréquences. Les termes sélectionnés sont ensuite lemmatisés, ce qui permet une certaine normalisation des termes dans l'index [9].

❖ **L'extraction des mots-clés du document**

Appelée Tokenization en anglais, l'extraction des mots-clés est une étape qui peut sembler triviale au premier abord, et qui pourtant constituera la base de tout le reste du processus d'indexation. Il faut donc que cette phase soit d'une qualité maximale [66]. Certains systèmes de recherche d'information utilisent une liste de mots-clés prédéfinie. Cette liste est conçue manuellement et, dans la plupart des cas, construite par rapport à un thème spécifique. Cette méthode permet entre autres de contrôler la taille des index. L'utilisation d'une extraction automatique des mots-clés ou l'utilisation d'une liste de mots-clés prédéfinie, détermine le type d'indexation. Orientée document dans le premier cas et orientée requête dans le second [13].

❖ **La normalisation des mots-clés du document**

Ce traitement consiste à retrouver pour un mot sa forme normalisée (généralement le masculin pour les noms, l'infinitif pour les verbes, le masculin singulier pour les adjectifs, etc.). Ainsi, dans l'index ne sont conservées que les formes normalisées, ce qui offre un gain de place

appréciable, mais surtout, si le même traitement est effectué sur la requête, cela permet d'être beaucoup plus souple et rapide dans la recherche. Par exemple, si l'utilisateur effectue une recherche avec un verbe, les documents comportant ce verbe dans toutes ses formes conjuguées seront pris en compte, et pas seulement les documents contenant le verbe dans la forme fournie par l'utilisateur. Cette étape est également appelée « traitement morphologique des mots-clés » [34].

Cette phase peut également être enrichie avec un traitement syntaxique et sémantique des mots-clés. Le premier consiste à identifier et regrouper un ensemble de mots dont la signification dépend de leur union. Par exemple, les mots « maison blanche » ne signifient habituellement pas qu'on a affaire à une maison qui est blanche, mais plutôt au siège de la présidence des Etats-Unis. Elle consiste aussi à éliminer des ambiguïtés comme par exemple les problèmes d'homographie. Le traitement sémantique a pour but de faire des distinctions entres les différents sens possibles d'un même mot (polysémie). Par exemple, cette phase permet de différencier le mot « pièce » qui peut correspondre à une pièce de monnaie par exemple, ou à une pièce dans une maison. C'est une tâche ardue qui n'est pas aujourd'hui bien maîtrisée et dont l'intérêt, en termes de hausse des performances des systèmes, n'est pas toujours démontré.

- **L'élimination des mots vides**

Cette étape revêt une importance certaine dans la mesure où elle constitue un facteur d'une grande influence dans la précision de la recherche. Le fait de ne pas éliminer les mots vides provoque inévitablement du bruit. L'élimination des mots vides qui sont des mots du langage courant et qui ne contiennent pas beaucoup d'information sémantique doit se faire aussi bien à l'indexation qu'à l'interrogation (élimination des mots vides de la requête).

Notons que ce procédé n'est valide que si on considère une seule langue, alors que pour un moteur de recherche qui tient compte de toutes les langues possibles, cette opération mènerait à des ambiguïtés et des omissions. Par exemple, le terme "the" pour la langue anglo-saxonne alors que "the" qui remplace la boisson "thé" a un sens bien défini en langue française.

Certains moteurs de recherche ignorent des termes de longueurs inférieures à certaines longueurs, par exemple 2 ou 3, alors que certaines abréviations peuvent avoir un sens bien défini. Par exemple, "NY" est une abréviation de "NewYork". On conclue que cette étape de traitement est un peu difficile à éviter ou à appliquer.

❖ La pondération des mots-clés

Cette étape est entièrement dépendante du modèle de recherche d'information utilisé. Elle permet de définir l'importance qu'a un terme dans un document donné, elle est également utilisée pour filtrer l'index résultant du processus d'indexation (c'est-à-dire : éliminer les index dont le poids est inférieur à un certain seuil). Et de manière générale, la majorité des formules de pondération des termes est construite par combinaison de deux facteurs. Un facteur de pondération locale quantifiant la représentativité locale d'un terme dans le document, et un second facteur de pondération globale mesurant la représentativité globale du terme vis-à-vis de la collection des documents.

a) Pondération locale

La pondération locale permet de mesurer la représentativité locale d'un terme. Elle prend en compte les informations locales du terme par rapport à un document donné. Elle indique l'importance du terme dans ce document. Les fonctions de pondération locales les plus utilisées sont les suivantes [92] :

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

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

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

– **fonction normalisée :** permet de réduire les différences entre les valeurs associées aux termes du document, et elle est donnée comme suit [4]:

$$0.5 + 0.5 * \frac{tf_{i,j}}{max_{t_{i \in D_j}}\, tf_{i,j}} \tag{1.1}$$

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

Cette approche consiste à choisir les mots les plus représentatifs selon leur fréquence d'occurrences car généralement on admet qu'un mot qui apparaît souvent dans un texte représente un concept important. Cependant, avec cette démarche, les mots les plus fréquents sont les mots fonctionnels (mots vides) tels que "de", "un", "les", etc. Ainsi, il devient évident qu'on ne peut pas garder tous les mots les plus fréquents comme des index. Pour résoudre ce problème, la pondération globale a été introduite.

b) Pondération globale [4]

La pondération globale prend en compte des informations concernant un terme par rapport à la collection de documents. Elle indique la représentativité globale du terme dans l'ensemble des documents de la collection. Un poids plus important doit être donné aux termes qui apparaissent moins fréquemment dans la collection : les termes qui sont utilisés dans de nombreux documents sont moins utiles pour la discrimination que ceux qui apparaissent dans peu de documents. Le facteur de pondération globale qui dépend de la fréquence inverse dans le document a été introduit, comme le facteur IDF (pour Inverted Document Frequency) donné par plusieurs formules parmi lesquelles on peut citer:

$$Idf = \log(\frac{N}{n_i}) \quad (1.2)$$

$$Idf = \log(\frac{N - n_i}{N}) \quad (1.3)$$

$$Idf = \log(1 + \frac{N}{n_i}) \quad (1.4)$$

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

Dans cette approche on s'intéresse essentiellement à la distribution d'un terme dans le corpus, en négligeant la fréquence d'apparition d'un terme dans un document. Ainsi, on évite les termes fonctionnels (tels que "de", "et", etc.) qui sont très fréquents dans un document.

Le calcul des poids des termes se base aussi bien sur la pondération locale et globale (on parle alors de pondération TF-IDF). Dans ce cas, le terme choisi combine deux caractéristiques: il est important dans le document et il apparaît peu dans les autres documents.

Les pondérations locales et globales ne tiennent pas compte d'un aspect important du document : sa longueur. En général, les documents les plus longs auront tendance à utiliser les mêmes termes de façon répétée, ou à utiliser plus de termes pour décrire un sujet. Par conséquent, les fréquences des termes dans les documents seront plus élevées, et les similarités avec la requête seront également plus grandes. En effet, certaines mesures normalisent la formulation de la fonction de pondération en intégrant la taille des documents, ce qu'on appelle facteur de normalisation. Robertson et Sparck-Jones [86] proposent de normaliser la fonction de pondération dans **BM25** de la façon suivante :

$$w_{i,j} = \frac{tf_{i,j} \times (K_1 + 1) \times \log(\frac{N}{n_i})}{K_1 \times \left((1 - b) + b \times \frac{dl_j}{\Delta l}\right) + tf_{i,j}} \quad (1.5)$$

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, sa valeur optimale dépend de la longueur et de l'hétérogénéité des documents dans la collection de documents (dans TREC3 [87], $K_1 = 2$) ;

b est une constante appartenant à l'intervalle [0, 1] et contrôle l'effet de la longueur du document (dans TREC [87], elle est fixée à 0.75) ; dl_j est la longueur du document D_j, et Δl est la longueur moyenne des documents dans la collection entière.

Ces paramètres ont été maintes fois mis à jour selon les différentes versions de OKAPI [84]. Cet ajustement dépend de la collection testée, ce problème met que le résultat de la recherche est toujours instable.

6.1.3. 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. Plusieurs études ont été menées sur les structures de fichier utilisées dans la recherche d'information. Nous nous basons sur [82] pour présenter les plus importantes.

- **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.

- **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 d'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. Le tableau suivant présente les termes t_1, t_2, t_3 et leurs signatures.

Terme	Signature du terme
t_1	0101
t_2	1010
t_3	0011

Tableau 1. 2 Signature des termes

On considère à présent le documentd_1 contenant le terme t_1, le document d_2 contenant les termes t_1 et t_3 et le document d_3 contenant les termes t_1 et t_2. Le fichier de signature pour les trois documents est alors présenté dans le tableau suivant.

Document	Signature du document
d_1	0101
d_2	0111
d_3	1111

Tableau 1. 3 Signature des documents

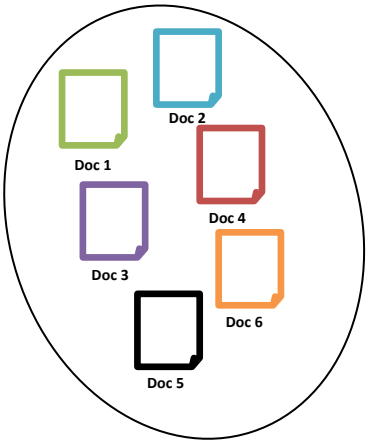
Une fois qu'un fichier signature est créé, la réponse à une requête consiste à calculer une chaîne pour la requête et à la comparer à celle calculée pour chaque document.

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.

❖ **Fichier inversé :**

La structure de fichier inversé 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 on inverse 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 d'informatiques actuels (commerciaux ou académiques).



Mot 1	3
Mot 2	5
Mot 3	1
. . .	
Mot k	4

Index

Mot 1	<Doc1, 3>, <Doc4, 8>, <Doc8, 1>
Mot 2	<Doc1, 1>, <Doc2, 3>, <Doc3, 5>, <Doc5, 3>, <Doc6, 6>
Mot 3	<Doc6, 3>
. . .	
Mot k	<Doc3, 4>, <Doc4, 2>, <Doc5, 1>, <Doc6, 2>

Fichier Inverse

Figure 1.6 Structure de fichier Inverse

Un fichier inversé 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 "posting liste" de la forme: $t_i \rightarrow (d_1, tf_{i1}), (d_2, tf_{i2}), \ldots, (d_j, tf_{ij})$, où d_j indique l'identificateur du document j et tf_{ij} indique la fréquence du terme i dans le document j [116].

Les fichiers inversés 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 utilisent un fichier inversé. 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.

6.2. Le processus d'interrogation

La phase d'interrogation est la phase d'interaction entre le système et l'utilisateur. Ce dernier exprime son besoin d'information via un langage de requête que le système va se charger de traduire. Cette traduction se fait selon le modèle de requête et a pour but de comprendre les besoins de l'utilisateur et de les exprimer dans un formalisme similaire à celui mis en œuvre lors de l'indexation des documents. Ce processus fournit une requête interne. Suite à cette phase de compréhension de la requête, un modèle de correspondance calcule la correspondance entre la requête interne et chaque index des documents. Ce calcul, établi par la fonction de correspondance, a classiquement pour résultat une liste ordonnée des documents de la base. La comparaison entre requête et document aboutit rarement à des équivalences strictes, mais plutôt à des équivalences partielles : le document correspond à une partie seulement de la requête. Le premier document de la liste renvoyée par le système est celui qui est considéré par le système comme le plus pertinent, c'est-à-dire celui qui répond le mieux à la requête, toujours d'après le système. Le dernier document est celui qui est considéré par le système comme le moins pertinent. Cette notion de pertinence repose sur la proximité entre les besoins exprimés par l'utilisateur et les résultats fournis par le système. On différencie alors la pertinence utilisateur de la pertinence système. (Dans la section 5.1 du chapitre III, nous donnerons notre propre formulation de la pertinence ainsi que deux nouvelles).

6.3. Reformulation de requête:

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

1. Comment peut-on retrouver plus de documents pertinents vis-à-vis d'une requête donnée?

2. Comment peut-on mieux exprimer la requête de l'utilisateur de manière à mieux répondre à son besoin?

La figure 1.7, présente les principales techniques d'amélioration des SRI par reformulation de la requête initiale en y ajoutant de nouveaux termes. La reformulation peut se faire par

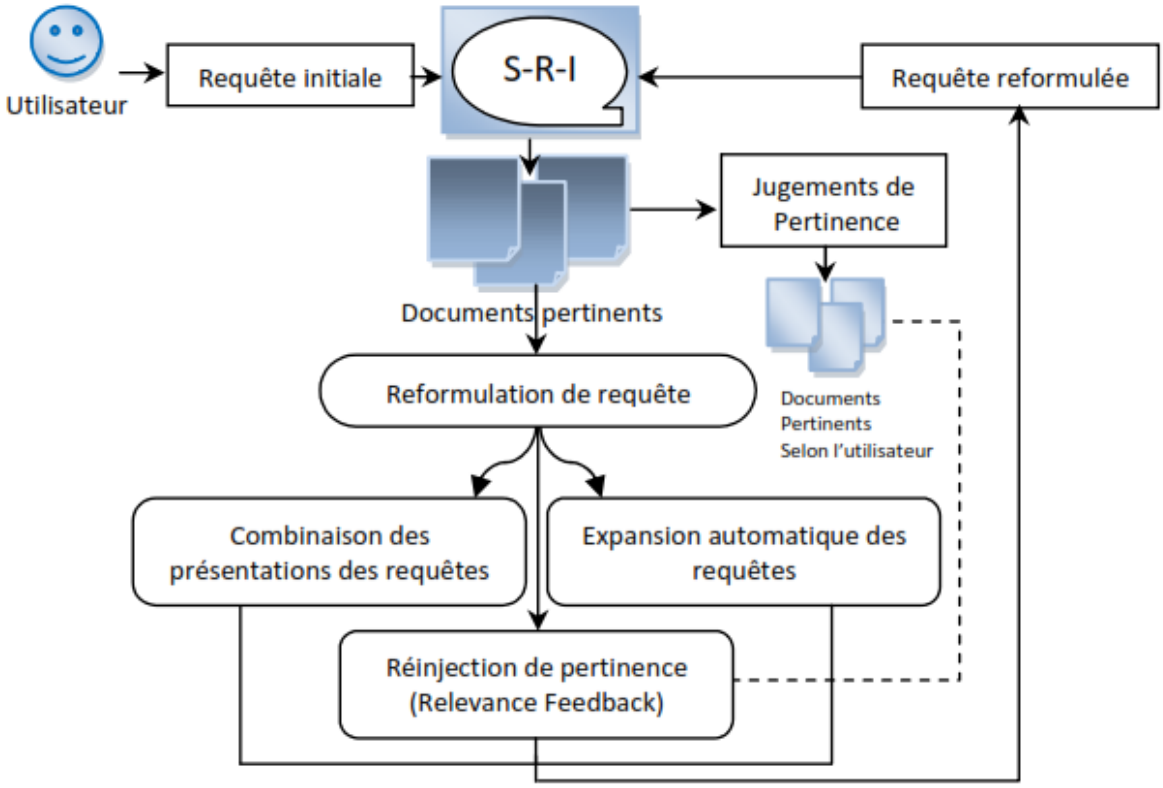


Figure 1.7Techniques d'amélioration des SRI par reformulation de requêtes [19].

expansion automatique de la requête, par combinaison de différentes présentations de la requête ou par réinjection de pertinence. Nous présentons dans ce qui suit ces trois principales techniques.

6.3.1. Expansion automatique des requêtes

[19] L'expansion directe de la requête consiste à rajouter à la requête initiale des termes issus de ressources linguistiques existantes ou bien de ressources construites à partir des collections. Plus précisément, un niveau des ressources linguistiques, le but est d'utiliser un vocabulaire contrôlé issu de ressources externes. On peut alors utiliser des ontologies linguistiques (citons par exemple Wordnet [67]). On peut également ajouter à la requête des variantes morphologiques des termes employés par l'utilisateur. Le but de ce mécanisme est d'assurer la restitution des documents indexés par des variantes des termes composant la requête.

Les associations établies manuellement traduisent généralement des relations de synonymie et de hiérarchie. Les thésaurus construits manuellement sont un moyen efficace pour l'expansion de requête. Cependant, leur construction et la maintenance des informations sémantiques qu'ils contiennent sont coûteuses en temps et nécessitent le recours à des experts des domaines considérés. Pour cette raison, ils restent peu utilisés par les SRI.

En ce qui concerne la seconde catégorie de ressources, elles sont construites en s'appuyant sur une analyse statistique des collections. Il s'agit de chercher des associations de termes afin d'ajouter des termes voisins à la requête. Il existe aussi d'autres méthodes entièrement automatiques telles que le calcul des liens contextuels entre termes [30] et la classification automatique de documents [29].

Les associations créées automatiquement sont généralement basées sur la cooccurrence des termes dans les documents. Les liens inter-termes renforcent la notion de pertinence des documents par rapport aux requêtes. Ce procédé est souvent appliqué par les moteurs de recherche.

6.3.2.Combinaison des présentations des requêtes

Plusieurs approches de recherche d'information [101] utilisent une seule représentation de requête comparée à plusieurs représentations de document.

Il a été montré dans [52] qu'une recherche plus efficace peut être atteinte en exploitant des représentations multiples de requêtes ou des algorithmes de recherche différents ou encore en utilisant différentes techniques de réinjection.

Une combinaison des représentations de requêtes peut augmenter le rappel d'une requête, tandis que la combinaison des algorithmes de recherche peut augmenter la précision. La base théorique de la combinaison des évidences a été présentée par Ingwersen [43]. Il a en particulier montré que des représentations multiples d'un même objet, par exemple une requête, permettent une meilleure perception de l'objet qu'une seule bonne représentation. Cependant, il est important que chacune des sources d'évidences utilisées fournisse non seulement un point de vue différent sur l'objet, mais que ces points de vue aient différentes bases cognitives. Les représentations multiples d'une requête peuvent donner différentes interprétations du besoin en information.

Une des approches de combinaison de multiples représentations de requêtes est proposée dans [10]. Elle consiste à calculer les scores des documents directement depuis la fonction d'appariement document-requête en utilisant le même système de recherche mais différentes versions de la requête. Ensuite, les résultats obtenus par chacune des versions sont combinés pour avoir une seule liste finale. Ces versions sont issues soit des expressions d'une même requête par des chercheurs différents, soit des présentations d'une même requête dans des langages différents.

Tamine et al, proposent dans [106] une technique de recherche d'information basée sur les algorithmes génétiques, plus précisément, elle propose d'utiliser une population de requêtes

qui évolue à chaque étape de la recherche et tente de récupérer le maximum de documents pertinents.

6.3.2.Réinjection de la pertinence

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

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

- L'extraction des évidences est l'étape la plus importante, elle consiste en général à extraire les termes pertinents qui serviront à l'enrichissement de la requête initiale. Plusieurs approches ont été développées. La plus reconnue est celle de Rocchio [88] adaptée au modèle vectoriel.

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

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

D'une manière générale, la phase d'échantillonnage ne présente pas de problématique spécifique. Le seul point abordé à ce niveau concerne le nombre d'éléments à évaluer pour pouvoir effectivement constituer un échantillon représentatif.

La problématique principale de la réinjection de pertinence réside dans les deux autres phases: l'extraction des termes (ils sont alors pondérés pour sélectionner les éléments les plus pertinents) et la réécriture de la requête avec repondération des termes.

Dans la plupart des approches de la littérature, les deux phases sont effectuées avec des méthodes de pondération des termes similaires. Cependant certaines méthodes, et particulièrement celles basées sur le modèle probabiliste, utilisent des méthodes de pondération différentes.

- La notion de pertinence

Pertinence est la notion centrale dans la Recherche d'Information car toutes les évaluations s'articulent autour de cette notion. Mais c'est aussi la notion la plus mal connue, malgré de

nombreuses études portant sur cette notion telles que celles figurant dans [33]. Voyons quelques définitions de la pertinence pour avoir une idée de la divergence.

La pertinence est:

1. la correspondance entre un document et une requête, une mesure d'informativité du document à la requête;

2. un degré de relation (chevauchement, relativité, …) entre le document et la requête;

3. un degré de la surprise qu'apporte un document, qui a un rapport avec le besoin de l'utilisateur;

4. une mesure d'utilité du document pour l'utilisateur; …etc.

Même dans ces définitions, les notions utilisées (informativité, relativité, surprise, …) restent très vagues et ceci, parce que les utilisateurs d'un SRI ont des besoins très variés. Ils ont aussi des critères très différents pour juger si un document est pertinent. Donc, la notion de pertinence est utilisée pour recouvrir un très vaste éventail de critères et de relations. nous appelons pertinence système l'ensemble des principes qui sous-tendent la fonction de correspondance dans un système de recherche d'information (donc le document est jugé pertinent par le système) [33], par opposition à la pertinence utilisateur, qui correspond à l'ensemble des jugements de pertinence que produit l'utilisateur qui utilise le système (le document est jugé pertinent par l'utilisateur). Voir section 7.1.

- Fonction de similarité

La comparaison entre le document et la requête revient à calculer un score, supposé représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction ou d'une probabilité de similarité notée *RSV(d,q)* (RetrievalStatus Value)[6], où Q est une requête et d un document et dont la formule de calcul dépend entièrement du modèle de recherche d'information utilisé. Cette mesure tient compte du poids des termes dans les documents, déterminé en fonction d'analyses statistiques et probabilistes. La fonction d'appariement est très étroitement liée aux opérations d'indexation et de pondération des termes de la requête et des documents du corpus. D'une façon générale, l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'information. La fonction de similarité permet ensuite d'ordonner les documents renvoyés à l'utilisateur. La qualité de cet ordonnancement est primordiale. En effet, l'utilisateur se contente généralement d'examiner les premiers documents renvoyés (les 10 ou 20 premiers). Si les documents recherchés ne sont pas présents dans cette tranche,

l'utilisateur considérera le SRI comme mauvais vis-à-vis de sa requête. Le but de tout SRI est donc évidemment de rapprocher la pertinence système de la pertinence utilisateur [95].

7. Les modèles théoriques de la recherche d'information

7.1. La notion de modèle de RI

Le modèle joue un rôle central dans la Recherche d'Information dans la mesure que c'est lui qui détermine le comportement clé d'un SRI. Il doit fournir une formalisation du processus de recherche d'information et accomplir plusieurs rôles dont le plus important est de fournir un cadre théorique pour la modélisation de la mesure de pertinence [95]. Il doit d'abord donner une signification au résultat de l'indexation. Un document est représenté par un ensemble de termes index et leurs poids. Si un terme index est censé représenter un concept important décrit dans un document, il existe différentes façons d'interpréter son poids. Un modèle théorique doit donner une interprétation précise à ce poids. Le modèle doit aussi interpréter les relations possibles entre les termes d'indexation. Ces deux fonctions amènent à la représentation d'un document. Une représentation similaire peut être créée pour une requête [68]. Finalement, un modèle de recherche d'information doit déterminer la relation entre un document et une requête à partir de leurs représentations. Ceci se fait souvent avec un calcul de similarité.

7.2. Classification des modèles de la RI

On retrouve dans la littérature plusieurs classifications des modèles de recherche d'information selon les besoins de l'auteur. La plus utilisée dans les ouvrages traitants la recherche d'information est celle mentionnée dans [39] qui classe les modèles de Recherche d'information en deux grandes catégories : les modèles sémantiques et les modèles statistiques. Les modèles sémantiques essayent de mettre en œuvre l'analyse syntaxique et sémantique; autrement dit, ils essayent de reproduire, à un certain point, la compréhension du texte en langage naturel qu'un utilisateur humain fournirait. Dans les modèles statistiques, les documents qui sont retrouvés sont ceux qui correspondent le plus étroitement à la question en termes de quelques mesures statistiques. On retrouve au niveau des modèles sémantiques tous les modèles traitant la reconnaissance de la syntaxe et de la sémantique du langage naturel dans lequel le texte du document est écrit. Et dans les modèles statistiques, on retrouve les modèles connus dans la recherche d'information classique tels que les modèles booléens, vectoriels ou probabilistes. Dans la suite de notre travail, on ne s'intéressera qu'à la deuxième catégorie, à savoir, les modèles statistiques, vu que les modèles sémantiques relèvent plus du domaine du Traitement Automatique du Langage Naturel. Les modèles statistiques sont

présentés dans [80] selon deux grandes catégories (Figure 1.8) qui sont les modèles classiques et les modèles structurés. Dans cette section on ne présentera que les modèles classiques.

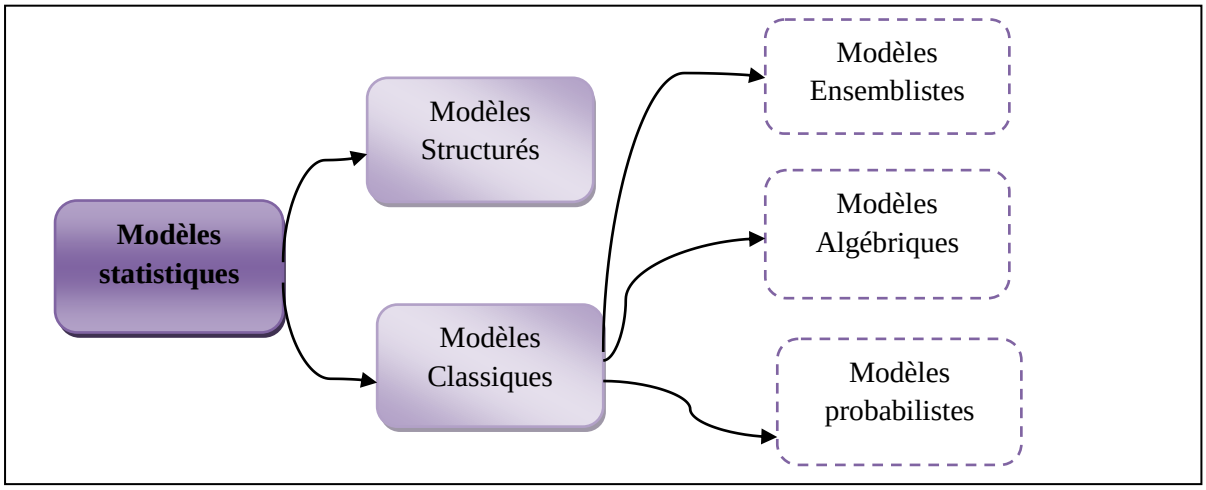


Figure 1.8Classement des modèles de RI

7.2.1.Les modèles classiques de la RI

Les modèles basés sur la théorie des ensembles

Cet ensemble de modèles se basent sur la théorie des ensembles, ainsi une requête est représentée par un ensemble de termes séparés par des opérateurs logiques (OR, AND, NOT). Le document est, quant à lui, représenté par une liste de mots-clés. Ces modèles permettent d'effectuer des opérations d'union, d'intersection et de différence lors de l'interrogation. Le modèle le plus connu et le plus simple de cette catégorie est le modèle booléen. On y retrouve également le modèle booléen étendu et le modèle flou [19].

- **Le modèle booléen**

Le modèle booléen est le plus simple et le plus répandu des modèles de la recherche d'information. C'est également le premier à s'imposer dans le domaine de la recherche d'information. Il s'appuie sur l'utilisation des opérateurs logiques manipulés grâce à l'algèbre de Boole. Il consiste à formuler une question avec une liste de termes séparés par des opérateurs logiques (ET, OU, NON), et à rechercher les documents correspondant à cette requête. Dans le modèle booléen simple, les poids $w_{i,j}$ sont tous égaux à 0 ou 1. Ainsi, un document est représenté par un ensemble de termes, ou encore par un vecteur booléen x, ou encore par une conjonction de termes, et une requête doit être représentée par une expression booléenne utilisant les termes. Un cadre utile, à cause en particulier des possibilités de généralisation, est la logique des propositions. *T* est l'ensemble des symboles propositionnels. Un descripteur est la conjonction des termes à coefficient 1 et des négations des termes à coefficient 0. Une requête est une expression quelconque, utilisant généralement des

conjonctions, des disjonctions et des négations (sur le même ensemble *T* de symboles propositionnels). Ainsi, les langages de description de documents et de requêtes sont des fragments de la logique des propositions sur T. On peut alors définir un document pertinent pour une requête q comme étant un document dont la description d implique, au sens logique, la requête ($d => q$). Une méthode de preuve en logique des propositions fournit donc un algorithme d'appariement: une réponse est un document qui implique logiquement la requête. Toutes les techniques de la logique des propositions sont alors utilisables. La similarité entre un document et une requête, est alors une fonction à valeurs 0 ou 1 qui peut être définie, en général, de la manière suivante ;

$$RSV(d,q) = \begin{cases} 1 & Si \quad d \ appartient \text{ à } l'ensemble\ décrit\ par\ q \\ 0 & Sinon \end{cases} \qquad (1.6)$$

Les différentes variantes de cette formule selon l'utilisation des opérateurs logiques sont les suivantes :

$RSV(d, q_1 \wedge q_2) = 1 \ SSI \ RSV\ (q_1, d) = \ RSV\ (q_2, d) = 1$
$RSV(d, q_1 \vee q_2) = 0 \ SSI \ RSV\ (q_1, d) = \ RSV\ (q_2, d) = 0$
$RSV(d, \neg q) = 1 \ SSI \ RSV\ (q, d) = \ 0$

Les différentes variantes de cette formule selon l'utilisation des opérateurs logiques sont les suivantes :

- La conjonction peut être interprétée comme une intersection
- La disjonction comme une union.
- La négation comme une différence.

Donc on peut implémenter ce modèle à partir d'une programmation du type de données «ensemble». Les principaux avantages du modèle booléen sont, d'une part, son formalisme précis, et que les langages de description de documents et de requêtes sont des parties d'un même langage, la logique des propositions, et que celle-ci est bien étudiée sous tous ses aspects. Et d'autre part, sa transparence. En effet, la raison pour laquelle un document a été sélectionné par le système est claire pour l'utilisateur : elle répond exactement à sa requête. Le modèle booléen a ses inconvénients : le critère de décision est binaire, un document est soit pertinent soit non pertinent (la similarité d'un document et d'une requête est 1 ou 0), il n'y aucune possibilité de classement. Un document ne contenant pas tous les termes de la requête ne sera pas retourné. Ainsi ce modèle est plus un modèle de données qu'un modèle de recherche d'information.

❖ **Les modèles algébriques**

Les modèles algébriques regroupent tous les modèles de RI qui utilisent une représentation vectorielle des documents et des requêtes [76] dans lesquels, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. La similarité est calculée algébriquement en se basant sur la représentation du document et de la requête. Le représentant le plus connu de cette catégorie est le modèle vectoriel. Des modèles plus complexes utilisant le formalisme des réseaux de neurones pour garantir une certaine interaction avec l'utilisateur, ont également été utilisés dans la recherche d'information [32]. Nous nous contentons, à ce niveau, de présenter brièvement le modèle vectoriel.

❖ **Le modèle vectoriel**

Les modèles vectoriels ont été créés pour compenser la limitation de la pondération binaire des modèles booléens. Ils sont basés sur l'attribution de poids non binaires aux termes indexés dans les documents et aux termes des requêtes. Plus de détails sont donnés dans le chapitre suivant.

8. Evaluation des performances des SRI

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

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

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

Ces deux objectifs sont évalués par les mesures de précision et de rappel définis ci-dessous.

8.1. Les mesures de Précision/Rappel

Les mesures de précision/rappel sont obtenues en partitionnant l'ensemble des documents restitués par le SRI en deux catégories : les documents pertinents et les documents non pertinents [58] [23]. Ces deux catégories se définissent comme suit :

❖ Taux de précision : La précision mesure la capacité du système de rejeter tous les documents non pertinents à une requête. Il est donné par le rapport entre l'ensemble des documents sélectionnés pertinents et l'ensemble des documents sélectionnés.

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

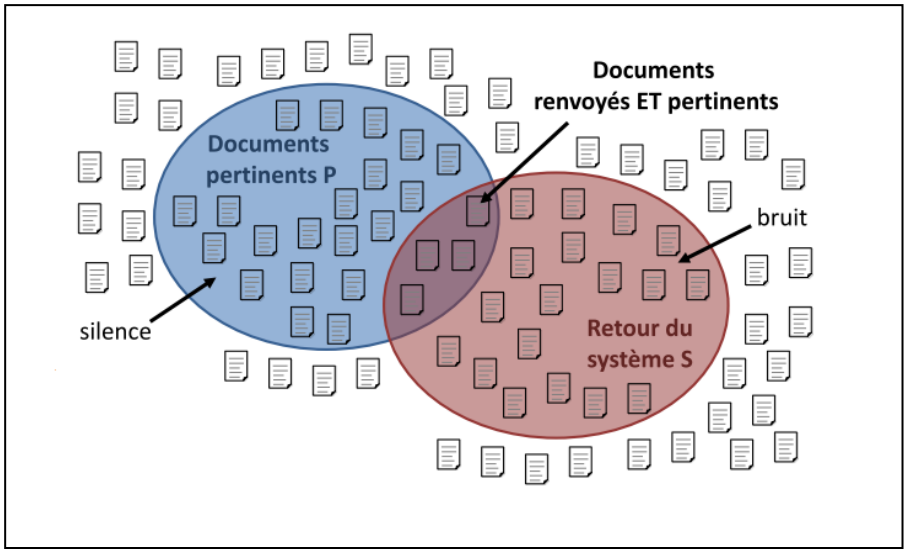


Figure 1.9Mesures d'évaluation d'un SRI

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

Soient P l'ensemble de documents pertinents, S l'ensemble des documents que retourne le système.

$$\text{Rappel} = \frac{|P \cap S|}{|P|} \quad ; \quad \text{Silence = 1-rappel}$$

$$\text{Précision} = \frac{|P \cap S|}{|S|} \quad ; \quad \text{Bruit = 1- précision}$$

Le silence signifie l'existence des documents pertinents qui ne sont pas retournés par le SRI, alors que le bruit signifie l'existence des documents non pertinents qui sont retournés par le SRI. La figure 1.9 donne un schéma représentant ces concepts.

8.2. Collections de tests

Les collections de tests ont été traditionnellement utilisées en recherche d'information pour évaluer les stratégies de recherche. Plus particulièrement, une collection de référence doit traduire la subjectivité de pertinence des utilisateurs. D'autre part, elle doit contenir une masse d'information assez importante et variée pour constituer un environnement standard d'interrogation.

Différentes collections de test sont utilisées en recherche d'information. Parmi elles nous citons :

- les collections TREC[84] [87]
- les collections Amaryllis.

8.2.1. Les collections TREC

Le projet TREC est un programme international initie au début des années 90 par le NIST (National Institute of Standards and Technology) et du DARPA (Defense Advanced Reserach Projet Agency). Ce programme offre des moyens homogènes d'évaluation des systèmes de recherche d'information. Il est devenu la référence en recherche d'information pour diverses raisons, il a permis de définir les tâches en recherche d'information et de construire de larges collections de test.

Dans ce qui suit, nous allons définir les différents éléments qui constituent le projet TREC. Parmi ces éléments : les tâches, les participants, la source d'information et enfin la structure et le principe de construction de la collection TREC.

- **Tâches** : Différentes taches de RI sont définies chaque année dans TREC. Depuis des tâches spécifiques ont été progressivement introduites (TREC en1996). L'objectif est de permettre l'évaluation d'approches spécifiques en recherche d'information concernant le filtrage, le croisement de langues, la recherche dans de très large corpus (100 giga octets et plus), les modèles d'interactions, etc.
- **Les participants** : 25 groupes ont participé à la première édition de TREC en 1992 et 66 groupes de 16 pays différents ont également participé à TREC8 [84].
- **Source d'information** : Les documents de la collection sont issus de la presse écrite en 1999 (Financial Time, Résumes de publication USDOE, SAN jose Mercury news, etc.). 5 CD-ROM contenant environ 5 giga octets de données textuelles et 450 besoins d'information (requêtes) sont déjà disponibles.
- **Structure d'un document TREC**

  Les documents TREC sont organisés en fichiers ; chaque fichier contient un ensemble de documents. Un document TREC est généralement présenté sous le format SGML. Il est identifié par un numéro et décrit par un auteur, une date de production et un contenu textuel. La figure 1.10 donne un exemple de document TREC.

```
<doc>
<docno> WSJ880406- </docno>
<h1> AT&T Unveils Services to Upgrade Phone
Networks Under
Global Plan </h1>
<author> Janet Guyon (WSJ Staff) </author>
<dateline> New York </dateline>
<text> American Telephone & Telegraph Co.
Introduced the first
of a new Generation of phone services with
broad...
</text>
</top>
```

Figure 1.10 Exemple d'un document TREC

- **Structure d'une requête TREC**

Une requête TREC est également identifiée par un numéro. Elle est décrite par un sujet générique, une description brève et une description étendue sur les caractéristiques des documents pertinents associés à la requête. La figure 1.11 suivante présente à titre d'exemple une requête TREC.

```
<top>
<num> Number : 168</num>
<title> Topic: Financing AMTRAK<title>
<desc> description :A document will address the rôle of the
Federal Government in financing the,operation of the national
Railroad Transportation Corporation(AMTRAK)
</desc>
<narr> Narrative: A relevant document must provide information
on the
government'sresponsability to make AMTRAK an economically
viable
entity. It could also discuss the privatization of AMTRAK as an
alternative
to continuing government subsidies. Documents comparing
governemnt
subsidies given to air and bus transportation with those
provided to
AmTRAK
</narr>
</top>
```

Figure 1.11 La structure d'une requête TREC

Le processus de construction d'une collection TREC consiste à constituer un groupe d'assesseurs de pertinence. Chacun d'eux gère un ensemble d'une moyenne de 10 sujets de requêtes. Il détermine aussi les documents pertinents associes dans la collection. On sélectionne finalement 50 sujets de requêtes sur la base du nombre de documents pertinents estimés.

8.2.2. Les collections Amaryllis

Amaryllis est la version française du projet TREC. Il a pour objectif principal d'évaluer des logiciels de recherche d'information dans des corpus de texte en français. Des collections de textes sont également fournies dans ce cadre.

Amaryllis a suivi la méthodologie TREC pour le cycle exploratoire. Cependant les volumes de documents et le nombre de termes étaient inférieurs.

9. Conclusion

Ce premier chapitre a porté essentiellement sur l'étude des SRI de manière générale. Nous avons présenté l'architecture commune à tous les systèmes de recherche d'information ; notamment l'appariement document/requête et la reformulation des requêtes ; puis nous avons présentés les étapes essentielles d'une bonne indexation.

Dans le chapitre suivant, nous présenterons les différents modèles de RI tout en insistant sur le fonctionnement des systèmes CORI [27] et CS[2]ainsi que d'autres systèmes de recherche d'information liés à notre problématique.

CHAPITRE II

Les systèmes et les modèles de Recherche d'Information

Sommaire

1. Introduction :

Pour répondre à un besoin d'information, un humain peut s'adresser à des personnes de son entourage ou rechercher l'information sur des supports physiques dans les bibliothèques, les journaux, livres, revues, etc.

Dans le cadre d'un système de recherche d'information, la recherche d'information (RI) est basée sur quatre entités, à savoir, l'homme, le besoin d'information, l'information et l'outil d'acquisition d'information. Le besoin d'information est exprimé par l'homme sous forme d'une requête. Suivant le développement technologique, l'apparition des ordinateurs et de l'Internet, les outils développés dans le domaine de la RI ont aussi évolué. Cette évolution touche la manière d'indexer, de stocker et de rechercher l'information contenue dans les documents. Un document est un objet qui véhicule des informations pouvant prendre plusieurs formes notamment texte, son, images, vidéo, etc.

Nous nous intéressons exclusivement à la RI textuelle où les documents sont sous forme de texte (ou du moins leur représentation), et pour lesquels l'outil servant à effectuer la recherche est un système informatique. Un système de recherche d'information n'est pas un système Question/Réponse: il n'est pas sensé répondre explicitement à une requête, mais simplement sur l'existence (ou non) et la localisation de documents ayant rapport avec sa demande. Un SRI est donc un outil informatique qui permet à l'utilisateur d'exprimer son besoin d'information à l'aide d'une requête et qui retrouve les documents pertinents à cette requête parmi l'ensemble des documents qu'il gère, ensemble appelé corpus du système. Très souvent, les systèmes de recherche d'information retournent des listes de liens vers des documents. Cette liste est triée selon le degré de pertinence (calculé par le SRI), appelé aussi score, de chacun des documents qu'elle contient.

Les problèmes dans le domaine de la RI sont de deux sortes:

- des problèmes liés à l'utilisateur : sa capacité à déterminer ses besoins d'information et sa capacité à exprimer ceux-ci par une requête. La conséquence est que cette dernière n'est qu'une description partielle de son besoin d'information ;

- des problèmes liés au système : essentiellement sa capacité à identifier les documents pertinents à la requête, c'est le problème de la sélection de collections de documents

pertinents, le problème auquel on s'intéresse dans cette thèse. Car une mauvaise sélection au préalable mène à une pénalisation de certaines collections jugées pertinentes et, par la suite, un nombre très important de documents pertinents ne sera pas recommandé à l'utilisateur.

En effet, on peut dire qu'un système de recherche d'information n'a que trois notions principales : en premier lieu le **document**, en deuxième lieu **la requête;** ces deux premiers éléments constituent les données. Et, en troisième lieu, le processus de traitement qui établit la relation entre les deux premiers éléments : selon la requête de l'utilisateur, le processus sélectionne les collections contenant les **documents** pertinents.

Le processus de la sélection de collections se base réellement sur plusieurs points selon la méthode utilisée ; mais le point primordial est la satisfaction de l'appariement totale ou partielle.

L'appariement: l'appariement consiste à associer pour une requête donnée la liste des documents qui lui sont pertinents. Deux paradigmes peuvent se distinguer :

- **appariement exact** où un document est jugé pertinent s'il vérifie tous les critères spécifiés dans la requête [14] et ce jugement est binaire.

- **appariement avec classement** où un degré de pertinence est attribué à chaque document en fonction de sa similarité sémantique avec la requête. Les documents sont alors ordonnés avant d'être présentés à 1'utilisateur.

Une fonction d'appariement (correspondance):une fois l'index construit, l'appariement entre la requête et l'index peut désormais s'effectuer. L'appariement consiste à détecter les collections pertinentes pour la requête posée, et éventuellement calculer le degré de cette pertinence appelé aussi score. Ce score sert à trier la liste des collections retournées. Dans notre thèse, on propose une nouvelle fonction de scoring qui sera détaillée dans le prochain chapitre.

2. Types des Systèmes de recherche d'information (SRIs)

Les systèmes de recherche d'information se divisent en deux grandes catégories selon l'architecture sous-jacente. On distingue alors, les systèmes de recherche d'information centralisés, où une entité centrale s'occupe de toutes les tâches liées à la collecte, d'indexation, de sauvegarde et de traitement des requêtes, et les systèmes de recherche d'information décentralisés, où un ensemble d'entités du même ordre se partagent les

différentes fonctionnalités de système. Dans la suite de cette section, nous présentons en détails ces différents systèmes.

2.1 Système de recherche d'information centralisée (SRIC)

Un SRIC fonctionne d'une manière totalement centralisée. Sa particularité est l'unicité de son index [2]. Tous les documents de son corpus sont indexés dans le même fichier ou la même structure qui sert de base pour la recherche.

La particularité d'un SRIC est l'unicité de son index, comme il est illustré sur la figure 2.1. Bien que la recherche d'information soit exhaustive, c'est-à-dire, si l'information est stockée alors elle sera certainement localisé, le système centralisé est toutefois non extensible et souffre du problème de passage à l'échelle. Ceci est en réalité son majeur inconvénient outre le fait que la recherche n'est pas assez exhaustive suite à la dynamique de ce type de système.

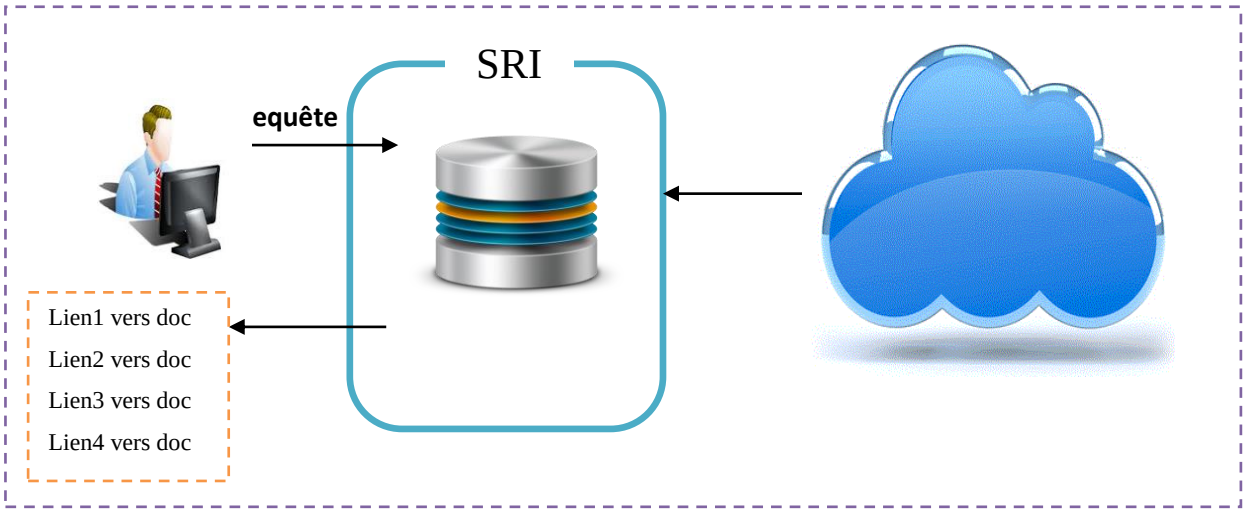


Figure 2. 1 Système de recherche d'information centralisé

En effet, vu la quantité phénoménale des données sur Internet, et face au rythme de croissance de ces données, il est impensable d'employer un index unique pour toute cette quantité de données à cause des problèmes suivants :

1. la puissance et les capacités de sauvegarde n'évoluent pas avec la même vitesse que les données sur Internet ;

2. il est impossible de se procurer tous les documents existants. En effet, il existe sur Internet des documents inaccessibles comme par exemple:

- ❖ les documents protégés par mot de passe ;

- les documents des sites qui interdisent 1'indexation de leurs documents par le biais du fichier robots.txt ;
- les documents de la partie cachée d'Internet, correspondant souvent à des documents générés dynamiquement comme, par exemple, les horaires de trains ou les cours boursiers ;
- ou tout simplement, les documents qui ne sont pas publiés sur le Web tels que les fichiers sur les disques dur, etc.

3. il est très coûteux de construire un index pour la totalité des documents collectés, car il est nécessaire de les charger localement afin de les indexer. Or, ce chargement ralentit le fonctionnement du réseau et les serveurs à partir desquels le chargement s'effectue;

4. la mise à jour de l'index unique n'est pas facile, et cette tâche peut durer longtemps. Or, certains sites, comme, par exemple les sites d'information, mettent leurs documents à jour quotidiennement ;

5. l'unicité de la méthode d'indexation malgré l'hétérogénéité des sources des documents est inadéquate. En effet, des documents rédigés dans des langues différentes sont indexés de la même façon.

2.2 Les Systèmes de recherche d'information distribués (SRID) :

A la différence des systèmes centralisés, un ensemble de serveurs sont attachés à un courtier (broker) qui leur achemine la requête selon la vue qu'il a sur chacun. En recevant la requête, ces serveurs la traitent localement et retournent leurs résultats au courtier. La figure 2.2 suivante donne un schéma global qui illustre l'architecture globale de ce type de système. Dans la suite, nous donnons une terminologie importante. Cette terminologie permet de mettre en clair certaines notions importantes et sert d'explication de ce que nous présenterons dans le chapitre suivant.

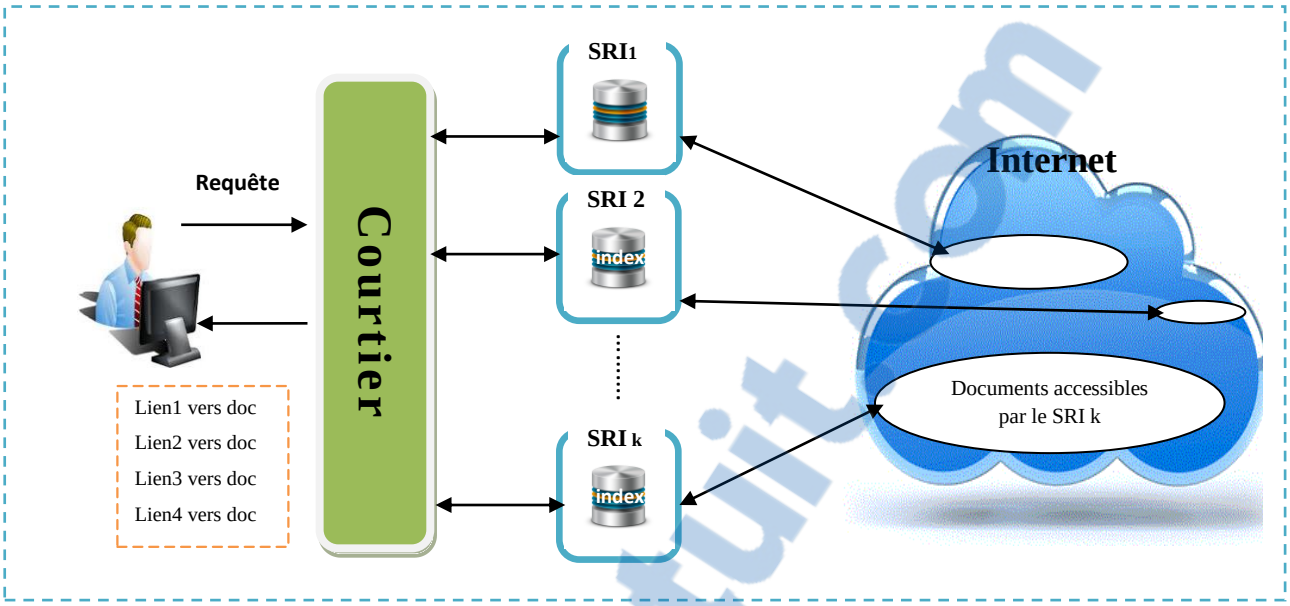


Figure 2. 2 Système de recherche d'information distribué (SRID)

2.2.1 Terminologie :

Définissons d'abord les termes qui vont nous servir à présenter le domaine de la recherche d'information distribuée RID.

1. **Une source d'informations** est un ensemble de documents mis à disposition par un particulier ou une organisation désirant les publier [2].

2. **Une collection** est un ensemble de documents appartenant à une ou plusieurs sources d'information.

3. **Une liste de résultats** est une liste triée de références vers des documents qui sont jugés pertinents par le système à une requête donnée. Ces listes peuvent avoir des syntaxes très différentes et peuvent contenir des informations variées telles que :

- les degrés de pertinence des documents. Un degré de pertinence peut être une valeur numérique ou une valeur symbolique telle que des graphiques ou des étoiles etc.
- les passages qui contiennent le plus de termes communs avec la requête;
- les résumés des documents ;
- etc.

4. **Un serveur** est une machine qui offre à un utilisateur le service de rechercher de l'information dans une collection qu'il héberge. Un serveur contient un SRI local qui effectue cette tâche. L'utilisateur peut interroger directement le serveur ou par le biais d'une autre machine en passant par un réseau.

5. **Un courtier** est un module qui joue le rôle d'intermédiaire entre l'utilisateur et les différentes sources d'information. Le courtier agit comme un pseudo-moteur de recherche qui reçoit en entrée une requête et fournit en sortie une liste de résultats.
6. **Un représentant** de serveur est un ensemble d'informations détenues par le courtier et décrivant certaines caractéristiques du serveur comme par exemple les termes apparaissant dans la collection du serveur, leurs fréquences d'apparition, ainsi que des informations utiles à son interrogation (sa localisation, son coût, ses droits d'accès, etc.).
7. **L'utilité d'un serveur** à une requête se traduit par sa capacité à contenir et retourner des documents pertinents à cette requête.

2.2.2 Fonctionnement d'un SRID :

Un système de recherche d'information distribué (SRID) [35] a le même but qu'un SRIC, à savoir, satisfaire le besoin d'information de l'utilisateur. Couvrir au maximum la masse d'information existante sur Internet est un premier pas vers ce but. En effet, la performance d'un SRI dépend de deux critères. Le premier est sa couverture de l'information disponible sur Internet tant dit que le deuxième critère concerne le Modèle de Recherche d'Information(MRI) utilisé.

Effectivement, un grand corpus riche en documents pertinents pour une requête donnée est inexploitable si son MRI affiche des lacunes dans la quête de ces documents. De plus, un MRI efficace est incapable de désigner des documents pertinents préalablement inexistants dans le corpus qu'il a indexé. Les SRIDs ont été conçus pour assurer une grande couverture de l'Internet et pour pallier les problèmes rencontrés par les SRICs. Un SRID est constitué d'un courtier qui communique avec un ensemble de serveurs comme il est illustré dans la figure 2.2. chaque serveur correspond à un SRI. Chaque serveur contient donc un index et un moteur de recherche qui exploite cet index.

Un SRID est un ensemble de serveurs qui coopèrent afin d'atteindre un but commun, celui de répondre au mieux aux besoins d'information des utilisateurs. L'idéal est que le fonctionnement interne de cet ensemble de serveurs reste transparent à l'utilisateur qui garde l'impression de travailler avec un seul serveur, c'est à dire le courtier.
Le courtier est le cœur d'un SRID, il contient cinq composants logiciels ; un module de gestion, un module frontal (une interface utilisateur), un module de sélection de serveurs (collections), un module de communication et un module de fusion de résultats. Le tableau Table 2.1 donne les composantes du module courtier.

Composants logiciels	Rôle
Un Module de gestion	Construit les représentants des serveurs du système dont il connait les spécificités.
Un Module Frontal	Permettre à l'utilisateur de soumettre sa requête.
Un Module de Sélection	Choisir les collections qui sont susceptibles de répondre avec des documents pertinents.
Un Module de communication	Adapter la requête au langage d'interrogation de chaque collection sélectionnée, acheminer la requête et réceptionner la réponse.
Un Module de fusion	Grouper les réponses reçues par le module de communication et rendre via l'interface utilisateur une liste unique de résultats à l'utilisateur.

Tableau 2. 1 Le Rôle du courtier

La figure 2.3 présente le fonctionnement global du module et courtier ainsi que les différentes interactions entre ses composantes. Elle montre aussi l'interaction entre le module communication et les différentes entités maitresses dans le système distribué global.

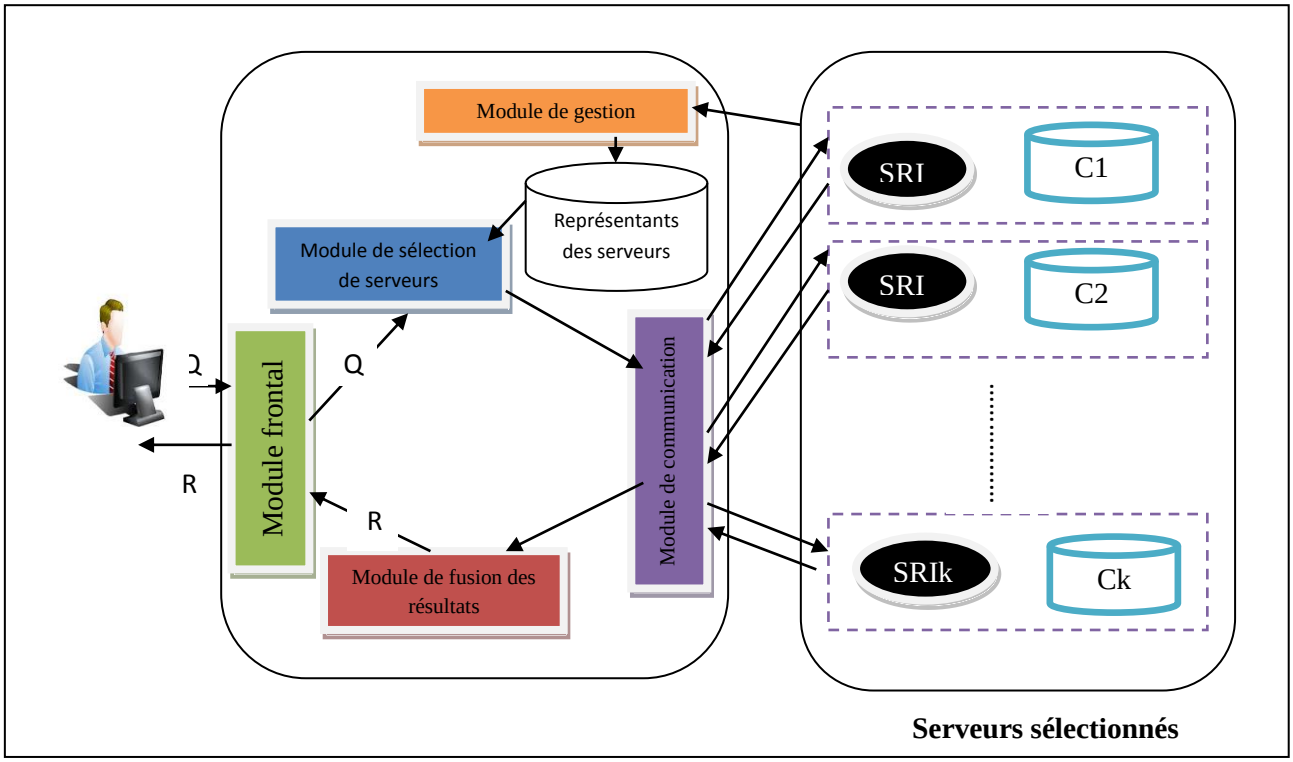


Figure 2. 3Les modules fonctionnels d'un SRID

2.3 Les SRIDs: une solution pour les problèmes des SRICs :

Les SRIDs, en distribuant le processus d'indexation et de recherche, apportent des solutions aux problèmes posés par les SRICs [2]:

1. La couverture maximale de l'information disponible sur Internet n'est plus une tâche qui incombe à un seul serveur. La couverture obtenue est l'union des couvertures des serveurs du SRID. Un SRID permet également d'exploiter efficacement les ressources matérielles puisque la tâche de stockage est distribuée sur un ensemble de serveurs.
2. Une source de documents ne souhaitant pas être indexée par d'autres serveurs distants, peut installer son propre SRI et s'insérer dans le SRID.
De cette façon, ses documents sont visibles sans qu'ils ne soient pour autant indexés par des serveurs "étrangers". C'est le cas par exemple desserveurs payants, qui souhaitent être référencés sans que le contenu de leurs documents ne soit consultable.
3. Les serveurs adhérant au SRID, sont responsables de la construction et de la mise à jour de leur corpus. La décentralisation de ces deux tâches les rend moins coûteuses.
4. Chaque serveur étant responsable de l'indexation de sa collection, il est libre de choisir la méthode d'indexation appropriée à son corpus (ou collection), par exemple en fonction de la langue.

2.4 Les principales difficultés dans la RID

Il existe un certain nombre de problèmes concernant les SRIDs. Nous en citons quelques-uns. Le premier est lié à la stratégie commerciale des serveurs et les autres sont d'ordre technique.

2.4.1. La coopération passive des serveurs

Par des serveurs qui coopèrent d'une façon passive, nous entendons des serveurs qui n'émettent pas d'objection à ce qu'un courtier les interroge et récupère leurs résultats.
Les administrateurs des serveurs peuvent ne pas approuver le fait qu'un courtier, considéré comme un concurrent, contourne leurs propres résultats. C'est le cas des serveurs Web (moteurs de recherche) qui retournent des résultats incluant des bordereaux publicitaires dont la consultation est plus bénéfique pour eux que la consultation des documents retournés.

De point de vue réseautique, ce problème s'appelle aussi déni de service. Les serveurs questionnés expriment un refus catégorique à la coopération de peur que cette coopération nuise à leur fonctionnement.

2.4.2. La coopération active des serveurs

La coopération active des serveurs signifie que ces derniers acceptent de fournir au courtier des informations qui sont nécessaires pour son fonctionnement.

Ces informations peuvent être, par exemple, la technique d'indexation, la fonction de similarité, l'ensemble des termes qui apparaissent dans la collection, leurs fréquences d'apparition, etc.

La plupart des méthodes proposées dans la littérature supposent la disponibilité, au niveau du courtier, de certaines informations sur le contenu des collections. Or, pour obtenir ces informations, la coopération active des serveurs est très souvent nécessaire, et elle n'est pas systématique pour des raisons de confidentialité ou pour des raisons commerciales.

2.4.3. L'interopérabilité

Le courtier est opérable avec différents serveurs qui sont autonomes et hétérogènes. L'une des implications qui en découle est la nécessité pour le courtier de pouvoir communiquer et fonctionner conjointement avec les serveurs. Or, ces derniers peuvent changer (langage, présentation, ajout/suppression d'information). Un intergiciel (ou middleware) est nécessaire dans ce cas.

2.4.4. La volatilité des documents [2]

Un document peut voir son contenu évoluer voire être supprimé. Les index locaux des serveurs sont basés sur le contenu des documents à un moment donné. Si certains documents, ont subi une éventuelle modification ou ont été supprimés, les moteurs locaux pourront retourner des documents qui ne s'accordent pas du tout à une requête ou qui n'existent plus. En outre, et à supposer que le moteur local ait mis à jour son index, le courtier a, pour sa part également, des informations concernant les collections à mettre à jour. Et dans ce cas, seule une coopération parfaite peut résoudre ce problème en informant le courtier des changements qui se sont produits.

L'un des problèmes relatifs à la volatilité des documents est illustré par le cas suivant : un document indexé par plusieurs serveurs subit une modification de son contenu ; certains serveurs ont mis à jour leurs index depuis la modification, d'autres non. Le courtier reçoit des scores différents pour le document modifié, même si tous les serveurs utilisent des techniques d'indexation identiques, de calcul de scores, etc. Le courtier dans ce cas n'est pas en mesure de déterminer le score de la version la plus récente du document.

3. Les modèles de la recherche d'information

Le modèle joue un rôle central dans la RI. C'est lui qui détermine le comportement clé d'un SRI. Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit les deux rôles suivants:

- Créer une représentation interne pour un document ou pour une requête basée sur ces termes.
-Définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance.

On distingue trois grandes catégories de modèles :

- ❖ le modèle booléen,
- ❖ le modèle vectoriel,
- ❖ le modèle probabiliste.

Nous étudions les deux premiers modèles et leurs variantes. Celles-ci sont les plus utilisés dans les SRI actuels. Nous enchaînons ensuite par la présentation du modèle probabiliste. Une brève actualisation de chaque modèle est aussi présentée.

Pour mieux comprendre le principe de chaque modèle on va utiliser un corpus exemple.

Corpus exemple :

Soit un corpus composé de quatre documents, d_1, d_2, d_3 et d_4. Leurs contenus se présentent comme suit :

Document : d_1

Ce livre est consacré à l'étude des **bases de données** et plus particulièrement à l'étude des **bases de données** relationnelles et des **bases de données XML**.

Document : d_2

Ce livre est consacré à la gestion des **bases de données XML**.

Document : d_3

Ce livre est consacré à l'étude des **réseaux** informatiques : **réseaux** locaux (LAN) et **réseaux** étendus (WAN) ainsi qu'à l'interconnexion des **bases de données** par des **réseaux**.

Document : *d4*

Ce livre est consacré à la technologie **XML** : édition de documents **XML**, transformation de documents **XML** et interrogation de documents **XML**.

Termes d'indexation :

Terme : bd

si le document parle de bases de données.

Terme : réseau

si le document parle des réseaux informatiques.

Terme : xml

si le document parle de la manipulation de documents XML.

3.1. Les modèles ensemblistes

3.1.1. Le modèle booléen strict

Le modèle booléen est le premier modèle utilisé en recherche d'information. Il est basé sur la théorie des ensembles et sur l'algèbre de Boole. Dans ce modèle, une requête est représentée sous forme d'une équation logique. Les termes sont reliés par des connecteurs logiques ET, OU et NON [94]. Le moteur de recherche retrouve les documents qui correspondent exactement à la requête, compte tenu de la présence ou de l'absence des termes de celle-ci dans la représentation des documents et de l'expression booléenne. Une requête booléenne représente une définition exacte d'un ensemble de documents.

Par exemple, la requête ***'réseaux'*** définit tout simplement les documents indexés avec le terme réseaux. En utilisant les opérateurs de Boole, les requêtes et les ensembles de documents correspondants peuvent être combinés pour former de nouveaux ensembles de documents.

Combiner deux termes avec l'opérateur ***'ET'*** définit un ensemble de documents plus petit que les ensembles de documents indexés par l'un de ces termes, par exemple, la requête ***'bd et XML'*** produit les documents indexés à la fois par les termes bd et xml. Combiner les termes avec l'opérateur ***'OU'*** définit un ensemble de documents plus grand que les ensembles de documents indexés par l'un de ces termes. A titre d'exemple, la requête ***'bd et xml'*** produit les documents indexés par le terme *bd* ou le terme *xml*.

Pour tout terme $t \in T$ et pour tout document $d \in D$, on a :

« *d* est indexé par *t* » est vrai ou faux.

Un document est représenté par l'ensemble des termes qui l'indexent.

Documents du corpus Exemple :

d_1 = {bd, xml}

d_2 = {bd, xml}

d_3 = {bd, réseau}

d_4 = {xml}

Ainsi, la similarité entre un document *d* et une requête *q* est définie par :

1) $sim : D \times Q \rightarrow \{0, 1\}$
2) $sim(d, t) = 1$ si $t \in d$, 0 sinon
3) $sim(d, q_1 \wedge q_2) = sim(d, q_1) \wedge sim(d, q_2)$
4) $sim(d, q_1 \vee q_2) = sim(d, q_1) \vee sim(d, q_2)$
5) $sim(d, \neg q) = \neg sim(d, q)$
6) $sim(d, (q)) = sim(d, q)$

Exemple de similarité :

$d_3 = \{bd, réseau\}$

$q = (bd \vee xml) \wedge \neg réseau$

$sim(d_3, q)$

$= sim(d_3, (bd \vee xml) \wedge \neg réseau)$

$= sim(d_3, bd \vee xml) \wedge sim(d_3, \neg réseau)$

$= (sim(d_3, bd) \vee sim(d_3, xml)) \wedge \neg sim(d_3, réseau)$

$= (1 \vee 1) \wedge \neg 1$

$= 0$

- Le document d_3 ne répond pas à la requête *q*.

La réponse à une requête est l'ensemble des documents qui sont similaires à cette requête.

1) $rep(q) = \{d \in D \mid sim(d, q) = vrai\}$
2) $rep(t) = \{d \in D \mid t \in d\}$
3) $rep(q_1 \wedge q_2) = rep(q_1) \cap rep(q_2)$
4) $rep(q_1 \vee q_2) = rep(q_1) \cup rep(q_2)$
5) $rep(\neg q) = D - rep(q)$
6) $rep((q)) = rep(q)$

Application:

$q = (bd \vee xml) \wedge \neg réseau$

$rep(bd) = \{d_1, d_2, d_3\}$

$rep(réseau) = \{d_2\}$

$rep(xml) = \{d_1, d_2, d_4\}$

$rep(q) =$

$= (rep(bd) \cup rep(xml)) \cap (D - rep(réseau))$

$= (\{d_1, d_2, d_3\} \cup \{d_1, d_2, d_4\}) \cap (\{d_1, d_2, d_3, d_4\} - \{d_3\})$

$= \{d_1, d_2, d_4\}$

Fichier inverse :

Pour améliorer les performances on peut pré-construire pour chaque terme $t \in T$, l'ensemble *rep(t)* qui est appelé liste inverse.

L'ensemble de ces listes inverses constitue le fichier inverse du corpus.

Si *card(D) = n*, une liste inverse peut être représentée par un vecteur de *n* bits, dont le $i^{éme}$bit est égal à 1 si le document d_i est indexé par *t* et 0 sinon.

Selon notre corpus test, la liste inverse sera comme le montre le tableau 2.2 suivant :

	D1	***D2***	***D3***	***D4***
bd	1	1	1	0
Réseau	0	0	1	0
xml	1	1	0	1
(bd ∨xml) ∧ ¬réseau	1	1	0	1

Tableau 2. 2Fichier inverse

Analyse du modèle :

Le modèle booléen présente le principal avantage de simplicité de mise en œuvre comme il se prête à une implantation performante avec les fichiers inverses. Il présente cependant les principaux inconvénients suivants :

– Les formules de requêtes sont complexes, non accessibles à un large public,

– La réponse du système dépend de l'ordre de traitement des opérateurs de la requête,

– La fonction d'ordre des documents n'est pas une fonction d'ordre (total ou partiel),

– Les modèles de représentation des requêtes et des documents ne sont pas uniformes.

Ceci rend le modèle inadapté à une recherche progressive.

Une extension de ce modèle a été effectuée par Fox et Salton [91].

Il s'agit du modèle booléen étendu. Ce dernier complète le modèle de base en intégrant des poids dans l'expression de la requête et des documents. Ceci induit la sélection de documents sur la base d'un appariement rapproche (fonction d'ordre) et non exact.

3.1.2. Le modèle booléen étendu

Le modèle booléen étendu est introduit en 1983 par Salton, Fox et Wu [91]. L'idée est de permettre l'utilisation des opérateurs logiques tout en proposant une pertinence graduée. Ce

modèle peut être vu comme une combinaison des modèles booléen et vectoriel. Au lieu d'estimer le poids d'un terme par son absence ou présence, la pondération des termes dans un document est basée sur le *tf-idf* normalisé c'est-à-dire que le poids d'un terme dans un document se trouve entre 0 et 1. Ce modèle consiste à calculer la distance entre les coordonnées d'un document et les coordonnées d'une requête.

Le modèle booléen étendu considère que les opérations booléennes ont une influence sur la façon dont il faut entreprendre la requête. La représentation d'un document contrairement au modèle booléen basique, tient compte des poids des termes. Chaque document est représenté par un vecteur de termes pondérés. Selon l'opérateur booléen utilisé, le modèle booléen étendu calcule le score d'un document selon deux cas principaux :

Requête de type disjonction :

Soient k_a et k_b deux termes. Une requête de type disjonction regroupant ces deux termes est de la forme : k_a ou k_b. Si on considère un document d dont les poids respectifs de k_a et k_b sont w_a et w_b alors le score de document d par rapport à la requête $k_a \vee k_b$ est estimé selon sa distance par rapport au point d'origine (0, 0) et est donné par l'équation 2.1 suivante:

$$score(k_a \, ou \, k_b, d) = \sqrt{\frac{score^2(k_a,\ d) + \ score^2(k_b, d)}{2}} = \sqrt{\frac{w_a^2 + w_b^2}{2}} \quad (2.1)$$

La figure 2.4 suivante montre que le document d_2 est plus pertinent que le document d_1 malgré qu'ils contiennent les termes $k_a$et k_b(les poids de chacun sont différents). Les quarts de cercles concentriques représentent les documents ayant le même score.

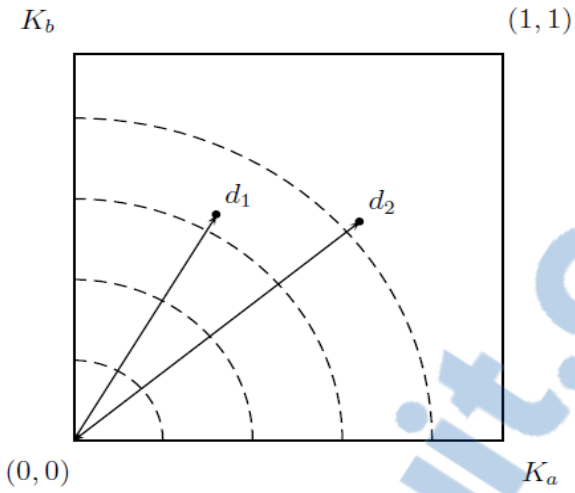


Figure 2. 4modèle booléen étendu : mesure de similarité entre un document et une requête de type ou

Requête de type Conjonction :

Une requête de conjonction sur les termes k_a et k_b est exprimée par : k_a et k_b. Si on considère un document *d* dont les poids respectifs de k_a et k_b sont w_a et w_b alors le score de *d* par rapport à la requête $k_a \wedge k_b$ est estimé selon sa distance par rapport au point (1, 1). La figure 2.5 montre que le document *d1* est plus pertinent que le document *d2* malgré qu'ils contiennent les termes k_a et k_b (les poids de chacun sont différents). Les quarts de cercles concentriques représentent les documents ayant le même score.

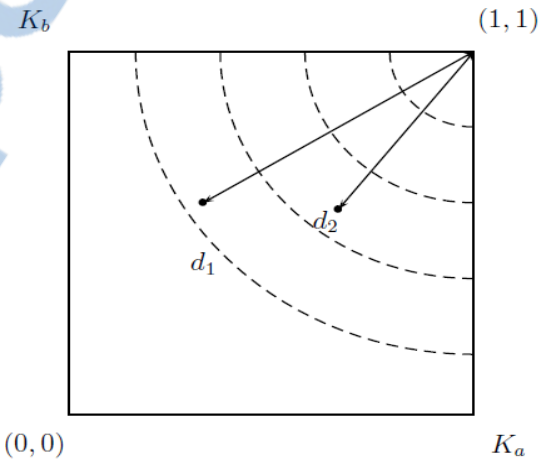


Figure 2. 5modèle booléen étendu : Mesure de similarité entre un document et une requête de type et

Le point (1, 1) représente la situation où les deux termes k_a et k_b sont présents dans le document. La mesure de similarité de cette requête par rapport à un document est donnée par l'équation 2.2 suivante:

$$score(k_a et\ k_b, d) = 1 - \sqrt{\frac{(1 - score(k_a, d))^2 + (1 - score(k_b, d))^2}{2}} \quad (2.2)$$

$$= 1 - \sqrt{\frac{(1 - w_a)^2 + (1 - w_b)^2}{2}}$$

Requête de type négation :

Une requête négation est de la forme : non k_a. Le score d'un document est estimé par l'équation 2.3 suivante:

$$score(non\ k_a, d) = 1 - score(k_a, d) \quad (2.3)$$

3.1.3.Le modèle flou

La représentation des documents et des requêtes reflète partiellement les contenus sémantiques des documents et des requêtes [16] [17]. Par conséquent la correspondance d'un document par rapport à une requête est approximative ou vague. Cette fonction

$\mu_A: U \rightarrow [0, 1]$ d'approximation est un sous-ensemble d'un univers de discours U caractérisée par la fonction qui associe à chaque élément u de U la valeur$\mu_A\ (u) \in [0, 1]$

Le degré d'appartenance est utilisé pour représenter l'incertitude ou l'ambiguïté [114]. Les trois opérations les plus couramment effectuées sur des ensembles flous sont :

$$\begin{aligned} \mu(a\ ou\ b) &= \max(\mu_a, \mu_b) \\ \mu(a\ et\ b) &= \min(\mu_a, \mu_b) \\ \mu(non\ a) &= 1 - \mu(a) \end{aligned} \quad (2.4)$$

Un ensemble flou est un ensemble dont les éléments sont affectés d'un degré d'appartenance : un réel dans l'intervalle [0, 1].

Notation

$E = \{\mu_1/x_1, ..., \mu_n/x_n\}$

$\mu_E: \{x_1, ..., x_n\} \rightarrow [0, 1]$ = fonction d'appartenance à E

$\mu_E(x_i) = \mu_i$= degré d'appartenance de l'élément x_i à l'ensemble E

Intersection :

$E \cap F = \{x \mid x \in E \wedge x \in F\}$ (stricte)

$E \cap F = \{\mu/x \mid \mu = min(\mu_E(x), \mu_F(x))\}$ (floue)

Union :

$E \cup F = \{\mu/x \mid \mu = max(\mu_E(x), \mu_F(x))\}$

Différence :

$E - F = \{\mu/x \mid \mu = min(\mu_E(x), 1 - \mu_F(x))\}$

Principe :

Pour tout terme $t \in T$ et pour tout document $d \in D$, on a :

d est indexé par t avec un degré de pertinence $\in [0, 1]$ qui traduit le poids du terme t dans le contenu de d. Ce poids peut être estimé par l'équation 2.5suivante :

$$poids(t, d) = \frac{fréquence\ de\ t\ dans\ d}{fréquence\ maximum\ d'un\ terme\ de\ d} \qquad (2.5)$$

Un document est représenté par l'ensemble flou de ses termes.

Documents du corpus Exemple :

$d_1 = \{1/bd, 0.33/xml\}$

$d_2 = \{1/bd, 1/xml\}$

$d_3 = \{0.25/bd, 1/réseau\}$

$d_4 = \{1/xml\}$

Similarité

$sim : D \times Q \rightarrow [0, 1]$

$sim(d, t) = poids(t, d)$

$sim(d, q_1 \wedge q_2) = min(sim(d, q_1), sim(d, q_2))$

$sim(d, q_1 \vee q_2) = max(sim(d, q_1), sim(d, q_2))$

$sim(d, \neg q) = 1 - sim(d, q)$

$sim(d, (q)) = sim(d, q)$

$d_1 = \{1/bd, 0.33/xml\}$

q = (bd ∨ réseau) ∧ ¬xml

$sim(d_1, q) = min(max(poids(bd, d_1), poids(réseau, d_1)), 1 - poids(xml, d_1)) = 0.67$

- Le document d_1 répond à la requête q avec un degré de pertinence de 0.67.

La réponse à une requête est l'ensemble flou des documents affectés chacun d'un degré d'appartenance égal à leur similarité avec la requête.

$rep(q) = \{sim(d, q)/d \mid d \in D\}$

Par exemple :

$rep((bd \vee réseau) \wedge \neg xml)$

$= \{0.67/d_1, 1/d_3\}$

d_1 qui parle un peu de xml (0.33) répond donc beaucoup (0.67) à la requête ¬xml

Comparaison entre booléens strict et flou :

Documents	
Strict	**flou**
d_1 = {bd, xml} d_2 = {bd, xml} d_3 = {réseau} d_4 = {xml}	d_1 = {1/bd, 0.33/xml} d_2 = {1/bd, 1/xml} d_3 = {0.25/bd, 1/réseau} d_4= {1/xml}

Requêtes			
bd ∧xml		**bd ∨xml**	
strict	**flou**	**strict**	**flou**
$\{d_1, d_2\}$	$\{0.33/d_1, 1/d_2\}$	$\{d_1, d_2, d_3, d_4\}$	$\{1/d_1, 1/d_2, 0.25/d_3, 1/d_4\}$

Tableau 2. 3Comparaison entre le modèle booléen strict et flou avec exemple.

Documents	
strict	**flou**
d_1= {bd, xml} d_2= {bd, xml} d_3 = {réseau} d_4= {xml}	d_1= {1/bd, 0.33/xml} d_2 = {1/bd, 1/xml} d_3= {0.25/bd, 1/réseau} d_4= {1/xml}
Requête	
(bd ∨réseau) ∧ ¬xml	
strict	**flou**
{d_3}	{0.67/d_1, 1/d_3}

Tableau 2. 4Comparaison entre le modèle booléen strict et flou avec exemple.

Jusqu'à nos jours, ce modèle suscite des intentions de la communauté des chercheurs. Vu que les bases de données ou collections de documents sont énormes, il devient inadéquat, voire impossible, de traverser toutes les bases de données pour répondre aux requêtes. On a vu alors la naissance d'algorithmes dits à arrêt au plutôt (early termination). Ce type d'algorithmes tente de localiser les meilleures réponses selon un mécanisme booléen combiné avec une fonction de scoring probabiliste. Sans doute l'algorithme WAND [22] est le plus populaire ces dernières années. Cet algorithme applique deux phases de calcul. Dans la première, il définit un seuil, qui varie au cours de l'exécution et initialisé à zéro. WAND définit aussi un prédicat propositionnel en fonction d'une certaine fonction du *tf-idf* qui donne une mesure de similarité approximative entre le document et le terme actuel de la requête. Si le prédicat est vérifié alors l'algorithme passe à la deuxième phase, où un calcul exact du score du document est calculé.

Bien que cet algorithme soit efficace en termes de réponses, il souffre du problème de l'effectivité. Dans le cadre de l'extension de ce modèle, [77] définit une nouvelle appelée *p*-norm (issue de la norme de Hölder) pour étendre le modèle booléen. Plus encore, les auteurs ont établi une nouvelle limite pour borner l'indépendance des termes.

3.2.Le modèle algébrique et ses dérivés

[11] Les modèles algébriques proposent une représentation vectorielle pour du document et de la requête. La mise en correspondance entre le document et la requête consiste à calculer la similarité entre les vecteurs représentant les documents et les requêtes.

Le modèle vectoriel est l'ancêtre de tous les modèles algébriques. Les premiers travaux de Salton [93] avaient pour finalité de concevoir la fonction d'appariement selon les propriétés et les opérations associées au concept d'espace vectoriel. Bien que simpliste, ce modèle reste le plus utilisé et le plus efficace.

3.2.1. Le modèle vectoriel basique

Le modèle vectoriel (nommé aussi VSM pour Vector Space Model), a été popularisé par Salton en 1971 [93] [90]. Ce modèle propose de représenter les documents et les requêtes par des vecteurs d'indexation dans un espace engendré par les termes d'indexation. Le modèle vectoriel représente les requêtes et les documents sous forme de vecteurs dans un même espace vectoriel. La mesure de similarité entre le document représenté par un vecteur $\vec{d}$= *(d_1, d_2 . . . d_n)*, où d_i est le poids d'un terme *i* dans le document *d*, et la requête définie par $\vec{q}$= *(q_1, q_2 . . . q_n)* où q_i est le poids (souvent 0 ou 1 selon que le terme appartient ou pas à la requête) du terme *i* dans la requête *q*, est estimée selon les propriétés et les mesures populaires issues de la théorie des espaces vectoriels. Ils existent plusieurs mesures pour calculer la similarité entre le document et la requête, la plus simple est le produit scalaire (équation 2.6) :

$$RSV(\vec{d},\vec{q}) = \sum_{k=1}^{n} d_k \times q_k \qquad (2.6)$$

Si les composantes des deux vecteurs sont binaires (1 si le terme existe dans le document, 0 si non) alors la mesure de similarité entre le document et la requête est égale au nombre de mots partagés entre eux. Le produit scalaire est très sensible à la norme des vecteurs ; document et requête (leurs longueurs). D'autres mesures ont été proposées; elles sont tout de même basées sur le produit scalaire. La mesure du cosinus, qui mesure le cosinus de l'angle formé par le document et la requête, est utilisé dans le modèle vectoriel: plus l'angle est petit, plus la requête est proche du document et par conséquent plus le cosinus de l'angle est élevé. La mesure cosinus est donnée par l'équation 2.7 suivante:

$$RSV(\overrightarrow{d},\overrightarrow{q}) = \cos(\vec{d},\vec{q}) = \frac{\vec{d} \times \vec{q}}{|\vec{d}| \times |\vec{q}|} = \frac{\sum_{k=1}^{n} d_k \times q_k}{\sqrt{\sum_{k=1}^{n} d_k^2 . \sum_{k=1}^{n} q_k^2}} \qquad (2.7)$$

La figure 2.6 illustre le cosinus de l'angle formé par une requête et deux documents d_1 et d_2. Le document d_2 est différent de la requête, le cosinus de l'angle résultant est égal à $cos(\theta) < 1$.

Le document d_1 est, contrairement à d_2, très proche de la requête, son RSV *est égal à 1 = cos(0).*

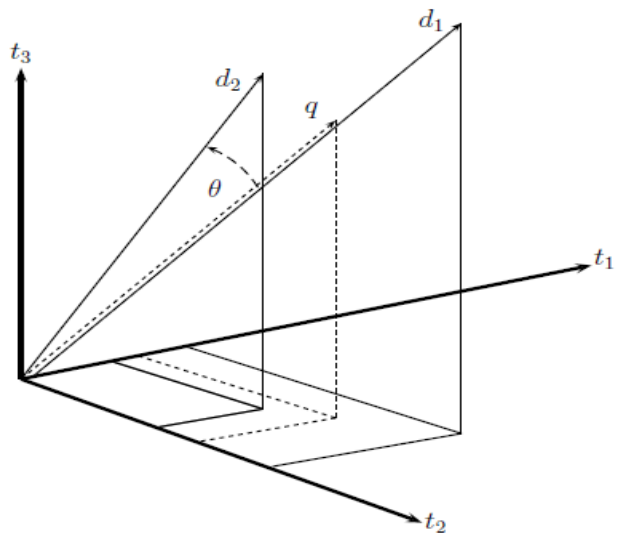


Figure 2. 6 La représentation selon le modèle vectoriel

D'autres mesures [81] sont utilisées pour percevoir le score d'un vecteur document par rapport à un vecteur requête. Parmi ces mesures, on rencontre le coefficient de similarité de Jaccard, donné par l'équation 2.8. Il exprime l'intersection entre une requête donnée et un document. Plus cette intersection est élevée plus elle est significative.

$$Jaccard: RSV(d,q) = \frac{\sum_{k=1}^{n} d_k . q_k}{\sum_{k=1}^{n} d_k^2 + \sum_{k=1}^{n} q_k^2 - \sum_{k=1}^{n} d_k . q_k} \quad (2.8)$$

Le coefficient de *Dice* permet de donner plus de valeur (deux fois) à l'intersection entre le document et la requête, comme le montre l'équation 2.9 suivante.

$$Dice: RSV(d,q) = \frac{2 \times \sum_{k=1}^{n} d_k . q_k}{\sum_{k=1}^{n} d_k^2 + \sum_{k=1}^{n} q_k^2} \quad (2.9)$$

Le coefficient de chevauchement *overlap*, donné par l'équation 2.10, donne beaucoup d'importance aux cas où q fait partie de d. Si q appartient à d alors ce coefficient vaut 1.

$$Overlap: RSV(d,q) = \frac{\sum_{k=1}^{n} d_k . q_k}{\min(\sum_{k=1}^{n} d_k^2, \sum_{k=1}^{n} q_k^2)} \quad (2.10)$$

Toutes ces mesures ont l'avantage de profiter des propriétés de l'espace vectoriel pour la perception de l'appariement utilisateur. Le principal intérêt de leur application est leur habilité à retourner des listes ordonnées de documents.

L'inconvénient majeur du modèle vectoriel est le fait qu'il suppose que les termes d'indexation forment une base. Or ils existent énormément de relations sémantiques qui font qu'un terme pourra s'exprimer en fonction des autres. Par ailleurs, il est très difficile voire impossible de traduire des relations par des combinaisons linéaires de termes, or ceci s'avère indispensable à la construction de vraie base de termes d'indexation.

3.3.Le modèle probabiliste et ses dérivés :

Le modèle probabiliste est basé sur la théorie de la décision. Le but est de calculer la probabilité de la pertinence d'un document *d* par rapport à une requête *q*.

3.3.1.Le modèle probabiliste de base :

Maron et Kuhn [59] étaient à l' origine de l'apparition du modèle probabiliste. Ce modèle utilise un modèle mathématique fondé sur la théorie des probabilités. La similarité entre un document et une requête est mesurée par le rapport entre la probabilité qu'un document *d* donné soit pertinent pour une requête *Q*, notée $p(d/Q)$ et la probabilité qu'il soit non pertinent, cette dernière probabilité est notée $p(\bar{d}/Q)$. Ces probabilités sont estimées par les probabilités conditionnelles selon qu'un terme de la requête est présent, dans un document pertinent ou dans un document non pertinent.

Soient, *P*(*terme_présent/pertinent*) la probabilité qu'un terme de la requête apparaisse dans un document sachant que celui-ci est pertinent et *P*(*terme présent/non pertinent*) la probabilité que ce terme apparaisse dans un document non pertinent.

Cette mesure est souvent donnée par la formulation 2.11suivante :

$$RSV(Q,d) = \frac{p(d/Q)}{p(\bar{d}/Q)} = \sum_{i=1}^{t} \log \frac{p(1-q)}{q(1-p)} \qquad (2.11)$$

Où:

p: est *P* (terme $t_i$présent/ *d* pertinent),
q: est *P* (terme $t_i$présent/ *d* non pertinent),
t: nombre total de termes dans la requête.

L'idée générale de cette approche est d'ordonner l'ensemble des documents de la collection selon leurs probabilités de pertinence. Robertson [85] appelle ce critère "principe de classement par probabilités".

Selon Robertson, deux hypothèses doivent être vérifiées pour garantir l'optimalité d'ordonnancement d'une collection de documents :

– La pertinence des documents doit être une variable aléatoire binaire prenant l'une des valeurs vrai ou faux,

– La pertinence d'un document ne doit pas influencer la pertinence d'un autre document.

Afin de définir des modèles non paramétrés tels qu'il est défini dans les modèles basés langage, [5] introduit de nouveaux modèles probabilistes qui mettent en œuvre de nouvelles normalisations du *tf* en mesurant la divergence entre la distribution en cours de la fréquence d'un terme et celle obtenue suivant un processus aléatoire.

Les auteurs étudient alors la loi binomiale et les statistiques de Bose-Einstein. Les auteurs définissent deux types de normalisation de la fréquence d'un terme. La première définition suppose que les documents ont la même longueur alors que la deuxième définition tient compte de la longueur du document en question ainsi que d'autres statistiques.

Le modèle probabiliste de Robertson définit dans sa formule la longueur des documents, les fréquences des termes ainsi qu'un ensemble de paramètres expérimentaux. Le problème du tf normalisé suscite beaucoup d'attention car les documents longs sont souvent pénalisés. Dans ce contexte de nouvelles tentatives et de nouveaux modèles sont actuellement proposés afin d'atténuer à ce problème.

[73], en étendant [72], se base sur la normalisation du la fréquence d'un terme dans un ensemble de documents. [73] introduit un nouveau modèle probabiliste qui tient compte de la distribution de la valeur maximale du poids d'un terme sur l'ensemble de documents qui le contiennent. Pour se faire, il combine deux nouvelles [84] normalisations du *tf*, à savoir, *ritf* (pour relative intra-document frequency d'un terme *t* dans le document *d*) et *lrt* (length normalized frequencyde *t* dans *d*).

Ces deux notions manipulent la moyenne des fréquences du terme *t* dans le document *d*, la moyenne des longueurs des documents dans la collection et la longueur du document en question. La fonction de scoring, définie dans ce travail, combine *rith* et *lrt* suivant un paramètre expérimental α. Les expérimentations comparées à BM25 montrent que ce nouveau modèle excelle de loin BM25 ;

3.3.2 Le modèle bayésien

Les réseaux bayésiens [75] [108] sont des graphes orientés acycliques dans lesquels les nœuds représentent des variables aléatoires, et les liens sont des dépendances entre ces variables.

On distingue deux principaux types de réseaux, les réseaux d'inférence et les réseaux de croyance.

3.3.2.1 Les réseaux bayésiens d'inférence : En attachant des probabilités initiales aux racines du graphe, on calcule de proche en proche le degré de croyance associé aux autres nœuds du graphe.

Les réseaux inférentiels bayésiens associent des variables aléatoires aux termes d'indexation, aux documents et aux requêtes de l'utilisateur.

Une variable aléatoire associée à un document représente l'évènement d'observer ce document. Les arcs sont dirigés du nœud document vers les nœuds termes, l'observation d'un document est alors conditionnée par l'augmentation de la valeur des variables associées aux termes d'indexation [18] [21]. Ceux-ci conditionnent à leur tour l'observation de la requête. Ainsi, les arcs sont orientés des nœuds associés aux termes d'indexation vers le nœud de la requête. La figure 2.7 illustre un réseau inférentiel bayésien simple de pertinence d'un document à une requête de trois termes.

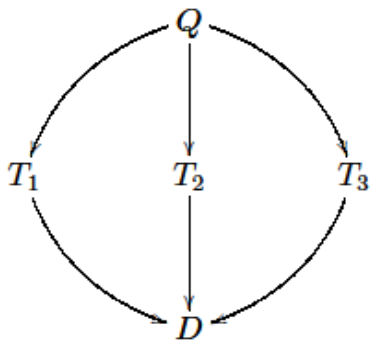


Figure 2. 7Modèle de réseau inférentiel

L'évènement la requête est accomplie (Q = 1) est réalisé si le sujet lié à un terme est vrai (T1 = 1, T2 = 1 ou T3 = 1), ou une combinaison de ces évènements.

Les trois sujets sont inférés par l'évènement le document est pertinent (D = 1).

Par l'enchaînement de règles de probabilités, la probabilité jointe des autres nœuds du graphe est :

$$P(D, T_1, T_2, T_3, Q) = P(D) \times P(T_1 \backslash D) \times P(T_2 \backslash D, T_1) \times P(T_3 \backslash D, T_1, T_2) \times P(Q \backslash D, T_1, T_2, T_3) \quad (2.12)$$

Le système Inquery [26] est l'un des modèles qui implémentent un réseau inférentiel bayésien. Il est utilisé pour formuler des requêtes simples basées sur des mots-clés, des requêtes booléennes ou ensemble. Il propose des opérateurs de moyenne et de moyenne pondérée, des opérateurs booléens probabilistes ou stricts, des opérateurs de proximité et de synonymie. Une procédure de prétraitement de la requête permet de générer une forme inférentielle prête à être évaluée.

3.3.2.2 Les réseaux bayésiens de croyance : Les réseaux bayésiens de croyance (belief network), sont introduits par Ribeiro-Neto et Muntz [79]. Ils sont fondés sur une étude épistémologique (Interprète la probabilité comme étant un degré de croyance dont les spécifications viennent des expérimentations statistiques) pour l'interprétation des probabilités. Mais ils s'écartent de l'inférence réseau par l'adoption d'un modèle bien défini, les réseaux de croyance [79] sont une généralisation des réseaux d'inférence.

4. Conclusion

Nous avons présenté dans ce chapitre une vue détaillée des systèmes de recherche d'information (SRI) de même que leurs différents modèles respectifs selon l'importance de chacun dans les SRIs.

Le chapitre suivant constitue le noyau de notre étude. Nous aborderons les motivations qui nous ont incité à travailler dans ce domaine puis nous déterminerons les objectifs à atteindre. Cette partie de thèse souligne l'aspect innovateur de la sélection de collections. Nous énumérerons les méthodes existantes. Ainsi que notre contribution sera dévoilée en détail.

CHAPITRE III

Contribution à la sélection de collections

Sommaire

1. Introduction :

Le but majeur de la sélection de collections est de réduire le nombre de serveurs interrogés tout en gardant, au maximum, la performance qu'on aurait obtenue si on interrogeait tous les serveurs. Dans ce domaine précis de recherche, on confond la sélection de collections à la sélection de serveurs, dans la mesure où on considère qu'un serveur maintient une seule collection. Le procédé de sélection réduit la charge inutile de messages qui pourraient circuler dans le système. La question devient donc à considérer un mécanisme efficace pour choisir les meilleurs serveurs pour une requête donnée. Par la suite, des règles équitables de classement des réponses reçues doivent être implémentées.

Dans ce chapitre, nous passons en revue les méthodes proposées dans la littérature, dans un premier temps, et nous dressons notre contribution détaillée dans ce domaine dans le second. Tout d'abord, nous revenons sur la définition de la notion de pertinence et sur le procédé de la sélection de collections.

2. Notion de pertinence[65]

La pertinence est une notion fondamentale et cruciale dans la recherche d'information. Elle tire cette importance des différents systèmes qui l'ont implémentée différemment ; jusqu'alors de nouvelles recherches se font proposée afin de satisfaire au mieux l'utilisateur. En réalité, c'est autour d'elles que toutes les évaluations s'articulent. Elle reste ainsi, très difficile à cerner, malgré les nombreuses études portant sur cette notion (voir Chapitre I, section 3.7).

La pertinence porte sur deux volets. Le premier est sur la partie système où elle est une notion précise et déterministe. Un algorithme décide de la pertinence d'un document pour une requête par un ensemble de calculs généralement fondés sur les termes contenus dans cette requête et dans un document. Dans ce cas, la pertinence est binaire ou est donnée par une valeur dans R ou [0, 1].

L'utilisateur est le deuxième volet sur lequel porte la notion de pertinence. Elle est, cette fois-ci une notion vague, complexe et subjective. Plusieurs critères sont pris en compte dans ces cas-là et qui ne sont pas toujours connus de l'usager lui-même, L'utilisateur apprécie la *topicalité* et *l'utilité* du document. En effet, dans le premier cas, le document doit permettre d'accéder à une information complémentaire, et dans le second cas, l'utilisateur doit retrouver

les sujets de sa requête dans le document. Outre l'analyse du sens, l'utilisateur, souvent inconsciemment, applique un algorithme pour juger la pertinence d'un document.

La question qu'on peut se poser est : à quoi sert d'étudier la notion de pertinence si on sait qu'elle est très variable, et même floue ? Une des raisons est de tenter de détecter certains comportements communs chez les utilisateurs, et d'essayer de les formaliser. Si on arrive à cerner une partie de la pertinence commune, on pourra l'implanter dans les systèmes pour au moins répondre à une partie commune des besoins.

3. La Sélection des collections

3.1. Motivations

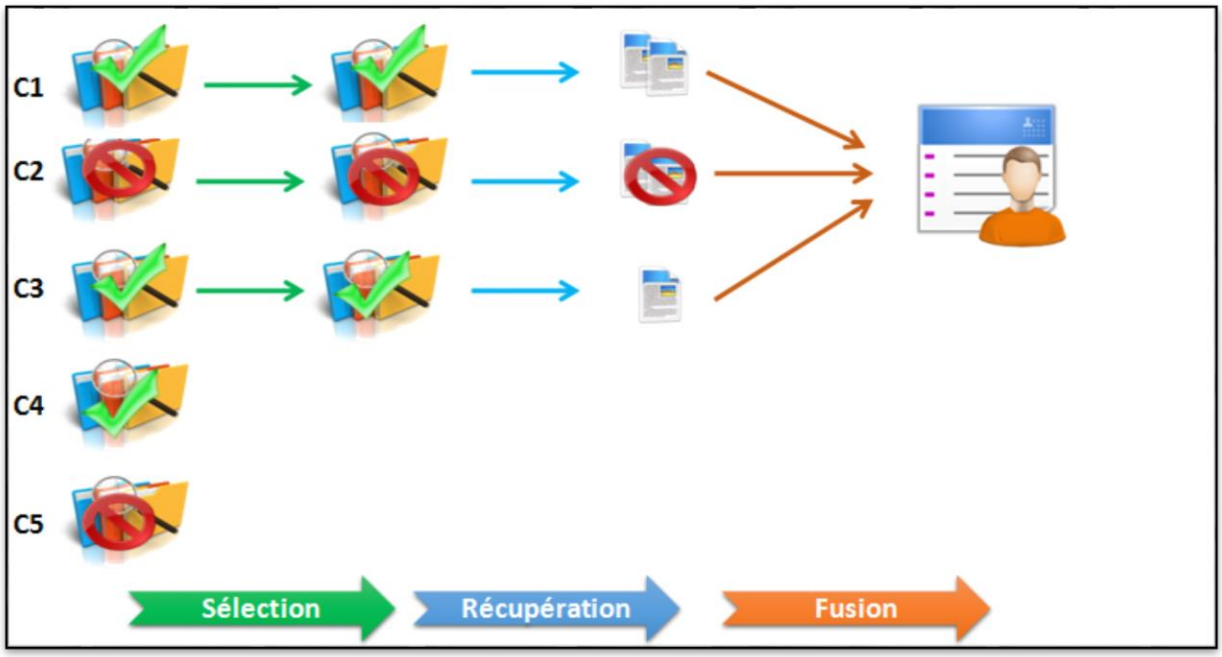


Figure 3. 1 les trois phases primordiales de SRI

Nous sommes motivés par ce domaine pour plusieurs raisons, nous ne citons que deux. La première est le domaine lui-même, où une concurrence acharnée est lancée pour réaliser un système performant. Dans cette partie, nous définissons ce domaine et nous justifions pourquoi un moteur de recherche général est inadéquat pour cette problématique. La deuxième raison est d'ordre conceptuel où nous expliquons les architectures existantes et nous justifions notre choix par une architecture précise.

Le problème de la sélection de collections est largement étudié en informatique [31][25][45][46][41]. En réalité, il a été étudié depuis la période des pharaons, lorsqu'on avait indexé la bibliothèque d'Alexandrie, jusqu'à nos jours [115] [50].

La problématique se pose comme suit : Ayant un ensemble de collections, où chaque collection est composée d'un ensemble de document, et d'une requête donnée. La question de sélection revient à trouver un mécanisme efficace, mieux encore s'il est effectif, pour choisir les meilleures collections à cette requête selon un ou plusieurs critères.

Le processus de sélection passe par trois phases [31]. La première consiste à router la requête vers les collections, les plus pertinentes. La seconde phase est une phase de récupération. Parmi les sélections présentes, le procédé tire les collections candidates, alors que dans la troisième phase, la liste des réponses est fusionnée et triée selon un ordre puis retournée à l'utilisateur. La figure 3.1 donne une vue globale de ce procédé.

La question la plus logique qui peut être posée est : pourquoi cherche-t-on un système pour ce type de problèmes ? Est-ce que les moteurs de recherche généraux tels que Google, Ask ou Yahoo ne suffisent pas ?

Effectivement, les moteurs de recherche généraux ne suffisent pas et n'apporte pas assez de réponses consistantes pour les requêtes qui ont un lien avec le domaine de la sélection de collections. Une collection est un assemblage de documents du même thème, souvent appelés "stories". Une revue, ou un ensemble de revues, ou un journal (tel que Le Quotidien d'Oran, El Watan,…), les récits reportés par les journalistes ou les bourses sont des exemples de ce type de collections. Ce domaine de recherche est très lié à la recherche d'information rapide comme les informations que retourne Yahoo.

La conférence TREC rassemble des ensembles de jeux de données (datasets) extrêmement diverses. On rencontre dans les documents TREC trois parties essentielles : les documents, les requêtes et les jugements. Un document est un assemblage de documents portant des numéros uniques et sont écrits en SGML. La figure 1.10 du chapitre I donne un exemple de document TREC.

Les requêtes sont aussi des documents, écrits dans le même format, possédant une syntaxe bien précise. La figure 1.11 du chapitre I présente un exemple de requête TREC.

Enfin, les jugements sont donnés par un ensemble d'experts de différents domaines, qui jugent pour chaque requête l'ensemble de documents pertinents. Le problème est que tous les documents et toutes les requêtes n'ont pas reçus forcément de jugements. Il s'en suit que, TREC est véritablement un concours où les chercheurs tentent de réaliser un matching maximal entre ces trois composantes.

En essayant alors d'utiliser un moteur général tel que Google pour ce type de problème, on pourra être sûr qu'une seule réponse correcte serait retournée, d'autant plus que les moteurs de recherche généraux n'indexent pas les digital libraries, mais indexent plutôt des documents séparés ayant une URL propre. Nous avons essayé la requête : "*korean merchant banking firm kimbaco*", en date du 25/11/2014 dans Google. La figure 3.2 présente une capture de la réponse retournée par Google.

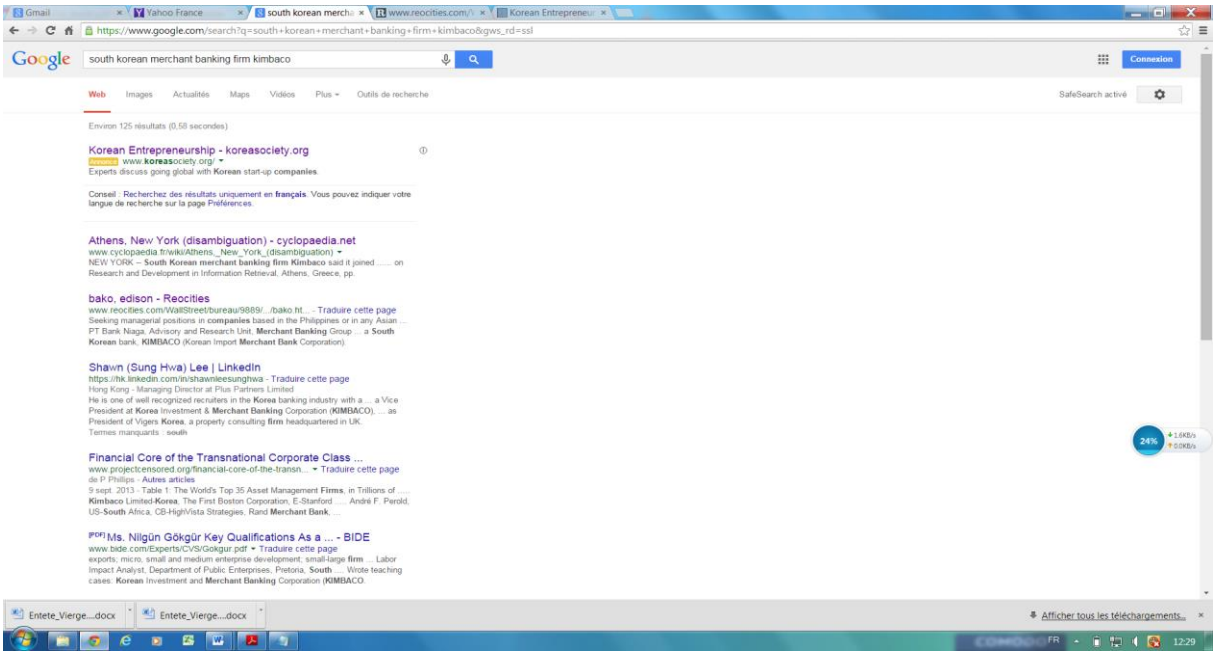


Figure 3. 2 Capture de la réponse retournée par Google.

La figure 3.3 présente le contenu de la meilleure réponse retournée. Bien sûr, nous avons scruté toutes les meilleures réponses.

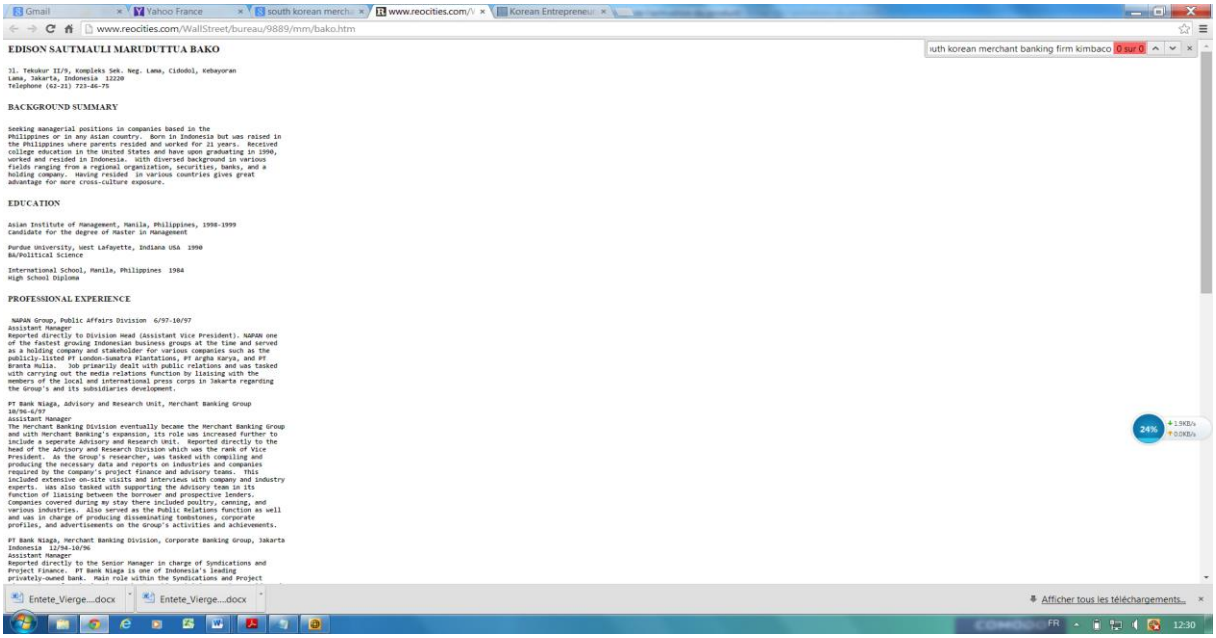


Figure 3. 3 Contenu de la meilleure réponse de Google

L'architecture sur laquelle tourne le système de recherche d'information est une source principale pour présenter des résultats performants. Actuellement, d'innombrables systèmes de recherche d'informations ont vu le jour pour faire face au développement spectaculaire de l'Internet. Ces systèmes reposent tous sur le même principe : la construction d'un index centralisé qui sert de base pour le processus de recherche. La centralisation de l'index constitue la limite de ce type de systèmes, vu la croissance exponentielle de l'information disponible enligne. Ce problème est communément appelé problème de scalabilité ou de passage à l'échelle.

Pour faire face à ce problème, distribuer le processus de stockage et de recherche constitue vraisemblablement le moyen le plus adapté. Un système d'information distribué est typiquement constitué d'un ensemble de serveurs et chaque serveur possède un ensemble de documents répartis sur des collections et d'un module qu'on appelle courtier. Lors de la réception d'une requête, le courtier joue le rôle principal, il sélectionne puis interroge un sous-ensemble de serveurs (collections). Chaque serveur sélectionné prend la main et effectue la tâche qui lui est demandée et retourne une liste de réponses. En deuxième lieu, le courtier fusionne les réponses dans une liste qu'il trie avant de la remettre à l'utilisateur. [1]

De point de vue architecture de système et afin de remédier au problème de passage à l'échelle, les systèmes pair-à-pair hybrides présentent une alternative de taille [83]. Initialement, l'architecture de ces systèmes pair-à-pair a été centralisée où un ensemble de pairs étaient liés à un serveur. Ils informaient de l'ensemble de documents qu'ils mettent en partage. Le serveur construit un répertoire où il enregistre pour chaque document, l'ensemble

des pairs qui le détiennent. Lorsqu'un pair a besoin d'un fichier, il émet une requête auprès du serveur ; celui-ci consulte son répertoire et retourne la liste des pairs actifs qui le possèdent. Ce pair établit une connexion avec un de ces pairs et un téléchargement est actionné. Il est à signaler que le serveur, dans ce contexte précis, ne sauvegarde pas les données, il sert plutôt d'index pour le système global. Un système reposant sur une telle architecture ne peut pas passer à l'échelle. On a remarqué qu'il crashe au bout de quelques centaines de pairs.

Napster[6] [3][83] est le système représentatif de ce type d'architecture. Il a été fermé à cause de procédures de copywright. OpenNap [70] est son héritier. Il est plus performant en matière de charge vu qu'il propose de réunir beaucoup de serveurs autour d'un cercle afin de mieux supporter la charge de communication et la basse de la bande passante.

L'architecture pair-à-pair pure était le deuxième support architectural sur lequel sont basés les nouveaux systèmes. Dans ce cas, les pairs sont liés entre sans qu'il n'y ait d'entité centrale. Tous les pairs sont égaux et permettent le partage et la recherche. Un pair devient serveur lorsqu'il fournit les documents et devient un pair lorsqu'il télécharge. De cette situation, naquit l'acronyme "servent", pour indiquer la fonctionnalité mi-serveur-mi-pair des différents acteurs. La diffusion est le mécanisme de localisation des informations. Le problème est qu'au bout de quelques milliers de messages échangés, le système tomberait dans une noyade.

Le système GNUtella [6] est le plus représentatif de systèmes reposant sur une architecture pure.

Afin de tirer le meilleur des architectures précédentes, on a vu la naissance des systèmes hybrides. Une telle architecture hybride est composée de deux parties. La partie supérieure inclue des pairs plus performants que les autres de points de vue système d'exploitation, bande passante, temps continu passé en activité et absence de pare feu. Ces pairs sont appelés super-pairs ou ultra-pairs. Ils sont liés entre eux selon une architecture pure et forment un graphe connexe. La partie basse se compose des pairs restants. Ceux-ci sont liés aux ultra-pairs et ne sont pas connectés entre eux. De cette manière, le système global devient composé d'un ensemble de clusters, où chaque ultra-pair est un serveur pour le cluster qu'il dirige.

Afin de localiser un document, un pair envoie la requête à l'ultra-pair auquel il est attaché. Ayant un index indiquant pour chaque document la liste de pairs qui le possèdent. Si le document est dans le cluster, l'ultra-pair retourne la liste au pair demandeur sinon il achemine la requête aux ultra-pairs voisins qui feront la même chose. Si le pair demandeur reçoit la liste des pairs ayant le fichier en question, à ce moment, il établit une connexion directe avec l'un parmi eux pour déclencher un téléchargement. Ce type de connexion se ferme une fois le

téléchargement est terminé. De points de vue performance, les systèmes pair-à-pair hybrides sont très performants et actuellement ils sont les plus utilisés.

Sans nul doute, GNUtelle 0.6 est le plus grand système, les autres tels que LimeWire [55] ou FastTrack [54] sont des clients. Skype [102] repose sur une telle architecture.

Par la même occasion, on a vu la naissance de systèmes pair-à-pair où l'architecture est structurée. Ce type de système repose sur ce qu'il convenu d'appeler les tables de hachage distribuées DHT. Une sorte d'administration est implémentée afin d'assurer l'affectation des clés à travers le réseau. Ce type d'architecture est performant sauf qu'il n'est pas assez puissant comme les systèmes hybrides. Il peut souffrir de la réputation des adresses ce qui cause des problèmes de virtualisation et d'accès aux ressources. Le système CHORD [104] est un bon représentatif de ce type de systèmes. Il réunit les pairs autour d'un cercle et chaque pair est doté d'une table de pointeurs ou fingers qui permettent aux requêtes de sauter vers des pairs lointains. La correspondance entre les pairs et les clés est bien calculée afin que les sauts soient corrects. La figure 3.4 présente un exemple de localisation d'une clé dans Chord. Les fingers permettent des branchements lointains pour rapprocher N8 du nœud N56 qui possède la clé K58.

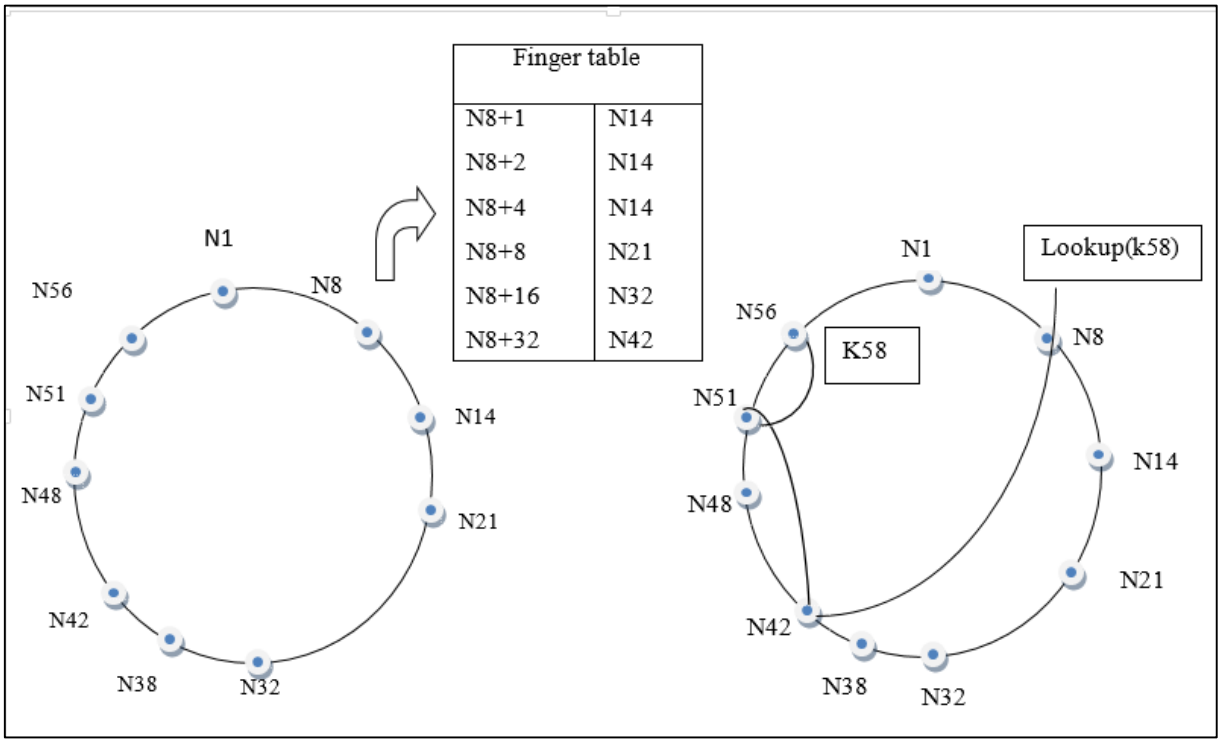


Figure 3. 4 Localisation d'une information dans Chord

3.2. Objectifs

La sélection de collections se réfère à la sélection automatique ou manuelle d'un sous-ensemble de collections susceptibles de contenir des documents pertinents pour une requête donnée. La négligence de cette étape et le transfert de la requête à toutes les collections disponibles au moment de l'interrogation est une opération très coûteuse en termes de flux circulant dans le système. Ce qui mène à l'augmentation de la latence. Ainsi, le but de la sélection est de réduire le nombre de collections questionnées autant que possible, sans diminuer de l'efficacité de la récupération [41] [36].

Notre objectif principal est contenu dans les motivations précédemment introduites. Il s'agit de réaliser un système permettant de localiser l'information dans un système possédant un ensemble de collections. Nous étions motivés vu que la recherche approximative permet de laisser une brèche où nous pouvons proposer une solution compétitive. D'autant plus que beaucoup de méthode existantes définissent des paramètres issus des expérimentations menées et omettent une certaine anatomie du système de collections qui pourrait exister. Ces paramètres peuvent induire de mauvais résultats si la situation qui a permis la déduction des valeurs de ces paramètres n'a pas lieu. Un second objectif est de coupler la recherche des collections et la recherche de documents. Ceci veut dire, que nous tentons de réaliser un système permettant de sélectionner les documents indépendamment des collections à la manière des machines de recherche généraux. Un troisième objectif est de réaliser ce système sur une architecture pair-à-pair hybride vu que nous disposons d'un système offrant cette possibilité.

Dans la partie qui suit, nous présentons des systèmes qui ont trait direct avec la sélection de collections et nous enchaînerons par la suite par notre contribution.

4. Méthodes de sélection de collections

Le problème de la sélection des collections est très important et suscite des travaux intenses. Dans cette partie, nous présentons les méthodes CVV, Gloss ainsi que les deux méthodes CS et CORI. Ces deux méthodes sont connues comme étant des références dans ce domaine. Ces systèmes sont centralisés. Nous enchaînons alors la présentation de deux systèmes décentralisés qui tournent sur des architectures pair-à-pair.

4.1.Méthodes basée sur l'approche naïve

Il est sans doute plus simple d'ignorer l'étape de sélection de serveur et d'envoyer la requête à tous les serveurs connus du système. D'une part, ce choix, souvent adopté par les usagers débutants, s'avère très onéreux en termes de ressources et de communication, ce qui génère des temps d'attente important. D'autre part, interroger systématiquement tous les serveurs risque de dégrader la qualité de la recherche d'information dans le système. En effet, interroger un serveur qui ne contient pas de document pertinent augmente le bruit dans la réponse. Effectuer la sélection revient donc à diminuer le coût de la recherche et augmenter, éventuellement, la performance du système.

4.2. La sélection manuelle

L'utilisateur se voit confié la tâche de choisir les serveurs sur lesquels il voudra effectuer sa recherche. Par exemple, dans West-Law[3] , l'utilisateur sélectionne les sources de droit qu'il désire consulter. Cette sélection se fera sur la base des descriptions des serveurs ou sur des connaissances *a priori* qui, en général, sont très restreintes.

4.3. Méthode basée sur la CVV :

Par définition, la Cue validity est la probabilité conditionnelle qu'un objet appartient à une catégorie particulière connaissant un attribut de cet objet. Cet attribut sert comme indice sur l'objet.

Cette notion a été initialement adaptée dans le contexte de la catégorisation des textes. Dans la recherche d'information, un attribut pour un texte c'est le terme alors que le document est un attribut de la collection. La CVV a été utilisée initialement en 1995 pour la catégorisation de textes. En 1997, Yuwono et al. [113] l'ont utilisé pour la sélection de serveurs, où ces auteurs [113] présentent un SRID appelé D-WISE dont la partie sélection de serveurs est nommée sélection par Cue Validity Variance (CVV) car elle est basée sur la CVV des termes de la requête. Pour une collection donnée, la CV (Cue Validity) d'un terme indique sa capacité à distinguer les documents entre eux. Notons que cette valeur est similaire à 1'*Idf* classique sauf que l'*Idf* porte sur le pouvoir d'un terme à distinguer les documents où il apparaît des autres documents d'une collection. La CVV est la variance de CV à travers toutes les collections du système.

[3]Service en ligne pour la recherche dans le domaine juridique: http:/ /www.westlaw.com

Un serveur *S* est représenté au niveau du courtier par le nombre de documents que sa collection contient, et la fréquence documents de chaque terme qu'il contient. Le score d'un serveur est obtenu en combinant les CVV des termes de la requête et leurs fréquences-documents. Le score d'un serveur fournit des indications sur la concentration des termes de la requête dans la collection du serveur.

4.4. Gloss

Gravano et al. [38] ont introduit Gloss (Glossary of Servers Server). La première version de ce système fonctionne dans un environnement booléen où chaque serveur utilise un système de recherche d'information booléen. Gloss utilise la fréquence documents et la taille des collections pour calculer le score d'un serveur. Ce dernier est le nombre estimé de documents pertinents que contient le serveur.

Gloss a été généralisé par la suite à un environnement vectoriel où chaque serveur utilise un SRI vectoriel, cette nouvelle version de Gloss est appelée vGloss [38]. On suppose dans ce cas que le même SRI est utilisé par tous les serveurs. Pour une requête donnée, le score d'un serveur est la somme des scores de ses documents supérieurs à un seuil donné. Bien sûr, ces scores n'étant pas connus par le courtier, ils sont estimés. L'estimation des scores des documents est basée sur les informations que le courtier détient concernant les collections. Les informations liées au serveur sont représentées par deux vecteurs, celui des valeurs de fréquence documents pour chaque terme de la collection et celui de la somme des poids de chaque terme dans les documents de la collection.

4.5. La méthode CORI

CORI [27] est un algorithme de classement pour le système de récupération INQUERY [26]. CORI s'exécute sur les réseaux d'inférences bayésiens avec des gigantesques documents, dans la mesure où chaque collection est la fusion de tous les documents qu'elle contient [27]. Dans ce réseau bayésien, les feuilles *d* représentent les documents de chaque collection et les termes qui apparaissent dans la collection représentent les nœuds. Les valeurs des arcs sont des probabilités. Ces dernières sont basées sur des statistiques comme le *df* (document frequency) et *icf* (inverted collection frequency). La figure 3.7 représente un tel schéma [27].

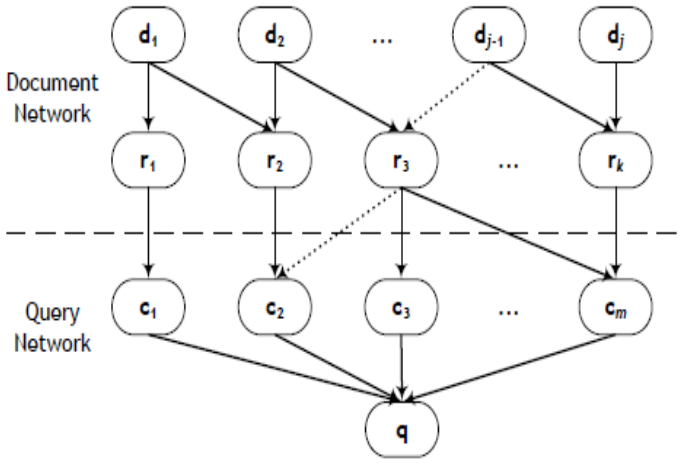


Figure 3. 5 Un simple réseau d'inférence de recherche de documents [27]

Dans ce système, la similarité entre la requête d'un utilisateur et un ensemble de collections de documents est calculée afin de classer les collections selon leurs pertinences. La requête est ensuite soumise à chaque collection sélectionnée (bien classées) pour récupérer l'ensemble des documents les mieux classés. Ces ensembles de documents séparés sont ensuite fusionnés.

CORI considère que la similarité entre une requête *Q* et une collection donnée est la somme des probabilités des termes de la requête apparaissant dans la collection. Le problème avec cette méthode est qu'elle privilège les collections contenant le plus de documents concernant un terme donné. Cette manière de calcul, montre que CORI s'intéresse plus à la quantité qu'à la qualité des réponses même si la collection ne répond qu'à un seul terme de *Q*. Il est important de soulever que CORI répond au requête disjonctives.

Dans ce système, la similarité entre une requête et une collection donnée est la somme (la disjonction) des probabilités des termes de la requête apparaissant dans la collection. La similarité est calculée comme par l'équation 3.1 suivante [27]:

$$CORI(q,c) = \frac{\sum_{t \in q\&c}(d_b + (1-d_b)\cdot T_{c,t}\cdot I_{c,t})}{|q|} \qquad (3.1)$$

Où :

d_b : est une valeur obtenue de façon heuristique, généralement elle vaut 0.4.

$T_{c,t}$: représente le poids du terme *t* dans la collection *c*. Il est donné par l'équation 3.2.

$I_{c,t}$: est l'inverse de la fréquence de la collection *c* qui contient le terme *t* (inverted collection frequency). L'équation 3.3 calcule cette formule.

$|q|$: est le nombre de tous les mots de la requête sans redondance.

Le poids $T_{c,t}$ est calculé par la formule suivante :

$$T_{c,t} = d_t + (1 - d_t) \cdot \frac{f_{c,t}}{f_{c,t} + 50 + 150 \cdot F_c / \overline{F_c}} \quad (3.2)$$

Où :

d_t : est une valeur obtenue à partir d'un ensemble de heuristiques et vaut 0.4.d_t est la fréquence du terme par défaut.

$f_{c,t}$: le nombre de documents contenant le terme *t* dans la collection *c*.

Fc: Le nombre de termes dans la collection c.

Le facteur$I_{c,t}$ est calculé comme suit :

$$I_{c,t} = \frac{\log((N + 0.5)/f_t)}{\log(N + 1.0)} \quad (3.3)$$

Où :

f_t:le nombre de collections contenant le terme *t*.

N : le nombre de collections dans tous le système.

Discussion

Le score d'une collection *c* dans CORI, tel que donné par l'équation 3.1, dépend finalement du facteur *Tc,t* (de l'équation 3.2). Le facteur *Ic,t*, pour une requête et, au moment du calcul, dépend de *ft* et *N*. Ces deux valeurs sont communes à toutes les collections contenant les termes de la requête. Donc, *Ic,t* n'a pas d'impact sur le calcul final.

L'analyse donc de l'équation 3.2 montre que $T_{c,t}$ dépend largement de la valeur de *dt*. Cette constante est obtenue en analysant le jeu de données sur lequel va se faire. Puisque *dt* est la valeur minimale des fréquences des termes et puisqu'elle est fixée dans le système, il s'en suit que toutes les collections sont considérées égales vis-à-vis les termes. En d'autres termes, le nombre d'occurrences des termes n'ont aucune signification. A cause de cette logique, on ne peut conclure qu'une collection *c1* qui est plus pertinente qu'une autre collection *c2* même si les termes de *Q* apparaissent plus dans *c1* que dans *c2*. On conclue que les collections ne peuvent avoir une valeur par défaut car il est impossible de prédire la valeur de *dt*. Dans les

travaux de Callan lui-même [25],[36], *dt* a mis à 0. Donc, il n'y a aucun consensus sur cette valeur.

Encore à partir de l'équation 3.2, *Tc,t* dépend de *fc,t* et *Fc*. Le score d'une collection est corrélé avec *fc,t* ; plus ce facteur est élevé plus le score est élevé. Il résulte que pour deux collections *c1* et *c2,* si *fc1,t>>fc2,t* alors *Tc1,t>Tc2,t* même si la fréquence du terme *t* dans *c2* est supérieure à sa fréquence dans *c1*. Ceci n'est pas très adéquat avec les travaux qui portent sur l'étude des profils. Nous pensons que ceci n'est pas assez réel puisque les utilisateurs sont plus intéressés par les collections qui ont un maximum d'occurrences des termes de la requête. C'est de cette remarque, qu'on conclue que CORI s'intéresse plus à la quantité de réponses qu'à leur qualité.

Tc,t est aussi affecté par *Fc* le nombre de termes, ou longueur, contenus dans la collection. Cette équation montre clairement que le score *CORI(Q,c)*est inversement proportionnel avec *Fc*. Il s'en suit que les collections larges sont pénalisées. Callan, le maître d'orchestre de CORI, lui-même reconnaît ce problème dans [100]. Dans [99] et dans [98], il est montré que lorsque la taille des documents varie, CORI ne donne pas de bons résultats. Ces travaux donnent des méthodes pour estimer la pertinence des documents en introduisant de nouveaux paramètres ; le problème est que ces travaux n'ont pas discuté le choix des paramètres nouvellement introduits. En réalité, les méthodes qui se basent sur les heuristiques souffrent confrontées aux problèmes de paramétrisation, puisque si les circonstances qui ont permis l'instanciation des constantes ne sont pas présentes alors les résultats ne seront pas sûrs. On note que le reste de l'équation est invariant au moment du calcul, d'où le score final dépend de l'équation (3.2). [103] a montré que pour chaque dataset, les paramètres d_b et d_t doivent être calibrés. Ces calibrages posent un grand problème à CORI, puisqu'on ne peut les prédire et les changer à chaque fois.

CORI reste toujours une méthode de sélection des collections importante dans ce domaine de recherche d'information dans les réseaux bayésiens. Holz et al dans [42] ont proposé un Framework pour l'évaluation de l'information dans un système pair-à-pair, ils incorporent CORI comme fonction de classement. Ce travail a été étendu dans [111] où une étude expérimentale a entrepris le classement des ressources d'information dans un environnement pair-à-pair où il utilise une forme modifiée de CORI pour le ranking. Cette modification a été justifiée dans la mesure où c'était difficile de l'appliquer telle qu'elle est définie dans son origine.

[96] est un travail de la recherche d'information qui se base sur les langages. Ce travail, que nous présenterons avec plus de détails dans la section5, a proposé une nouvelle fonction de classement des ressources. Cette fonction a été comparée à CORI.

De leur part, [89] présente un système pour la classification de patents afin de sélectionner les collections et d'intégrer des outils de recherche dans leur système à base de cette classification. La méthode de sélection proposer a été comparée à CORI et semble donner de meilleurs résultats.

[47] démontre que la similarité inter-document est un moyen efficace pour fédérer les recherches. Ils proposent ainsi une fonction basée sur l'heuristique et qui évalue les requêtes en tenant compte des clusters auxquels ces documents appartiennent. Les résultats fournis par cette stratégie ont été comparés avec CORI.

4.6. La méthode Classement des Serveurs CS

[66] le principe de la méthode Classement des Serveurs est de ne pas analyser tous les documents de chaque collection (serveur). Elle propose d'analyser seulement les *n_doc* premiers documents de chaque collection. Encore plus, toujours pour des besoins d'économie, le système ne prend pas en charge tous les termes de la requête ; plutôt, il ne considère que les deux premiers termes car, selon [13][51],ils sont les plus importants mots clés Inspirée.

Largement inspiré du système Inquirus[51], pour calculer le score d'un serveur (collection), CS commence par attribuer un score à chacun des *n_doc* documents, comme le montre l'équation 3.4 suivante:

$$\text{score}(d,q) = (c1.nb_d) + \big(c2.ind\_dis(d,q)\big) + \frac{nb\_occ}{c3} \qquad (3.4)$$

Où, pour tout document *d* :

nb_d : est le nombre de mots clés de la requête contenus dans *d*,

nb_occ : est le nombre total d'occurrences des mots clés dans *d*,

ind_dis : est un indicateur de distance entre deux termes de la requête dans *d*. Cet indicateur est donné par l'équation 3.5 suivante[1][51]**.**

$$score(d,q) = (c1.nb_d) + \big(c2.ind\_dis(d,q)\big) + \frac{nb\_occ}{c3} \qquad (3.5)$$

c1, c2,c3 sont des constantes positionnées à c1=100, c2=1000, c3=1000 selon [51][66].

Le score d'un serveur est que la somme des scores des documents, comme suit :

$$S_i = \sum_{d \in n\_doc\_i} score(d, q) \quad (3.6)$$

Discussion :

CS retourne rapidement des réponses avec une minimum quantité de messages, cependant cette approche souffre de deux problèmes :

Le premier réside dans la manière avec laquelle les documents sont sélectionnés. Le fait de limiter les collections à *n_doc* documents seulement à tester, permet de négliger un nombre très important de documents pertinents quand ce dernier sera supérieur à *n_doc*, surtout quand une collection contient des documents pertinents rarement trouvés dans une autre collection, comme nous allons monter dans la partie expérimentation..

Le deuxième problème réside dans la restriction faite où ils ne considèrent que la distance entre les deux premiers termes. Pour les requêtes composées de trois termes ou plus, les auteurs choisissent d'omettre les termes à partir de la troisième position. Intuitivement, les nombre de faux positifs et de faux négatifs vont augmenter ; ce qui engendre une ambigüité dans les A titre d'exemple, soit la requête Q=<composition, chimique, eau>. Selon CS, la seule distance à retenir est celle entre les termes : "composition", "chimique". Le tableau 3.1 donne trois réponses pertinentes où la plus pertinente est moins classée. Si on considère *n_doc=2* alors cette réponse sera écartée.

Résultats de la recherche
Composition chimique mercure
Composition chimique eau
Composition chimique air

Tableau 3. 1 Résultats retournés selon CS

4.7. Fusion de résultats dans une machine pair-à-pair de recherche dans le Web

Les systèmes précédents tournaient sur des architectures centralisées où un serveur central s'occupe de l'indexation, la sélection, la fusion et la réponse. Afin d'améliorer le système en terme de minimisation de charge, les systèmes pair-à-pair ont été présentés comme une alternative au modèle centralisé.

Dans cette optique, [97] est un travail qui tourne sur le système MINERVA[4]. [12] Celui-ci est un système pair à pair fondé sur une table de hachage distribué DHT de type Chord [104]. Chaque pair explore le Web et compose un index local de l'ensemble des pages Web qu'il a crawlé. Les pairs coopèrent pour construire un répertoire global dans lequel chaque pair

[4]http://www.minerva-project.org

inscrit un sommaire de ce qu'il partage. Le sommaire contient les *df* (fréquences des documents), les *tf* (terme frequencies) et le nombre total des documents et leurs longueurs. Une procédure de hachage est appliquée sur les termes. La finalité de la coopération entre les pairs génère une méta-machine pour la recherche à travers le Web.

[97] s'appuie sur le feedback de la pseudo-pertinence et se base aussi sur le modèle basé sur langage pour extraire l'information, ainsi, les auteurs considère le Global English language model *GE*. La méthode de sélection consiste à calculer le score d'un pair P_i pour une requête *Q*. Cette méthode passe par deux étapes. Dans la première, le $Score(Q, P_i)$ se calcule par l'équation 3.7. On note que t_k est le $k^{ième}$ terme de la requête *Q*.

$$Score(q, P_i) = -\sum_{k=1}^{|q|} \log p(t_k \backslash P_i) \tag{3.7}$$

La probabilité conditionnelle $p(t_k \backslash P_i)$que le terme t_k est dans le langage de P_i selon *GE* est donnée par l'équation 3.8.

$$p(t_k \backslash P_i) = \lambda \,.\, \frac{ctf_{tk}}{|P_i|} + (1-\lambda).p(t_k \backslash GE) \tag{3.8}$$

Avec λ une valeur empirique appartenant à l'intervalle [0,1]. Le facteur ctf_{tk} est la fréquence du terme t_k dans la collection du pair P_i ; $p(t_k/GE)$ est une probabilité conditionnelle que t_k soit dans le langage GE. L'équation 3.9 donne une évaluation de cette probabilité :

$$p(t_k \backslash GE) = \frac{\sum_{i=1}^{|p|} ctf_{t_k}^{P_i}}{\sum_{k=1}^{|T|} \sum_{i=1}^{|P|} ctf_{t_k}^{P_i}} \tag{3.9}$$

Dans la seconde étape pour calculer la sélection finale des pairs répondants au mieux à la requête Q, les k meilleures réponses sont étendues par une méthode d'expansion ; où des termes sont rajoutés à la requête. Ce procédé permet de se rapprocher du profil des utilisateurs. Plus de détails sont dans [28].

5. La méthode Collection Selection-Based Relevance Degree CSRD

Dans cette partie, nous présentons notre contribution dans le domaine de la sélection de collections. Vu que les méthodes précédentes, et bien d'autres, définissent toutes dans leurs formules de calcul des paramètres qui obtiennent leurs valeurs par des méthodes empiriques. Nous leur reprochons ce procédé empirique, car si les conditions qui ont fourni ces valeurs ne

sont pas réunies alors les résultats ne seront pas corrects. Par exemple, Callan, le pionner de la méthode CORI, a donné plusieurs formulations à sa fonction de calcul, et jusqu'alors il la perfectionne dans ses travaux. Spark Jones, qui est le maitre incontestable de la célèbre méthode BM25[84], [87], a lui aussi posé ce problème et a donné plusieurs versions à sa formules BM25(BM3, BM11 et bien d'autres).

Ce constat nous a incités à définir une formule libre de tout paramètre heuristique et ne tient compte que de l'anatomie des collections. Afin de mieux expliquer notre objectif ici, nous considérons l'exemple suivant, tout en gardant la même notation introduite dans la section 4.6 :

Soient deux collections c_1, c_2 et un terme t_1 appartenant à ces deux collections. Le tableau 3.2 donne des statistiques de t_1 dans c_1 et c_2.

Attribut	**Désignation**	**Valeur**
$f_{c1,t1}$	Nombre de documents dans c_1 contenant t_1	5
F_{c1t1}	Les occurrences de t_1 dans c_1	12
f_{c2t1}	Nombre de documents dans c_2 contenant t_1	17
F_{c2t1}	Les occurrences de t_1 dans c_2	17

Tableau 3. 2 Un exemple introductif

Dans cette situation, il résulte, d'une part, que la collection c_2 continents 17 documents où $t_1$apparaît une seule exactement, seulement, par document, et, d'une autre part, la concentration de ce terme dans $c_1$est plus importante. Cette situation révèle qu'il y a des documents où t_1 apparait plusieurs fois. De point de vue utilisateur, $c_1$est plus pertinente que c_2. Il est sûr que CORI réponde par c_2, puisque le nombre de termes contenant $t_1$est plus élevé. Cette réponse n'est pas satisfaisante, car de point de vue sémantique la quantité informative dans $c_1$est plus importante que dans c_2.

Nous nous proposons de définir une fonction de sélection qui maximise le quotient informatif des collections. Ainsi, une collection recevra un meilleur rang si les degrés de ses documents pertinents sont élevés. Dans le but d'arriver à cette fin, nous donnons les définitions fines suivantes.

5.1.Définition de la pertinence et du degré de la pertinence :

Definition1 : pertinence d'un document :

Un document *d* est pertinent pour une requête *q* s'il partage au moins un terme avec elle. Ce qui est calculé par l'équation 3.10 [61] [60] [62].

$$\boldsymbol{d} \cap \boldsymbol{q} \neq \emptyset \tag{3.10}$$

Définition2 : **pertinence totale :**

Un document *d* est considéré totalement pertinent pour une requête *q* s'il contient tous les termes de cette requête. On calcule cette propriété à l'aide de l'équation 3.11 :

$$\boldsymbol{d} \cap \boldsymbol{Q} = \boldsymbol{Q} \tag{3.11}$$

Définition3 : degré de pertinence d'un document :

Le degré de pertinence d'un document *d* pour une requête *q*, noté d°(d), est le nombre de termes partagés par eux. L'équation 3.12 suivante le donne.

$$\boldsymbol{d}^{\circ}(\boldsymbol{d}) = |\boldsymbol{d} \cap \boldsymbol{Q}| \tag{3.12}$$

Définition4 : degré de pertinence total d'un document

Le degré de pertinence d'un document d°(d) est dit total, noté td°(d) s'il vérifie l'équation 3.13.

$$\boldsymbol{td}^{\circ}(\boldsymbol{d}) = |\boldsymbol{Q}| \tag{3.13}$$

Définition5 : pertinence d'une collection

Une collection *c* est pertinente pour une requête *q* si elle contient un, ou plus, document pertinent pour cette requête. Ce qui donne : $\boldsymbol{c} \cap \boldsymbol{q} \neq \emptyset \Rightarrow \exists \boldsymbol{d} \in \boldsymbol{c} / \boldsymbol{d} \cap \boldsymbol{q} \neq \emptyset$

L'équation 3.14 est une formulation de cette définition :

$$\boldsymbol{c} \cap \boldsymbol{Q} = \{\boldsymbol{d} \in \boldsymbol{C} \ / \ \boldsymbol{d} \cap \boldsymbol{Q} \neq \emptyset\} \tag{3.14}$$

Définition 6 : degré de pertinence d'une collection

Le degré de pertinence d'une collection c, noté *d°(c),* est le nombre de documents pertinents de c pour q. l'équation 3.15 donne ce nombre.

$$d^{\circ}(c) = |c \cap Q| \quad (3.15)$$

Nous étendons la pertinence entre une requête q et une collection c selon le nombre de documents pertinents appartenant à la collection c et répondant à la requête q. D'où une collection c est pertinente pour la requête q si et seulement si :

$$c \cap q \neq \emptyset$$

Définition 7 : niveau de pertinence de documents

Soient d_i etd_j deux documents pertinents pour la requête q. d_i est plus pertinent que d_j si et seulement si :

$$d^{\circ}(d_i) > d^{\circ}(d_j) \quad (3.16)$$

Définition 8 : niveau de pertinence de collections

Soient c_i et c_j deux collections pertinentes pour la requête q. c_i est plus pertinente que c_j si et seulement si:

$$d^{\circ}(c_i) > d^{\circ}(c_j) \quad (3.17)$$

5.2. Architecture du système

Afin de résister à l'effet passage à l'échelle, nous proposons un système distribué de type pair-à-pair hybride. Nous proposons un ultra-pair appelé broker auquel sont attachés d'autres serveurs appelé pairs. Les collections sont réparties une par pair. Chaque pair construit un index local qui indique pour chaque terme la liste des documents qu'ils le contiennent ainsi que leurs *tf*. Le broker maintient un index qui indique pour chaque terme la liste des pairs qu'ils le possèdent dans leurs index locaux.

Lorsqu'un utilisateur émit une requête au système, le broker la reçoit et l'analyse afin de l'acheminer au sous-ensemble des pairs pertinents pour elle. Chaque pair calcule son propre score selon la fonction que nous présenterons dans la section suivante. Les pairs envoient leurs scores et leurs listes de documents au broker qui réunit les résultats et les trie pour retourner le résultat de la fusion à l'utilisateur. Ce procédé est présenté dans la figure 3.8 suivante [61]

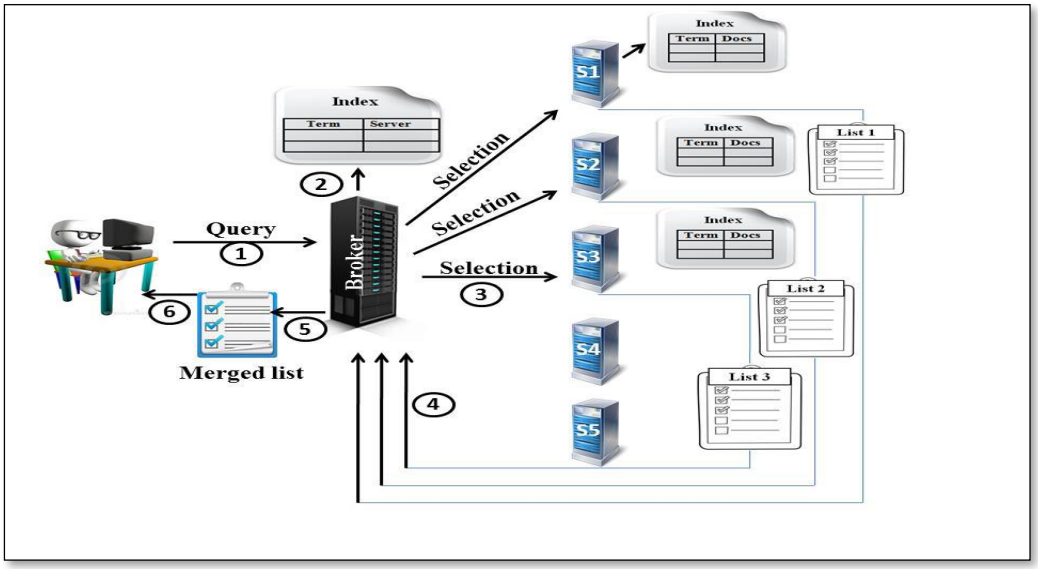


Figure 3. 6Processus d'évaluation d'une requête dans un environnement distribué

5.3. La méthode Collection Selection-Based Relevance Degree CSRD [57] [63] [60]

L'exemple introductif montre clairement l'idée de base que nous voulons mettre en œuvre. Nous proposons de classer les collections selon leur quotient informatif dans les meilleurs rangs. La fonction de sélection, proposée, privilégie les collections contenant les documents les plus pertinents pour une requête q; c'est-à-dire, les collections dont les degrés de pertinence de leurs documents sont maximaux : $\boldsymbol{d^{\circ}(d) = |d \cap q| = |q|}$**.** C'est à partir de ce principe que notre méthode porte le nom CSRD (pour : Collection Selection-Based Relevance Degree, en anglais) SDP (Selection basée sur le Degré de Pertinence, en français).

Principe de fonctionnement

En recevant la requête q, le serveur S_i, possédant la collection c_i, exécute les tâches suivantes :

1. Déterminer les documents $d_j / \ d_j \cap q \neq \emptyset$
2. Déterminer les documents $d_k / \ d_k \cap q = q$
3. S_i calcule le score de sa collection selon l'équation 3.18 :

$$\boldsymbol{f(q, c_i) = \left(\frac{1}{Nd_{j,i}} + Nd_{k,i}\right) * \sum_{t=1}^{|q|} TF_{t,i}} \qquad (3.18)$$

Où :

$Nd_{j,i}$: le nombre de documents de type d_j dans la collection i.

$Nd_{k,i}$: le nombre de documents de type d_k dans la collection i.

$TF_{t,i}$: La fréquence de terme *t* dans la collection *i*.

Les collections seront classées par la suite par ordre décroissant. La collection qui a le score le plus élevé est considérée la plus pertinente. De cette manière, les réponses contiendront, dans les premiers rangs, les résultats répondant aux requêtes conjonctives, où il est nécessaire de répondre exactement aux requêtes. Le reste correspond à toutes les combinaisons possibles des termes de la requête tenant compte de leurs concentrations dans les documents. CORI répond aux requêtes disjonctives sans tenir compte de la sémantique pouvant être contenue dans les réponses.

6. Conclusion :

Les méthodes de sélection de collections sont nombreuses et, chacune possède un principe sur lequel elle est opérationnelle. Nous leur reprochons l'utilisation de paramètres empiriques. Ces paramètres biaisent le résultat final. Les auteurs doivent calibrer ces paramètres selon les données mises en jeu.

Dans ce chapitre, nous avons défini la pertinence et nous avons présenté plusieurs méthodes pour la sélection de collections. Nous avons accentué l'explication sur les méthodes CORI et CS. Celles-ci ont fait leurs preuves mais souffrent du problème de paramètres empiriques.

Afin d'éviter toute inexactitude dans les calculs, nous avons proposé une méthode de localisation des données servant à estimer la pertinence d'une collection vis-à-vis d'une requête donnée dans un environnement de recherche d'informations distribué. Nous avons présenté certaines définitions importantes et nous avons décrit le système sur lequel nous avons fait l'estimation. Notre fonction ne définit pas des paramètres extra-collection et se base principalement sur le calcul des degrés de pertinence des collections.

Le chapitre suivant présente les résultats des expérimentations des différentes méthodes.

CHAPITRE IV

Expérimentations et Evaluation

Sommaire

1. Introduction

Dans le chapitre précédent, nous avons introduit quelques approches pour la sélection de collections ainsi que notre contribution dans le domaine. Dans ce chapitre, nous abordons l'environnement matériel et logiciel dans lequel nous avons réalisé notre travail. Nous présentons les résultats de la comparaison des trois méthodes CORI, CS et CSRD selon des métriques bien connues.

2. Environnement de développement

Pour mener nos expérimentations, nous avons mis au point les méthodes CORI, CS et SDP sur une machine (Processeur Intel(R) Core(TM)2 Duo CPU T6570 @ 2, 20 GHZ), dotée d'une capacité mémoire de 3 GB RAM sous Windows 7. L'application est développée en utilisant le langage de programmation JAVA sous l'environnement de développement NetBeans. Nous faisons appel au package JFreeChart (version) pour le traçage des courbes.

2.1. Description

Java est un langage typé à objet. Il est compilé et basé sur une architecture logicielle très particulière nécessitant une machine virtuelle Java. Il utilise les notions usuelles de la programmation orientée objet : la notion de classe, d'encapsulation, d'héritage, d'interface, de virtualité, de généricité, … Il est accompagné d'un ensemble énorme de bibliothèques standard couvrant de très nombreux domaines, notamment des bibliothèques graphiques. C'est un langage qui présente d'excellentes propriétés de portabilité du code. Son gros point faible est une relative lenteur, surtout si on le compare à des langages comme le C++. Cependant, ce défaut a été résolu en grande partie par l'introduction de la technologie JIT (compilateur *Just-In-Time*, en anglais « juste à temps »), qui compile le code à la première exécution, permettant une exécution quasiment aussi rapide qu'en C/C++.

2.2. Caractéristique

JAVA présente plusieurs qualités. Les principales sont les suivantes :

interprété	Le code source est compilé en pseudo code ou bytecode puis exécuté par un interpréteur Java : la Java Virtual Machine (JVM)
portable : indépendant de toute plate-forme	Pas de compilation spécifique pour chaque plate-forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine.
orienté objet.	Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs
simple	le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage

	multiple et la surcharge des opérateurs, ...
fortement typé	toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser un cast ou une méthode statique fournie en standard pour la réaliser.
assure la gestion de la mémoire	l'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.
sûr	Un programme Java planté ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet.
Econome	le pseudo code possède une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution.
multitâches	il permet l'utilisation de threads qui sont des unités d'exécutions isolées. La JVM, elle-même, utilise plusieurs threads.

Tableau 4. 1Caractéristiques du langage JAVA

2.3. L'IDE 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 License).

En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres (comme Python ou Ruby) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages 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) Un environnement Java Development Kit JDKest requis pour les développements en Java.

3. Environnement de l'application

Etant donné un ensemble de documents et une requête utilisateur, le but de la présente étude est de comparer la pertinence des réponses des trois méthodes qui répondent à ce besoin en information. Pour ce faire, nous avons besoin, comme tous processus de la recherche d'information, d'un index, d'une collection de test et d'un ensemble de documents qui

répondent pertinemment à chaque requête au sein de cette collection. L'objectif est l'évaluation des performances de chaque méthode.

Nous n'avons pas tenu compte de l'aspect distribué du moment que l'objectif de l'étude est la comparaison entre les trois méthodes en terme de pertinence des réponses et non pas en fonction du temps de réponse, ou tout autre critère, où l'architecture du système aura une influence sur les résultats.

3.1. Collection d'expérimentation

Pour évaluer la qualité des trois méthodes, nous avons effectué nos expérimentations sur les documents de corpus Reuters-21578[5].

Ce corpus est un ensemble de dépêches financières émises au cours de l'année1987 par l'agence Reuters, en langue anglaise, et disponible gratuitement sur le web. Il a été présenté dans [53].Les textes sont présentés dans le format SGML où chaque grand document est composé d'un ensemble de petits documents de longueurs différentes.

Ce corpus est une mise à jour du corpus Reuters-22173. Cette mise à jour, effectuée en 1996, a permis de supprimer les documents présents deux fois, de corriger des erreurs typographiques, de préciser certains formats, et de mieux définir le découpage à considérer pour l'apprentissage et le test. Du fait de ces corrections, il n'est pas possible de comparer les performances obtenues sur les différentes versions du corpus. Cependant, les caractéristiques globales du corpus sont restées identiques, et les remarques sur le comportement général des systèmes étudiés sont toujours valables.

3.2. Requêtes

Afin de tester les performances des différentes méthodes, nous disposons de 4*1000 requêtes de test. Leurs longueurs variaient entre deux et cinq termes.

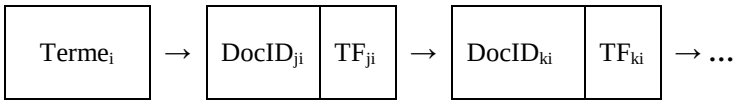
3.3. Les index

Une phase préliminaire et commune à tous les processus de recherche d'information est l'indexation. Pour la présente étude, nous avons utilisé les indexes de type inversé, où chaque entrée est un terme d'indexation suivi par la liste des couples [116]:

<document qui contient ce terme, fréquence du terme dans ce document>.

Il s'agit d'une liste dynamique ayant la structure suivante :

[5]http://www.daviddlewis.com/resources/testcollections/reuters21578/



Nous disposons d'un index global qui indexe 18103 documents qui font le sujet de l'étude. Comme nous l'avions indiqué dans le chapitre précédent, où nous avons construit un système pair-à-pair hybride (voir section 5.2 du chapitre III), nous avons distribué les 18103 documents sur 500 collections d'une manière aléatoire comme [35], engendrant ainsi 500 indexes locaux. La distribution est non uniforme c-à-dire la taille des collections est variable.

3.4. Méthodologie d'évaluation

Dans un premier temps, nous avons comparé la fidélité des trois méthodes à la définition de la pertinence telle que nous l'avions définie dans la section 5.1 du chapitre III. Dans ce stade de comparaison, nous avons exécuté une requête composée de trois termes. Cet exemple sert de prologue pour expliquer la méthodologie de comparaison du second temps. De ce fait, nous testons leurs pertinences et nous les comparons selon les métriques le rappel (recall), la précision et la F-mesure. Dans cette comparaison, nous quantifions les exécutions et nous commentons les résultats. Des détails sur ces définitions peuvent être consultés dans [58] et [23].

4. Résultats des expérimentations :

4.1. Comparaison sur un exemple

Dans cette expérience et dans un premier temps, nous avons lancé la requête Q=<*bahia, cocoa, july*> dans un système centralisé dans le but de calculé toutes les collections pertinentes. Nous avons collecté aussi les valeurs des degrés *d°* et *td°*(voir section 5.1 chapitre III). Nous appelons ces résultats obtenus le "réel existant". Le tableau 4.2 donne un fragment de six (06) résultats. La première colonne contient les collections pertinentes du système. La seconde correspond aux degrés de pertinence alors que la troisième colonne donne les degrés totaux.

Collection	***d°***	***td°***
153	2	3
47	1	2
141	3	2
40	7	1
170	6	1
6	6	1

Tableau 4. 2 les six meilleurs résultats pour la requête Q=< bahia, cocoa, july>

Le tableau 4.2 montre clairement que les collections totalement pertinentes sont données dans les meilleurs rangs (les trois premières lignes). En d'autres termes, l'importance est donnée aux réponses conjonctives. Le reste des réponses sont ordonnées selon leurs degrés de pertinence.

CORI		CS		CSRD	
Collection	**Score**	**Collection**	**Score**	**Collection**	**Score**
6	0.124397083129399	104	0.6275000140000001	153	202.9142857142857
153	0.12436885576294932	165	0.5776000190000001	47	126.21428571428571
40	0.124308533326096	3	0.5715000260000001	141	85.55555555555556
47	0.1243080548149238	114	0.389500024	155	83.53125
189	0.1243057804557923	116	0.326200002	40	76.17391304347827
82	0.12430570051613006	130	0.29230001399999997	102	74.825

Tableau 4. 3 les six meilleures collections et leurs scores retournés par les trois méthodes

Dans le second temps, nous avons injecté la requête Q dans le réseau simulé de 500 pairs. Le tableau 4.3 présente les résultats retournés par les trois méthodes.

Nous remarquons que CORI a retourné la collection 6 comme étant la meilleure réponse. En analysant cette collection, nous avons trouvé qu'elle ne contient pas un nombre élevé de termes F_c en comparaison avec le reste des collections et de même le nombre $f_{c,t}$ de cette collection est plus petit que les $f_{c,t}$ des autres collections (voir l'équation 3.2, page 79). Pour ces raisons, cette collection est bien classée. Globalement, CORI a fourni de bons résultats. Nous signalons que nous avons choisi le meilleur résultat fourni par cette méthode. Malheureusement, la méthode CS n'a pas retourné de bons résultats comme les deux autres. Cette méthode dépend largement de *n_doc*, qui délimite le nombre de documents à retourner, et de la distance entre les premiers termes de la requête. De cette manière, la méthode CS ne peut en aucun cas être assez exhaustive.

Ce tableau 4.3 affiche clairement que CSRD est très similaire au "réel existant". Notre approche a retourné les meilleures réponses dans les meilleurs rangs. La collection 47 est classée meilleure que 141 parce que nous avons choisi de diviser sur $Nd_{j,i}$ dans l'équation 3.18 de la page 87 . Cependant, malgré cette division, nous privilégions les réponses conjonctives, et c'est tout à fait égal à l'exemple introductif donné dans la section 5 page 84.

4.2. Comparaison selon la précision

Nous avons calculé la précision dans le système centralisé. Le tableau 4.4 donne les 10 meilleures valeurs selon la précision. La figure 4.1 présente la courbe de la précision dans le système centralisé.

Collection	Precision
153	0.0045
47	0.0030
141	0.0028
40	0.0016
156	0.0016
43	0.0015
53	0.0015
48	0.0015
82	0.0015
197	0.0015

Tableau 4. 4 Le Top-10 des collections et leurs valeurs de précision pour la requête <*bahia, cocoa, july*>

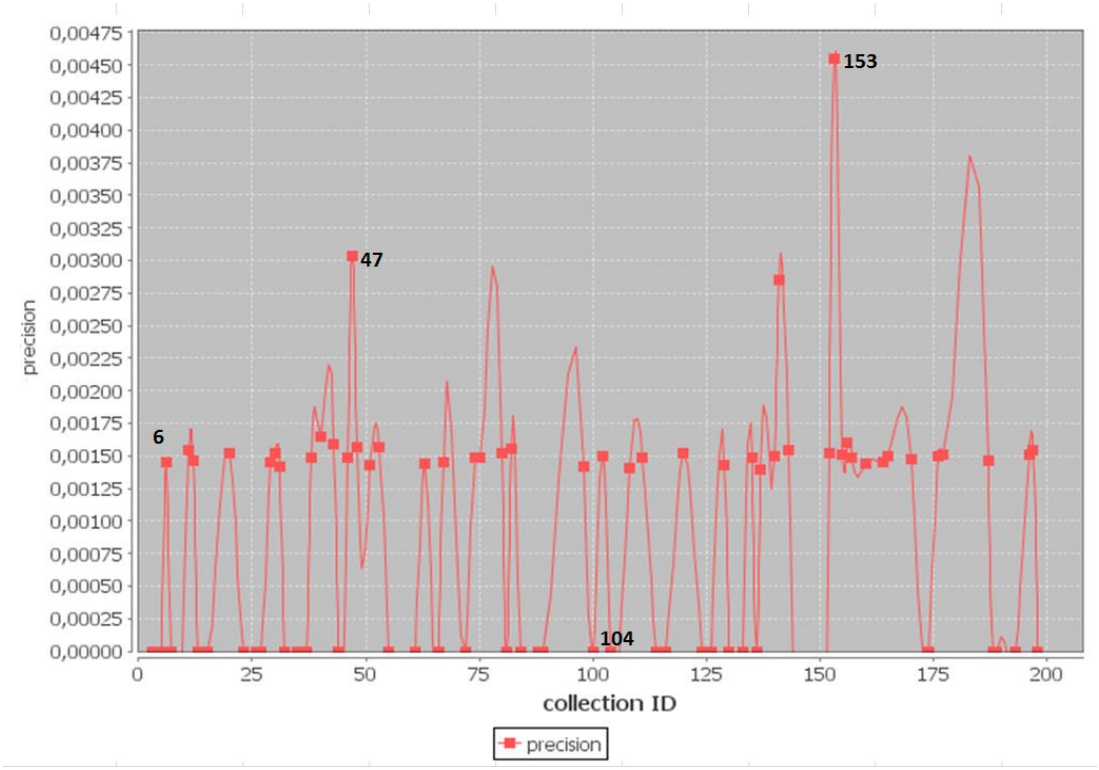


Figure 4. 1Courbe représentant les précisions des collections pour Q

La figure 4.1 montre que le Top-2 de CSRD possède les meilleures valeurs pour la précision (les collections 153 et 74), alors que la précision du Top-1 de CORI n'est globalement élevé. Dans la troisième place, nous avons trouvé que CS n'a pas retourné des collections possédant des valeurs de précision élevées.

4.3. Comparaison selon la F-mesure

Nous avons aussi calculé la F-mesure pour cette requête. Uniquement pour ce cas, nous avons omis le rappel, du moment que la F-mesure contient ces deux métriques dans sa définition. Par contre, nous avons calculé le rappel dans la suite des expérimentations.

La figure 4.2 montre clairement que CSRD a retourné les meilleures réponses avec la meilleure F-mesure. Dans le deuxième rang, CORI a aussi répondu de bons résultats alors que CS n'est pas au même niveau en terme de qualité de réponses comme les autres méthodes.

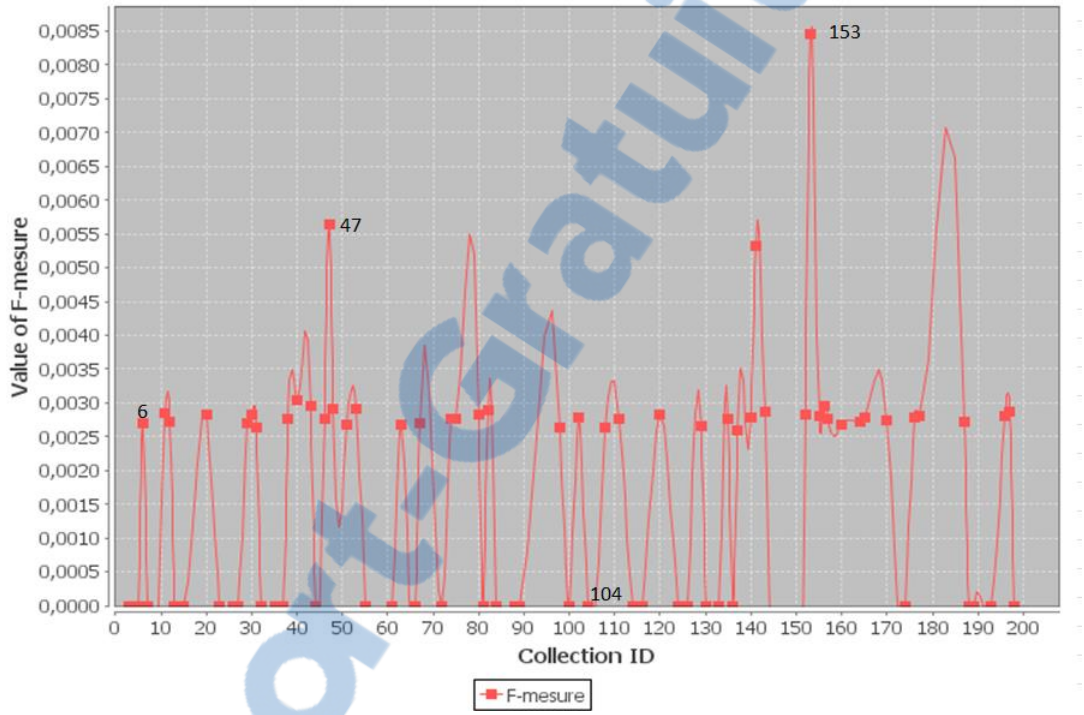


Figure 4. 2Courbe représentant la F-mesure pour la requête Q

Remarque : Pour l'éclaircissement, nous présentons dans l'appendice un exemple de comparaison où nous injectons une requête de deux termes.

4.4. Comparaison en quantifiant les expérimentations

Dans cette partie de la comparaison, nous avons mené des expérimentations intensives où nous avons lancé un nombre important de requêtes. Les informations collectées concernent la pertinence, la précision, le rappel et la F-mesure.

4.4.1 Comparaison selon la pertinence et la précision

Cette étude est faite en deux phases. Dans la première, nous avons étudié le "réel existant". Nous avons lancé la collection entière dans un système décentralisé où nous avons injecté 4*1000 requêtes. Les longueurs de ces dernières variaient entre deux et cinq termes. Nous désignons par Req2 une requête composée de deux termes. Nous avons collecté pour chaque requête Req l'ensemble des documents pertinents et leurs précisions. Les documents pertinents partagent, chacun, au moins un terme avec Req. Les listes obtenues sont triées par ordre décroissant selon la précision. Ce qui donne réellement les meilleures réponses à ces 4000 requêtes.

Dans la seconde phase de ces expérimentations, nous avons lancé ces 4*1000 requêtes dans le système distribué simulé. Nous avons étudié spécialement le Top-5 pour chaque méthode. Nous avons choisi de faire l'étude sur ces sous-ensembles réduits car leurs résultats étaient assez élevés. A partir de la sixième position, les écarts des scores se sont creusés. Par la suite, nous avons pris l'intersection entre du Top-5 de chaque méthode avec le Top-5 du "réel existant" pour chaque type de requêtes. En d'autres termes, nous étudions combien de documents parmi le Top-5 de CORI apparaissent dans le Top-5 du système centralisé. Plus le cardinal de l'intersection est grand plus la méthode est pertinente, indépendamment des rangs occupés par les documents.

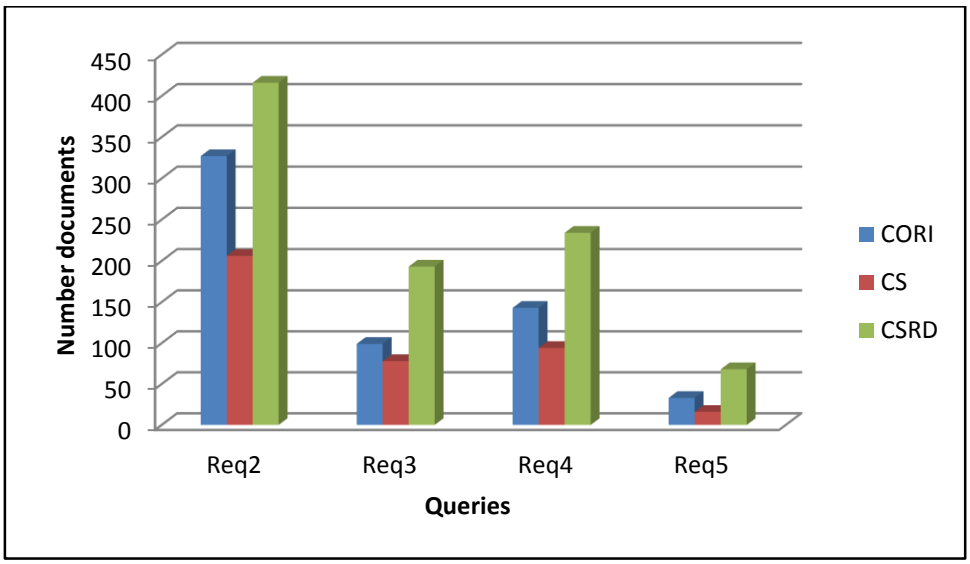


Figure 4. 3 Comparaison selon la précision en termes du nombre de documents partagés entre les méthodes et le système centralisé

Le résultat de cette expérience est illustré dans la figure 4.3. Nous pouvons remarquer que CS a retourné le plus petit nombre de documents en commun avec Top-5 système. Elle a localisé moins de documents ayant des précisions élevées. Cette figure montre que les résultats de CORI étaient satisfaisants. Pour le cas de CSRD, les résultats étaient meilleurs que les autres méthodes. Ces résultats étaient meilleurs de façon que les requêtes de longueurs cinq, par exemple, CSRD a fourni des fois plus de résultats que CORI (68 vs 33). Globalement, CSRD a fourni une moyenne de 77.25 documents de mieux que CORI.

4.4.2 Comparaison selon le rappel

De la même manière que l'expérience précédente, nous avons récolté les résultats pour le rappel. Sauf que pour cette fois-ci, nous varions Top-K de 1 à une valeur aussi élevé jusqu'à la détection de la valeur de K pour que le rappel atteigne 1, la valeur théoriquement maximale à atteindre.

Il est important de noter que cette valeur est purement théorique et qu'il est impossible de l'avoir dans un système de recherche d'information ouvert. Dans notre cas, il s'agit d'un système fermé où il n'y a pas injection de nouvelles collections et il n'y a pas de départ des pairs. Cette situation de système dynamique est un autre cas d'étude différent de celui que nous étudions dans cette thèse.

La figure 4.4 suivante présente le résultat de cette expérience. Afin de donner plus de chance à CS, nous lui avons varié *n_doc* de 2 à 5 ; ainsi, CS_2 veut dire que CS a été exécutée pour *n_doc*=2. Sur la figure 4.4, la courbe CS_3 en jaune n'apparait pas car pour *n_doc*=3, CS a fourni les mêmes résultats que CS_2.

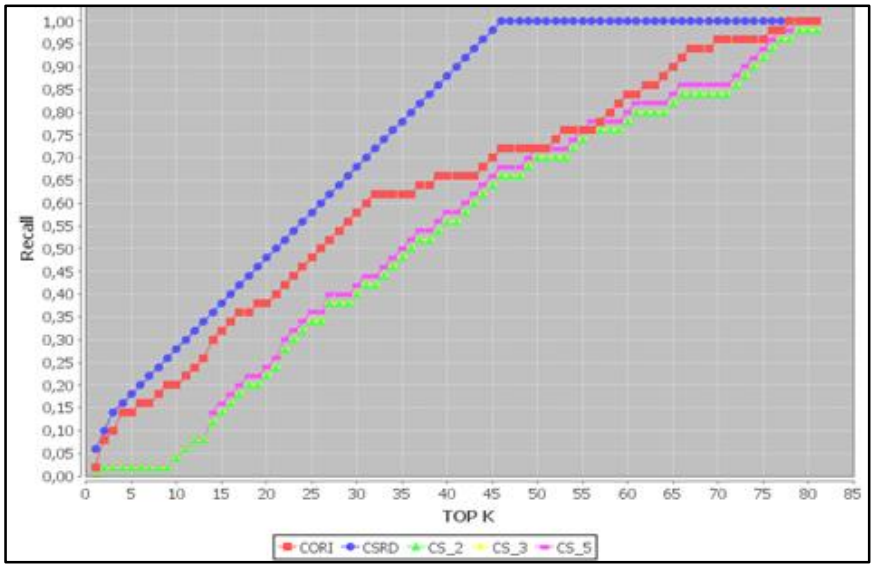


Figure 4. 4Comparaison selon le rappel

Cette figure montre que même en augmentant la valeur de *n_doc*, les performances de CS étaient faibles, le rappel n'a pas pu atteindre la valeur 1. Cette méthode est dans ce cas pénalisée par la manière de calcul où elle ne considère que la distance entre les deux premiers termes. De cette manière, beaucoup de documents pertinents n'ont pas été atteints.

La figure 4.4 montre que CORI est assez performante. Les documents retournés étaient plus pertinents que CS, du moment que à égalité de K, CORI retourne plus de meilleurs résultats. Le rappel de CORI a atteint la valeur 1 pour k=78.

Il est bien illustré dans cette figure CSRD fournit des performances meilleures que les deux autres méthodes. Son rappel était plus élevé et a pu atteindre 1 seulement pour K=46. CSRD est presque deux fois plus rapide que CORI.

4.4.3. Comparaison selon la F-mesure

Dans cette section, nous comparons les trois méthodes selon la métrique F-mesure. Cette métrique est intéressante et peut même remplacer les deux autres métriques [58] [23].

Dans cette série d'expérimentation, nous avons exécuté les 4*1000 requêtes dans le système centralisé. Les requêtes sont de longueurs différentes. Pour chaque type de requêtes, nous avons trié la liste des résultats dans un ordre décroissant selon la F-mesure. Là aussi, nous avons injecté les requêtes dans le système distribué. Nous avons récolté les résultats pour chaque méthode, puis nous avons calculé l'intersection entre la liste du système centralisé avec la liste de chaque méthode. Dans cette expérience, nous avons calculé le Top-10.

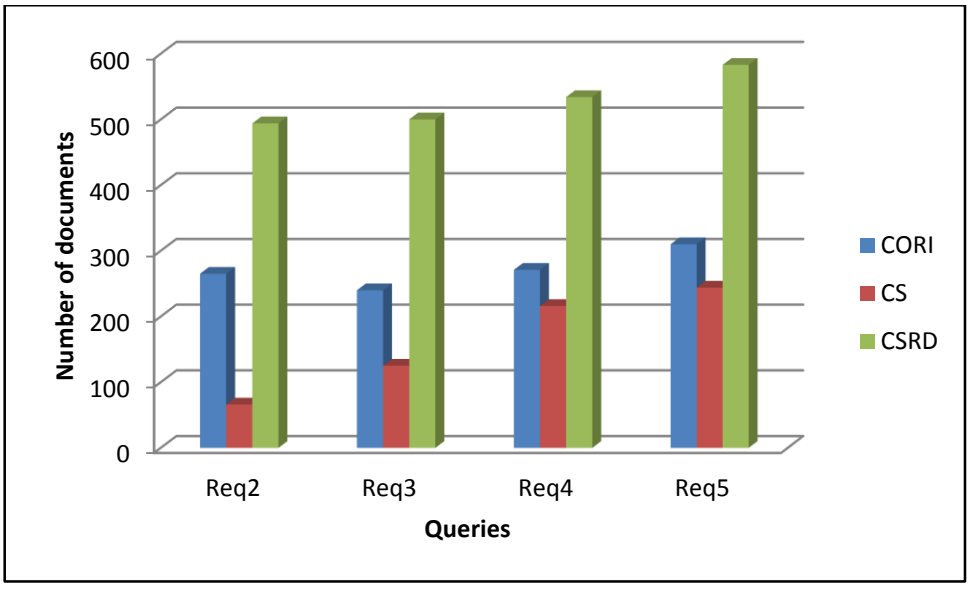


Figure 4. 5Comparaison selon la F-mesure

Cette figure montre l'efficacité et l'effectivité de CSRD. La méthode CS a toujours produit des résultats inférieurs en qualité par rapport à CORI et à CSRD. Pour les requêtes de type quatre termes et cinq termes, CS a été proche de CORI, alors que pour les autres type de requêtes, CORI a largement surpassé CS. A la différence des autres méthodes, CSRD a produit plus de résultats que ses adversaires. En réalité, CSRD a fourni deux fois plus de résultats que CORI, comparativement au système centralisé.

5. Conclusion

La sélection de collections est un domaine de recherche intensivement étudié. Jusqu'à nos jours, de nouveaux travaux de recherche sont publiés. Il s'agit de produire des fonctions de mesure qui tentent de répondre au mieux aux réels besoins des utilisateurs avec de grandes probabilités.

Dans ce chapitre, nous avons présenté l'environnement dans lequel nous avons implémenté les méthodes CORI, CS et CSRD ainsi que les métriques qui ont servi de moyen de comparaison entre ces méthodes. Les expérimentations ont été menées sur un dataset bien connu dans ce domaine. Les résultats obtenus ont montré que CSRD a largement dépassé ces adversaires en termes des métriques définies et en termes de la pertinence. Ces expérimentations ont bien montré que CS était plus faibles comparativement aux autres méthodes. Son grand problème réside dans sa manière prompte pour tirer des résultats. Ces expérimentations ont aussi montré que CORI est assez efficace.

Conclusion et Perspectives

L'explosion, sans précédent, du volume d'informations disponibles sous des formats hétérogènes produits par des sources d'informations distribuées, est l'un des résultats du développement de l'Internet et de l'informatique dans tous les secteurs d'activité.

Dans un contexte pareil. L'automatisation des systèmes responsable de la gestion de ces masses de données est devenue plus que nécessaire.

Notre travail se situe dans le contexte de ces outils automatisés et plus précisément dans le domaine de la Recherche d'Information (RI). La recherche d'information est une branche en informatique qui s'intéresse à l'acquisition, l'organisation, le stockage et la recherche des informations. Elle propose des outils, appelés systèmes de recherche d'information (SRI), dont l'objectif est de capitaliser un volume important d'information et d'offrir des moyens permettant de localiser les informations pertinentes relatives à un besoin en information d'un utilisateur exprimé à travers une requête.

Nous nous sommes intéressés dans le cadre de ce travail à l'information textuelle. Nous avons utilisé indifféremment les termes information ou document, pour désigner la portion de texte renvoyée à l'utilisateur.

Notre travail a porté essentiellement sur le problème de la sélection de collections. Ce sujet tient son importance au fait que les moteurs de recherche généraux sont incapable d'indexer et de traiter les collections. En réalité, les moteurs de recherche indexent les documents uniquement autant qu'entités isolées possédant leurs propres caractéristiques. Alors que l'assemblage des documents dans des collections demande l'implémentation d'autres mécanismes.

Dans cette thèse, nous avons présenté la théorie de la recherche de l'information en illustrant toutes les notions relatives à ce contexte. Nous avons présenté aussi la sélection de collections ainsi que les méthodes qui constituent son état de l'art.

Nous avons étudié principalement la pertinence des collections sélectionnées, en mettant le point sur les méthodes CS et CORI. Celles-ci définissent des paramètres dont les valeurs sont obtenues à partir des expérimentations. Ce qui compromet les résultats puisqu'il y aura un problème de standardisation de constantes. La pénalisation de certains types de collections est un autre problème dont souffrent beaucoup de méthodes.

A partir d'un ensemble de constats, nous avons proposé une nouvelle méthode de sélection de collections. Nous l'avons appelée CSRD, pour Collection Selection Based-Degree

Relevance. Son principe est d'attribuer le score le plus élevé à la collection contenant les documents ayant les degrés de pertinence les plus élevés. De ce fait, la pertinence d'une collection est étroitement liée à la pertinence des documents qu'elle contient. Cette fonction tire donc sa justification de l'anatomie de la sélection sans favoriser les petites collections, en nombre de documents et en nombre de termes, ni pénaliser les collections. Afin de mettre notre fonction en marche, nous avons proposé d'utiliser un système pair-à-pair hybride où chaque pair maintient une collection et un index sur ses termes. Les ultra-pairs maintiennent, quant à eux, des index sur les termes et les pairs qui les indexent. Cette proposition permet d'évite l'acheminement de la requête à des pairs qui non aucun lien de pertinence avec elle. Ce procédé favorise la réduction de la quantité des messages circulant dans le système.

Dans le but de comparer CSRD aux autres méthodes, nous avons utilisé un dataset bien connu. Dans les expérimentations, nous n'avons pas tenu compte des aspects de la réseautique. Le but était de les comparer selon la pertinence. Les expérimentations menées ont permis de montrer à quel point CSRD est pertinente selon des métriques bien connues.

Dans le futur, nous comptons étudier les clusters de collections (ceux-ci deviendront alors une couche sémantique qui s'implante au-dessus des collections). Ces clusters permettront d'offrir une vue sémantique d'un système pair-à-pair hybride que nous implémenterons aussi. Nous comptons aussi comparer CSRD à d'autres méthodes comme celle présentée dans [28].

Références bibliographiques

1. Abbaci, F. , Savoy, J. , Beigbeder, M. : Méthodes pour la sélection de collections dans un environnement distribué , Congrès Documents dans les systèmes d'information mobiles, pp 227 – 238, Tunisie,(2002).

2. Abbaci, F. : Méthodes de sélection de collections dans un environnement de recherche d'informations distribuée , thèse de doctorat de L'Université Jean Monnet de Saint-Etienne, (2003).

3. Aberer, K. , Hauswirth, M. : Peer-to-peer information systems: concepts and models, state-of-the-art and future systems , ACM SIGSOFT Software Engineering Notes Vol 26 Issue 5, Sept. 2001 pp 326-327 (2001)

4. Amrouche, K. : 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, (2008).

5. AMTI ,G. , Van Risjbergen , C.J. : Probabilistic models of information retrieval based on measuring the divergence from randomness, ACM Transactions on Information systems, Vol 20 n°4 october 2002, pp 357-39,(2002).

6. Anderson,D. : Chapitre 5: SETI@home, De: Peer-to-Peer: Harnessing the Power of Disruptive Technologies ,O'Reilly, (2001).

7. Anh ,V. , Moffat, A. : Inverted index compression using word-aligned binary codes, Inf. Retrieval, Volume 8, Issue 1, pp 151-166 (2005).

8. Bai, Q. , Ma, C. , Chen, X. : A New Index Model Based on Inverted Index, 3rd International Conference on Software Engineering and Service Science (ICSESS), IEEE pp 157-160 (2012).

9. Baziz , M. : Indexation conceptuelle guidée par ontologie pour la recherche d'information, thèse de doctorat , Université Paul Sabatier de Toulouse, (2005).

10. Belkin, N. ,Muresan, G. , Zhang, X. : Using User's Context for IR Personalization , Proceedings of the ACM/SIGIR Workshop on Information Retrieval in Context , pp 23-25 (2004).

11. Ben Aouicha, M. : Une approche algébrique pour la recherche d'information structurée, thèse de doctorat en informatique de l'Université Paul Sabatier de Toulouse (2009).

12. Bender, M. , Michel,S. , Weikum,G. , et Zimmer, C. : Bookmark-driven query routingin peer-to-peer web search, In: Callan, J., Fuhr, N., and Nejdl, W., WorkshopProceedings of the 27th Annual International Conference on Research and Developmentin Information Retrieval ACM SIGIR '04, pp 46–57, (2004).

13. Berrut, C. :Indexation des données multimédia, utilisation dans le cadre d'un système de recherche d'informations thèse pour obtenir le diplôme d'Habilitation à Diriger des Recherches , Université Joseph Fourier - Grenoble I, (1997).

14. Beza-Yates, R. : Modern Information Retrieval, (Livre) Addison Wesley, (1999).

15. Boubekeur-Amirouche, F. : Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets , thèse de doctorat l'Université Toulouse III - Paul Sabatier ,(2008).

16. Boughanem, M. , Loiseau, Y. , Prade , H. : Rank-ordering documents according to their relevance in information retrieval using refinements of ordered-weighted aggregations, Proc. of the 3rd International Workshop on Adaptive Multimedia Retrieval (AMR'05) , Glasgow, UK, pp 44–54, (2005).

17. Boughanem, M. , Loiseau, Y. ,Prade ,H. : Refining aggregation functions for improving document ranking in information retrieval,International Conference on Scalable Uncertainty Management (SUM 2007), Washington,DC, USA 2007:pp 255–267, (2007).

18. Boughanem, M., Brini, A. , Dubois , D. : Possibilistic networks for information retrieval, International ACM SIGIR Conference - Workshop Information retrieval and applications of graphical models, pp 67–89, (2007).

19. Bouramoul, A.: Recherche d'information contextuelle et semantique sur le web , thèse de doctorat, l'Université Mentouri ,Constantine,(2011).

20. Brini , A.H. : Un Modèle de Recherche d'Information basé sur les Réseaux Possibilistes , thèse de doctorat université de Paul Sabatier de Toulouse, (2005).

21. Brini, A. , Boughanem, M. , Dubois, D. : A model for information retrieval based on possibilistic networks, String Processing and Information Retrieval (SPIRE), pp 271–282, (2005).

22. Broder, A. Z. ,Carmel, D. , Herscovici, M. , Soffer, A. , Zien, J. Y. : Efficient query evaluation using a two-level retrieval process, In CIKM 2003. New Orleans, LA, USA, ACM 2003, pp 426-434, (2003).

23. Büttcher, S. , Clarke, C. L. A. , Cormack, G.V. : Information Retrieval: Implementing and Evaluating Search Engines, MIT Press 2010, pp 68, (2010).

24. Calabretto, S. : Recherche d'Information, thèse de doctorat, LIRIS, INSA Lyon,(2003).

25. Callan, I. P. : Distributed information retrieval, In Croft,W. B. , editor, Advances in Information Retrieval, Kluwer Academic Publishers, pp 127-150,(2000).

26. Callan, J.P. , Croft, W.B. , Harding, S.M. : The inquery retrieval system, Proceedings of the Third International Conference on Database and Expert Systems Applications, Springer-Verlag, pp 78-83 (1992).

27. Callan, J.P. , Lu, Z. ,Croft, W.B. : Searching distributed collections with inference network, Fox, E. A. , Ingwersen , P. , Fidel, R. , eds : Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval , ACM, pp 21-28, (1995).

28. Chernov , S. : Result Merging in a Peer-to-Peer Web Search,
DBISP2P'05/06 Proceedings of the 2005/2006 international conference on Databases, information systems, and peer-to-peer computing
pp 26-37

29. Chevallet, J.-P., Nie, J.Y. : Intégration des analyses du français dans la recherche d'informations , In Recherche d'Informations Assistée par Ordinateur (RIAO'97), Montreal, pp 761-772, jun (1997).

30. Claveau , V., Sébillot , P. : Vincent Claveau, Apprentissage semi-supervisé de patrons d'extraction de couples nom-verbe, TAL (traitement automatique des langues), Hermès, Vol. 45, No. 1, 2004

31. Craswell, N. E. : Methods for Distributed Information Retrieval, thesis submitted for the degree of Doctor of Philosophy at The Australian National University ,May (2000).

32. De Souza, A. F. , Pedroni, F. , Oliveira, E. , Ciarelli, P. M. , Henrique, W. F. , Veronese, L. P. , Badue , C. : Automated Multi-label Text Categorization with VG-RAM Weightless Neural Networks, Neurocomputing , v.72, pp 2209 - 2217, (2009).

33. Denos, N. : Modélisation de la pertinence en recherche d'information : modèle conceptuel, formalisation et application, thèse pour obtenir le grade de Docteur de l'Université Joseph Fourier-Grenoble I,(1997).

34. Denoyer, L. : Apprentissage et inférence statistique dans les bases de documents structurés : Application aux corpus de documents textuels ,thèse de doctorat de l'université de PARIS 6,(2004).

35. Doulkeridis, C., Norvag, K., Vazirgiannis, M., (2008). Peer-to-Peer Similarity Search over Widely Distributed Document Collections. LSDS-IR '08 Proceedings of the 2008 ACM workshop on Large-Scale distributed systems for information retrieval pp 35- 42, (2008).

36. French, J.C. , Powell, A. L. , Callan, J. , Viles, C. L. , Emmitt, T. , Prey, K. J. , Mou, Y. : Comparing the performance of database selection algorithms, Hearst, M. , Gey , F.,Tong, R. , eds, Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval, ACM, p 238–245,(1999).

37. Glory,V. , Dominic, S. : Inverted Index Compression Using Extended Golomb Code, IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012) pp 20 - 25, (2012).

38. Gravano, L. , Garcia-Molina, H. , Tomasic , A. : GlOSS: text-source discovery over the Internet , ACM Transactions on Database Systems, 24(2), pp 229-264, (1999).

39. Greengrass, Ed. : Information Retrieval, A Survey, (2000). http://www.cs.umbc.edu/cadip/readings/IR.report.120600.book.pdf

40. Haddad, H. : Utilisation des Syntagmes Nominaux dans un Système de Recherche d'Information,VTT Information Technology, pp 341-346 ,(2003).

41. Hawking, D. , Thistlewaite, P. : Methods for Information Server Selection, ACM Transactions on Information Systems, 17(1), pp 40-76,(1999).

42. Holz, F. , Witschel, H. F. , Heinrich, G. , Heyer, G. , Teresniak, S. : An Evaluation Framework for Semantic Search in P2P Networks, Proceedings of the I2CS 2007, (2007).

43. Ingwersen , P. : Poly representation of information needs and semantic entities: elements of a cognitive theory for information retrieval interaction, In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 101-110, (1994).

44. Jalam, R. , Chauchat, JH. : Pourquoi les n-grammes permettent de classer des textes? Recherche de mots-clefs pertinents à l'aide des n-grammes caractéristiques , 6th International Conference on Textual Data Statistical Analysis, France pp 381-390 (2002)

45. Jie, L. , Callan , J. : Merging retrieval results in hierarchical peer-to-peer networks, SIGIR'04: Proc of the 27th annual international conference on Research and development in information retrieval, ACM, pp 472-473,(2004).

46. Jie, L. , Callan , J. : User Modeling for Full-Text Federated Search in Peer-to-Peer Networks, SIGIR'06, pp 332-339 (2006).

47. Khalaman, S. , Kurland, O. : Utilizing Inter-Document Similarities in Federated Search, SIGIR'12, August 12–16, 2012, Portland, Oregon, USA, ACM, pp 1169-1170, (2012).

48. Kleinberg, J. M. : Authoritative Sources in a Hyperlinked Environment, Journal of the ACM, Vol. 46, No. 5, September 1999, pp 604 –632, (1999).

49. Kompaoré, N.D.Y. : Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes: vers un processus de RI adaptatif, thèse de doctorat en informatique, Université Paul Sabatier de Toulouse, (2008).

50. Kulkarni,A., Callan,J. : Selective Search: Efficient and Effective Search of Large Textual Collections. ACM Trans Volume 33 Issue 4, Article No. 17 (2015)

51. Lawrence, S. , Giles, C.L. : Inquirus, the NECI meta search engine,WWW'7, pp 95-105,(1998).

52. Lee ,J. H. : Combining the evidence of different relevance feedback methods for information retrieval, Information Processing and Management, 34(6) , pp 681-691, (1998).

53. Lewis,D. D., Yang, Y. T., Rose, G. & Li, F. RCV1: A new benchmark collection for text categorization research. *Journal of Machine LearningResearch*, 5: pp 361-397, (2004).

54. Liang, J. , Kumar, R. , Ross, K. : The FastTrack overlay: a measurement study, Computer Networks, volume 50, pp 842–858, (2006).

55. LimeWire, http://www.limewire.org.

56. Liu, X. : Efficient Maintenance Scheme of Inverted Index for Large-scale Full-Text Retrieval, 2nd International Conference on Future Computer and Communication (ICFCC), IEEE pp 565- 570 (2010).

57. Maisonnasse , L. : Les supports de vocabulaires pour les systèmes de recherche d'information orientés précision : application aux graphes pour la recherche d'information médicale, thèse de doctorat en informatique, Université Joseph Fourier- Grenoble I,(2008).

58. Manning, C. D. , Raghavan, P. , Schütze , H. : An Introduction to Information Retrieval, Cambridge University ,Press. 2008, pp 155, (2008).

59. Maron, M. E. , Kuhns, J. L. : On relevance, probabilistic indexing and information retrieval, J. ACM, 7(3), pp 216–244, (1960).

60. Mechach, K. , Zekri, L. , Abdi , M. K. : Une Nouvelle Approche pour la Sélection de Collections dans les Systèmes Distribués, In Proc of COSI'2014, Bejaia, Algeria, pp 25-27, (2014).

61. Mechach, K. , Zekri, L. , Abdi, M. K. : Collection and Selection Based Relevant Degrees Of Documents , In Journal JDIM volume 13 nombre 2, April 2015, pp 110- 119, (2015).

62. Mechach, K. , Zekri, L., Abdi, M. K. : Etude de La Pertinence lors de La Sélection de Collections dans les Systèmes Distribués , In EGC 2015, vol. RNTI-E-28, pp 489-490 , (2015).

63. Mechach, K. , Zekri, L., Abdi, M. K. : Sans paramètres extra-collections pour la sélection de collections , In PAIS 2015 , pp 1-7 , (2015).

64. Melnik, S. , Raghavan, S. ,Yang, B. , Garcia-Molina, H. : Building a Distributed Full-Text Index for the Web, Journal
ACM Transactions on Information Systems (TOIS) Volume 19 Issue 3, July 2001 pp 217-241 (2001).

65. Mercier, A. :Modélisation et prototypage d'un système de recherche d'informations basé sur la proximité des occurrences des termes de la requête dans les documents ,thèse de doctorat, l'Ecole Nationale Supérieure des Mines de Saint-Etienne , (2006).

66. Meylan, Ed. : Introduction théorique à la gestion de données textuelles, Haute Ecole Spécialisée de Suisse Occidentale, (2001).

67. Miller, G.A. : Wordnet ,A lexical database for english, In Communications of the ACM, Volume 38 Issue 11, pp 39-41 (1994).

68. Nie, J.Y. : Le domaine de recherche d'information – Un survol d'une longue histoire, Département d'informatique et recherche opérationnelle Université de Montréal, (2001).

69. Nottelman , H. , Fuhr , N. : From retrieval status value to probabilities of relevance for advanced in applications, Information retrieval, 6(3), pp 363–388, (2003).

70. Opennap, http ://opennap.sourceforge.net.

71. Page, L. , S. : The anatomy of a large-scale hypertextual web search engine, Proceedings of the Seventh International Web Conference (WWW 98), pp 107-117, (1998).

72. Paik, J.H. : A novel tf-idf weighting scheme for effective ranking ,In ACM SIGIR, pp 343-352, (2013).

73. Paik, J.H. : A Probabilistic model for information retrieval based on maximum value distribution, In ACM SIGIR ,pp 585-594, (2015).

74. Paradis, F. : Un modèle d'indexation pour les documents textuels structurés, thèse de doctorat de l'Université Joseph Fourier ,Grenoble 1, (1996).

75. Pearl, J. : Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference , Morgan Kaufman, San Mateo, CA, (1988).

76. Piwowarski, B. : Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information, thèse de doctorat de l'université de PARIS 6,(2003).

77. Pohl, S. , Moffat, A. , Zobel, J. : Efficient Extended Boolean Retrieval, IEEE Trans. Knowledge and Data Engineering, 24(6) , pp 1014-1024, June (2012).

78. Pouliquen, B. : Indexation de textes médicaux par extraction de concepts, et ses utilisations, thèse de doctorat de l'université de Rennes I, (2002).

79. Ribeiro, B. A. N. , Muntz, R. : A belief network model for ir,SIGIR, pp 253–260, (1996).

80. Ribeiro-Neto , B. : Recuperaçãon de Informaçãon, Présentation de cours,(1999).

81. Rijsbergen , C. J. : A theoritical basis for use of co-occurrence data information retrieval, Journal of Documentation, 33(2), pp 106–119, (1977).

82. Rijsbergen, C.V. : Information retrieval, In Information retrieval experiments

,Butterworths, London, 2nd edition,(1979).

83. Risson, J. ,Moors, T. : Survey of research towards robust peer-to-peer networks: Search methods, Computer Networks 50 , pp 3485–3521,(2006).

84. Robertson, S. E. , Walker , S. : Okapi/Keenbow at TREC–8, In proceedings NIST: http://trec.nist.gov/pubs/trec8/t8_proceedings.html, pp 151, 162 (2000).

85. Robertson, S. E. : THE PROBABILITY RANKING PRINCIPLE IN IR, Journal of Documentation, Vol. 33 Iss: 4, pp 294 – 304, (1977).

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

87. Robertson, S.E. ,Walker, S. , Jones,S. , Hancock-Beaulieu, M.M. , Gatford, M. : Okapi at TREC-3, The third Text REtrieval Conference (TREC-3), in Harman , D.K. ,NIST ,pp 109-126.

88. Rocchio, J.J. : Relevance feedback in information retrieval ,In The SMART retrieval system-experiments in automatic document processing, pp 313,323 , Prentice Hall Inc, (1971).

89. Salampasis, M. , Giachanou, A. , Paltoglou, G. : Multilayer Collection Selection and Search of Topically organized Patents, Proceedings of the Integrating IR technologies for Professional Search Workshop, pp 48-56, Moscow, Russia, March (2013).

90. Salton , G. , Lesk ,M. E. : Computer evaluation of indexing and text processing, J. ACM, 15(1), pp 8–36, (1968).

91. Salton, G. ,Fox, E. ,Wu , H. : Extended boolean information retrieval ,Commun. ACM, 26(11), pp 1022–1036, (1983).

92. Salton, G. : A Comparison between manual and automatic indexing methods , Journal of the American Documentation, 20(1), pp 61-71, (1971).

93. Salton, G. : The smart information retrieval system after 30 years - panel, SIGIR, pp 356–358, (1991).

94. Salton,G. , Allan, J. , Buckley, C. : Approaches to passage retrieval in full text information systems, SIGIR, pp 49–58, (1993).

95. Sauvagnat, K. : Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés, thèse en vue de l'obtention du Doctorat de l'Université Paul Sabatier,(2005).

96. Sergey, C. , Pavel, S. , Matthias, B. , Sebastian, M. , Gerhard, W. et Christian, Z. : database selection and result merging in P2P web search, G. Moro et al. (eds): DBISP2P 2005/2006, LNCS 4125, pp 26-37, (2007).

97. Sergey, C. , Serdyukov,P. , Bender ,M. , Michel , S. , Weikum, G. et Christian, Z. : Database Selection and Result Merging in P2PWeb Search, DBISP2P 2005/2006, LNCS 4125, pp 26–37, (2007).

98. Si, L. , Jie, L. ,Callan , J. : Distributed Information Retrieval With Skewed Database Size Distributions, In Proceedings of the National Conference on Digital Government Research (dg.o2003), Boston, pp 1-6, (2003).

99. Si, L. , Callan , J. : B. The Effect of Database Size Distribution on Resource Selection Algorithms, Callan , J. et al.(Eds), SIGIR 2003 Workshop on Distributed Information Retrieval, LNCS 2924, pp 31-42, (2003).

100. Si, L. , Callan , J. : A. Relevant Document Distribution Estimation Method for Resource Selection, In Proc. of the 26th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval, pp 298-305 (2003).

101. Simonnot , B. : Modélisation multi-agents d'un système de recherche d'information multimédia à forte composante vidéo, (Multi-Agent Modelling of a multimedia information retrieval system for still images and videos collections) ,Phd thesis, Henri Poincaré University,(1996).

102. Skype, http ://www.skype.com/intl/fr/home.

103. Souza, D. D'. , Zobel, J. , Thom, J. A. : Is CORI Effective for Collection Selection? An Exploration of Parameters, Queries, and Data. In Proc of the 9th Australasian Document Computing Symposium pp 41-46 (2004).

104. Stoica, I. , Morris , R. , Karger ,D. ,Kaashoek , F. , Balakrishnan , H. : Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '01, pp 149–160, New York, USA, ACM, (2001).

105. Tambellini, C. : Un système de recherche d'information adapté aux données incertaines: adaptation du modèle de langue , thèse de doctorat en informatique, Université de Nice-Sophia Antipolis-UFR sciences, (2007).

106. Tamine, L. : Optimisation de requêtes dans un système de recherche d'information approche basée sur l'exploitation de techniques avancées de l'algorithmique génétique , pp 14-28, Décembre (2000).

107. Trutle, H. , Croft , W. B. : Inference networks for document retrieval, Proc. SIGIR, pp 1–24, (1989).

108. Trutle, H. , Croft, W. B. : Evaluation of an inference network-based retrieval model, TOIS, 9(3), pp 187–222, (1991).

109. Vishwakarma, S. K. , Lakhtaria, K.I. , Bhatnagar, D. , Sharma, A.K. : An efficient approach for inverted index pruning based on document relevance, Fourth International Conference on Communication Systems and Network Technologies,

pp 487-490 (2014).

110. Wang, B. , Song, W., Lou,W. , Thomas Hou, Y. : Inverted Index Based Multi-Keyword Public-keySearchable Encryption with Strong PrivacyGuarantee, IEEE Conference on computer Communications INFOCOM Hong Kong China, pp 2092 - 2100 (2015).

111. Witschel, H.F. : Ranking Information Resources in Peer-to-Peer Text Retrieval: an Experimental Study, LSDS-IR'08, Napa Valley, California, USA, (2008).

112. Yan,H. , Ding , S. , Suel, T. : Inverted Index Compression and query processing with optimized document ordering, Madrid,Spain,(2009),

113. Yumono , B. , Lee , D. L. : Server ranking for distributed text retrieval systems on the internet, In The fifth International Confernece on Database Systems for Advanced application, pp 41-50 (1997).

114. Zadeh, L. A. : Fuzzy sets, Information and Control, 8(3), pp 338–353, (1965).

115. Zheng, G., Callan, J.: Learning to Reweight Terms with Distributed Representations. SIGIR'15, pp 575-584 (2015).

116. Zobel, J. , Moffat, A. : Inverted Files for Text Search Engines, ACM Computing Surveys, Vol. 38, No. 2, Article 6, July (2006).