THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *Institut National Polytechnique de Toulouse (INPT)*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Elayeb Bilel*
Le *26 juin 2009*

Titre : *SARIPOD: Système multi-Agent de Recherche Intelligente POssibiliste de Documents Web*

JURY
M. Jean-Paul Haton: Président de jury
M.Fabrice Evrard: Examinateur
M. Montaceur Zaghdoud: Examinateur

Ecole doctorale : *Mathématiques, Informatique, Télécommunications de Toulouse (MITT)*
Unité de recherche : *Institut de Recherche en Informatique de Toulouse (IRIT)*
Directeur(s) de Thèse : *M. Mohamed Ben Ahmed et M. Andréas Herzig*
Rapporteurs : *M. Salem Benferhat et Mme. Henda Hajjami Ben Ghezala*

A ma femme Myriam et ma petite fille Issrâ,
En témoignage de leurs respects et de mon amour…

A ma grande mère, A ma mère, mon père et ma tante Aicha
En témoignage de leurs sacrifices et de mon amour…

A Mes frères et soeurs,
En témoignage de mon amour…

Remerciements

Je suis très reconnaissant à mon directeur de thèse, le Professeur Mohamed BEN AHMED pour ses précieuses directives, ses idées scientifiques, sa disponibilité et son soutient perpétuel. Qu'il trouve ici le fruit de mes efforts comme témoignage de mon très grand respect.

Je tiens à exprimer ma profonde gratitude à Monsieur Fabrice EVRARD pour l'intérêt et la disponibilité qu'il a manifesté à l'égard de mes recherches ainsi que pour son soutien et sa patience au cours de mes nombreux séjours à Toulouse. Qu'il soit ici assuré de mon très grand respect et du plaisir que j'ai à travailler avec lui.

Je souhaite également exprimer toute ma reconnaissance à Monsieur Montaceur ZAGHDOUD pour l'intérêt et la disponibilité qu'il a manifesté à l'égard de mes recherches ainsi que pour ses encouragements continus durant ces trois années de thèse. Qu'il soit ici assuré de mon très grand respect et du plaisir que j'ai à travailler avec lui depuis mon PFE à l'ENSI.

Je suis aussi très reconnaissant à Monsieur Andréas HERZIG de m'avoir accueillie au sein de l'équipe LILAC de l'Institut de Recherche en Informatique de Toulouse (IRIT).

Mes remerciements s'adressent également à tous les membres du jury qui m'ont honoré d'avoir accepté d'évaluer ce travail. En particulier, je remercie:

Professeur Jean-Paul HATON d'avoir accepté de présider le jury de cette thèse.
Professeure Henda HAJJAMI BEN GHEZALA, la Présidente de L'Université de la Manouba en Tunisie, pour l'honneur qu'elle m'a fait en acceptant d'être le rapporteur de cette thèse.
Professeur Salem BENFERHAT pour l'honneur qu'il m'a fait en acceptant d'être le rapporteur de cette thèse.

J'adresse aussi mes remerciements au Professeur Michel DAYDE de m'avoir bien accueillie au sein du Laboratoire IRIT site ENSEEIHT, ainsi qu'au Professeur Louis FERAUD, le Directeur de l'Ecole Doctorale MITT pour ses encouragements continus.

Mes remerciements s'adressent également au Professeur Khaled GHEDIRA, l'ex-directeur de l'ENSI pour ses encouragements.

Je tiens à remercier aussi tous les enseignants de l'ENSI, particulièrement Monsieur Yassine JAMOUSSI et Madame Narjès BELLAMINE-BENSAOUD pour leur soutien et encouragements.

Remercier tous ceux qui m'ont aidé à finaliser ce travail est pour moi un plaisir, je pense particulièrement à Madame Myriam BOUNHAS et à Monsieur Ibrahim BOUNHAS. Qu'ils trouvent ici ma reconnaissance pour leurs disponibilités et encouragements.

Je tiens à remercier aussi tous les membres du laboratoire RIADI-GDL, en particulier Messieurs Tarek BEN MENA, Youssef BEN HALIMA, Riadh HADJ M'TIR et Ahmed HADAD et Mesdames Olfa DRIDI et Samia Sonia SELLMI pour leur collaboration, soutien et encouragements continus.

Je remercie aussi tous les personnels de l'IRIT site ENSEEIHT ainsi que de l'INPT, je pense particulièrement à Madame Sylvie EICHEN et à Monsieur Jean-Claude DARCOS de l'IRIT site ENSEEIHT, Mesdames Catherine GARCIA, Sylvie CARCASSES et Martine LACOSTE de

l'INPT ainsi que Madame Martine LABRUYERE, la secrétaire de l'EDMITT et Madame Christiane JOURDAA, chargée des relations internationales à l'ENSEEIHT pour leurs aides et encouragements.

Je remercie aussi tous les personnels de l'ENSI, je pense particulièrement à Madame Hédia OMRANI, la secrétaire du laboratoire RIADI-GDL, et Monsieur Rachid MZOUGHI, chargé de finance à l'ENSI, pour leurs aides et encouragements continus durant toutes mes missions scientifiques à l'étranger.

Mes remerciements vont également vers mes amis de toujours, Hédi SMIDA, Houcine SGHAIER, Abdelrahman ETTIH, Taieb ETTIH, Ramzi LAMLOUM, Mohamed KHELFA, Soufien ELMOKHTAR, Mohamed MOUTIA, Salah HADJ SALEM, Khaled BEN SMIDA, Malek BEN SMIDA, Moncef BEN SMIDA, Hichem OMRANI et Imed BALTI qui ont toujours été là pour partager les soucis, les joies et les moments de détente.

Un grand merci à la famille BOUNHAS, je pense particulièrement à Madame Aicha ELGATTOUFI et Messieurs Said, Salem, Mohamed, Ahmed et le petit Khalil (Lil !) et Mesdames Yemna, Mariem, Jazia et Khédija pour leurs aides et encouragements durant toutes les périodes de mes absences à l'étranger.

Je tiens à accorder une pensée particulière à ma petite fille Issrâ (Cha !), ma femme, mes parents, mes frères Sami et Nidhal et mes sœurs Samiha, Imen et les deux Meriems Elayeb et Handoura. Que vous soyez ici récompensés pour toutes vos écoutes et vos encouragements durant toutes mes études.

Résumé

La présente thèse de doctorat en informatique propose un modèle pour une recherche d'information intelligente possibiliste des documents Web et son implémentation. Ce modèle est à base de deux Réseaux Petits Mondes Hiérarchiques (RPMH) et d'un Réseau Possibiliste (RP) : Le premier RPMH consiste à structurer les documents retrouvés en zones denses de pages Web thématiquement liées les unes aux autres. Nous faisons ainsi apparaître des nuages denses de pages qui traitent d'un sujet et des sujets connexes (assez similaires sémantiquement) et qui répondent toutes fortement à une requête. Le second RPMH est celui qui consiste à ne pas prendre les mots-clés tels quels mais à considérer une requête comme multiple en ce sens qu'on ne cherche pas seulement le mot-clé dans les pages Web mais aussi les substantifs qui lui sont sémantiquement proches. Les Réseaux Possibilistes combinent les deux RPMH afin d'organiser les documents recherchés selon les préférences de l'utilisateur. En effet, l'originalité du modèle proposé se décline selon les trois volets suivants qui synthétisent nos contributions :

Le premier volet s'intéresse au processus itératif de la reformulation sémantique de requêtes. Cette technique est à base de relations de dépendance entre les termes de la requête. Nous évaluons notamment les proximités des mots du dictionnaire français « *Le Grand Robert* » par rapport aux termes de la requête. Ces proximités sont calculées par le biais de notre approche de recherche des composantes de sens dans un RPMH de dictionnaire de mots par application d'une méthode basée sur le dénombrement des circuits dans le réseau. En fait, l'utilisateur du système proposé choisit le nombre de mots sémantiquement proches qu'il désire ajouter à chaque terme de sa requête originelle pour construire sa requête reformulée sémantiquement. Cette dernière représente la première partie de son profil qu'il propose au système. La seconde partie de son profil est constituée des choix des coefficients de pertinence possibilistes affectés aux entités logiques des documents de la collection. Ainsi, notre système tient compte des profils dynamiques des utilisateurs au fur et à mesure que ces derniers utilisent le système. Ce dernier est caractérisé par son intelligence, son adaptativité, sa flexibilité et sa dynamicité.

Le second volet consiste à proposer des relations de dépendance entre les documents recherchés dans un cadre ordinal. Ces relations de dépendance entre ces documents traduisent les liens sémantiques ou statistiques évaluant les distributions des termes communs à des paires ou ensembles de documents. Afin de quantifier ces relations, nous nous sommes basés sur les calculs des proximités entres ces documents par application d'une méthode de dénombrement de circuits dans le RPMH de pages Web. En effet, les documents peuvent ainsi être regroupés dans des classes communes (groupes de documents thématiquement proches).

Le troisième volet concerne la définition des relations de dépendance, entre les termes de la requête et les documents recherchés, dans un cadre qualitatif. Les valeurs affectées à ces relations traduisent des ordres partiels de préférence. En fait, la théorie des possibilités offre deux cadres de travail : le cadre qualitatif ou ordinal et le cadre quantitatif. Nous avons proposé notre modèle dans un cadre ordinal. Ainsi, des préférences entre les termes de la requête se sont ajoutées à notre modèle de base. Ces préférences permettent de restituer des documents classés par préférence de pertinence. Nous avons mesuré aussi l'apport de ces facteurs de préférence dans l'augmentation des scores de pertinence des documents contenant

ces termes dans le but de pénaliser les scores de pertinence des documents ne les contenant pas.

Pour la mise en place de ce modèle nous avons choisi les systèmes multi-agents. L'avantage de l'architecture que nous proposons est qu'elle offre un cadre pour une collaboration entre les différents acteurs et la mise en œuvre de toutes les fonctionnalités du système de recherche d'information (SRI). L'architecture s'accorde parfaitement avec le caractère intelligent possibiliste et permet de bénéficier des capacités de synergie inhérente entre les différentes composantes du modèle proposé.

Dans le présent travail, nous avons donc pu mettre en exergue à travers les expérimentations effectuées l'intérêt de faire combiner les deux RPMH via un réseau possibiliste dans un SRI, ce qui permet d'enrichir le niveau d'exploration d'une collection. Ce dernier n'est pas limité aux documents mais l'étend en considérant les requêtes. En effet, la phase de reformulation sémantique de requête permet à l'utilisateur de profiter des autres documents correspondants aux termes sémantiquement proches des termes de la requête originelle. Ces documents peuvent exister dans d'autres classes des thèmes. En conséquence, une reclassification proposée par le système s'avère pertinente afin d'adapter les résultats d'une requête aux nouveaux besoins des utilisateurs.

Mots-clés : Recherche Intelligente d'Informations, Réseaux Petits Mondes Hiérarchiques, Réseaux Possibilistes, Pertinence Possibiliste, Préférences Utilisateur, Document Pertinent, Système Multi-Agent.

Abstract

This Ph.D. thesis proposes a new model for a multiagent possibilistic Web information retrieval and its implementation. This model is based on two Hierarchical Small-Worlds (HSW) Networks and a Possibilistic Networks (PN): The first HSW consists in structuring the founded documents in dense zones of Web pages which strongly depend on each other. We thus reveal dense clouds of pages which "speak" more or less about the same subject and related subjects (semantically similar) and which all strongly answer user's query. The second HSW consists in considering the query as multiple in the sense that we don't seek only the keyword in the Web pages but also its semantically close substantives. The PN generates the mixing of these two HSW in order to organize the searched documents according to user's preferences. Indeed, the originality of the suggested model is declined according to three following shutters' which synthesize our contributions:

The first shutter is interested in the iterative process of query semantic reformulation. This technique is based on relationship dependence between query's terms. We evaluate in particular the semantics proximities between the words of the French dictionary "*Le Grand Robert*" and query's terms. These proximities are calculated via our approach of research of the semantics components in the HSW of dictionary of words by application of our method of enumeration of circuits in the HSW of dictionary. In fact, the user of the suggested system chooses the number of close words that he desire to add to each word of his initial query to build his semantically reformulated query. This one represents the first part of user's profile which he proposes to the system. The second part of its profile makes up of its choices of the coefficients of relevance possibilistic of the logical entities of the documents of the collection. Thus, our system takes account of the dynamic profiles of its users progressively they use the system, which proves its intelligence, its adaptability, its flexibility and its dynamicity.

The second shutter consists in proposing relationship dependence between documents of the collection within an ordinal framework. These relationships dependence between these documents represent the semantic or statistical links evaluating the distributions of the general terms to pairs or sets of documents. In order to quantify these relationships, we are based on the calculations of the proximities between these documents by application of a method enumerating of circuits in the HSW of Web pages. Indeed, the documents can thus be clustered in common classes (groups of close documents).

The third shutter is related to the definition of the relationships dependence between query's terms and documents of the collection, within a qualitative framework. The assigned values to these relations translate preferably partial orders. In fact, possibilistic theory offers two working frameworks: the qualitative or ordinal framework and the numerical framework. We proposed our model within an ordinal framework. Thus, we add to our basic model preferences between query's terms. These preferences make it possible to restore documents classified by relevance's preference. We also measured the contribution of these preferably factors in the increase of the relevance's scores of documents containing these terms with an aim of penalizing the relevance's scores of the documents not containing them.

For the installation of this model we chose multiagent systems. The advantage of the proposed architecture is that it offers a framework for collaboration between the various actors and the implementation of all the functionalities of the information retrieval system.

Architecture agrees perfectly with the possibilistic intelligent character and makes it possible to profit from the capacities of inherent synergy in the suggested model.

We thus could put forward, through the carried out experiments, the goal of combining the two HSW via a possibilistic network in an information retrieval system, which makes it possible to enrich the exploration level of a collection. This exploration is not only limited to the documents but it extends by considering also the query. Indeed, the semantic query reformulation phase makes it possible to benefit user from other documents which contain some close terms of the initial query. These documents can exist in other topics classes. Consequently, a reclassification suggested by the system proves its relevance in order to adapt query's results to new user's needs.

Keywords: Intelligent Information Retrieval, Hierarchical Small-Worlds, Possibilistic Networks, Possibilistic Relevance, User's Preferences, Relevant Document, Multiagent System.

Table des matières

Table des figures

Table des tableaux

Introduction générale

La Recherche d'Information (RI) est une branche de l'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 au besoin d'un utilisateur exprimé à travers une requête.

En fait, un SRI est un système qui gère une collection d'informations organisées sous forme d'une représentation intermédiaire reflétant aussi fidèlement que possible le contenu des documents grâce à un processus préalable d'indexation, manuelle ou automatique. La recherche d'information désigne alors le processus qui permet, à partir d'une expression des besoins d'information d'un utilisateur, de retrouver l'ensemble des documents contenant l'information recherchée [Abbadeni et al., 1998] et ce par la mise en oeuvre d'un mécanisme d'appariement entre la requête de l'utilisateur et les documents ou plus exactement entre la représentation de la requête et la représentation des documents. La notion de document est prise ici au sens large et peut représenter une combinaison multimédia (documents hétérogènes intégrant du texte, du son, des graphiques et de la vidéo).

Afin d'effectuer une recherche pertinente, le SRI ne doit plus se contenter d'une analyse simple de la collection de documents et d'une mise en correspondance directe entre les requêtes et les documents pris de manière isolée. Dans le but d'améliorer la qualité de la recherche, des techniques plus élaborées incluant celles de reformulation et d'interaction, et tributaires du processus de recherche lui-même, sont introduites. D'une part, ces techniques sont en rapport avec la manière d'intégrer de la façon la plus efficace possible l'utilisateur dans le processus de recherche. D'autre part, ces techniques sont liées à la manière d'analyser et de représenter le contenu d'une collection en exploitant au mieux les relations qu'entretiennent les documents entre eux, les concepts du domaine entre eux ou même les descripteurs des documents entre eux

L'interaction entre l'utilisateur et le SRI permet à l'utilisateur de transmettre ses jugements en terme de pertinence, ce qui peut résoudre en partie le problème de la complexité de la requête. Grâce à ce mécanisme, il est possible au système d'acquérir des connaissances liées aux utilisateurs et de construire des profils permettant de représenter leurs centres d'intérêt, et d'effectuer un filtrage et un routage d'information. Les SRI classiques considéraient cette composante comme figée et définie a priori.

L'utilisateur peut présenter pour sa part des besoins de nature diverse (précise, exploratoire, thématique ou connotative). Le SRI doit donc présenter l'information sous plusieurs formes notamment en proposant des vues thématiques à l'aide d'un support de navigation [Kammoun-Bouzaïene, 2006] ou via des graphes de type Réseaux Petits Mondes Hiérarchiques (RPMH) [Elayeb et al., 2006].

Pour mener à bien cette recherche, plusieurs connaissances interviennent et se complètent, et des opérations interagissent dans un contexte qui évolue et qui doit s'adapter à des exigences liées aux utilisateurs ainsi qu'à la connaissance elle même.

Le cadre du Web constitue le champ d'application des SRI le plus répandu et le plus important en terme de documents disponibles vue qu'il représente aujourd'hui une source

importante d'information. Par ailleurs, contrairement aux moteurs de recherches conventionnels qui utilisent généralement des techniques d'indexation de l'information disponible sur Internet, le système proposé dans le cadre de cette thèse utilise des techniques de modélisation de la requête et des profils des utilisateurs, d'une part et de modélisation de documents de la collection d'autre part, tout en permettant à des agents logiciels d'interagir selon des stratégies coopératives pour trouver l'information qui répond aux besoins des utilisateurs.

1. Problématique de la thèse

La problématique majeure de la Recherche d'Information consiste à extraire à partir d'une collection de documents, ceux qui répondent à un besoin utilisateur en se basant souvent sur des informations pauvres provenant des requêtes des utilisateurs. Les différents modèles connus de la RI (booléen, vectoriel, probabiliste, bayésien) représentent les documents et les requêtes sous forme de listes de termes pondérés puis mesurent une valeur de pertinence (similarité vectorielle, probabilité de pertinence) en se basant sur ces termes et leurs poids. La pondération des termes est à notre sens l'élément fondamental de tous les modèles de RI actuels [Sparck-Jones, 1988] [Ribeiro-Neto et al., 1996]. Lorsqu'elle est calculée automatiquement, cette pondération est obtenue à partir de la combinaison des fréquences d'occurrences des termes dans les documents (*tf*), des fréquences d'apparition des termes dans la collection (*idf*) et de la longueur des documents (*dl*) [Salton et al., 1994] [Singhal et al., 1996ab]. Quel que soit le modèle, la réponse à une requête est une liste de documents ordonnés selon cette valeur de pertinence. Certaines approches considèrent les poids des termes comme des degrés de pertinence. Dans ces modèles, l'incomplétude de l'information, intrinsèque à la représentation vectorielle d'un document, n'est pas considérée lors de son évaluation pour une requête donnée. En réalité, on ne distingue pas entre les notions de possibilité ou de certitude lors des calculs de la pertinence. Les méthodes actuelles, relativement pauvres, utilisées pour représenter les documents (ensemble de termes et de leurs poids) ainsi que pour représenter le besoin utilisateur ne sont pas totalement compatibles avec une définition précise de la pertinence.

La présente thèse propose le développement d'un système multi-agent de recherche d'information sur Internet, baptisée "SARIPOD", combinant deux Réseaux Petits Mondes Hiérarchiques (RPMH) via des Réseaux Possibilistes (RP) :

Le premier RPMH consiste à structurer les documents retrouvés en zones denses de pages Web liées les unes aux autres. Nous faisons ainsi apparaître des nuages denses de pages qui traitent du même sujet et qui répondent à une requête. Ainsi chaque page Web serait un noeud d'un gigantesque graphe dont les arcs seraient les liens hypertextuels d'une page vers une autre. Certains calculs sur ce graphe sont à même de faire apparaître des regroupements thématiques (pages Web qui font partie du même thème). Ainsi chercher une information sur le Web ne se ferait plus au hasard. Mieux encore : une requête sous forme d'une description même approximative de ce que l'on cherche ferait aboutir dans un groupe (cluster) thématique et même sur la plus pertinente page Web de ce "cluster".

Le second RPMH est celui qui consiste à ne pas prendre les mots-clés proposé par l'utilisateur tels quels mais à considérer une requête comme multiple en ce sens qu'on ne cherche pas seulement le mot-clé dans les pages Web mais aussi les mots sémantiquement proches. En effet, il existe un arc entre deux sommets si l'un apparaît dans la définition dictionnairique de l'autre. Nous proposerons une organisation de ces termes en plusieurs "clusters" selon leurs proximités sémantiques déterminées grâce à l'étude des circuits entre les mots du dictionnaire français *« Le Grand Robert »*. Nous proposons grâce à ce RPMH une nouvelle réformulation

sémantique de la requête utilisateur. Les Réseaux Possibilistes combinent ces deux RPMH afin d'organiser les documents recherchés selon le profil de l'utilisateur.

En effet, ce système présente une nouvelle approche possibiliste pour un système de Recherche d'Information. Ce système, qui voit la Recherche d'Information comme un problème de diagnostic, traduit à l'aide de réseaux possibilistes naïfs des relations de dépendance entre les documents et les termes de la requête. Ces relations sont quantifiables par deux mesures : la possibilité et la nécessité de pertinence. La mesure de possibilité est utile pour filtrer les documents et la mesure de nécessité pour renforcer la pertinence des documents restants. Le processus de recherche restitue les documents plausiblement ou nécessairement pertinents à un utilisateur. De plus, si l'approche de base tient compte ici de l'aspect quantitatif et ne tient pas compte de la dépendance entre les termes de la requête, notre système permet de l'étendre au cadre qualitatif possibiliste, en introduisant des préférences entre les termes de la requête.

En fait, un tel modèle possibiliste devrait être capable de répondre à des propositions du type :

- Est-il plausible à un certain degré que le document d_i constitue une bonne réponse à la requête R_j?
- Est-il nécessaire, certain (dans le sens possibiliste), que le document d_i réponde à la requête R_j?
- Le document d_i est-il préférable au document d_j ou l'ensemble $\{d_i, d_j\}$ est-il préférable à l'ensemble $\{d_k, d_l\}$?

Le premier type de proposition vise à éliminer les documents faiblement plausibles de la réponse. La seconde réponse se focalise sur les documents qui seraient réellement pertinents. Le dernier type de proposition suggère que la liste ordonnée des documents en réponse à un besoin utilisateur peut être traitée d'une manière qualitative, et que des approches ordinales pourraient être utilisées dans la représentation des documents et des requêtes. La définition de la pertinence d'un document vis-à-vis d'une requête, en fonction des données dont nous disposons, est difficilement exprimable (ou traduisible) par une unique mesure de probabilité. En effet, celle-ci ne tient pas compte des caractères imprécis et vagues qui sont intrinsèques à la pertinence [Brini et Boughanem, 2003]. En réalité, une mesure de probabilité portant sur un événement et son contraire est quelque peu restrictive. Dans le modèle proposé par ces auteurs, un document contenant tous les termes de la requête constitue une réponse possiblement pertinente à la requête. Cette plausibilité doit être renforcée par une certitude provenant de la mesure de nécessité. La mesure de possibilité est utile pour éliminer les documents non pertinents et la mesure de nécessité pour renforcer la pertinence des documents non éliminés par la possibilité. L'usage de la théorie des possibilités en RI avait déjà été suggéré par [Prade et Testemale, 1987] qui proposaient un nouveau modèle d'indexation sous forme de groupes de mots-clés, pondérés par des degrés de possibilité et de nécessité.

Afin de combler la complexité du problème de RI, faisant intervenir des processus qui interagissent via un ensemble de connaissances, nous proposons un modèle qui se base sur une architecture multi-agent contribuant à la résolution du problématique posé. En fait, le modèle multi-agent que nous proposons permet d'inclure un certain nombre de connaissances nécessaires, fédérées par un ensemble d'agents (dont chacun est chargé d'une tâche spécifique) qui coopèrent pour satisfaire l'utilisateur.

2. Organisation de la thèse

La présente thèse est organisée en six chapitres :

Le premier chapitre présente les systèmes de Recherche d'Information. Ainsi, nous commençons par définir les notions de base de la RI. Puis, nous intéressons à la représentation des documents et de la requête et à la mise en correspondance entre la requête et les documents afin de sélectionner les documents pertinents. D'autre part, une phase de reformulation de la requête est associée au processus de la recherche dont le but est de combler le fossé existant entre la pertinence liée à l'évaluation de l'utilisateur et la pertinence jugée par le système. Nous présentons à la fin de ce chapitre les techniques utilisées pour l'évaluation des SRI.

Le second chapitre s'intéresse aux modèles de Recherche d'Information (RI). Nous étudions les modèles les plus connus de la RI. Nous nous intéressons particulièrement au sens de la pertinence donné par ces modèles. Nous nous sommes focalisés sur les approches proposées par ces modèles pour modéliser la requête utilisateur et les documents. Ces modèles sont discutés tout en identifiant leurs avantages et inconvénients dans la perspectives d'introduire des modèles capables de pallier ces limites.

Dans ce cadre, nous présentons dans le troisième chapitre une étude comparative entre les modèles de la RI à base de Réseaux Bayésiens (RBs) et ceux à base de Réseaux Possibilistes. Nous commençons par rappeler les définitions des Réseaux Bayésiens et leur utilité. Nous décrivons ensuite l'utilisation des RBs dans la RI. Nous présentons les Réseaux Possibilistes ainsi que leur application dans un cadre quantitatif de RI. Nous clôturons ce chapitre par un bilan comparatif de ces deux approches dont nous nous sommes inspirés de modèle.

Suite aux limites des systèmes existants identifiés dans les chapitres précédents, nous proposons dans un quatrième chapitre un modèle d'un SRI à base de Réseaux Petits Mondes Hiérarchiques (RPMH) et de Réseaux Possibilistes (RP). Un premier RPMH pour les mots du dictionnaire français « *Le Grand Robert* » est utilisé pour dégager les proximités entre les mots de la langue française. Le second RPMH est consacré aux pages Web recherchées et traduit de même les proximités entre ces pages. En fait, le modèle proposé détient son originalité du fait qu'il propose une nouvelle modélisation de la requête à base d'une reformulation sémantique ainsi qu'une nouvelle modélisation des documents permettant une classification à base des « petits mondes ». Les RP combinent les deux RPMH afin de proposer une nouvelle approche possibiliste qualitative pour la RI. Cette approche définie d'une nouvelle manière les deux notions de base dans un SRI : la pertinence et le profil.

Dans le cinquième chapitre nous proposons de mettre en place les différentes composantes du modèle proposé dans une architecture, baptisée : **S**ystème multi-**A**gent de **R**echerche **I**ntelligente **PO**ssibiliste de **D**ocuments Web (SARIPOD). Dans cette architecture, nous choisissons de mettre en place des modules qui sont dédiés à des tâches différentes qui sont complémentaires pour certaines et concurrentes pour d'autres. Par ailleurs, certaines tâches sont coordonnées en parallèle et d'autres sont séquentielles. Une spécification complète du système SARIPOD qui décrit les différents composants de son architecture est présentée. Enfin, une conception orientée-objet UML de ce système est exposée.

Finalement, le sixième chapitre concerne l'implantation informatique du système SARIPOD. Nous proposons un ensemble d'agents coopératifs assurant le parallèlisme de traitement exigé par le système. Ensuite, les outils de sa réalisation sont présentés avec des extraits de résultats d'expérimentations.

En guise de conclusion, nous dressons un bilan de nos travaux, en mettant en exergue nos propositions, nous finissons par la proposition de nombreuses perspectives possibles à ces travaux.

Première Partie :

Etat de l'art sur la Recherche d'Information

Chapitre 1

Les Systèmes de Recherche d'Information

Un Système de Recherche d'Information (SRI) est un système qui permet de retrouver les documents pertinents à une requête d'utilisateur, à partir d'une base de documents volumineuse. Le processus de recherche d'information pertinente que le SRI est sensé restituer à un utilisateur, consiste en la mise en correspondance des représentations des informations contenues dans un fond documentaire et des besoins de cet utilisateur exprimés par une requête. En fait, l'objectif de l'utilisateur est de compléter son état de connaissance par l'acquisition d'informations contenues dans des documents pertinents.

Dans la définition d'un SRI, il y a trois notions clés: documents, requête et pertinence. En effet, un document peut être un texte, un morceau de texte, une page Web, une image, une bande vidéo, etc. On appelle document toute unité qui peut constituer une réponse à une requête d'utilisateur. Une requête exprime une interprétation du besoin d'information d'un utilisateur. Le but de la RI est de trouver seulement les documents pertinents. La notion de pertinence est très complexe. De façon générale, dans un document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin. C'est sur cette notion de pertinence que le système doit juger si un document doit être donné à l'utilisateur comme réponse. Cette notion de pertinence peut être appréhendée à deux niveaux : Au *niveau utilisateur*, ce dernier a un besoin d'information dans sa tête, et il espère obtenir les documents pertinents pour répondre à ce besoin. La relation entre le besoin d'information et les documents attendus est la relation de pertinence (idéale, absolue, …). Au *niveau système*, ce dernier répond à la requête formulée par l'utilisateur par un ensemble de documents trouvés dans la base de documents qu'il possède [Cleverdon, 1960] [Cleverdon, 1970] [Cleverdon, 1977].

Nous nous intéressons particulièrement dans cette thèse à la pertinence utilisateur que nous désignerons par pertinence. Les modèles de RI définis dans la littérature (détaillés dans la suite de cette thèse) 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), une probabilité de pertinence des documents étant donnée une requête (modèle probabiliste).

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

Nous détaillons dans la première section de ce chapitre les composants d'un système de recherche d'information. Nous présentons dans la deuxième section l'utilisateur, son besoin en information, son profil et sa requête. Dans la troisième section, nous nous intéressons à la phase d'analyse et d'indexation des documents et des requêtes. La notion de pertinence est présentée dans la quatrième section. La phase d'une évaluation d'un SRI fera l'objet de la cinquième section. Dans la dernière section, nous mettons l'accent sur la phase de reformulation de la requête.

1. Les composants d'un SRI

Un système de Recherche d'Information est composé de différents acteurs tels que : la requête ou besoin d'information d'un utilisateur, le corpus documentaire, ainsi que les différents étapes qui permettent d'aboutir à résultat répondant au besoin de l'utilisateur. Ces étapes sont : l'analyse et l'indexation, les modélisations de la requête et des documents, la mise en correspondance entre ces deux modèles (de requête et des documents) et l'évaluation et la rétroaction (voir figure 1.1).

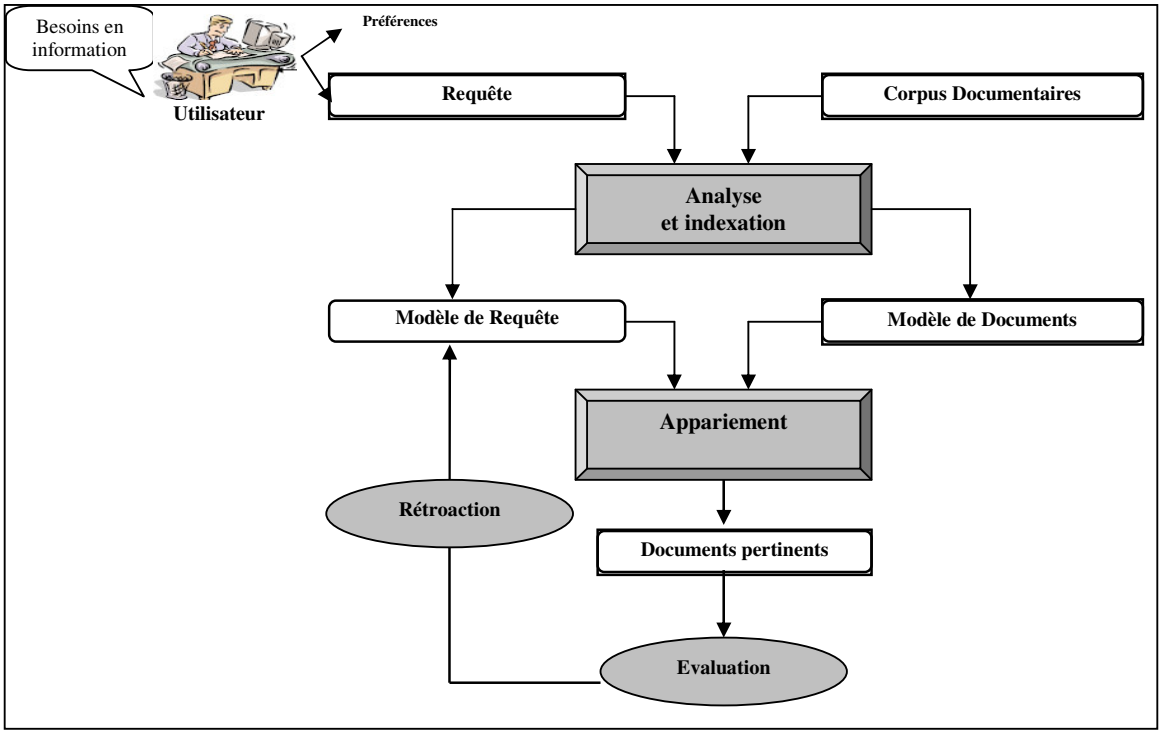


Figure 1.1 : Les composants d'un Système de Recherche d'Information

Nous détaillons dans la suite ces différents acteurs et étapes d'un SRI.

2. Utilisateur, besoin d'information, profil et requête

Dans les années 1980, le paradigme de la recherche d'information a commencé à s'élargir pour inclure les utilisateurs et leur intéraction avec le système. Il s'agit de *paradigme cognitif orienté-utilisateur* introduit par [Ingwersen, 1992]. Les systèmes d'information sont alors considérés comme des systèmes de communication entre un producteur d'information (l'auteur) et un utilisateur, le système informatique ayant pour objectif de faciliter cette communication.

Pour satisfaire au mieux l'utilisateur, il est essentiel de comprendre ses mécanismes cognitifs. Il est donc essentiel de le modéliser. Dans la majorité des travaux qui se sont intéressés à l'utilisateur et sa modélisation dans un processus de recherche d'information on introduit le terme usager pour dire utilisateur.

En fait, les travaux liés à la RI modélisent le comportement de l'usager, mais ne permettent pas une compréhension de son système cognitif (domaine de la psychologie cognitive).

Selon [Daniels, 1986], deux classes de modèles d'usagers ont été proposées :

- Les *modèles analytiques* qui modélisent le comportement interne de l'usager : connaissances, processus cognitif, etc.
- Les *modèles quantitatifs et empiriques* qui modélisent le comportement externe de l'usager.

Ces modèles ont été classés suivant trois dimensions :

- Le *modèle canonique* opposé à une collection de modèles d'utilisateurs individuels.
- Le *modèle explicite* construit par l'utilisateur ou le concepteur du système qu'on oppose au modèle implicite construit par l'ordinateur sur la base du comportement de l'utilisateur.
- Le *modèle basé sur des caractéristiques persistantes* (à long terme) à l'opposé du modèle basé sur des caractéristiques ponctuelles (à court terme).

[Daniels, 1986] propose de modéliser l'utilisateur avec les paramètres suivants :

- *USER :* correspond au statut de l'usager.
- *UGOAL :* correspond aux buts de l'usager (ses préférences ou stratégies de recherche).
- *KNOW :* définit le niveau d'expertise ou le degré de connaisssance de l'utilisateur dans le domaine.
- *IRS :* définit la familiarité de l'usager avec les systèmes documentaires.
- *BACK :* correspond à l'expérience de l'usager vis-à-vis du système concerné.

Ces différents éléments peuvent être regroupés dans un *profil utilisateur*. Ils appartiennent au système cognitif de l'utilisateur et permettent d'effectuer un filtrage initial sur les documents et de choisir des présentations personnalisées, adaptées au profil.

Selon [Cluzeau-Ciry, 1988], quatre catégories de demande ou stratégies de recherche ont été proposées :

- Une *demande précise* exprimée lorsque l'utilisateur sait exactement ce qu'il cherche.
- Une *demande thématique* utilisée lorsque l'utilisateur cherche à explorer le corpus sur un thème particulier.
- Une *demande connotative* exprimée dans le contexte de la recherche d'image par l'expression d'un visage par exemple, par métaphore dans le contexte de la recherche textuelle.
- Une *demande exploratoire* définit quand l'utilisateur veut se faire une idée du contenu du corpus ; et c'est après une consultation préalable que seront définis plus précisément ses besoins.

2.1 Requête en RI

Une requête désigne une interrogation d'une base d'informations, portant sur les éléments qu'elle contient. Une requête peut être exprimée de différentes manières :

- En *langage naturel* en utilisant des mots non-contrôlés ;
- En utilisant des *phrases courtes en langage naturel* ;
- Sous forme de *textes ou de documents en langage naturel*. On qualifie ceci par la requête par l'exemple ou par similarité (*QBE : Query By Example*) ;

- Sous forme de *grille ou formulaire* sur les champs de catalogag[1] ou plus généralement sur des champs issus d'une structure logique.

En recherche d'information, deux types de requêtes ont été utilisés : une requête vectorielle exprimée à travers des termes pondérés et une requête booléenne exprimée via des termes connectés par des opérateurs booléens. Ces deux types des requêtes seront détaillés dans le chapitre suivant.

D'autre part, vue que les requêtes sont parfois complexes, il est conseillé de les sauvegarder pour des réutilisations ultérieures. [Kammoun-Bouzaïene, 2006] a introduit donc en plus du *profil utilisateur* le *profil de requêtes*. Cette dernière perspective est appliquée dans le domaine de la diffusion sélective de l'information ou la diffusion ciblée. En fait, les profils de requêtes seront utilisés pour scruter systématiquement et en temps réel les nouvelles informations entrées dans la base pour les diffuser aux utilisateurs concernés. De notre part, nous proposons un SRI qui tient compte des requêtes déjà joué par le système. En effet, l'enregistrement de ces requêtes ainsi que leurs réponses retournées par le système dans une base d'historique servira d'avantage pour des réutilisations ultérieures. Nous détaillons le processus de gestion de l'historique dans le système SARIPOD dans le chapitre 5.

2.2 Représentation des résulats de requêtes

La majorité des SRI permettent de restituer les documents en les classant par ordre de pertinence décroissante par rapport à la requête. Ceci permet aux utilisateurs de ne s'intéresser qu'à ceux qui ont un score de pertinence supérieur à un certain seuil. La notion de pertinence d'un document vis-à-vis une requête sera détaillée dans la section 4 de ce chapitre.

Cette restitution des documents par rapport à une requête peut se faire sous plusieurs formes :

- Soit en proposant des *résumés automatiques* du document restitué, dont la taille est variable. Ces résumés sont construits en attribuant une importance aux phrases qui contiennent les termes de la requête afin qu'ils soient adaptés aux sujets de recherche de chaque utilisateur [Tombros et Sanderson, 1998].
- Soit sous forme d'une *liste de titres ou de passages* qui contiennent les termes de la requête mais qu'il n'est pas envisageable de présenter le document dans son intégralité sauf s'il est suffisamment court.

Dans d'autres systèmes, plutôt que d'interroger le SRI par le biais d'une requête exprimée de l'une des manières décrites ci-dessus, l'intérrogation est basée sur la visualisation globale de l'ensemble des documents du corpus et sur des outils qui permettent d'exploiter cet ensemble en utilisant notamment une approche classificatoire, ou encore la navigation à travers une carte explicitant ces classes et les différentes relations qui peuvent les liers.

D'autre modalité de restitution se fait sous forme graphique :

- Soit par une *représentation graphique globale*, issue généralement des méthodes de classification et particulièrement des cartes auto-organisatrices de Kohonen. Parmi ces systèmes citons : NEURODOC [Lelu et François, 1992], WEBSOM [Kohonen et al., 1996], MULTISOM [Lamirel, 1995] [François et al., 2003].

[1] Le *catalogage* correspond à l'identification des références de chaque document (nom d'auteurs, titre, éditeur, nom de revue, date, etc.) et à la saisie dans une notice documentaire ou *FID* (*Fiche d'Identification du Document*). Pour un livre on parle de fiche ou de notice bibliographique. La structuration se fait habituellement en utilisant les normes *MARC* (*Machine Readable Cataloging*) et *UNIMARC* (*UNIversal MARC*).

- Soit par une *représentation graphique individuelle* permettant de représenter les documents et éventuellement les liens qui existent entre eux. Cette méthode est peu intéressante quand la taille du corpus augmente. Parmi ces systèmes citons : AIR [Belew, 1989], WWWD [Snowdon et al., 1996], TETRALOGIE [Mothe et Dkaki, 1998].

Cette diversité dans la manière de présenter les résultats des requêtes permet aux utilisateurs :

- de donner une aperçu sur le contenu de documents afin d'éviter un accès direct ;
- de repérer les documents pertinents en montrant dans quel contexte sont utilisés les termes présents dans la requête.

3. Analyse et indexation des documents et des requêtes

L'objectif de l'analyse et de l'indexation est de d'abord trouver des concepts les plus importants dans le document, et de créer une représentation interne en utilisant ces concepts (intensions). Pour trouver des concepts, il est nécessaire de procéder une analyse sémantique pour déterminer ce qui est un concept dans un texte. Cette analyse n'est pas disponible pour la RI. Les techniques existantes sont souvent restreintes à un domaine très spécialisé, et l'analyse est très complexe.

Ainsi, en pratique, on cherche plutôt des *représentants* (instances ou extensions) des concepts. Ces représentants peuvent être de forme différentes: des mots simples, des termes (éventuellement composés), ou des doublets de mots (groupes de deux mots). En fait, le choix de représentants dépend de deux critères essentiellement: la facilité de traitement; la précision de représentation de sens.

Étant donné le grand nombre de documents à traiter, il est nécessaire que le traitement pour la reconnaissance des représentants soit plus faisable. Cependant, les représentants trouvés doivent permettre à décrire le contenu (la sémantique) du document et de la requête de façon assez précise.

L'idée d'utiliser des mots comme des représentants de concepts est assez naturelle. En effet, les mots sont des unités linguistiques qui sont les plus faciles à reconnaître, et qu'elles sont assez porteuses de sens. Ce sont ces unités qu'on utilise le plus souvent dans les systèmes actuels.

Cependant, les mots ne donnent pas une description toujours très précise. Par exemple, le concept de "recherche d'information", une fois représenté par les mots "recherche" et "information", perd beaucoup de sens, car les mots "recherche" et "information" sont très courants en français, et ils sont des sens très imprécis. Ainsi, les chercheurs ont aussi proposé des approches visant à regrouper des mots pour former des termes composés. Ces approches utilisent soit une analyse syntaxique et/ou statistique, soit un dictionnaire de termes composés, soit une terminologie (vocabulaire contrôlé, taxonomies, thésaurus), soit une ontologie (modèle de représentation des connaissances). Nous allons considérer des mots comme des représentants de concept. Ces représentants sont aussi appelés des index, en rapport avec leur rôle qu'ils joueront dans la recherche.

3.1 Approche basée sur la fréquence d'occurrences

L'objectif ici est de trouver les mots qui représentent le mieux le contenu d'un document. On admet généralement qu'un mot qui apparaît souvent dans un texte représente un concept important. Ainsi, la première approche consiste à choisir les mots représentants selon leur

fréquence d'occurrence. La façon la plus simple consiste à définir un seuil sur la fréquence : si la fréquence d'occurrence d'un mot dépasse ce seuil, alors il est considéré important pour le document.

Cependant, les statistiques des occurrences montrent que les mots les plus fréquents sont des mots fonctionnels (ou mots outils, mots vides). En français, les mots "de", "un", "les", etc. sont les plus fréquents. En anglais, ce sont "of", "the", etc. Ce phénomène n'est pas étrange si on connaît la loi de Zipf [Zipf, 1949] qui stipule que :

« Si on classe les mots dans l'ordre décroissant de leur fréquence, et on leur donne un numéro de rang (1, 2, …), alors : *Rang * fréquence ≈ constante* ».

Voyons un exemple en anglais :

Rang	Mot	Fréquence	Rang* Fréquence
1	the	69 971	69 971
2	of	36 411	72 822
3	and	28 852	86 556
4	to	26 149	104 596
5	a	23 237	116 185
6	in	21 341	128 046
7	that	10 595	76 165

Tableau 1.1 : Exemple de calcul de la fréquence d'occurrences

Il devient évident que nous ne pouvons pas garder tous les mots les plus fréquents comme des index d'un document. En restant dans la même lignée, un autre seuil maximal a été défini. En effet, si la fréquence d'un mot dans le document dépasse ce seuil, alors il n'est pas considéré comme index de ce document.

L'utilisation de ces deux seuils correspond à ce qu'on croit sur l'informativité de mot. L'informativité mesure la quantité de sens qu'un mot porte. Cette notion n'est pas définie très précisément dans la RI. Elle est utilisée seulement de façon intuitive. Cependant, on peut trouver son équivalent dans la théorie de l'information (par exemple, la théorie de Shannon, ou l'entropie)

La correspondance entre l'informativité et la fréquence est illustrée dans la figure 1.2:

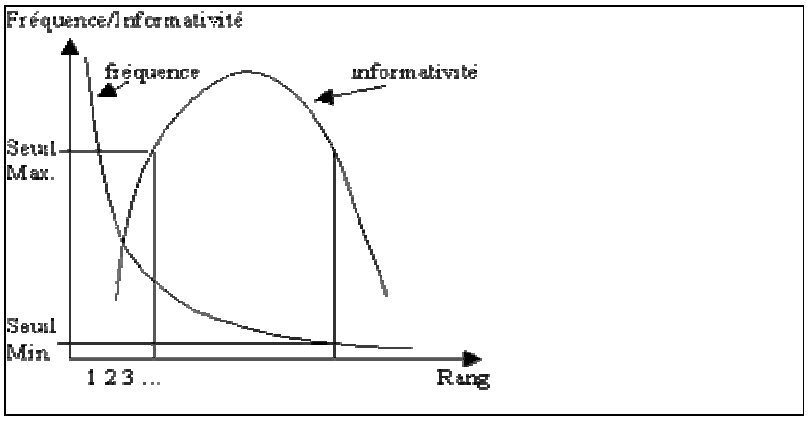


Figure 1.2 : La correspondance entre l'informativité et la fréquence

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

3.2 Approche basée sur la valeur de discrimination

Par "discrimination", on se réfère au fait qu'un terme distingue bien un document des autres documents. C'est-à-dire, un terme qui a une valeur de discrimination élevée doit être apparaître seulement dans un petit nombre de documents. Un terme qui apparaît dans tous les documents n'est pas discriminant.

Le pouvoir de discrimination d'un terme est important dans le choix de termes index qu'on veut garder. L'idée est de garder seulement les termes discriminants, et éliminer ceux qui ne le sont pas.

Le calcul de la valeur de discrimination a été développé dans le modèle vectoriel. Ainsi, nous nous situons dans ce modèle. Une description plus détaillée du modèle sera donnée dans le prochain chapitre.

Dans le modèle vectoriel, chaque document est représenté par un vecteur de poids comme suit:

$$\begin{array}{cccccc} & t_1 & t_2 & t_3 & \ldots & t_n \\ d_i \rightarrow & <p_{i1} & p_{i2} & p_{i3} & \ldots & p_{in}> \end{array}$$

Où p_{ij} est le poids du terme t_j dans le document d_i.

Étant donné un corpus (un ensemble de documents), on a donc une matrice. Pour calculer la valeur de discrimination d'un terme, on doit comparer une sorte d'uniformité au sein du corpus avec celle du corpus transformé dans lequel le terme en question a été uniformisé (mis au même poids). L'idée est que, si on uniformisant le poids d'un terme dans tous les documents, on obtient une grande amélioration dans l'uniformité du corpus, ce terme était donc très différent (non uniformément distribué) dans différents documents. Il a donc une grande valeur de discrimination. En revanche, si en uniformisant le poids du terme, on n'obtient pas beaucoup d'amélioration sur l'uniformité, ce terme était donc déjà distribué de façon uniforme, donc peu discriminant.

Le calcul de la valeur de discrimination d'un terme se fait comme suit:

1. On calcule d'abord le vecteur centroïde (ou le vecteur moyen) du corpus comme suit:

   Pour chaque terme, son poids dans le vecteur centroïde V est le poids moyen de ses poids dans les documents. C'est-à-dire:

   $$p_j = \sum_i p_{ij} / N \qquad (1.1)$$

   où N est le nombre de documents dans le corpus.

2. On calcule l'uniformité du corpus comme la similarité moyenne des documents avec le centroïde:

   $$U_1 = C * \sum_j Sim(d_i, V) \qquad (1.2)$$

   où C est une constante de normalisation (par exemple $1/N$), et $Sim(d_i, V)$ est la similarité entre le document d_i et le vecteur centroïde V. Ici, Sim doit être une formule normalisée qui donne une valeur dans [0,1] (voir la description sur le modèle vectoriel dans le chapitre suivant).

3. On uniformise le poids du terme en question à 0, et on répète les deux étapes ci-dessus pour obtenir une nouvelle valeur d'uniformité U_2.

4. La valeur de discrimination du terme est :

$$V = U_2 - U_1. \quad (1.3)$$

Dans ce calcul de la discrimination, on ne préoccupe pas beaucoup de la fréquence d'un terme dans un document particulier, mais beaucoup plus à sa distribution dans le corpus. En utilisant la valeur de discrimination, on peut éliminer les mots fonctionnels comme "de", "à", etc. qui apparaissent dans tous les documents en langue française.

3.3 Approche basée sur *tf* x *idf*

Le nom *tf* x *idf* est très connu dans le milieu de la RI. Cela désigne un ensemble de schémas de pondération (et de sélection) de termes. *tf* signifie "term frequency" et *idf* "inverted document frequency". Par *tf*, on désigne une mesure qui a rapport à l'importance d'un terme pour un document. En général, cette valeur est déterminée par la fréquence du terme dans le document. Par *idf*, on mesure si le terme est discriminant (ou non-uniformément distribué). Ici, on donne quelques formules de *tf* et d'*idf* souvent utilisées.

1. *tf* = fréquence d'occurrence du terme dans un document $f(t, d)$;

   $tf = f(t,d) / Max[f(t, d)]$ où $Max[f(t,d)]$ est la fréquence maximale des termes dans d;

   $tf = log(f(t, d))$;

   $tf = log(f(t, d) + 1)$;

2. $idf = log(N/n)$ où N est le nombre de documents dans le corpus, et n ceux qui contient le terme.

3. Finalement, on peut aussi imposer certaine normalisation sur les valeurs calculées.

Une formule de *tf* x *idf* est donc la multiplication d'une *tf* par une *idf*. Par exemple:

$$tf \times idf = [f(t, d) / Max[f(t, d)]] * log(N/n)$$

Une formule *tf* x *idf* combine les deux critères ci-dessus :

1. L'importance du terme pour un document (par *tf*),
2. Le pouvoir de discrimination de ce terme (par *idf*).

Ainsi, un terme qui a une valeur de *tf* x *idf* élevée doit être à la fois important dans ce document, et aussi il doit apparaître peu dans les autres documents. C'est le cas où un terme correspond à une caractéristique importante et unique d'un document.

Avec une telle formule, on peut donc choisir à garder seulement les termes dont la valeur de *tf* x *idf* dépasse certain seuil.

3.4 La pondération de termes

La pondération qu'un terme possède peut aussi être de diverses natures. Elle peut être simplement la fréquence d'occurrence, ou bien une mesure dérivant de cette fréquence (par exemple, normalisée). Elle peut être également une formule de *tf* x *idf*. Des comparaisons ont montré qu'en utilisant seulement la fréquence d'occurrence ne donne pas une performance satisfaisante (même si on élimine les mots fonctionnels d'une certaine façon). En général, les formules de *tf* x *idf* donnent de meilleures performances.

En réalité, si on utilise la valeur de *tf* x *idf* pour filtrer les termes index, on peut utiliser la même valeur de *tf* x *idf* comme la pondération de terme. C'est de cette manière qu'on procède généralement. Donc, le filtrage et la pondération ne sont pas deux processus nécessairement séparés.

3.5 Filtrage des mots fonctionnels

Certains mots fonctionnels, comme le mot "auparavant", "ès", etc. n'apparaissent pas très souvent dans des textes. Par le calcul de valeur de discrimination ou par l'utilisation de *idf*, on n'arrive pas nécessairement à les éliminer. Or, on ne veut pas les garder comme index parce qu'ils sont vides de sens.

Afin d'éliminer ces mots de force, on utilise une liste, appelée stoplist (ou parfois anti-dictionnaire) qui contient tous les mots qu'on ne veut pas garder. Ces mots sont souvent des prépositions (e.g. "de", "à"), prénom ("aucun", "tout", "on"), certains adverbes ("ailleurs", "maintenant"), adjectifs ("certain", "possible"), etc.

Certains mots inclus dans cette liste ne sont pas nécessairement vides de sens (ça dépend du domaine. Ils ne sont pas vides de sens en linguistiques). Mais leur sens importe très peu pour des besoins de RI.

La liste utilisée dans un système peut aussi varier. Cela dépend su domaine d'application. Par exemple, le mot "article" est inclus dans certains systèmes comme mot vide parce qu'on reçoit beaucoup de requête d'utilisateur qui contient le mot "papier", comme "des papier sur l'informatiques". Cependant, ce mot peut être très significatif dans certaines applications (par exemple, pour une base de documents en papeterie).

Le traitement lié à une stoplist est très simple. Quand on rencontre un mot dans un texte, on doit d'abord examiner s'il apparaît dans cette liste. Si oui, on ne le considère pas comme un index.

3.6 Lemmatisation

Nous remarquons que plusieurs mots ont des formes légèrement différentes, mais leur sens restent le même ou très similaire. C'est notamment le cas des mots conjugués. Par exemple, les mots transformer, transforme, transforment, transformation, transformateur, …ont des sens très similaires.

La différence de forme entre ces mots n'est pas utile à considérer pour la RI. Au contraire, on voudrait trouver des documents sur "transformation" à partir d'une requête sur "transformer". Ainsi, il faut éliminer ces différences non-significatices, c'est-à-dire de ramener ces mots à une forme identique.

Ces mots ont la même racine (lemme). Ainsi, on arrive à éliminer les terminaisons de mots, et garder seulement la racine, on a donc une forme identique pour eux. C'est l'idée qui conduit à utiliser la lemmatisation.

Il existe plusieurs façons de lemmatiser des mots.

1. Une première façon consiste à examiner seulement la forme de mot, et selon la forme, on essaie de déduire ce qui est la racine. C'est cette approche que Porter utilise dans [Porter, 1980]. En effet, cet algorithme élimine les terminaisons de mot en anglais en 5 grandes étapes: la première étape essaie de transformer le pluriel en singulier. Les étapes subséquentes essaient d'éliminer au fur et à mesure les dérivations (e.g. -ness qu'on ajoute derrière certains adjectifs (happiness), -able ajouté derrière un verbe (adjustable)).

   Cet algorithme transforme parfois deux mots différents en une même forme. Par exemple en anglais, *derivate*/*derive*, *activate*/*active*. Cependant, pour la plupart, la transformation semble raisonnable.

Porter a comparé son algorithme avec un autre disponible à l'époque qui utilisait un algorithme beaucoup plus compliqué. Il s'est avéré que cet algorithme simple fonctionne mieux pour la RI. Maintenant, cet algorithme est considéré comme un algorithme classique. La plupart de procédures de lemmatisation l'utilise, ou utilise une variante.

2. On peut aussi utiliser un dictionnaire dans la lemmatisation. Pour savoir si une séquence de lettres à la fin correspond à une terminaison d'un mot, il suffit de faire une élimination ou une transformation tentative, et de voir si la forme obtenue existe dans le dictionnaire. Sinon, ce n'est pas une terminaison correcte, et d'autres possibilités sont ensuite envisagées. Par exemple, on peut accepter la règle qui remplace -ation par -er. Par exemple, transform-ation, élimin-ation, etc. Cependant, pour "vocation", si on applique cette règle, on obtiendra "vocer". Ce n'est pas une transformation correcte. Pour éviter cela, on peut vérifier dans le dictionnaire si le mot "vocer" existe. Sinon, on ne le transforme pas. Cette approche a été utilisée pour le français dans [Savoy, 1993].

   L'utilisation d'un dictionnaire ajoute certains avantages, mais elle est au prix de disposer d'un dictionnaire. La plupart de systèmes de RI n'en disposent pas, et un tel dictionnaire électronique n'était pas encore peu accessible.

3. Une lemmatisation correcte requièrt souvent une reconnaissance correcte de catégorie grammaticale. Ainsi, on peut penser à utiliser un taggeur (ou un analyseur de catégorie) automatique dans un processus de lemmatisation. Plusieurs méthodes de taggages ont été proposées. Une des approches possibles est de déterminer la catégorie d'un mot de façon probabiliste. Pour cela, il faut d'abord qu'on entraîne un modèle probabiliste en utilisant un ensemble de textes catégorisés manuellement (le corpus d'entraînement). Ce modèle détermine la probabilité d'un mot d'être dans une catégorie selon sa forme, et selon les mots qui l'entourent.

   Avec ce mécanisme de reconnaissance de catégorie, on peut se permettre de transformer une forme de mot en une forme standard - la forme de citation (par exemple, nom singulier, adjectif masculin singulier), au lieu de couper simplement la terminaison.

3.7 L'approche basée sur une indexation

Durant l'indexation, on doit transformer les mots (lemmatisation), sélectionner un ensemble d'index et les quantifier. Le résultat d'une indexation est donc un ensemble de *termes* qui peut être constitué soit d'un mot, soit d'une racine de mot, soit d'un terme composé si on possède un mécanisme pour reconnaître des termes composés.

$d \rightarrow \{ \ldots (t_i, p_i), \ldots \}$, où t_i est un terme, et p_i est son poids.

Cet ensemble de termes pondérés sera utilisé pour constituer une représentation du contenu du document. En fait, l'organisation de ces termes en une représentation dépend du modèle de RI utilisé. Ainsi, dans différents modèles, le même ensemble de termes aura une signification différente.

Une requête peut être maintenant une expression plus complexe, incluant des opérateurs logiques (ET, OU, …) ou d'autres types d'opérateurs. L'évaluation est compositionnelle, c'est-à-dire, on commence par évaluer les éléments de base (par exemple, des mots) dans la requête, obtenant ainsi des listes de documents; ensuite, on combine ces listes selon l'opérateur qui relie ces éléments pour obtenir finalement une seule liste de documents.

En fait, cette approche possède les avantages suivants :

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

Le prix à payer pour ces avantages est le besoin de l'espace de stockage supplémentaire pour la structure d'index. En général, cet espace correspond à 40% à 200% de la taille de collection de documents, selon la complexité de l'indexation. Mais ce besoin d'espace pose de moins en moins de problème maintenant.

Utilisant cette approche, on peut voir les opérations et l'environnement de la RI comme l'indique la figure 1.3 :

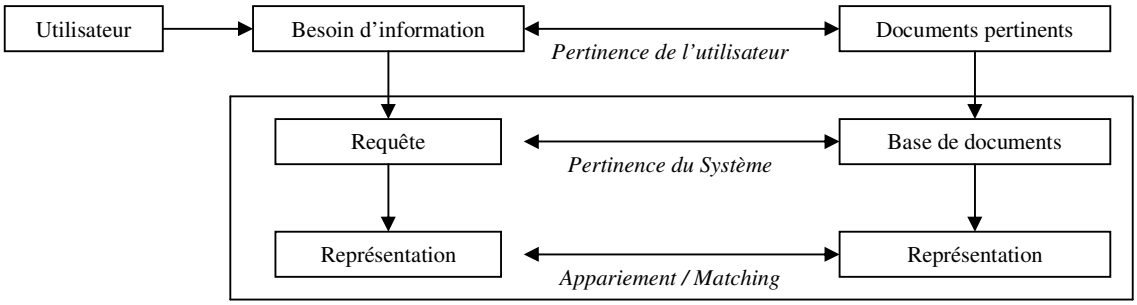


Figure 1.3 : Opérations et environnement de la RI

Nous distinguons trois niveaux différents :

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

La relation entre le besoin d'information et les documents attendus est la relation de pertinence (idéale, absolue, …).

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

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

(iii) Le niveau interne du système: La requête formulée par l'utilisateur (souvent en langue naturelle) ne peut pas se comparer directement avec des documents en langue naturelle eux aussi. Il faut donc créer des représentations internes pour la requête et pour les documents. Ces représentations doivent être manipulables par l'ordinateur. Le processus de création de ces représentations est appelé l'indexation. Il est aussi à noter que les représentations créées ne reflètent qu'une partie des contenus de la requête et des documents. La technologie de nos jours ne permet pas encore de créer une représentation complète.

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

Par ailleurs, nous remarquons qu'il y a des différences entre deux niveaux différents. En ce qui concerne le besoin d'information, il est transformé en une requête, puis en une représentation de cette dernière aux niveaux inférieurs. Du côté document, il y a des changements similaires. Les relations que nous pouvons déterminer à chaque niveau ne sont pas pareilles non plus. En fait, l'objectif ultime espèré est qu'un bon système de RI puisse aboutir à une bonne *correspondance* (*Appariement / Matching*) qui reflète bien la *pertinence du système*, laquelle, correspond bien au jugement de *pertinence de l'utilisateur.* Cependant, étant donné la différence entre les niveaux, il y a nécessairement une dégradation. Ainsi, une autre tâche de la RI est d'évaluer un système de RI une fois construit. Cette évaluation du système tente de mesurer l'écart entre les niveaux (surtout entre le niveau système et le niveau interne du système).

4. Notion de pertinence

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

- la correspondance entre un document et une requête, une mesure d'informativité du document à la requête;
- un degré de relation (chevauchement, relativité, …) entre le document et la requête;
- un degré de surprise qu'apporte un document, qui a un rapport avec le besoin de l'utilisateur;
- une mesure d'utilité du document pour l'utilisateur.

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

De nombreux travaux ont été menés sur cette notion. En effet, la pertinence n'est pas une relation isolée entre un document et une requête. Elle fait appel aussi au contexte de jugement. Ainsi, Tefko Saracevic [Saracevic, 1970] propose la définition suivante pour tenir compte de cette influence multiple du contexte sur la pertinence :

« *La pertinence est la A d'un B existant entre un C et un D jugé par un E, où :*

- *A = intervalle de la mesure ;*
- *B = aspect de la pertinence (la pertinence absolue) ;*
- *C = un document ;*
- *D = contexte dans lequel la pertinence est mesurée (y compris le besoin d'information) ;*
- *E = le juge (l'utilisateur)* ».

Il reconnaît déjà l'importance du contexte sur la pertinence, ainsi que l'utilisateur lui-même. Si on varie ces facteurs, la notion de pertinence change aussi.

Selon [Schamber et al., 1990] la pertinence est fonction de la qualité d'information, elle est toujours liée à un utilisateur alors que la quantité d'information ne l'est pas. Ces auteurs ont défini la pertinence de la manière suivante :

« *La pertinence est un concept dynamique qui dépend du jugement de l'utilisateur sur la proximité de l'information lue et celle qui est nécessaire. La pertinence est un concept mesurable* ».

Pour tenir compte de cette influence multiple, ces auteurs ont décomposé la problématique de pertinence selon les trois axes suivants :

- Le *comportement* regroupe la description et l'analyse du comportement de l'utilisateur : le contenu, la description du document, le savoir de l'utilisateur, la manière de formuler sa requête et la possibilité d'interactivité en particulier la possibilité de reformuler sa requête ;
- La *mesure* concerne l'étape de construction de mesure et particulièrement le jugement de valeur par l'utilisateur : la dichotomie (oui/non), la grille de collecte selon une échelle de valeurs prédéfinies ou la note libre ;
- La *terminologie* concerne la définition du concept de pertinence.

Par ailleurs, Mizzaro [Mizzaro, 1997] propose un modèle élaboré à partir de son étude récapitulative des différents aspects de la pertinence, il recense et classifie un ensemble de liens. Il définit la pertinence comme une relation entre deux entités de deux groupes. D'un coté, on trouve le *document*, la *description* et l'*information*, et d'un autre, on trouve le *problème*, le *besoin d'information*, la *question* et la *requête*. Les entités mentionnées peuvent être décomposées selon trois composantes :

- Le *sujet* qui correspond au sujet qui intéresse l'utilisateur,
- La *tâche* qui correspond à l'activité pour laquelle l'utilisateur effectue sa recherche,
- Le *contexte* qui correspond à n'importe quelle autre composante affectant la manière d'effectuer la recherche et l'évaluation.

La pertinence selon le même auteur peut être perçue comme un point dans un espace à quatre dimensions :

1. La première dimension est relative au document, sa description et à l'information ;
2. La deuxième comporte le problème, le besoin d'information, la question et la requête ;
3. La troisième comporte le sujet, la tâche, le contexte et toute combinaison possible entre eux ;
4. La quatrième correspond au temps qui s'écoule entre l'apparition du problème et l'obtention de la solution.

Les différents types de pertinences sont représentés dans la Figure 1.4.

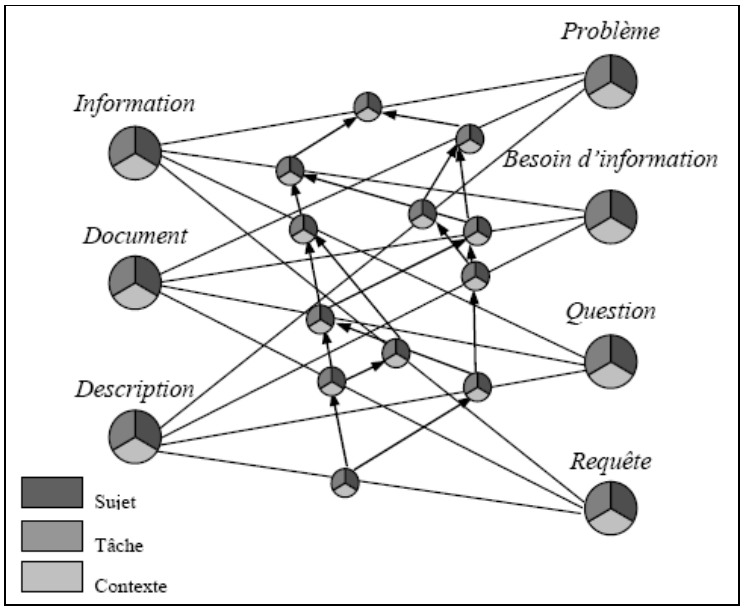


Figure 1.4 : Ordre partiel de pertinence

Chaque ligne joignant les objets correspond à une pertinence. La troisième dimension est représentée par les niveaux de gris utilisés. La dimension temps n'est pas représentée pour simplifier le schéma. Les flèches représentent dans quelle mesure une pertinence est proche de la pertinence de l'information reçue au problème de l'utilisateur et dans quelle mesure il est difficile de la mesurer.

Par ailleurs, la question qu'on peut se poser est : à quoi sert d'étudier la notion de pertinence si on sait qu'elle est très variable ? Une des raisons est de tenter de trouver certains comportements communs entre les utilisateurs, et essayer de les formaliser. Si on arrive à cerner une partie de pertinence commune, on pourra l'implanter dans les systèmes pour répondre au moins à une partie commune des besoins. On connaît maintenant certains facteurs communs. Par exemple, le *sujet* (ou en anglais *topic*) est le facteur le plus important dans la pertinence. Ainsi, on peut construire des systèmes en utilisant uniquement le critère de sujet, ce qui conduit à l'approche basée sur la *topicalité*. Une autre raison des études de la pertinence est d'essayer de comprendre exactement comment le contexte influence sur elle. Si on arrive à comprendre cela, par exemple, à trouver des contextes typiques dans lesquels un facteur devient très important, on pourra implanter des systèmes spécialisés en conséquence. Derrière ces études, il y a aussi des motivations philosophiques comme celle de comprendre comment l'humain raisonne.

5. Evaluation d'un système de RI

Le but de la RI est de trouver des documents pertinents à une requête, et donc utiles pour l'utilisateur. La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses idéales que l'utilisateur espère recevoir. Plus les réponses du système correspondent à celles que l'utilisateur espère, mieux est le système.

5.1 Corpus de test (références)

Pour arriver à une telle évaluation, on doit connaître d'abord les réponses idéales de l'utilisateur. Ainsi, l'évaluation d'un système s'est faite souvent avec certains corpus de test. Dans un corpus de test, il y a:

- un ensemble de documents;
- un ensemble de requêtes;
- la liste de documents pertinents pour chaque requête.

Pour qu'un corpus de test soit significatif, il faut qu'il possède un nombre de documents assez élevé. Les premiers corpus de test développés dans les années 1970 renferment quelques milliers de documents. Les corpus de test plus récents (par exemple, ceux de TREC[2]) contiennent en général plus 100 000 documents (considérés maintenant comme un corpus de taille moyenne), voir des millions de documents (corpus de grande taille). Parmi les collections de documents de test les plus utilisées en RI citons :

- La collection CACM regroupant les titres et les résumés triés du journal CACM ;
- La collection Cranfield traitant des résumés du domaine « *Aeronautical Engineering* » ;
- La collection Medline traitant les articles triés du journal « *Medical Journal* » ;
- La collection *Time* constituant les articles triés du journal *Time*.

Le tableau 1.2 récapitule ces collections.

	Nombre de documents	**Nombre de requêtes**
CACM[3]	3240	64
CISI[4]	1460	112
CRAN[5]	1400	225
MED[6]	1033	30
TIME[7]	425	83

Tableau 1.2 : Quelques collections de documents de test en RI

L'évaluation d'un système ne doit pas se reposer seulement sur une requête. Pour avoir une évaluation assez objective, un ensemble de quelques dizaines de requêtes, traitant des sujets variés, est nécessaire. L'évaluation du système doit tenir compte des réponses du système pour toutes ces requêtes.

Finalement, il faut avoir les réponses idéales pour l'utilisateur pour chaque requête. Le dernier élément d'un corpus de test fournit cette information. Pour établir ces listes de documents pour toutes les requêtes, les utilisateurs (ou des testeurs simulant des utilisateurs) doivent examiner chaque document de la base de document, et juger s'il est pertinent. Après cet exercice, on connaît exactement quels documents sont pertinents pour chaque requête. Pour la construction d'un corpus de test, les jugements de pertinence constituent la tâche la plus difficile.

[2] http://trec.nist.gov/
[3] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/cacm/
[4] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/cisi/
[5] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/cran/
[6] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/medl/
[7] http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/time/

5.2 Rappel et Précision

La comparaison des réponses d'un système pour une requête avec les réponses idéales nous permet d'évaluer les métriques suivantes :

5.2.1 Le Rappel

Le rappel mesure la proportion de documents pertinents retrouvés parmi tous les documents pertinents dans la base. La proportion complémentaire est le ***Silence*** qui correspond à la proportion de documents pertinents non retrouvés.

$$Rappel = \frac{|P \cap R|}{|P|} \in [0, 1] \text{ et } Silence = 1 - Rappel \quad (1.4)$$

Avec : *P* représente le nombre de documents pertinents dans tout le corpus.
R représente le nombre de documents retrouvés.

5.2.2 La Précision

La précision mesure la proportion de document pertinent retrouvé parmi tous les documents retrouvés par le système. La proportion complémentaire est le ***Bruit*** qui correspond à la proportion de documents retrouvés qui ne sont pas pertinents.

$$précision = \frac{|P \cap R|}{|R|} \in [0, 1] \text{ et } Bruit = 1 - précision \quad (1.5)$$

5.2.3 La F-mesure

Plusieurs indicateurs de synthèse ont été créés à partir de deux mesures de Rappel et de la Précision, mais le plus célèbre est la F-mesure. Cette mesure correspond à une moyenne harmonique de la précision et du rappel. Cette moyenne diminue lorsque l'un de ses paramètres est petit et augmente lorsque les deux paramètres sont proches tout en étant élevés [Rijsbergen, 1979].

$$F - mesure = \frac{(1 + \beta^2)\, précision \times rappel}{(\beta^2 \times précision) + rappel} \quad (1.6)$$

Le paramètre β permet de pondérer la précision ou le rappel, il est égal généralement à la valeur 1. Pour effectuer ces mesures, il faut disposer des réponses idéales aux requêtes en question. La Figure 1.5 illustre ces formules.

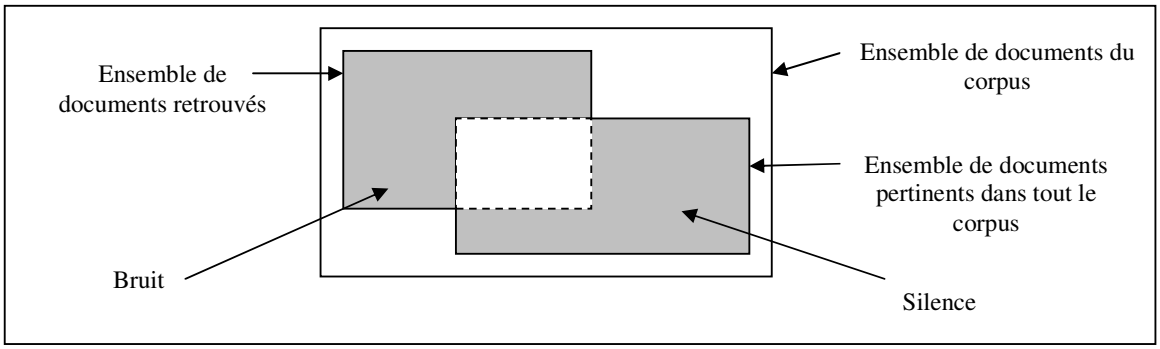


Figure 1.5 : Rapprochement de pertinences système et utilisateur

Par ailleurs, un système idéal est un système donne de bons taux de précision et de rappel en même temps. Un système qui aurait 100% pour la précision et pour le rappel signifie qu'il trouve tous les documents pertinents, et rien que les documents pertinents. Cela veut dire que les réponses du système à chaque requête sont constituées de tous et seulement les documents idéaux que l'utilisateur a identifiés. En pratique, cette situation n'arrive pas. Plus souvent, nous pouvons obtenir un taux de précision et de rappel aux alentours de 30%.

En fait, les deux métriques ne sont pas indépendantes. Il y a une forte relation entre elles : quand l'une augmente, l'autre diminue. Il ne signifie rien de parler de la qualité d'un système en utilisant seulement une des métrique. En effet, il est facile d'avoir 100% de rappel: il suffirait de donner toute la base comme la réponse à chaque requête. Cependant, la précision dans ce cas-ci serait très basse. De même, on peut augmenter la précision en donnant très peu de documents en réponse, mais le rappel souffrira. Il faut donc utiliser les deux métriques ensemble.

Les mesures de précision-rappel ne sont pas statiques non plus (c'est-à-dire qu'un système n'a pas qu'une mesure de précision et de rappel). Le comportement d'un système peut varier en faveur de précision ou en faveur de rappel (en détriment de l'autre métrique).

Pour comparer deux systèmes de RI, il faut les tester avec le même corpus de test (ou plusieurs corpus de test). Un système dont la courbe dépasse (c'est-à-dire qu'elle se situe en haut à droite de) celle d'un autre est considéré comme un meilleur système.

Il arrive parfois que les deux courbes se croisent. Dans ce cas, il est difficile de dire quel système est meilleur. Pour résoudre ce problème, nous pouvons utiliser aussi la *précision moyenne* comme une mesure de performance. En effet, la précision moyenne est une moyenne de précision sur un ensemble de points de rappel. Cette précision moyenne pouura être utilisée soit sur 10 points de rappel (0.1, …, 1.0), soit sur 11 points de rappel (0.0, 0.1, …, 1.0). Cette dernière est possible seulement avec la polarisation.

La précision moyenne décrit bien la performance d'un système. C'est la mesure souvent utilisée en RI.

6. Reformulation de la requête

La recherche d'information est un processus qui se base essentiellement sur la requête exprimée par l'utilisateur pour répondre à ses besoins. Quel que soit le système de recherche utilisé, le résultat d'une recherche ne peut être intéressant si la requête ne décrit pas explicitement et clairement les besoins de l'utilisateur. En général, l'utilisateur se contente de donner quelques mots-clés. Ces derniers sont issus d'une connaissance générale sur un domaine donné. Par conséquent, les documents renvoyés par le système de recherche peuvent appartenir à des domaines et disciplines différents par lesquels l'utilisateur n'est pas concerné.

La reformulation de requêtes est une phase importante du processus de recherche d'information. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin [Efthimiadis, 2000]. En effet, les techniques de reformulation consistent à modifier les requêtes pour ressembler davantage aux documents jugés pertinents et s'éloigner des documents non pertinents. Plus la distance entre la requête initiale et la requête reformulée est grande, plus il y a de nouveau documents qui vont apparaître comme résultat de la nouvelle recherche. Ces techniques peuvent être assistées par l'utilisateur (interactives), comme elles peuvent être menées d'une manière automatique.

La première technique est la plus répandue en RI. Il s'agit de la reformulation par réinjection (rétroaction) de la pertinence, appelée aussi *Relevance Feedback (RF)*. Elle consiste à extraire à partir des documents jugés pertinents par l'utilisateur les mots-clés les plus expressifs, et à les ajouter à la requête [Rocchio, 1971] [Robertson et Sparck-Jones, 1976]. Dans la deuxième technique, il s'agit de l'expansion de requête.

6.1 Rétroaction de pertinence (*Relevance Feedback)*

Le RF (Relevance Feedback) ou technique de modification des requêtes par analyse et incorporation des retours, est un processus de reformulation automatique de requêtes dont le but est de générer des requêtes optimales proches des besoins des utilisateurs. Cette reformulation qui se fait par interaction entre l'utilisateur et le système consiste en générale à modifier la pondération des termes de la requête initiale ou à leur substituer d'autres termes choisis pour leur caractère, notamment associatif, générique ou spécifique. Ces opérations de reformulation s'effectuent sur la base des indices fournis par l'utilisateur à travers, d'une part, la requête initiale et, d'autre part, les documents pertinents et non pertinents sélectionnés. Ce processus de recherche, de sélection de documents pertinents et non pertinents puis de génération automatique de requête se fait de façon itérative jusqu'à l'atteinte des objectifs à la satisfaction de l'utilisateur.

En fait, cette technique a pour but de simplifier la tâche de l'utilisateur qui n'a pas à déterminer dans les documents pertinents les termes importants, avant d'effectuer une nouvelle requête.

[Yuwono et al., 1997] distinguent deux techniques principales du RF : la technique semi-automatique basée sur le modèle de Rocchio et la technique automatique.

6.1.1 La technique du RF semi-automatique

Cette technique nécessite l'intervention de l'utilisateur qui doit identifier et sélectionner les documents pertinents et les documents non pertinents. Les travaux sur cette technique ont été menés par Rocchio à la fin des années 1970. Ces travaux ont été publiés en 1971 [Rocchio, 1971] et ont été suivis de ceux de Ide [Ide, 1971]. Plus tard, les travaux sur le RF semi-automatique ont été enrichis par l'apport de la méthode probabiliste. Cette approche a été implémentée par Harper, Hamian, Croft, Spark Jones et Van Rijisbergen [Yuwono et al., 1997].

Notons que la fonction de Rocchio dérive de l'hypothèse qu'une requête idéale Q^{new} doit maximiser la différence de sa distance Cosinus moyenne de ses documents pertinents et de sa distance Cosinus moyenne de ses documents non pertinents (la distance cosinus sera présentée dans le modèle vectoriel du chapitre suivant). Le système effectue la reformulation selon l'équation suivante :

$$Q^{new} = \alpha Q^{old} + \beta \frac{1}{|reldocs|} \sum_{reldocs} w_{t_i} - \gamma \frac{1}{|nonreldocs|} \sum_{nonreldocs} w_{t_i} \qquad (1.7)$$

α permet de moduler l'importance de la requête précédente Q^{old}.

β permet de moduler le vecteur profil moyen des documents choisis.

γ permet de moduler le vecteur profil des documents rejetés.

α, β et γ représentent des paramètres positifs. Leurs valeurs sont à fixer dans l'intervalle [0, 1].

|*reldocs*| représente le cardinal de l'ensemble des documents pertinents.

|*nonreldocs*| représente le cardinal de l'ensemble des documents non pertinents.

Le paramètre α n'était pas initialement pris en compte dans la formule de Rocchio. Salton, l'a introduit ultérieurement et c'est la forme générale définie qui est souvent considérée.

6.1.2 La technique de RF automatique

Selon [Aliane et al., 2004], lorsque le *feedback* de pertinence s'accompagne d'une adjonction (et/ou) suppression de termes, il s'agit de la reformulation automatique. La requête de l'utilisateur est remaniée automatiquement, pour intégrer les descripteurs des documents jugés pertinents ou rejetés.

En fait, il existe différentes variantes de cette technique : celles qui sont utilisées automatiquement pour reformuler la requête en augmentant le poids des termes présents dans les documents jugés pertinents et inversement pour diminuer les poids des termes jugés non pertinents.

Le problème avec la reformulation automatique est l'estimation des « bons » termes qui peuvent conduire effectivement à une amélioration du processus de recherche car l'introduction des termes inappropriés peut entraîner un *silence* ou au contraire augmenter un *bruit*.

Selon [Yuwono et al., 1997], dans les environnements où la technique du RF automatique est implémentée, un nombre prédéfini de documents extraits par la requête initiale sont réputés pertinents. Les procédures et formules utilisées dans l'approche du RF automatique sont des variantes des formules Rocchio et Ide qui permettent de faire abstraction des documents non pertinents.

En effet, le modèle de Ide [Ide, 1971] est une variante du modèle de Rocchio. Du modèle de Rocchio elle déduit la formule suivante qui lui sert de base dans ses travaux :

$$Q_{i+1} = \pi Q_i + \omega Q_0 + \alpha \sum_{i=1}^{\min(n_a, n'_p)} p_i + \mu \sum_{i=1}^{\min(n_b, n'_s)} NP_i \qquad (1.8)$$

Où $(n'_p + n'_s) = N$ le nombre de document extraits et servant au processus du "*feedback*". Les variables expérimentales étant : a, ω, μ, π, n_a, n_b *et* N.

Le paramètre α est positif et permet de pondérer tous les documents jugés pertinents par rapport à tous les éléments contribuant à la formation de ta requête (requête précédente Q_i, requête initiale Q_0 et documents non pertinents).

Le paramètre π permet d'augmenter la pondération de la requête précédente en fonction des documents du *feedback*. Q_0 est la requête initiale, Q_i est la requête de la précédente itération, ω permet d'utiliser la requête initiale comme partie intégrante de la nouvelle requête, μ doit être théoriquement négatif pour tenir compte des documents non pertinents extraits. Les paramètres n_a, n_b permettent d'utiliser un nombre spécifique de documents pertinents et non pertinents dans la requête même quand les valeurs des paramètres n_a, n_b sont plus grands (utilisation de la fonction *min*()).

La flexibilité de cette formule a permis à Ide non seulement de confirmer les résultats positifs obtenus par Rocchio, mais aussi d'étudier trois variantes de ce modèle [Ide, 1971]:

- Modèle basé sur l'utilisation exclusive de documents pertinents ;
- Modèle basé sur le nombre de documents N à extraire et à réintégrer dans le système à chaque itération du RF.
- Modèle basé sur l'intégration d'un ou de deux documents non pertinents aux documents pertinents et à la requête initiale.

6.2 Expansion de requêtes

Une expansion de requête peut être vue comme un traitement pour "élargir" le champ de recherche pour cette requête. Une requête étendue va contenir plus de termes reliés. En utilisant le modèle vectoriel, par exemple, plus de documents seront repérés. Ainsi, ce traitement est souvent vu comme un moyen d'augmenter le taux de rappel. Cependant, nous savons qu'il n'a pas de sens de parler du rappel sans considérer en même temps la précision. Ainsi, cette affirmation que l'expansion de requête va conduire à un meilleur rappel n'est pas tout à fait juste. Il faut plutôt dire que, en sélectionnant les documents selon un seuil de similarité entre un document et une requête, nous avons la chance de sélectionner plus de documents pertinents avec une requête étendue.

L'utilité de l'expansion de requête dépend fortement de deux facteurs:

1. Quels mots doit-on utiliser pour étendre la requête?
2. Comment les nouveaux mots doivent-ils être ajoutés dans la requête?

Les mots utilisés pour faire l'expansion de requête doivent être fortement reliés à la requête. Typiquement, on utilise un dictionnaire de synonyme, ou un thésaurus. Les mots reliés avec des mots de la requête par certains types de relation (e.g. IS_A) sont choisis pour étendre la requête.

Il y a aussi des études qui essaient de trouver automatiquement les mots fortement reliés. La plupart de ces approches exploitent les co-occurrences: Plus deux mots co-occurent dans des textes, plus on suppose qu'ils sont fortement reliés. Une fois ces relations statistiques choisies, on peut les utiliser dans un processus d'expansion de requête.

Il est aussi suggéré que le processus d'expansion soit interactif : L'usager peut filtrer les mots proposés par le système. Cette approche est utilisée dans certains systèmes, par exemple, Medline qui intègre un thésaurus du domaine médical [Joubert et al., 1991].

6.3 Les problèmes posés par la reformulation de la requête

La rétroaction de pertinence est d'un emploi souvent lourd pour l'utilisateur qui doit interagir avec la système, tandis que les termes ajoutés lors d'une expansion automatique ne sont pas toujours appropriés et peuvent par conséquent engendrer du bruit puisqu'il est possible d'introduire dans la requête des termes qui n'ont pas de rapport avec le besoin de l'utilisateur. D'autre part les termes de la requête sont généralement considérés de manière isolée dans l'expansion de la requête.

En cas de courtes requêtes, l'expansion de requêtes peut ne pas être efficace car l'ambiguïté eventuelle de la requête risque d'être prolongée dans l'expansion. Ce problème est résolu par la combinaison d'un filtrage et d'une expansion nommée le « *Query By Example* ».

La plupart d'approches d'expansion considèrent chaque mot de la requête isolé. [Qiu et Frei, 1993] pensent qu'il vaut mieux choisir des mots qui sont reliés à la requête qu'aux mots individuels de la requête. Autrement dit, ils calculent la relation entre un mot et la requête dans son ensemble, et choisissent d'utiliser les mots les plus fortement reliés. Ils montrent que cette approche est meilleure que celle de l'expansion de mots.

D'autre part, il est possible qu'un document ne concernant qu'un seul terme de la requête soit mieux classé qu'un autre document concernant tous les termes de la requête : le premier contient plusieurs représentation du même terme. Dans ce cadre [Salton et McGill, 1983] ont introduit les deux notions d'*exhaustivité* (*exh*) et de *spécificité* (*spec*). L'exhaustivité détermine si tous les aspects de la requête ont été abordés dans le document. Quant à la

spécificité, elle détermine si tout le contenu du document *D* traite du thème de la requête *Q*. En fait, ces deux mesures peuvent être calculées de plusieurs façons, parmi les quelles citons :

$$exh(D,Q) = \frac{\sum_j d_j q_j}{\sum_j q_j} \quad \text{et} \quad spec(D,Q) = \frac{\sum_j d_j q_j}{\sum_j d_j} \tag{1.9}$$

ou encore :

$$exh(D,Q) = \sqrt{\frac{\sum_j d_j q_j}{\sum_j q_j}} \quad \text{et} \quad spec(D,Q) = \sqrt{\frac{\sum_j d_j q_j}{\sum_j d_j}} \tag{1.10}$$

Avec : $D = (d_1, d_2, \ldots, d_n)$ vecteur des poids associés aux termes descripteurs dans le document *D*.

$Q = (q_1, q_2, \ldots, q_n)$ vecteur des poids associés aux termes descripteurs dans la requête *Q*.

Notons que la reformulation de la requête ne permet d'améliorer la recherche que relativement aux résultats obtenus à partir de la requête initiale. Ces améliorations de requêtes sont variables d'une base documentaire à une autre et peuvent dépendre, d'une part du nombre de termes ajoutés et de leur sélection et d'autre part de la manière avec laquelle ils sont ajoutés.

7. Conclusion

Nous avons détaillé dans ce chapitre les différents acteurs qui interviennent dans un Système de Recherche d'Information. Nous avons montré que le besoin de l'utilisateur ne doit pas se limiter uniquement à sa requête. Son intéraction avec le système est aussi une composante essentielle pour améliorer la qualité de la recherche. En fait, pour satisfaire davantage le besoin d'information d'un utilisateur et pour l'intégrer dans le processus de recherche, d'autres techniques ont été introduites telles que la reformulation de requêtes et les techniques de classification. Il s'est avéré aussi que la phase d'analyse et d'indexation est très utile pour construire une représentation riche, cohérente et proche du contenu du document. Cette phase permet aussi une accélération du processus de recherche. En effet, la réussite d'un tel processus est étroitement liée à la qualité du système d'indexation.

Nous allons présenter dans le chapitre suivant les modèles les plus connus de la RI. Nous nous intéressons particulièrement à la reformulmation de la requête ainsi qu'au sens de la pertinence donné par ces modèles.

Chapitre 2

Les modèles de la Recherche d'Information

La RI est un ensemble de techniques et d'outils informatiques dont la finalité initiale était bibliographique : il s'agissait d'aider les usagers à trouver, dans des fonds documentaires, les références concernant un sujet particulier. L'amélioration des capacités de stockage des ordinateurs a changé la nature du problème, qui n'est désormais plus d'exploiter des notices bibliographiques mais de conserver et d'accéder directement aux informations textuelles contenues dans les documents qui constituent les fonds.

Le champ de la recherche d'information moderne couvre ainsi plus largement la catégorisation des documents textuels, leur indexation, leur classification, leur catalogage et l'accès à leur contenu. Longtemps réservée à une petite communauté de spécialistes, la RI est aujourd'hui connue et utilisée par un public plus large à travers les moteurs de recherche sur Internet.

Le problème général de la RI est de retrouver dans un ensemble de documents ceux qui contiennent des informations qui constituent des réponses à la requête d'un utilisateur. Les systèmes de RI disposent ainsi d'une représentation des informations contenues dans les fonds documentaires et d'une procédure permettant de déterminer leur pertinence comme réponses à une requête particulière. Idéalement, ces systèmes devraient « comprendre » les informations textuelles contenues dans les documents et les requêtes (ces dernières, généralement formulées en langue naturelle, peuvent être considérées comme des documents supplémentaires). Une telle compréhension est malheureusement hors de la portée des systèmes de Traitement Automatique en Langage Naturel (TALN) ; étant donné les volumes des documents, une analyse sémantique de leur contenu par des opérateurs humains n'est pas non plus envisageable. Les systèmes de RI peuvent donc au mieux calculer une approximation du sens de ces informations, et évaluer leur proximité avec celui de la requête, de façon à classer les documents en fonction de leur pertinence comme réponses à la requête.

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

- Créer une représentation interne d'un document ou d'une requête basée sur ces termes ;
- Définir une méthode d'appariement (ou *matching*) entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité).

Le modèle joue un rôle central dans la Recherche d'Information ; c'est celui qui détermine le comportement clé d'un système de RI. En fait, il existe deux techniques d'appariement de sens qui utilise la linguistique : l'inférence et paraphrasage.

L'appariement à base d'inférence est une technique statistique, utilisée pour la manipulation des connaissances en Intelligence Artificielle (IA). Cette approche utilise les réseaux sémantiques afin d'introduire un niveau sémantique dans les traitements. Alors que le paraphrasage s'appui sur des connaissances purement linguistiques, indépendantes du domaine d'application ; les transformations sont par conséquent générales et n'invoquent pas un niveau de représentation des connaissances d'ordre conceptuel.

Les techniques linguistiques sont lourdes à mettre en œuvre dans un cadre qui se veut pratique et efficace. De l'autre côté, les techniques statistiques sont plus performantes en percevant la valeur de pertinence qu'on associe aux termes comme une dimension du sens. Dans ces perspectives, nous nous intéressons dans ce chapitre qu'aux techniques statistiques.

Nous décrivons dans la première section le modèle de "Matching score", premier modèle utilisé dans la RI. Dans la deuxième section, nous détaillons le modèle booléen ainsi que ses deux extensions : le modèle booléen basé sur des ensembles flous et le modèle booléen étendu ou p-norme. Le modèle vectoriel et ses deux extensions (le modèle vectoriel généralisé et modèle vectoriel et domaine sémantique) feront l'objet de la troisième section. Dans la quatrième section, nous présentons le modèle probabiliste. Dans la dernière section, nous mettons l'accent sur la reformulation de requêtes dans ces modèles.

1. Modèle "Matching score"

C'est peut-être le premier "modèle" utilisé dans la RI. L'idée est assez primitive et intuitive : Un document est représenté par un ensemble de termes pondérés par leur fréquence. Une requête est aussi un ensemble de termes, pondérés à 1. Le degré de correspondance est la somme des fréquences des termes de la requête dans le document : $R(d,q) = \sum_i f_i$

Où f_i est la fréquence d'un terme de q dans le document d.

La valeur R ainsi calculée est appelée la "matching score". En réalité, cela est équivalent à parcourir le document et à voir combien de fois les termes de la requête apparaissent dans ce document. Plus ce "matching score" est élevé, plus on considère que le document correspond à la requête, et donc plus il sera classé haut dans la réponse.

Ce modèle est primitif car il utilise directement le résultat de l'indexation sans aucune réorganisation ou modélisation [Salton et al., 1983a].

2. Modèle booléen

Dans ce modèle, un document est représenté comme une conjonction logique de termes (non pondérés), par exemple : $d = t_1 \wedge t_2 \wedge \ldots \wedge t_n$

Une requête peut être vue comme une formule logique contenant un certain nombre de termes reliés par des opérateurs logiques tels que : ET (noté $\wedge$), OU (noté $\vee$) et NON (noté $\neg$). Par exemple : $q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$

Pour qu'un document corresponde à une requête, il faut que l'implication suivante soit valide : $d \Rightarrow q$. Cette évaluation peut être aussi définie de la façon suivante :

Un document peut être représenté comme un ensemble de termes, et une requête comme une expression logique de termes. La correspondance ou pertinence système $R(d, q)$ entre une requête q et un document d est déterminée de la façon suivante, si q_i et q_j sont des termes de la requête :

$$\begin{cases} R(d, q_i) = 1 \textbf{ si } q_i \in d \text{ ; } \textbf{0 sinon}. \\ R(d, q_i \wedge q_j) = 1 \textbf{ si } R(d, q_i) = 1 \textbf{ et } R(d, q_j) = 1 \text{ ; } \textbf{0 sinon}. \\ R(d, q_i \vee q_j) = 1 \textbf{ si } R(d, q_i) = 1 \textbf{ ou } R(d, q_j) = 1 \text{ ; } \textbf{0 sinon}. \\ R(d, \neg q_i) = 1 \textbf{ si } R(d, q_i) = 0 \text{ ; } \textbf{0 sinon}. \end{cases} \quad (2.1)$$

Dans sa version simple, le modèle présente les trois problèmes suivants [Waller et Kraft, 1979]:

1. La correspondance entre un document et une requête est soit 1, soit 0. En conséquence, le système détermine un ensemble de documents non ordonnés comme réponse à une requête. Il n'est pas possible de dire quel document est meilleur qu'un autre. Cela crée beaucoup de problèmes aux usagers, car ils doivent encore fouiller dans cet ensemble de documents non ordonnés pour trouver des documents qui les intéressent. C'est difficile dans le cas ou beaucoup de documents répondent aux critères de la requête.
2. Tous les termes dans un document ou dans une requête étant pondérés de la même façon simple (0 ou 1), il est difficile d'exprimer qu'un terme est plus important qu'un autre dans leur représentation. Ainsi, un document qui décrit en détail "informatique", mais mentionne un peu "commerce" se trouve être représenté par {informatique, commerce} dans laquelle les deux termes deviennent aussi important l'un que l'autre. Cela ne correspond pas à ce qu'on souhaite avoir.
3. Le langage d'interrogation est une expression quelconque de la logique de propositions (un terme étant une proposition). Cela offre une très grande flexibilité aux usagers d'exprimer leurs besoins. Cependant, un problème en pratique est que les usagers manipulent très mal les opérateurs logiques, surtout dans beaucoup de cas, les mots "et" et "ou" ne correspondent pas tout à fait aux opérateurs logique $\wedge$ et $\vee$. En partie à cause de cela, les expressions logiques données par un usager correspondent souvent mal à son besoin. La qualité de la recherche souffre donc en conséquence.

Nous récapitulons dans la suite les avantages et les inconvénients de ce modèle :

Les avantages du modèle booléen :

- Le modèle est plus facile à implanter et nécessite relativement peu de ressources ;
- Le langage de requête booléen est plus expressif que celui des autres modèles ;
- Ce modèle convient aux utilisateurs connaissant exactement leurs besoins et en mesure de les formuler précisément avec le vocabulaire qu'ils maîtrisent.

Les inconvénients du modèle booléen :

- Il est difficile aux novices de formuler une requête combinant plusieurs opérateurs logiques, notamment pour les requêtes complexes. L'importance relative des mots-clés ne peut pas être exprimée ;
- Le classement des documents extraits par ordre de pertinence est difficile ;
- La reformulation automatique des requêtes par la technique du "Relevance Feedback" est plus ardue.

Notons que le modèle booléen standard n'est utilisé que dans très peu de systèmes de nos jours. Parmi les SRI classiques basés sur le modèle booléen, nous citons MEDLARS[8] (1970) devenu MEDLINE[9] et TEXTO[10] (1982). En fait, si nous utilisons un modèle booléen, c'est plutôt une extension de ce modèle. Les extensions de ce modèle, présentées dans la suite, essaient justement de corriger ces lacunes.

[8] MEDLARS : *MEDical Literature Analysis and Retrieval System*. C'est la version Medline de la *National Library of Medecine*.
[9] MEDLINE : MEDLARS on LINE : C'est une base qui indexe des articles de périodiques qui se rapportent au domaine médical.
[10] TEXTO : C'est un logiciel de manipulation de fonds documentaire.

Pour réduire les limites posées par ce modèle proposé pour la RI et dans le but d'augmenter leurs performances, deux extensions ont été proposées. Celles-ci sont décrites ci-après.

2.1 Modèle Booléen basé sur des ensembles flous

Cette extension au modèle booléen standard consiste à affecter des pondérations (a_i) aux termes dans la représentation des documents. Du côté requête, elle reste toujours une expression booléenne. Avec cette extension, un document est représenté comme un ensemble de termes (t_i) pondérés (a_i) comme suit [Kraft et al., 1983] :

$d = \{\ldots, (t_i, a_i), \ldots\}$

La correspondance entre un document d et une requête exprimée par les deux termes q_i et q_j, peut prendre plusieurs formes. Une d'elles est la suivante :

$$\begin{cases} R(d, q_i) = a_i \\ R(d, q_i \wedge q_j) = \min (R(d, q_i), R(d, q_j)) \\ R(d, q_i \vee q_j) = \max (R(d, q_i), R(d, q_j)) \\ R(d, \neg q_i) = 1 - R(d, q_i) \end{cases} \quad (2.2)$$

Dans cette évaluation, les opérateurs logiques $\wedge$ et $\vee$ sont évalués par min et max respectivement. C'est une des évaluations classiques proposées par L. Zadeh [Zadeh, 1965] dans le cadre des ensembles flous. Dans cette évaluation on s'intéresse soit à la partie la plus facile quand il s'agit d'une disjonction, soit à la pqrtie la plus difficille quand il s'agit d'une conjonction. Par conséquent, les opérateurs *min* et *max* se rapprochent respectivement des opérateurs ET et OU, puisque le rang d'un document retrouvé dépend du terme de plus faible poids pour une requête ET et de plus fort poids pour une requête OU. D'après [Salton, 1983], cette méthode manque de pouvoir de descrimination.

Ainsi, plusieurs autres formes d'évaluation ont été proposées. Une des formes est celle de Lukaswicz qui fait intervenir dans l'évaluation les deux parties $R(d, q_i)$ et $R(d, q_j)$:

$$\begin{cases} R(d, q_i) = a_i \\ R(d, q_i \wedge q_j) = R(d, q_i) * R(d, q_j) \\ R(d, q_i \vee q_j) = R(d, q_i) + R(d, q_j) - R(d, q_i) * R(d, q_j) \\ R(d, \neg q_i) = 1 - R(d, q_i) \end{cases} \quad (2.3)$$

Dans cette évaluation, les deux parties d'une conjonction ou d'une disjonction contribuent en même temps, contrairement à celle de Zadeh [Zadeh, 1965]. Cependant, elle a le même problème qui est $R(d, q \wedge \neg q) \neq 0$ et $R(d, q \vee \neg q) \neq 1$. En plus, $R(d, q \wedge q) \neq R(d, q) \neq R(d, q \vee q)$.

En comparant ces extensions avec le modèle standard, il est assez facile de voir les avantages. Le plus important est la possibilité de mesurer le degré de correspondance entre un document et une requête dans [0, 1]. Ainsi, les documents puissent être ordonné dans l'ordre décroissant de leur correspondance avec la requête. L'usager peut parcourir cette liste ordonnée et décider où s'arrêter. En outre, cette représentation plus raffinée, car on peut exprimer dans quelle mesure un terme est important (représentatif) dans un document.

2.2 Modèle booléen étendu ou P-Norme

Le modèle p-norme [Salton et al., 1983a] est proposé pour résoudre certains problèmes observés dans le modèle booléen standard :

- La taille des réponses est non contrôlable ;
- Les réponses sont non-ordonnées ;
- Tous les termes ont la même importance ;
- Pour une requête qui est une longue conjonction, un document qui satisfait la majorité des termes est aussi mauvais qu'un document qui ne satisfait aucun terme ;
- pour une requête qui est une longue disjonction, un document qui satisfait un terme est aussi bon qu'un document qui satisfait tous les termes ;

L'approche proposée tente d'étendre le modèle booléen standard sur plusieurs aspects. D'abord, observons la table de vérité utilisée pour l'évaluation booléenne standard donnée par le tableau 2.1:

A	B	$A \wedge B$	$A \vee B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Tableau 2.1 : Table de vérité pour l'évaluation booléenne standard

Dans la colonne de $A \wedge B$, l'objectif est d'atteindre le cas de la dernière ligne. Dans la colonne de $A \vee B$, c'est plutôt la première ligne qu'il faut éviter. Ainsi, une façon de rendre flou l'évaluation stricte consiste à calculer une sorte de distance entre les points à éviter ou à atteindre. Selon cette distance, on va déterminer l'évaluation de la conjonction ou de la disjonction. L'idée de base correspond à la figure 2.1 :

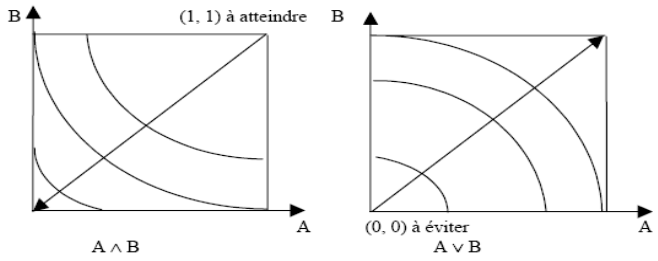


Figure 2.1 : Evaluation de la conjonction et de la disjonction

Dans la figure 2.1, étant donné une évaluation de A et de B, on détermine un point dans l'espace A-B. Dans le cas de la conjonction, on cherche à évaluer dans quelle mesure ce point est proche de (1, 1) - le point à atteindre. Ce rapprochement peut être mesurée par le complément de la distance entre le point et le point (1, 1) : plus cette distance est grande, moins $A \wedge B$ est satisfaite à ce point. Pour les points qui se situent sur une même courbe, ils ont la même distance avec (1, 1). Dans le cas de $A \vee B$, on cherche plutôt à éviter le point (0, 0). Plus on est loin de (0, 0), plus $A \vee B$ est satisfaite.

Basée sur cette intuition, l'évaluation suivante est proposée par Salton et al. [Salton et al., 1983a]. On admet la pondération de termes dans les documents : a_i est le poids de q_i dans d.

$$\begin{cases} R(d, q_i) = a_i \\ R(d, q_i \wedge q_j) = 1 - ([(1- R(d, q_i))^2 + (1- R(d, q_j))^2] / 2)^{1/2} \\ R(d, q_i \vee q_j) = [(R(d, q_i)^2 + R(d, q_j)^2) / 2]^{1/2} \\ R(d, \neg q_i) = 1 - R(d, q_i) \end{cases} \quad (2.4)$$

Dans cette évaluation, la distance est normalisée (divisée par $2^{1/2}$).

Première généralisation

Une première généralisation de l'évaluation précédente a consisté à permettre aussi à associer une pondération aux termes de la requête. Dans cette approche, la signification de cette pondération est bien définie : elle mesure l'importance du terme pour le besoin de l'usager. Plus un terme a une pondération forte, plus il est important. Ainsi, une requête (q_i, q_j, q_k) pondérés respectivement par (b_i, b_j, b_k) prend la forme suivante :

$$q = q_i^{bi} \wedge (q_j^{bj} \vee q_k^{bk}) \quad (2.5)$$

L'évaluation devient la suivante (supposons que a_i soit le poids de q_i dans d) :

$$\begin{cases} R(d, q_i^{bi}) = a_i * b_i \\ R(d, q_i^{bi} \wedge q_j^{bj}) = 1 - [[b_i^2 * (1-R(d, q_i))^2 + b_j^2 * (1-R(d, q_j))^2] / (b_i^2 + b_j^2)]^{1/2} \\ R(d, q_i^{bi} \vee q_j^{bj}) = [b_i^2 * (R(d, q_i)^2 + b_j^2 * R(d, q_j)^2) / (b_i^2 + b_j^2)]^{1/2} \\ R(d, \neg q_i) = 1 - R(d, q_i) \end{cases} \quad (2.6)$$

Deuxième généralisation

Pour généraliser ce dernier modèle on peut attribuer une pondération aux opérateurs logiques pour déterminer dans quelle mesure un opérateur doit être évalué de façon stricte. C'est le rôle du facteur p qu'on ajoute sur un opérateur. Les opérateurs $\wedge^p$ et $\vee^p$ sont évalués comme suit :

$$\begin{cases} R(d, q_i^{bi} \wedge^p q_j^{bj}) = 1 - [[b_i^p * (1-R(d, q_i))^p + b_j^p * (1-R(d, q_j))^p] / (b_i^p + b_j^p)]^{1/p} \\ R(d, q_i^{bi} \vee^p q_j^{bj}) = [b_i^p * (R(d, q_i)^p + b_j^p * R(d, q_j)^p) / (b_i^p + b_j^p)]^{1/p} \end{cases} \quad (2.7)$$

Cela correspond à remplacer une distance euclidienne par une distance "*p-norme*". La valeur de p peut varier dans [1, ∞[. Plus p est grand, plus l'évaluation est stricte. Cela peut se comprendre par l'examen de deux cas extrêmes : quand $p = 1$, on retrouve une évaluation équivalente à celle du modèle vectoriel (qui sera détaillé dans la section suivante) ; quand $p \to \infty$, l'évaluation est équivalente à celle du modèle booléen standard ou basée sur des ensembles flous. Nous ne donnons pas de preuve ici, mais il est assez facile de prouver les suivants [Salton et al., 1983a] :

$$\begin{cases} R(d, q_i^{bi} \wedge^1 q_j^{bj}) = [b_i * R(d, q_i) + b_j * R(d, q_j)] / (b_i + b_j) \\ R(d, q_i^{bi} \vee^1 q_j^{bj}) = [b_i * R(d, q_i) + b_j * R(d, q_j)] / (b_i + b_j) \end{cases} \quad (2.8)$$

La conjonction et la disjonction deviennent identiques. Cette évaluation correspond à sim0 du modèle vectoriel.

$$\begin{cases} R(d, q_i^{bi} \wedge^\infty q_j^{bj}) = \min (R(d, q_i), R(d, q_j)) \\ R(d, q_i^{bi} \vee^\infty q_j^{bj}) = \max (R(d, q_i), R(d, q_j)) \end{cases} \quad (2.9)$$

Cette évaluation est la même que celle du modèle basée sur des ensembles flous.

Basé sur cette comparaison, le comportement du modèle p-norme varie entre le modèle booléen et le modèle vectoriel comme illustré dans la figure 2.2 :

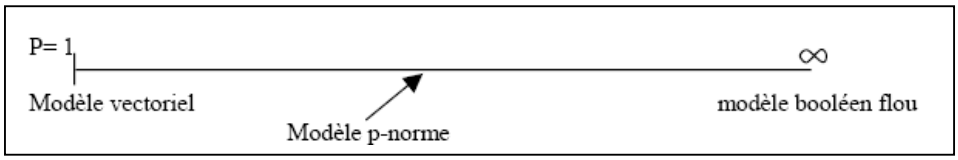


Figure 2.2 : Comportement du modèle p-norme

Le modèle p-norme est intéressant non pas pour sa performance en pratique (bien que les expérimentations montrent qu'il est meilleur que le modèle vectoriel et le modèle booléen flou), mais pour son cadre unificateur. Cela nous aide à comprendre la différence entre le modèle vectoriel et le modèle booléen : un modèle vectoriel peut être considéré comme un modèle booléen dans lequel la différence entre la conjonction et la disjonction est annulée.

3. Modèle vectoriel

Le modèle vectoriel constitue une alternative au modèle booléen. Dans ce modèle, les documents et les requêtes sont représentés par des vecteurs de poids des termes descripteurs. Chaque poids dans le vecteur désigne l'importance du terme correspondant dans le document ou dans la requête. Les vecteurs sont exprimés dans un espace vectoriel défini par l'ensemble des termes construits lors de la phase d'indexation.

Nous définissons :

T : l'univers des termes descripteurs des documents d'un corpus d ;
V(T) : l'espace vectoriel de dimension |T| = n sur IR.

le modèle a été proposé par Salton [Salton, 1971] dans le cadre du système SMART. En fait, ce modèle répose sur :

- l'hypothèse que les documents les plus pertinents sont ceux qui sont les plus proches des requêtes (c'est-à-dire qui contiennent les mêmes termes) ;
- une représentation similaire des documents et des requêtes par des vecteurs de même type ;
- le degré de correspondance entre les deux vecteurs, déterminé par leur mesure de similarité. Plusieurs manières de calculer la similarité (*Sim*) entre deux vecteurs (*d* et *q*) ont été proposées, parmi lesquelles nous citons :

(i) Produit scalaire

Le produit scalaire brut, donné par l'équation (2.10), avantage les documents contenant de nombreux termes descripteurs. C'est pourquoi la plupart des mesures qui l'utilise comprennent un facteur de normalisation.

$$Sim(d,q) = \vec{d}.\vec{q} = \sum_{i=1}^{n} d_i q_i \qquad (2.10)$$

(ii) Mesure du Cosinus

La mesure de cosinus, donnée par l'équation (2.11) et initialement proposée par Salton [Salton, 1989], mesure l'angle que forme le vecteur document et le vecteur requête. Cette mesure permet de trouver les documents situés dans un cône dont le sommet est situé à l'origine, la requête représente l'axe central, et le rayon, le seuil de mise en correspondance. En fait, l'avantage de cette mesure est son indépendance vis-à-vis de la norme des vecteurs à comparer. Ce qui évite de procéder à une normalisation préalable. Le cosinus vaut 1 si les vecteurs sont parallèles et 0 s'ils sont orthogonaux. La mesure de cosinus peut être également interprétée comme une valeur de corrélation.

$$Sim_{Cos}(d,q) = \frac{\vec{d}.\vec{q}}{\left\|\vec{d}\right\| \times \left\|\vec{q}\right\|} = \frac{\sum_{i=1}^{n} d_i q_i}{\sqrt{\sum_{i=1}^{n} {d_i}^2 \sum_{i=1}^{n} {q_i}^2}} \tag{2.11}$$

(iii) Mesure de Tanimoto ou S-mesure

La mesure de Tanimoto [Tanimoto, 1958], donnée par l'équation (2.12), dérive de l'indice de Jaccard (équation (2.13)) utilisé pour effectuer des classifications. Cet indice permet d'évaluer la ressemblance entre deux ensembles en évaluant le quotient entre leur nombre d'éléments communs et leur nombre d'éléments distincts.

$$Sim_{Tan}(d,q) = \frac{\vec{d}.\vec{q}}{\left\|\vec{d}\right\|^2 + \left\|\vec{q}\right\|^2 - \vec{d}.\vec{q}} = \frac{\sum_{i=1}^{n} d_i q_i}{\sum_{i=1}^{n} {d_i}^2 + \sum_{i=1}^{n} {q_i}^2 - \sum_{i=1}^{n} d_i q_i} \tag{2.12}$$

$$Indice_{Jac}(d,q) = \frac{\left|d \cap q\right|}{\left|d \cup q\right| - \left|d \cap q\right|} \tag{2.13}$$

(iv) Mesure de Dice

La mesure de Dice, donnée par l'équation (2.14), dérive de l'indice de Dice (équation (2.15)). Cet indice permet de quantifier l'écart entre deux ensembles en comptabilisant le nombre de termes qu'ils ont en commun.

$$Sim_{Dice}(d,q) = \frac{2\vec{d}.\vec{q}}{\left\|\vec{d}\right\|^2 + \left\|\vec{q}\right\|^2} = \frac{2\sum_{i=1}^{n} d_i q_i}{\sum_{i=1}^{n} {d_i}^2 + \sum_{i=1}^{n} {q_i}^2} \tag{2.14}$$

$$Indice_{Dice}(d,q) = 2 \times \frac{\left|d \cap q\right|}{\left|d\right| + \left|q\right|} \tag{2.15}$$

Avec :

$d = (d_1, d_2, \ldots, d_n)$ vecteur des poids associés aux termes descripteurs dans le document $d \in V(T)$.

$q = (q_1, q_2, \ldots, q_n)$ vecteur des poids associés aux termes descripteurs dans la requête $q \in V(T)$.

$\vec{d}.\vec{q}$ *représente le produit scalaire entre les deux vecteurs* $\vec{d}$ *et* $\vec{q}$.

$\left\|\vec{d}\right\|$ *et* $\left\|\vec{q}\right\|$ *représente les normes Euclidiennes des vecteurs* $\vec{d}$ *et* $\vec{q}$, *avec* $\left\|\vec{d}\right\|^2 = \vec{d}.\vec{d}$

$|d|$ *représente le cardinal de l'ensemble d.*

Si les termes possèdent des poids négatifs[11] ils vont participer à diminuer le score des documents indexés par ces termes. La mesure de similarité varie dans ce cas dans l'intervalle [-1, 1].

Malgré qu'il existe des corrélations entre ces différentes mesures, ces dernières ne donnent pas le même résultat pour le classement des documents. En effet, [Hamers et al., 1989] a démontré que $Sim_{Jac}(d, q) \approx 2Sim_{Cos}(d, q)$, ce qui veut dire qu'en considérant un seuil endeças duquel les documents sont considérés pertinents, la méthode du cosinus tranche plus brutalement que les autres.

Il est également possible d'utiliser une mesure de distance, la plus élémentaire correspond à la distance euclidienne :

$$Dist(d,q) = \sqrt{\sum_{i=1}^{n}(d_i - q_i)^2} \qquad (2.16)$$

Les mesures de similarité décrites précédemment expriment le degré de correspondance entre deux vecteurs. La pertinence système qu'on note R peut ainsi être assimilée à une similarité : $R(d, q) = Sim\ (d, q)$.

Dans le cas de l'utilisation d'une distance, cette dernière est inversement proportionnelle à la pertinence système, la formule précédente devient donc : $R(d, q) = 1/\ Dist\ (d, q)$.

En utilisant ces mesures de similarité, il est possible d'obtenir un classement des documents par degré de pertinence. Le nombre de documents à présenter peut être fixé d'une manière approximative et/ou peut être défini par une valeur de similarité arbitraire (valeur de coupure ou seuil) auquel cas, seuls les documents dont la mesure de similarité est supérieure à la valeur de coupure sont pris en compte.

D'autre part, les mesures de similarité peuvent être utilisées pour comparer des documents entre eux ou des requêtes entre elles.

Dans sa version initiale, le modèle vectoriel se base sur l'hypothèse que les termes sont indépendants, contrairement au modèle booléen, qui permet dans la représentation de la requête d'utiliser des opérateurs logiques qui prennent en considération les connexions sémantiques entre les termes, ce qui pose le problème de manque d'expressivité de la requête vectorielle. Pour remédier à ce problème, l'utilisation conjointe d'un thesaurus pour améliorer considérablement les performances de ce modèle.

Afin de pallier les limites posées par le modèle vectoriel, des extensions ont été proposées. Parmi lesquelles nous citons :

3.1 Modèle vectoriel généralisé

Dans sa version initiale le modèle vectoriel ne permet pas d'exprimer des liens sémantiques entre les termes descripteurs. C'est-à-dire que les vecteurs de base associés aux descripteurs sont deux à deux orthogonaux. Afin de résoudre ce problème [Wong et Raghavan, 1984] ont proposé de transformer l'espace vectoriel initial B pour introduire d'éventuelles corrélations. La description vectorielle d'un document d s'exprime dans une base B', constituée par un

[11] Un poids négatif exprime l'importance de l'absence du critère de recherche dans les documents recherchés.

ensemble de vecteurs représentant des discripteurs virtuels[12] et se substitue à la pseudo-base *B*, de la manière suivante :

$$d = \sum_{x_i \in B'} a_i x_i \tag{2.17}$$

Avec :

- *B' représente la nouvelle base représentant des vecteurs qui vont se substituer aux vecteurs de la base initiale B définie par l'espace de description du corpus.*
- *a_i est le degré de pertinence dans le document d du descripteur virtuel associé au vecteur x_i.*

D'autre part, [Lamirel, 1995] a proposé une nouvelle métrique spécifique qui peut être associée à l'espace vectoriel des descripteurs afin d'introduire l'effet de corrélation dans la mesure de similarité entre un document *d* et une requête *q*.

Cette contribution se base sur l'expression générale du produit scalaire entre deux vecteurs $\vec{u}$ et $\vec{v}$ qui s'écrit : $\langle \vec{u}, \vec{v} \rangle_M = \vec{u}^t M \vec{v}$.

M est la matice associée au produit scalaire, elle définit une métrique non euclidienne dans l'espace qui tient compte d'une manière approfondie, à la fois des liens de synonymie et des liens d'hiérarchie entre les descripteurs. Elle peut être assimilée à la matrice de passage de la base *B* à la base *B'*. Elle a la forme suivante :

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Où chaque coefficient C_{ij} peut être interprété comme le degré de certitude associé à l'inférence floue[13] $i \rightarrow j$.

Le produit scalaire n'étant plus systématiquement commutatif, il fallait alors définir une mesure orientée. La mesure de similarité Cosinus devient en conséquence [Lamirel, 1995] :

$$Sim(d,q) = \frac{d^t M q}{\sqrt{q^t M q} \times \sqrt{d^t M d}} \tag{2.18}$$

d^t et q^t peuvent être assimilés à des vecteurs déduits respectivement de *d* et de *q* en utilisant les inférences floues entre les termes descripteurs. La mesure de similarité devient :

$$Sim(d,q) = \frac{d^t q}{\sqrt{q^t M q} \times \sqrt{d^t d}} \tag{2.19}$$

Puisqu'il s'agit d'une mesure de similarité orientée, Lamirel, à l'image de [Nie, 1988] propose de la prendre globalement en compte de la manière suivante :

$$Sim_{Nie}(d,q) = \alpha Sim(q,d) + \beta Sim(d,q) \tag{2.20}$$

Avec :

- $Sim(q, d)$ peut être assimilée au degré de certitude de l'inférence floue $q \rightarrow d$.

- $Sim(d, q)$ peut être assimilée au degré de certitude de l'inférence floue $d \rightarrow q$.

[12] Deux descripteurs fortement synonymes apparaissant dans *B* pourraient être réduits à un seul descripteur dans *B'*.

[13] Si $P(d \rightarrow q) = 1$, alors le document *d* contient tous les termes de la requête *q*.
Si $P(q \rightarrow d) = 1$, alors le document *d* ne concerne que la requête *q*.

- α et β peuvent être initialisés à ½.

3.2 Modèle vectoriel et domaines sémantiques

[Lamirel, 1995] a introduit la dimension sémantique dans le modèle vectoriel de RI. En effet, son approche consiste à prendre en compte un niveau supplémentaire dans la recherche d'information en considérant des points de vue sur les documents par l'intermédiaire de leurs domaines sémantiques. Dans la méthode proposée, le besoin de l'utilisateur s'exprime généralement par un nombre réduit de descripteurs dans un domaine ciblé afin d'éviter le bruit apparaissant systématiquement lors d'une mise en correspondance globale.

L'idée consiste à partitionner l'espace vectoriel de description des documents en sous-espaces vectoriels associés à ces domaines sémantiques, ainsi :

$$\forall s_i \in P(T), \exists V(s_i) \in V(T) \text{ où } \bigcup_{i=1}^{n} V(s_i) = V(T) \tag{2.21}$$

Avec :

- *$V(S_i)$ correspond au sous-espace vectoriel de dimension $|S_i|$ sur IR associé au domaine sémantique S_i.*
- *L'union des sous-espaces vectoriels associés aux domaines forme l'espace vectoriel de description.*

En cas où l'utilisateur s'intéresse à un domaine sémantique caractérisé par un sous-espace, alors la mesure de similarité entre une requête q et un document d est rapportée au sous-espace (mesure partielle) et s'exprime de la façon suivante :

$$Sim(d,q) = Sim(q_{V(S_i)}, d_{V(S_i)}) \tag{2.22}$$

Pour procéder à un classement global de pertinence à partir des classements partiels il faut vérifier si les valeurs de pertinence sont compatibles. Dans ce cas il est possible de considérer la somme pondérée des valeurs de pertinences partielles entre la requête et le document, données par chacune des mises en correspondance. Les poids utilisés peuvent être considérés comme une mesure d'importance donnée au domaine lors de l'interrogation.

Dans le cas général, le calcul de pertinence global se base sur le calcul d'un rang moyen pondéré ; ce qui correspond à la somme pondérée des rangs d'un document *d* pour une requête *q*, s'exprimant comme suit :

$$Rang(d,q) = \frac{\sum_{S_i} \alpha_{S_i} r_{S_i}(d,q)}{\sum_{S_i} \alpha_{S_i}} \tag{2.23}$$

Avec :

- *α_{S_i} correspond aux poids de pertinence du domaine S_i lors de la mise en correspondance.*
- *r_{S_i} correspond au rang du document d pour la requête q selon la mise en correspondance partielle associée au domaine S_i.*

4. Modèle probabiliste

Cette approche s'intéresse à la probabilité de pertinence des documents. Il n'est plus question de chercher si un document est plus ou moins pertinent mais de chercher une probabilité de pertinence qui est plus ou moins importante. Il s'agit de déterminer pour un document *d*

trouvé la probabilité qu'on obtienne l'information pertinente avec celle qu'on obtienne l'information non pertinente.

Soient *R* et *NR* représentant respectivement la pertinence (*Relevance*) et la non-pertinence (*Non Relevance*). L'idée de base dans un modèle probabiliste est de tenter de déterminer les probabilités $P(R|d)$ et $P(NR|d)$ pour une requête donnée. Ces deux probabilités signifient respectivement que : si on retrouve le document *d*, elles indiquent la probabilité de pertinence et non-pertinence de l'information obtenue [Fuhr, 1992] [Jones et al., 2000].

Une première hypothèse consiste à ne considérer que la présence et l'absence des termes dans le document et la requête dont les termes ne sont pas pondérés. Il s'agit de déterminer les caractéristiques de *R* et *NR* pour une requête donnée. Donc, implicitement, $P(R|d)$ et $P(NR|d)$ correspondent plutôt à $P(R_q|d)$ et $P(NR_q|d)$ pour la requête *q*, mais cet index peut être ignoré pour l'instant.

Il est donc possible de classer les documents selon les deux mesures $P(R|d)$ et $P(NR|d)$ en utilisant la formule suivante :

$$P(R|d) / P(NR|d) \tag{2.24}$$

Ces deux probabilités ne sont pas directement calculables. Ainsi, l'utilisation du théorème de Bayes donne :

$$P(R|d) = P(d|R)\ P(R) / P(d) \tag{2.25}$$

$$P(NR|d) = P(d|NR)\ P(NR) / P(d) \tag{2.26}$$

Ce qui donne :

$$P(R|d) / P(NR|d) = [P(d|R)\ P(R)] / [P(d|NR)\ P(NR)] \tag{2.27}$$

Or, pour une même requête $P(R)$ et $P(NR)$ sont des constantes, la formule devient :

$$P(R|d) / P(NR|d) = P(d|R) / P(d|NR) \tag{2.28}$$

Avec :

- *P(R|d) que la probabilité que d fasse partie de l'ensemble des documents pertinents.*
- *P(R) est la probabilité de pertinence, c'est-à-dire, si on choisit un document au hasard dans le corpus, la probabilité qu'il soit pertinent.*
- *P(d|R) est la probabilité que le document soit choisi (si on prend au hasard un document dans un corpus correspondant alors à la chance qu'il soit d).*

Hypothèse d'indépendance et le modèle de recherche indépendant

Pour estimer $P(d|R)$ et $P(d|NR)$, le document sera décomposé en un ensemble d'"événements". Un événement dénote soit la présence ou l'absence d'un terme dans ce document, c'est-à-dire une série d'éléments $(t_i = x_i)$ où x_i représente l'absence ou la présence du terme t_i dans le document *d*. Ainsi [Fuhr, 1992]:

$$P(d|R) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3, \ldots |R) \tag{2.29}$$

$$P(d|NR) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3, \ldots |NR) \tag{2.30}$$

Dans la théorie des probabilités, la probabilité de la combinaison de plusieurs événements est déterminée comme suit :

$$P(a, b, c, d \ldots|R) = P(a|R) * P(b|a,R) * P(c|a,b,R) * P(d|a,b,c,R) * \ldots \tag{2.31}$$

C'est-à-dire qu'il faut tenir compte des dépendances entre les événements, représentées dans cette formule par des probabilités conditionnelles. Il est vrai que dans le contexte de la RI, les présences et les absences de termes sont dépendantes. Par exemple, si le terme «informatique» apparaît dans un document, il y a plus de chance que le terme « ordinateur » apparaîsse aussi. Ainsi, nous avons : $P(ordinateur = 1 \mid informatique = 1) > P(ordinateur = 1)$.

Le calcul de $P(d|R)$ et de $P(d|NR)$ est complexe, car il faut tenir compte des dépendances suivantes :

$P(t_2 = x_2 \mid t_1 = x_1, R)$, $P(t_3 = x_3 \mid t_1 = x_1, t_2 = x_2, R)$, *etc.*

En conséquence le nombre de documents pertinents d'apprentissage doit être très élevé. Ainsi, l'hypothèse d'indépendance est supposée pour simplifier le calcul de de $P(d|R)$ et de $P(d|NR)$:

Hypothèse d'indépendance : les événements liés aux différents termes sont indépendants. Ainsi [Fuhr, 1992]:

$$P(d|R) = \Pi_{(ti=xi)\in d} P(t_i = x_i \mid R) \quad (2.32)$$

$$P(d|NR) = \Pi_{(ti=xi)\in d} P(t_i = x_i \mid NR) \quad (2.33)$$

Le problème est réduit donc à l'estimation de $P(t_i = x_i|R)$ et $P(t_i = x_i|NR)$, ce qui est beaucoup plus faisable. [Roberston et Sparck-Jones, 1976] considèrent que les termes descripteurs sont attribués sans poids. Seules les termes de la requête sont pondérés à partir du bouclage de pertinence, ce qui permet de calculer une probabilité de pertinence pour chaque terme en fonction de sa distribution parmi les documents pertinents et les documents non pertinent. Ils partent de l'hypothèse que les termes non liés sémantiquement sont distribués également parmi les documents. En fait, la probabilité pour un terme de se retrouver un certain nombre de fois dans un élément d'information suit une loi de Poisson. L'ensemble des documents pertinents et non pertinents jugés par l'utilisateur donnent deux distributions de Poisson. Il est également possible de partir d'une étude priliminaire qui utilise un échantillon de documents.

En effet, avec ces échantillons, il est possible d'estimer $P(t_i = x_i|R)$ et $P(t_i = x_i|NR)$ où R et NR correspondent maintenant respectivement à l'ensemble des documents pertinents et non pertinents parmi les échantillons. Il suffit de construire la table de distribution suivante pour chaque terme t_i :

#doc. pert. contenant t_i	#doc. pert. ne contenant pas t_i	#doc. pert.
#doc. non-pert. contenant t_i	#doc. non-pert. ne contenant pas t_i	#doc. non-pert.
#doc. contenant t_i	#doc. ne contenant pas t_i	#échantillons

Tableau 2.2 : Table de distribution pour chaque terme t_i

Supposons qu'on ait les valeurs suivantes pour t_i:

r_i	$n - r_i$	n
$R_i - r_i$	$N - R_i - n + r_i$	N - n
R_i	$N - R_i$	N

Tableau 2.3 : Table de valeurs du terme t_i

Ainsi :

$p_i = P(t_i = 1|R) = r_i/n$ **et** $(1 - p_i) = P(t_i = 0|R) = (n - r_i)/n$ (2.34)

$q_i = P(t_i = 1|NR) = (R - r_i)/(N - n)$ **et** $(1-q_i) = P(t_i = 0|NR) = (N–R- n + r_i)/(N - n)$ (2.35)

Ici, pour simplifier les formules (2.34) et (2.35), on dénote $P(t_i = 1|R)$ par p_i, $P(t_i = 0|R)$ par $(1-p_i)$, et $P(t_i = 1|NR)$ par q_i, $P(t_i = 0|NR)$ par $(1-q_i)$.

La formule (2.28) précédente s'écrit donc :

$$P(d|R) / P(d|NR) = \frac{\prod_{t_i} P(t_i = 1|R)^{x_i} P(t_i = 0|R)^{(1-x_i)}}{\prod_{t_i} P(t_i = 1|NR)^{x_i} P(t_i = 0|NR)^{(1-x_i)}} = \frac{\prod_{t_i} p_i^{x_i} (1 - p_i)^{(1-x_i)}}{\prod_{t_i} q_i^{x_i} (1 - q_i)^{(1-x_i)}} \quad (2.36)$$

Soit $g(d) = log[P(d|R) / P(d|NR)]$, alors :

$$\begin{aligned} g(d) &= \sum_{t_i} [x_i \log p_i + (1 - x_i) \log(1 - p_i) - x_i \log q_i + (1 - x_i) \log(1 - q_i)] \\ &= \sum_{t_i} x_i [\log(\frac{p_i}{1 - p_i}) - \log(\frac{q_i}{1 - q_i})] + \sum_{t_i} \log(\frac{1 - p_i}{1 - q_i}) \end{aligned} \quad (2.37)$$

Remarquons que la partie $\sum_{t_i} \log(\frac{1 - p_i}{1 - q_i})$ ne dépend pas du document (i.e. x_i). C'est une constante (notée C) pour n'importe quel document. La formule (2.37) s'écrit :

$$g(D) = \sum_{t_i} x_i \log \frac{p_i (1 - q_i)}{q_i (1 - p_i)} + C \quad (2.38)$$

Le poids du terme t_i, noté w_i s'écrit:

$$w_i = \sum_{t_i} \log \frac{p_i (1 - q_i)}{q_i (1 - p_i)} = \log \frac{\frac{r_i}{n} \frac{N - R_i - n + r_i}{N - n}}{\frac{n - r_i}{n} \frac{R - r_i}{N - n}} = \log \frac{r_i / (n - r_i)}{(R - r_i) / (N - R_i - n + r_i)} \quad (2.39)$$

Ainsi, le poids pour un document est déterminé par $p(d) \propto g(d)$:

$$p(d) = \sum_{t_i} x_i w_i \quad (2.40)$$

Une généralisation de la fonction a été proposée par [Croft, 1981] en introduisant une pondération non binaire des termes. Une espérance mathématique est alors exprimée par l'équation (2.41) :

$$E(d) = \sum_{i=1}^{n} P(\delta_i) \log \frac{p_i (1 - q_i)}{q_i (1 - p_i)} \quad (2.41)$$

$P(\delta_i)$ est la probabilité d'indexation du document d par le terme i.

Dans [Chen, 1995], le modèle probabiliste est défini principalement par deux types de stratégies de recherche pour l'apprentissage : *document oriented strategy* et *query oriented strategy*.

En effet, la *document oriented strategy* correspond à la recherche des requêtes pertinentes pour un document donné (probabilité que $d \rightarrow q$), ce qui revient donc à estimer la pertinence des autres requêtes pour le même document. La *query oriented strategy* permet de determiner les documents pertinents pour une requête donnée (probabilité que $q \rightarrow d$), ce qui revient à estimer la pertinence pour le reste de la collection mais toujours pour la même requête.

Dans les deux stratégies, l'information fournie est exploitée par le *Relevance Feedback* effectué sur une collection. Cette information permet d'estimer la probabilité de pertinence soit pour le reste de la collection soit pour l'ensemble des requêtes, mais elle ne peut pas être généralisée à tous les deux. C'est la raison pour laquelle [Fuhr et Buckley, 1991] ont proposé la *feature oriented strategy*, une troisième stratégie de recherche pour l'apprentissage basée sur le modèle probabiliste. En effet, cette stratégie consiste à adopter des caractéristiques (longueur des documents, nombre de termes, etc.) au lieu des termes qui correspondent aux documents ou aux requêtes. En conséquence, cette stratégie donne une forme plus générale pour l'apprentissage probabiliste pour l'ensemble des documents et des requêtes. Pour ce faire, Christian Fuhr utilise des méthodes de régression et l'algorithme ID3 pour la construction de l'arbre de décision pour l'indexation et la recherche d'information.

Cependant, l'hypothèse d'indépendance de termes n'est pas toujours vérifiée. Par exemple, un document pertinent qui possède le terme « artificielle » en réponse à une requête incluant le terme « intelligence artificielle », a plus de chance d'inclure le terme intelligence que n'importe quel autre terme choisi au hasard.

Plusieurs extensions au modèle probabiliste originel ont donné des meilleurs résultats dans la sélection des documents pertinents que ce dernier. Parmi ces modèles citons : les modèles à base de réseaux d'inférence et des règles de Bayes et le modèle de Poisson (CONSTRUCTOR [Crawford et al., 1991], INQUERY [Turtle et Croft, 1991], OKAPI [Walker et al., 1997]).

5. Reformulation de requête dans ces modèles

Il est souvent difficile, pour l'utilisateur, de formuler exactement son besoin en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent parfois pas. Retrouver des informations pertinentes en utilisant la seule requête initiale de l'utilisateur est toujours difficile, et ce à cause de l'imprécision de la requête. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence du système, une étape de reformulation de la requête est souvent utilisée. La requête initiale est traitée comme un essai pour retrouver de l'information désirée ou ciblée. Les documents initialement présentés sont examinés et une formulation améliorée de la requête est construite, dans l'espoir de retrouver des documents plus pertinents. La reformulation de la requête se fait en deux étapes principales : trouver des termes d'extension à la requête initiale, et pondérer les termes dans la nouvelle requête.

[Dunlop, 1997] distingue deux types de rétroaction :

- La *rétroaction négative*, si un document pertinent est jugé non pertinent ; dans ce cas un grand changement va se produire. Inversement, si un document non pertinent est jugé non pertinent, la requête initiale ne changera pas vraiment.
- La *rétroaction positive*, si un document pertinent est jugé pertinent ; dans ce cas la transformation de la requête initiale est minime. Inversement, si un document non pertinent est jugé pertinent, alors un grand changement va se produire dans la requête

initiale. Autrement dit, l'effet du *feedback* est sensé être inversement proportionnel à l'importance du document retrouvé par rapport à la requête.

5.1 Reformulation de la requête dans le modèle booléen

Selon [Nie, 2004], il a été observé qu'une requête qui est formulée comme une longue conjonction est très difficile à satisfaire. En effet, la réponse est généralement vide exprimant un silence. Par contre, une longue disjonction est très facile à satisfaire : nombreux sont les documents qui vont être récupérés dans la réponse. C'est surtout pour résoudre le premier problème que la reformulation de requête est introduite.

Dans le cas d'une longue requête en conjonction, si un document satisfait la plupart des termes de la requête, on peut penser qu'il satisfait en partie le besoin de l'utilisateur. Il est dans ce cas préférable de proposer un ensemble de documents partiellement satisfaisants que de ne pas donner de réponse. Ainsi, le processus de reformulation consiste à examiner le nombre de documents en réponse. Si le nombre est très peu élevé, alors on peut assouplir la requête initiale en supprimant un terme selon le schéma suivant :

- Soit la requête initiale $q = (t_1 \wedge t_2 \wedge t_3 \wedge \ldots \wedge t_n)$. Si aucun document n'a été trouvé, une première methode consiste à relaxer la requête de la manière suivante :

  $q' = (t_2 \wedge t_3 \wedge t_4 \wedge \ldots \wedge t_n) \vee (t_1 \wedge t_3 \wedge t_4 \wedge \ldots \wedge t_n) \vee \ldots (t_1 \wedge t_2 \wedge t_3 \wedge \ldots \wedge t_{n-1})$

- Si aucun résultat satisfaisant n'est obtenu, il est possible de poursuivre cette démarche en relaxant chaque sous-requête par ablation supplémentaire de termes.

Une seconde methode envisageable consiste à supprimer le terme le plus difficile à satisfaire (celui qui correspond au minimum de documents récupérés).

En fait, ces deux types de méthodes ne sont cependant justifiés que par des besoins pratiques.

Une autre méthode reposant sur de meilleurs fondements théoriques consiste à considérer la relation que peuvent entretenir les termes comme une relation d'implication (nous avons soit $a \rightarrow b$ soit $b \rightarrow a$). Dans ce cas, le calcul de correspondance est basé sur le calcul d'une implication entre le document et la requête.

Pour l'extension de la requête le schéma suivant est proposé :

- Si b apparaît dans une requête q et si nous avons $a \rightarrow b$, alors la requête peut être étendue en remplaçant b par $(b \vee a)$;
- Si $\neg a$ apparaît dans une requête q et si nous avons $a \rightarrow b$, alors la requête peut être étendue en remplaçant $\neg a$ par $\neg(a \wedge b)$;

Ce processus est en accord avec la logique et l'idée intuitive liée à la notion d'implication. En effet :

- Si $q = b \wedge c$ et si on est en présence de l'implication $a \rightarrow b$ alors la requête étendue devient $q' = (b \vee a) \wedge c$; Par ailleurs, puisque $(b \vee a) \rightarrow b$, nous avons bien $q' \rightarrow q$.
- Si $q = \neg a \wedge c$, alors la requête étendue devient $q' = \neg(a \wedge b) \wedge c = (\neg a \vee \neg b) \wedge c$. De la même manière, nous avons bien $q' \rightarrow q$ puisque si $a \rightarrow b$ alors $\neg a \rightarrow \neg b$.

D'autre part, certains auteurs suggèrent d'associer une importance à chaque terme de la requête pour que l'utilisateur puisse différencier des termes très importants de ceux qui le sont moins. On peut voir certaines propositions dans les travaux des [Waller et Kraft, 1979], [Radecki, 1979] et [Kraft et al., 1983].

5.2 Reformulation de la requête dans le modèle vectoriel

Les techniques du *Relevance feedback* appliquées au modèle de recherche vectoriel ont été dominées par les travaux de Rocchio [Rocchio, 1971] puis de Ide [Ide, 1971]. Comme son nom l'indique, le *feedback* dans le cadre de ce modèle part du principe que la requête initiale formulée par l'utilisateur sert au système à identifier une zone ou région de l'espace d'index de termes qui contient des documents pertinents. N'ayant pas d'autres informations sur les caractéristiques des documents enregistrés, la requête initiale constitue l'unique indice de départ. En introduisant dans le cycle la requête initiale et les documents pertinents et non pertinents courants sélectionnés, l'utilisateur, ce faisant, fournit des informations au système qui lui permettent de reformuler automatiquement le profil de la requête de sorte que les documents générés au fur et à mesure des itérations tendent de plus en plus à se rapprocher des besoins de l'utilisateur.

En effet, dans le modèle vectoriel, le *feedback* négatif se comporte sensiblement de manière opposée au *feedback* positif et ce, quand il s'agit d'introduire des poids négatifs pour les termes que l'utilisateur ne désire pas retrouver dans les documents pertinents recherchés. Il agit comme un filtre pour ne garder que les documents qui répondent à certains termes. Par contre le *feedback* positif agit conformément à l'idée proposée par [Dunlop, 1997] dans les deux cas où il s'agit d'introduire des poids positifs ou des poids négatifs.

Pour réaliser la reformulation de requête deux méthodes sont possibles :

- Créer un vecteur additionnel pour la requête ;
- Ajouter des termes dans le même vecteur correspondant à la requête initiale.

Dans la première méthode, tous les termes ajoutés forment un nouveau vecteur. Le calcul de correspondance va se faire en deux temps : d'abord le vecteur de similarité avec le vecteur initial puis le calcul avec le vecteur d'extension. Le résultat final est alors une combinaison pondérée des deux vecteurs. Dans la seconde méthode, les termes nuls peuvent être transformés en valeurs non nulles. Les documents contenant ces termes verront alors leur mesure de similarité augmenter.

Le modèle vectoriel est basé sur une hypothèse d'indépendance entre les termes du vecteur. L'extension de requête par ajout d'un terme dans le même vecteur peut, dans certains cas, contredire cette hypothèse car le terme ajouté peut être éventuellement relié à un terme de la requête. Le resultat de cette interdépendance conduit à considérer le terme avec un poids deux fois plus élevés car il apparaît en quelque sorte deux fois dans le vecteur. L'extension va donc fortement altérer le sens de la requête initiale.

Le calcul séparé de l'extension de la requête par rapport à celle effectuée dans le vecteur initial tend à minimiser ce problème de dépendance. Cependant, il reste à déterminer la combinaison pondérée des deux résultats de calcul avec les deux vecteurs. Cette dernière ne peut être déterminée que d'une manière empirique.

5.3 Reformulation de la requête dans le modèle probabiliste

Robertson et Sparck-Jones [Robertson et Sparck-Jones, 1976] ont développé une formule de pondération des termes (donnée par l'équation (2.39)) basée sur la distribution des termes de la requête dans les documents jugés pertinents et les documents jugés non pertinents par l'utilisateur. Une variation de cette formule de base a été définie dans le but de calculer les nouveaux poids pour les termes de la nouvelle requête lors du processus de réinjection de pertinence :

$$w_i = \log \frac{p_i(1-q_i)}{q_i(1-p_i)} = \log \frac{(r_i + 0.5) \,/\, (n_i - r_i + 0.5)}{(R - r_i + 0.5) \,/\, (N - R - n_i + r_i + 0.5)} \quad (2.42)$$

Avec :

$$p_i = \frac{r_i + 0.5}{R+1} \text{ et } q_i = \frac{n_i - r_i + 0.5}{N - R + 1}$$

r_i correspond au nombre de documents pertinents qui sont indexés par le terme t_i ;

n_i correspond au nombre de documents qui sont indexés par le terme t_i ;

R correspond au nombre de documents pertinents ;

N correspond au nombre de tous les documents dans la collection ;

0.5 est un facteur d'ajustement.

Harman [Harman, 1992] a montré que l'utilisation du coefficient 0,5 dans la formule (2.42) comme facteur d'ajustement permet d'augmenter la précision. Cette augmentation a été de l'ordre de 25% sur la base Cranfield.

Haines et Croft [Haines et Croft, 1993] ont défini une méthodologie de repondération en utilisant une version révisée de la formule de pondération de Sparck-Jones :

Recherche initiale : $w_{ijk} = (C + idf_i).f_{ik}$ (2.43)

Feedback : $w_{ijk} = \left[C + \log \frac{p_{ij}(1-q_{ij})}{q_{ij}(1-p_{ij})} \right].f_{ik}$ (2.44)

Avec :

w_{ijk} : le poids du terme t_i dans la requête j et le document k ;

idf_i : fréquence absolue du terme t_i dans la collection ;

p_{ij} : probabilité que le terme t_i soit assigné à un ensemble de documents pertinents pour une requête j.

$p_{ij} = (r + 0.5)/(r + 1)$ si $r > 0$, $p_{ij} = 0.01$ si $r = 0$;

q_{ij} : probabilité que le terme t_i soit apparaisse dans un ensemble de documents non pertinents pour une requête j.

$q_{ij} = (n - r + 0.5)/(N - R + 1)$;

$$f_{ik} = K + (1-K).\frac{freq_{ik}}{\max(freq_k)}$$

$freq_{ik}$: la fréquence du terme t_i dans le document k ;

$freq_k$: la fréquence maximale d'un terme dans le document k ;

C, K : constantes.

Dans le modèle probabiliste, le feedback positif agit conformément à l'idée intuitive proposée par [Dunlop, 1997] s'appuyant sur le principe d'extrapolation à partir des documents jugés *a priori* pertinents pour déterminer la pertinence du reste des documents.

D'autre part, dans [Salton et Buckly, 1988] les méthodes d'activation/propagation ont été recommandées pour l'expansion du vocabulaire de la recherche et pour compléter les documents trouvés. Il s'agit d'exploiter les associations de paires de termes, les citations et les

indicateurs bibliographiques dans une représentation en réseaux de nœuds liés. L'efficacité de la méthode proposée dépend de la qualité effective de la représentation (nœuds et liens) ainsi que des règles d'activation. Le modèle proposé dans INQUERY [Turtle et Croft, 1991] est basé sur des réseaux d'inférence utilisant des règles de Bayes pour la RI (document network et query network) et constitue un exemple approprié de ces méthodes. Ce modèle sera détaillé dans le chapitre suivant.

5.4 Autres approches de reformulation de requêtes

Plusieurs méthodes de *relevance feedback* (RF) ont été incorporées dans SMART par [Salton et al., 1983b] [Salton et Buckley, 1990], à savoir la version de la formule de Rocchio et des extensions de celle-ci en ce qui concerne le fragment qui fait intervenir le *feedback* négatif. Par ailleurs, plusieurs travaux de recherche ont prouvé que le RF améliorait les résultats de la recherche selon les deux critères d'évaluation souvent utilisés dans les SRI à savoir les taux de rappel et de précision ; quoi que le *feedback* négatif, c'est-à-dire la mention des documents non pertinents, ne donne pas de résultats aussi satisfaisants que le *feedback* positif.

D'après [Harman, 1988], les techniques de reformulation de requêtes sont moins coûteuses quand il s'agit de guider l'utilisateur que quand il s'agit de procéder automatiquement. Dans ces perspectives, l'auteur propose de guider l'utilisateur désirant améliorer sa requête, dans le cadre d'une recherche en ligne, en lui fournissant une liste de mots qui correspondent aux voisins proches et aux différentes variantes des termes de la requête initiale. Le nombre de termes à ajouter et le nombre d'itérations à effectuer sont des paramètres à fixer automatiquement. Les expérimentations ont été menées en utilisant la collection Cranfield. Le même auteur [Harman et al., 1992] propose d'ajouter des descripteurs pertinents à la requête à partir des documents pertinents et de réduire le poids des termes non pertinents à partir des documents non pertinents. En fait, l'auteur affirme que si la pertinence (utilisateur) est affectée aux documents non pertinents (système), alors la requête transformée tend vers la généralisation. Par contre, si la pertinence (utilisateur) est affectée aux documents jugés pertinents (système), alors la requête transformée permet d'augmenter la recherche dans une direction.

De leur part, [Buckley et Salton, 1995] ont proposé une démarche qui se base sur la formule de Rocchio et sur l'amélioration dynamique des poids en testant leur changement possible sur le resultat de la recherche. Le processus est répété pour chaque terme dont le poids a été modifié et ce pour tester s'il y a effectivement une amélioration du résultat après changement. Selon Salton, les requêtes optimisées permettent d'obtenir un résultat meilleur (10-15%) par rapport aux requêtes initiales.

Le problème posé est celui de l'incohérence de comportement du *feedback* négatif vis-à-vis du *feedback* positif, ainsi que celui des mauvais résultats généraux. Dans ces perspectives, [Lamirel, 1995] a proposé une solution utilisant la transformée orthogonale afin de résoudre le problème de *feedback* négatif. En fait, cette solution consiste à redéfinir globalement les équations vectorielles du RF. L'auteur propose de considérer des informations nouvelles par rapport aux documents rejetés (jugés non pertinents). Ces informations correspondent à la partie complémentaire par rapport à celle contenue dans les documents rejetés. La reformulation négative vectorielle d'une requête consiste alors à approcher celle-ci d'une direction complémentaire à la direction des rejets plutôt que de l'éloigner de la direction des rejets elle-même, qui est le cas du *feedback* négatif usuel.

Pour sa part, [Lee, 1998] propose de combiner plusieurs méthodes pour le RF. En effet, l'auteur a proposé de fusionner les différents résultats, provenant de plusieurs requêtes reformulées, pour améliorer le degré de pertinence [Porter, 1982][Smeaton, 1983][Belkin et

al., 1993] [Fox et Shaw, 1994]. Les méthodes dont Lee a proposé de fusionner se basent sur des formules dérivées du modèle probabiliste en plus de celle de Rocchio, comme suit :

- La formule de [Ide, 1971] qui est dérivée de celle de Rocchio mais en éliminant les facteurs normalisation exprimés respectivement par les nombres de documents pertinents et non pertinents tout en limitant le nombre de documents non pertinents :

$$Q^{new} = \alpha.Q^{old} + \beta.\sum_{reldocs} w_{t_i} - \gamma T_{nonreldocs} \tag{2.45}$$

  $T_{nonreldocs}$ *correspond au vecteur des documents qui sont classés les plus moins pertinents.*

- La formule Pr_cl [Croft, 1979] liée au modèle probabiliste est déjà donnée par l'équation (2.42).

- La formule Pr_adj [Roberston, 1986] qui correspond à une version modifiée de la formule Pr_cl :

$$w_i = \log \frac{p_i(1-q_i)}{q_i(1-p_i)}, \text{ avec } p_i = \frac{r_i + n_i/N}{R+1} \text{ et } q_i = \frac{n_i - r_i + n_i/N}{N-R+1} \tag{2.46}$$

- La formule S_rpi [Fuhr et Buckley, 1991] qui correspond aussi à une version modifiée de la formule Pr_cl :

$$w_i = \log \frac{p_i(1-q_i)}{q_i(1-p_i)}, \text{ avec } p_i = \sum_{reldocs} \frac{w_{t_i}}{|reldocs|} \text{ et } q_i = \sum_{nonreldocs} \frac{w_{t_i}}{|nonreldocs|} \tag{2.47}$$

Sur un extrait de la collection TREC, [Lee, 1998] a montré que la requête reformulée à partir de la formule initiale de Rocchio est la plus proche de la requête originelle que toutes les autres. Il a montré également que les différentes requêtes étendues donnent des résultats différents et qu'une amélioration peut être apportée au résultat de la recherche, si les résultats respectifs sont combinés.

6. Conclusion

Les modèles présentés dans ce chapitre ont considéré le corpus de documents comme une composante statique, ce qui est loin d'être le cas. De sa part, l'utilisateur a été considéré aussi comme un acteur passif ou partiellement actif, en exploitant son évolution pour opérer une reformulation de la requête, dans le processus de recherche. Afin d'améliorer leurs performances, ces modèles ont été explorés en les combinant avec d'autres moèles, qui prennent en considération le caractère interactif entre le système et l'utilisateur et la nature évolutive de la collection.

En fait, l'idée consiste à combiner plusieurs méthodes de représentation à la fois des requêtes et des documents pour la Recherche d'Information. Cette combinaison permet d'améliorer le degré de pertinence des documents retrouvés. Il s'agit également de combiner les différentes stratégies de recherche et d'explorer plusieurs méthodes de *relevance feedback*.

D'autre part, pour tenir compte des liens sémantiques entre les termes, une extension du modèle probabiliste a été proposée en utilisant les réseaux Bayésiens, d'autres modèles utilisent les réseaux possibilistes. Ces deux modèles de RI seront détaillés et comparés dans le chapitre suivant.

Chapitre 3

Modèle Bayésien versus Modèle Possibiste de Recherche d'Information

Avec la croissance de la taille des bases de données de nos jours est née la nécessité d'automatiser le traitement de cette grande masse de données, automatiser le raisonnement et la prise de décision. Il serait donc intéressant d'avoir un ou plusieurs systèmes permettant de faire le lien entre les observations et la réalité pour un objectif précis (aide à la prise de décision), et cela, même lorsque les observations sont incomplètes et/ou imprécises. Les réseaux bayésiens (RB) apportent des solutions efficaces à ces insuffisances par leurs représentations graphiques compactes des problèmes réels complexes et leur rapidité en temps de calcul. En effet, l'utilisation des réseaux bayésiens (RB) en RI est apparue dans les années 1980 [Frisse, 1988][Frisse et Cousins, 1989] mais elle s'est largement développée par les travaux de Turtle [Turtle et Croft, 1990][Turtle et Croft, 1991] suivis d'autres [Ribeiro-Neto et al., 1996][Silva et al., 2000][De Campos et al., 2002][Calado et al., 2003].

Néanmoins, ce formalisme a ses limites. En effet, il modélise l'univers de manière causale et ne permet pas la modélisation de relations d'interdépendances, alors que le besoin de telles relations s'en fait souvent ressentir. De plus la complexité de la modélisation et des calculs augmente de manière exponentielle par rapport à la taille de l'univers.

Le modèle possibiliste quantitatif de RI, proposé par [Brini et al., 2004abc], tente de répondre en partie à ces limites. Tout d'abord, la pertinence est interprétée dans un cadre possibiliste. Ce cadre est plus à même de prendre en compte l'ignorance partielle qui peut affecter les informations utilisées dans les différents calculs. En fait, le modèle sépare les raisons de sélectionner un document pertinent de celles de le rejeter, en utilisant deux mesures : la nécessité et la possibilité. Les documents nécessairement pertinents sont ceux qui doivent figurer en haut de la liste des documents restitués et doivent permettre une certaine efficacité du système. Les documents possiblement pertinents sont ceux qui répondraient éventuellement à la requête utilisateur. Ils figurent dans la liste des documents restitués classés à la suite des documents nécessairement pertinents ou à défaut (si le système n'en trouve pas) ils sont considérés comme une réponse plausible.

Afin de permettre cette interprétation de la pertinence, la pondération des termes dans les documents doit être également réinterprétée. Il a été montré dans [Bookstein et Swanson, 1974] [Harter, 1975] que tous les termes d'indexation ne se comportent pas de la même manière dans une collection de documents. Harter fait une distinction entre les mots informatifs appelés aussi mots « spécialisés », qui se focalisent sur un type de documents et les mots non informatifs, non spécialisés qui sont distribués de manière normale sur l'ensemble des documents de la collection. Ceci va dans le sens de l'interprétation de la pertinence ; en effet les termes des documents jouent des rôles différents. Dans un document, il existe des termes fréquents importants (informatifs), nécessaires dans la représentation du document, donc nécessaires, pour décider de la pertinence de ce document vis-à-vis de la requête, et d'autres termes moins informatifs, qui ne sont que possiblement intéressants pour représenter le contenu du document.

La logique possibiliste offre un bon cadre pour représenter ces deux notions. En effet, le modèle possibiliste affecte à chaque terme d'indexation deux valeurs qui traduisent respectivement la certitude et la possibilité qu'un terme d'indexation soit "bon". Le dernier avantage (spécificité) de ce modèle réside dans sa prise en compte explicite de l'absence des termes de la requête dans le document lors de l'évaluation de la pertinence de ce document vis-à-vis de la requête.

Notre apport consiste à proposer une extension de ce modèle pour permettre une transition de l'approche quantitative (numérique) à une nouvelle approche qualitative (ordinale) pour un modèle possibiliste de Recherche d'Information. Cette extension sera détaillée dans le chapitre 4 de cette thèse.

Nous commençons, dans la première section, par introduire la notion des réseaux bayésiens (RB) ainsi que leur principe de raisonnement. Dans la deuxième section nous mettrons l'accent en particulier sur le modèle de RI basé sur les réseaux Bayésiens. La reformulation de requêtes dans les modèles de RI basé sur les RB fera l'objet d'une quatrième section. Nous decrivons ensuite, dans une cinquième section, le cadre théorique sur lequel repose l'approche possibiliste, à savoir les Réseaux Possibilistes (RP). La sixième section présente le modèle possibiliste quantitatif de RI. La reformulation de requêtes dans les modèles de RI basé sur les RP fera l'objet d'une septième section. La dernière section expose un bilan comparatif résumant les différences entre ces deux modèles de RI.

1. Les Réseaux Bayésiens

L'un des enjeux principaux dans le domaine de la recherche en Intelligence Artificielle est d'être capable de concevoir et de développer des systèmes dynamiques et évolutifs. De ce fait, ces derniers doivent être équipés de comportements intelligents qui peuvent apprendre et raisonner.

Mais dans la plupart des cas, la connaissance acquise n'est pas toujours adéquate pour permettre au système de prendre la décision la plus appropriée. Pour répondre à ce genre de questions, plusieurs méthodologies ont été proposées, mais seules les approches probabilistes s'adaptent mieux non seulement au raisonnement avec la connaissance et la croyance incertaine, mais aussi à la structure de la représentation de la connaissance. Ces approches probabilistes sont appelées "*Réseaux Bayésiens"* [Howard et Matheson, 1981][Pearl, 1988], mais sont aussi connues sous le nom de "*Belief Networks*", "*Causal Networks*".

Les Réseaux Bayésiens (RB) sont la combinaison des approches probabilistes et de la théorie de graphes. Autrement dit, ce sont des modèles qui permettent de représenter des situations de raisonnement probabiliste à partir de connaissances incertaines. Ils sont une représentation efficace pour les calculs d'une distribution de probabilités [Cornuéjols et Miclet, 2002].

Par ailleurs, les Réseaux Bayésiens doivent leurs noms aux travaux de Thomas Bayes (1702, 1761) au dix-huitième siècle sur « la probabilité des causes », travaux repris plus tard par LAPLACE et CONDORCET. Ils visent à faciliter la description d'une collection de croyance en rendant explicite les relations de causalité et de l'indépendance conditionnelle parmi ces croyances et à fournir un moyen plus efficace pour mettre à jour les forces de croyances (distribution conjointe de probabilité) lorsque des nouvelles évidences sont observées [Kim et Pearl, 1987].

1.1 Définition

Un Réseau Bayésien est défini par [Pearl, 1988] :

- Un graphe acyclique orienté G, $G = (V, E)$, où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G,
- Un espace probabiliste fini (Ω, Z, p),
- Un ensemble de variables aléatoires définies sur (Ω, Z, p), tel que :

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid Parents(V_i)) \quad (3.1)$$

Où $Parents(V_i)$ est l'ensemble des parents (causes) de V_i dans le graphe.

Un Réseau Bayésien est alors constitué de deux composantes :

- Un graphe causal orienté acyclique : il est la représentation qualitative de la connaissance. S'il y a un arc du nœud X vers le nœud Y, c'est que la variable X a une influence directe sur la variable Y (X cause Y).
- Un ensemble de distributions locales de probabilités : il est la représentation quantitative de la connaissance (paramètres du réseau). A chaque nœud est associé une Table de Probabilités Conditionnelles (TPC) qui quantifie les effets de des parents.

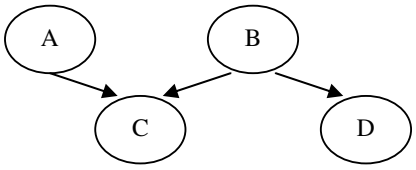
Exemple : [Pearl, 1988]

Ce matin-là le temps est clair et sec, M.X sort de sa maison. Il s'aperçoit que la pelouse de son jardin est humide. Il se demande s'il a plu la nuit, ou s'il a simplement oublié de débrancher son arroseur automatique. Il jette un coup d'œil à la pelouse de son voisin, et s'aperçoit qu'elle est également humide. Il en déduit alors qu'il a plu, et il décide de partir au travail sans vérifier son arroseur automatique.

La représentation graphique du modèle causal utilisé est dans la figure 3.1. Cette figure représente un Réseau Bayésien simple contenant quatre variables binaires, on peut écrire aussi :

$$P(A, B, C, D) = P(A).P(B).P(C \mid A, B).\ P(D \mid B) \quad (3.2)$$

Où :
- *A : Arroseur en marche ;*
- *B : Il a plu pendant la nuit ;*
- *C : Herbe du jardin humide ;*
- *D : Herbe du jardin voisin humide ;*



P(A = V) = 0,4
P(A = F) = 0,6
P(B = V) = 0,7
P(B = F) = 0,3

A	B	P(C = F)	P(C = V)
V	V	0,3	0,7
V	F	0,05	0,95
F	V	0,4	0,6
F	F	0,78	0,22

B	P(D = V)	P(D = F)
F	0,5	0,5
V	0,58	0,42

Figure 3.1 : Exemple de Réseau Bayésien

1.2 Principe du Réseau Bayésien

Les Réseaux Bayésiens (RB) sont des modèles probabilistes qui s'appuient sur des graphes traduisant par des nœuds les variables du système et par des arcs l'existence de liaisons directes entre ces variables.

L'étude d'un modèle de Réseau Bayésien nécessite une base de données et cherche à fournir à cette base une modélisation sous forme de graphe caractérisant les dépendances conditionnelles des différentes variables. Elle se déroule en deux phases [Hallouli, 2004] :

- ***Apprentissage ou constitution du réseau* :** Il s'agit ici de trouver la structure et les probabilités associées du réseau, à partir des données de la base et de traitements principalement statistiques.
- ***Inférence Bayésien* :** A partir des résultats de la première phase, le réseau permet la propagation d'information à l'intérieur de la structure, permettant toute interrogation sur la base et peut fournir pour chaque état partiel ou complet de la base (instanciation partielle ou complète des variables de celle-ci) des probabilités d'occurrence de toutes les valeurs possibles de toutes les variables.

1.3 Construction de la structure du RB par apprentissage

La structure d'un RB est l'ensemble des arcs du graphe orienté sous-jacent au réseau. Dans certaines situations, la structure est fournie par un expert. Si ce n'est pas le cas, on fait l'apprentissage à partir des données complètes ou incomplètes. La recherche de la structure est un problème difficile principalement à cause du fait que l'espace de recherche est de taille super-exponentielle en fonction du nombre de variables. Le problème confronté est : comment choisir la meilleure structure d'un Réseau Bayésien ?

Il y a deux approches générales de construction de la structure d'un Réseau Bayésien par apprentissage [François et Leray, 2004]. L'une est basée sur la recherche et des méthodes de marquage (search and scoring), l'autre est basée sur des méthodes d'analyses de dépendances.

La première approche est de nature heuristique, elle consiste à chercher la meilleure structure qui s'adapte aux données. Elle commence avec un graphe déconnecté, utilise des méthodes de recherche pour ajouter des arcs et teste par l'usage d'un score si la nouvelle structure est meilleure que l'ancienne. Dans la deuxième approche, le problème est vu différemment. Les algorithmes de cette approche essayent de découvrir les dépendances des données et puis emploient ces dépendances pour impliquer la structure.

Chacun des deux approches admet des avantages et des inconvénients. Généralement l'approche basée sur l'analyse des dépendances est plus efficace pour un réseau dont la structure n'est pas trop compliquée, mais la majorité de ces algorithmes nécessitent un nombre exponentiel de tests sur l'indépendance conditionnelle[14].

François et Leray [François et Leray, 2003] ont développé une étude comparative des algorithmes de construction de la structure d'un Réseau Bayésien par apprentissage [Meganck, 2006]. Cette étude porte sur les algorithmes MWST (arbre de recouvrement maximale), PC, K2 et GS (recherche gloutonne). Les auteurs ont déclaré que, l'algorithme MWST donne un graphe proche du graphe d'origine, malgré le fait que cette méthode ne

[14] Soit deux variables aléatoires X et Y. On dit que X et Y sont indépendantes conditionnellement à Z et on note ($X \perp Y|Z$) si l'une des propriétés suivantes sont vérifiées :

- $P(X|Y, Z) = P(X|Z)$
- $P(X, Y|Z) = P(X|Z).\ P(Y|Z)$

parcourt que l'espace (plus pauvre) des arbres. L'heuristique PC donne également de bons résultats. Cette méthode construit des structures avec peu d'arcs, mais qui sont presque tous pertinents. La méthode K2 est très rapide et est souvent utilisée dans la littérature. Elle reste cependant trop sensible à l'initialisation. Deux ordonnancements différents donnent deux Réseaux Bayésiens différents. Pour un ordre fixé, K2 trouve toujours le même graphe. Par contre en changeant d'ordonnancement, le graphe final change radicalement. K2 est employé avec l'algorithme MWST afin de donner de bons résultats. L'algorithme GS est également robuste face à la variation de la taille de l a base d'exemples surtout s'il est initialisé avec l'arbre obtenu par MWST.

1.4 Inférence dans les Réseaux Bayésiens

Le Réseau Bayésien permet de représenter un ensemble de variables aléatoires pour lesquelles on connaît un certain nombre de relations de dépendances. Appelons U l'ensemble des variables et P(U) la distribution de probabilités sur cet ensemble. Si nous disposons d'une nouvelle information sur une ou plusieurs variables, alors on souhaiterait remettre à jour la connaissance que représente le RB à travers P(U) à la lumière de cette nouvelle information. Cette remise à jour, qui se fera bien sûr en utilisant la règle de Bayes, est appelée l'inférence. Mathématiquement parlant, l'inférence dans un RB est le calcul de $P(U|\varepsilon)$, c'est-à-dire le calcul de la probabilité a posteriori du réseau sachant ε.

Les premiers algorithmes d'inférence exacte (par opposition à "approchée") pour les Réseaux Bayésiens ont été proposés dans [Pearl, 1982] et dans [Kim et Pearl, 1983] : il s'agissait d'une architecture à passage de messages et ils étaient limités aux arbres. Dans cette technique, à chaque nœud est associé un processeur qui peut envoyer des messages de façon asynchrone à ses voisins jusqu'à ce qu'un équilibre soit atteint, en un nombre fini d'étapes. Cette méthode a été depuis étendue aux réseaux quelconques pour donner l'algorithme JLO. Cette méthode est aussi appelée algorithme de l'arbre de jonction et a été développée dans [Lauritzen, 1988] et [Jensen et al., 1990].

Une autre méthode, développée dans [Pearl, 1988] et dans [Jensen, 1996], s'appelle le « cat-set-conditioning » : elle consiste à instancier un certain nombre de variables de manière à ce que le graphe restant forme un arbre. On procède à une propagation par messages sur cet arbre. Puis une nouvelle instanciation est choisie. On réitère ce processus jusqu'à ce que toutes les instanciations possibles aient été utilisées.

Un autre algorithme est apparu pour la première fois par Zhang et Poole dans [Zhang et Poole, 1994]. C'est essentiellement l'algorithme « d'élimination de variables » de Dechter [Dechter, 1996] ainsi appelée parce qu'il élimine par marginalisation (c'est-à-dire intégration) les variables les une après les autres. Un ordre dans lequel les variables doivent être marginalisées est exigé comme entrée de cet algorithme ; on l'appelle l'ordre d'élimination. Le calcul dépend de cet ordre. La complexité de l'algorithme d'élimination de variables peut être mesurée par le nombre d'opérations d'additions et de multiplications numériques qu'il exécute. Trouver un ordre d'élimination optimal est un problème NP-difficile.

L'inférence dans des réseaux quelconques est NP-difficile [Cooper, 1990], la complexité de l'inférence peut conduire à des temps de calculs prohibitifs pour des réseaux complexes. Il est impossible de calculer directement la loi de probabilité d'un nœud ou d'effectuer une inférence plus complexe, d'où l'utilité d'introduire un nouveau type d'inférence nommé inférence approximative. Les méthodes d'approximation cherchent à estimer la distribution de probabilité complète représentée par le réseau, en effectuant des tirages aléatoires avec des lois simples [Jordan et al., 1999][Mackay, 1999][Jaakkola et Jordan, 1999]. Les deux grandes

classes d'algorithme d'inférence approximative sont l'algorithme de Monte Carlo [Mackay, 1999] et l'algorithme variationnel [Jordan et Weiss, 2001].

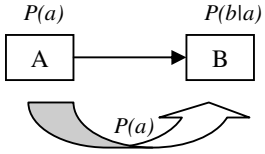
Nous proposons de détailler dans cette partie, deux algorithmes d'inférence exacte : l'algorithme « message passing » de Pearl, dans le cas de Réseau Bayésien à structure d'arbre, et l'algorithme « arbre de jonction », pour des réseaux à structures quelconques.

1.4.1 Algorithme "Message Passing" de Pearl

Cette technique est utilisée dans les réseaux acycliques orientés. Elle consiste, comme le porte son nom, à l'envoie et la réception de messages portant des valeurs ou encore des coefficients pour la mise à jour des tables de probabilités de chaque nœud.

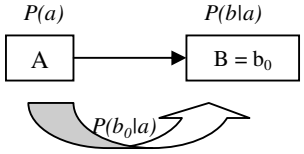
Le principe de cet algorithme consiste dans le fait que chaque nœud envoie des messages vers tous ses fils et tous ses parents afin de les informer du changement de sa table de probabilité. Les nœuds fils et parents révisent à ce fait, leurs propres tables de probabilités.

Le nœud *A* peut envoyer un message à son fils *B*, $\pi_B(A)$, pour que ce dernier puisse calculer sa valeur :



$$p(b) = \sum_a p(b,a) = \sum_a p(b|a).p(a)$$

Le nœud *B* peut envoyer un message à son père *A*. $\lambda_B(A)$, pour qu'il puisse calculer sa valeur :



$$p(a|b_0) = p(b_0|a).p(a)$$

Application du « message passing » dans le cas d'un arbre :

Soient un nœud *X*, Y_i le $i^{ème}$ enfant de *X*, *A* le seul parent de *X* (puisqu'il s'agit d'un arbre) et sachant *E* l'évidence, alors nous avons les assertions suivantes :

- $P(X|E) = \lambda(X)\Pi(X)$
- $\lambda(X) = \Pi_i\lambda_{Yi}(X)$
- $\Pi(X) = \sum_a p(X|a)\Pi_X(a)$

X envoie les informations suivantes :

- à son père : $\lambda_X(A) = \sum_X \lambda(X)\, p(X|a)$
- à ses enfants : $\Pi_{Yi}(X) = \Pi(X)\Pi_{j\neq i}\lambda_{Yj}(X)$

Etant donnés les cas particuliers suivants :

- X racine : $\Pi(X) = P(X)$
- X feuille non instanciée : $\lambda(X) = [1\ldots1]$
- X nœud instancié : $\lambda(X) = [001\ldots0]$ (la position du 1 correspond à la valeur donnée à X).

1.4.2 Algorithme "Arbre de Jonction"

L'algorithme de l'arbre de jonction dit JLO, des noms de ses auteurs : Jensen F. V., Lauritzen S. L. et Olesen K. G. s'applique à deux réseaux ne comprenant que des variables à valeurs discrètes [Lauritzen, 1988] [Jensen et al., 1990]. L'algorithme se comporte de la façon suivante :

- ***La phase de construction :*** elle consiste à appliquer des transformations graphiques sur un Graphe Acyclique Orienté (GAO) afin d'obtenir la structure finale : Arbre de Jonction. Ces transformations impliquent un certain nombre de structures intermédiaires et peuvent être récapitulées par :

  1. *Moralisation* : construction d'un graphe non dirigé, appelé *graphe* moral ;
  2. *Triangulation* : ajout sélectif des arcs au graphe moral pour former un *graphe triangulé* ;
  3. A partir du graphe triangulé, on construit des ensembles de nœuds appelés cliques. Chaque nœud contient une ou plusieurs variables du Réseau Bayésien original.
  4. Pour construire l'arbre de jonction, on connecte les cliques pour former un arbre non dirigé.

- ***La phase de propagation :*** elle consiste à la propagation des nouvelles informations concernant une ou plusieurs variables à l'ensemble du réseau, de manière à mettre à jour l'ensemble des distribution de probabilités du réseau.

a. Moralisation

L'étape de la moralisation consiste à marier les parents de chaque noeud deux à deux, puis à éliminer les directions dans le graphe obtenu. Soit la définition suivante d'un graphe moral :

Définition 3.1 **:** Soit $G = (V, A)$ un graphe orienté. On dit que le graphe $M = (V, E_M)$ est le graphe moral de G si et seulement :

- M n'est pas orienté.
- $A \subset E_M$.
- $\forall (u, v) \in V \times V, F(u) \cap F(v) \neq \varnothing \Rightarrow (u, v) \in E_M$, où $F(u)$ est l'ensemble des enfants de u.

Soit G un graphe acyclique orienté d'un Réseau Bayésien. Le graphe moral G_M qui correspond à G est construit de la façon suivante :

1. Création d'un graphe non orienté G_u en copiant G sans les directions des arcs.
2. Création de G_M à partir de G_u : pour chaque nœud V et ses parents $Pa(V)$ dans G nous connectons chaque paire de nœuds dans $Pa(V)$ en ajoutant un arc non orianté à G_u.

La figure 3.3 montre la moralisation du graphe acyclique orienté de la figure 3.2.

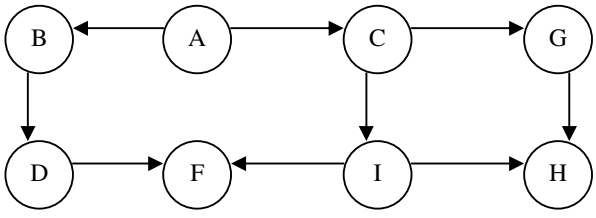


Figure 3.2 : Graphe acyclique orienté

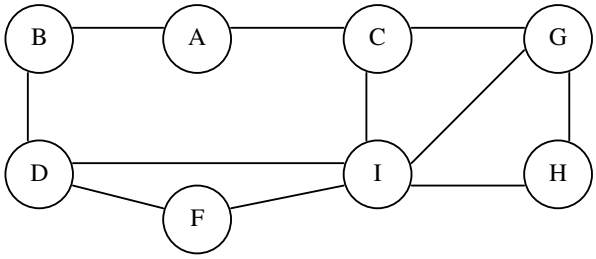


Figure 3.3 : Graphe moral

b. Triangulation

Un graphe non orienté est triangulé si chaque cycle de longueur quatre ou plus, contient un arc qui relie deux nœuds non adjacents dans le cycle. Soit le définition suivante d'un graphe triangulé :

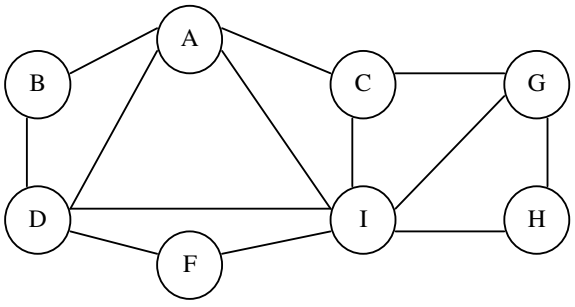
Définition 3.2 : Soit $G(V, E)$ un graphe non orienté. Un graphe $T = (V, E_T)$ est un graphe triangulé de G si et seulement si :

- T n'est pas orienté ;
- $E \subset E_T$;
- Pour tout cycle $[v_0, v_1, \ldots, v_n, v_0]$ de longueur supérieure ou égale à 4, il existe $i > j+1$ tel que $(v_i, v_j) \in E_T$ est un arc.

La procédure de triangulation (algorithme d'élimination) élaborée par [Kjaerulf, 1990] et décrite par :

1. Faire une copie du graphe morale G_M qu'on appelle G'_M.
2. Tant qu'il reste des nœuds dans G'_M on fait les étapes suivantes :
   a. Sélectionner un nœud V de G'M.
   b. Ce nœud V et ses voisins dans G'_M forment une clique. Connecter tous les nœuds de cette clique. Pour chaque arc ajouté dans G'_M, ajouter le même arc dans G_M.
   c. Enlever V de G'_M.
3. G_M, modifié par les arcs ajoutés dans les étapes précédentes, est triangulé.

La figure 3.4 montre les étapes de la triangularisation par l'algorithme précédent du graphe moral de la figure 3.3.



Graphe triangulé

Variable éliminée	Clique	Arc ajouté
H	IGH	Aucun
G	CIG	Aucun
F	DIF	Aucun
C	ACI	(A, I)
B	ADB	(A, D)
D	ADI	Aucun
I	AI	Aucun
A	A	Aucun

Algorithme d'élimination

Figure 3.4 : Triangularisation du graphe moral

c. Cliques

Une clique est un sous-graphe dont les nœuds sont complètement connectés. Nous proposons la définition suivante :

Définition 3.3 : Soit $G(V, E)$ un graphe et $W \subset V$. W est une clique si et seulement si :

$\forall (u, v) \in W \times W, (u, v) \in E$

Définition 3.4 : Soit $G(V, E)$ un graphe et W une clique. W est une clique maximale si et seulement s'il n'existe aucun sur-ensemble $U \supset W$, tel que U soit une clique.

Ainsi on peut dire qu'un clique dans un graphe non orienté G et complète et maximale si elle est un sous-graphe complet et maximal tel que :

- *Complet* signifie que chaque paire de nœuds (variable) distincts est connectée par un arc.
- *Maximal* signifie que la clique n'est pas complètement contenue dans un sous-graphe complet.

Dans la figure 3.4, les cliques du graphe triangulé sont (IGH), (CIG), (DIF), (ACI), (ADB) et (ADI).

d. Arbre de jonction

A présent nous avons un graphe non orienté, nous cherchons à construire un graphe optimal de jonction en connectant les cliques obtenues dans le paragraphe précédent. Nous proposons les deux définitions suivantes :

Définition 3.5 : Soit G un ensemble de cliques à partir d'un graphe non orienté et que ces cliques de G sont rangées dans un arbre T.

T est un arbre de regroupement si pour chaque paire de nœuds (u, v) de T, tous les nœuds dans le chemin entre v et u contiennent l'intersection $v \cap u$.

La figure 3.5 montre un exemple de l'arbre de regroupement. Nous remarquons que (a) est un arbre de regroupement par contre (b) n'est le pas car l'intersection de deux cliques (BCDE) et (CHGJ) est C qui n'appartient pas à la clique (DEFI).

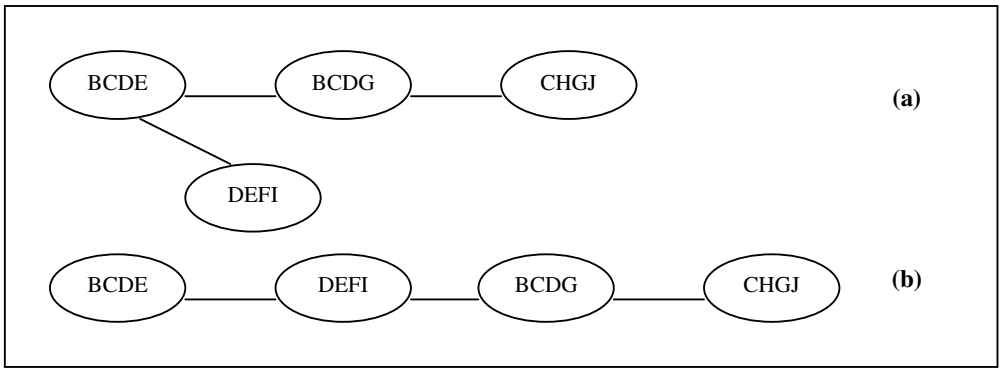


Figure 3.5 : (a)- arbre de regroupement (b)- n'est pas un arbre de regroupement

Définition 3.6 **:** Soit $G = (V, E)$ un graphe orienté acyclique. Soit $M = (V, E_M)$ le graphe moral associé à G, $T = (V, E_T)$ le graphe triangulé associé à M. On dit que $J = (V, A_i)$ est un arbre de jonction associé à G si et seulement si :

- J est un arbre de regroupement sur *V*.
- Toute clique maximale dans *T* est un nœud de *J*.

Etant donné un ensemble de n cliques, on peut construire un arbre de cliques en connectant itérativement chaque paire de cliques par un arc jusqu'à ce que les cliques soient toutes connectées par *n-1* arcs. Nous rappelons que les séparateurs contiennent les variables communes à deux cliques connectées dans l'arbre de jonction.

1. Construction de l'arbre des cliques

a. On commence avec un ensemble de *n* arbres, chaque arbre se compose d'une simple clique et d'un ensemble de séparateurs vide *S*.

b. Pour chaque paire distincte de cliques *X* et *Y* :

- Nous créons un séparateur $S_{XY} = X \cap Y$,
- On insère $S_{XY} = X \cap Y$ dans *S*.

c. On répète l'instruction (b) jusqu'à obtenir *n-1* séparateurs tels que :

- On sélectionne un séparateur S_{XY} de *S* suivant le critère indiqué ci-dessous. Puis on élimine S_{XY} de *S*.
- On insère S_{XY} entre *X* et *Y* uniquement si *X* et *Y* sont dans des arbres différents.

2. Choix des séparateurs appropriés : on décrit comment choisir le future séparateur en se basant sur les deux notions de masse et de coût.

- La masse d'un séparateur S_{XY} est le nombre des variables de $X \cap Y$.
- Le coût d'un séparateur S_{XY} est la somme des poids de *X* et *Y* où le poids est défini par :
  - Le poids d'une variable *V* est le nombre de ses valeurs d'états possibles.
  - Le poids d'un ensemble de variables *X* est le produit des poids des variables de l'ensemble *X*.

Nous pouvons maintenant sélectionner le futur séparateur de l'ensemble *S*, quand nous exécutons l'étape (c) :

- L'arbre de clique résultant doit satisfaire à la propriété (Join tree property) présentée ci-dessous et aussi on doit choisir le séparateur ayant la plus grande masse.
- Quand deux séparateurs ou plus ont la même masse, on choisit le futur séparateur ayant le plus petit coût.

Propriété 3.1 (Join tree property) : Etant donné un Réseau Bayésien (V, A) avec $V = (V_1,\ldots, V_n)$, un arbre non dirigé *T* et deux cliques *X* et *Y* dans *T*. Toutes les cliques sur le chemin *X* et *Y* contiennent $X \cap Y$, et aussi pour chaque variable $V_i \in V$, la famille de V_i notée F_{Vi} (V_i et ses parents $Pa(V_i)$) est au moins incluse dans un groupement.

Ceci termine la construction de l'arbre de jonction. Il est à noter que la complexité dans le pire des cas de l'heuristique de la triangulation est de l'ordre de $O(N^3)$ et que la création de l'arbre est de l'ordre de $O(N^2 logN)$ [Bellot, 2002].

La figure 3.6 montre l'arbre de jonction obtenue à partir des graphes des figures 3.3, 3.4 et 3.5.

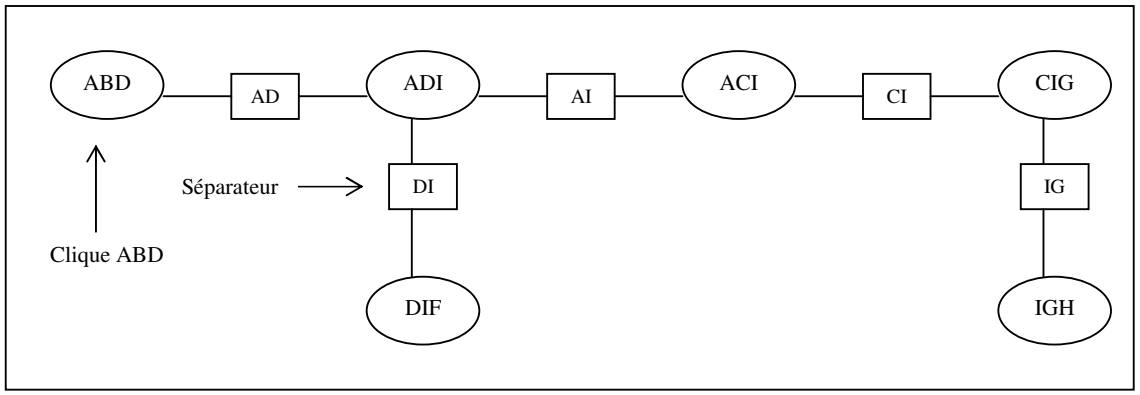


Figure 3.6 : Arbre de jonction

e. Structure secondaire d'un Réseau Bayésien

Nous proposons, de définir une fonction de potentiel comme suit :

Définition 3.7 : Soit un ensemble de variables U, on définit la fonction potentiel par :

$\Phi_U : U \rightarrow IR^+$
$u \rightarrow \Phi_U(u)$

U est appelé domaine de la fonction Φ_U. Il est aussi noté $dom(\Phi)$.

Nous définissons deux opérations sur le potentiel : la *marginalisation* et la *multiplication*.

Soient deux ensembles de variables X et Y tels que $X \subset Y$ et son potentiel Φ_Y.

La marginalisation de Φ_Y dans X est un potentiel noté Φ_X tel que :

$$\Phi_X = \sum_{Y|X} \Phi_Y \qquad (3.3)$$

La multiplication de deux potentiels Φ_1 et Φ_2 est un potentiel qui a les propriétés suivantes :

1. $dom(\Phi_1 \Phi_2) = dom(\Phi_1) \cup dom(\Phi_2)$.
2. $\Phi_1 \Phi_2 = \Phi_2 \Phi_1$.
3. $(\Phi_1 \Phi_2)\Phi_3 = \Phi_1 (\Phi_2 \Phi_3)$.
4. $\Phi_\varnothing = 1$ et 1. $\Phi = \Phi$.

Etant donné un Réseau Bayésien (V, A) et son arbre de jonction est construite, on définit sa structure secondaire par deux composante graphique et numérique :

1. La composante graphique nommée Arbre de jonction se compose de:

- Un arbre non dirigé T tel que chaque clique (nœud dans T) doit satisfaire à la propriété « Joint tree property ».
- Des séparateurs : c'est un ensemble constitué de l'intersection de deux cliques adjacentes.

2. La composante numérique est décrite en utilisant les fonctions de potentiels associées aux cliques et aux séparateurs de l'arbre de jonction de la manière suivante :

- Pour chaque clique X et son séparateur voisin S on a :

$$\Phi_S = \sum_{X|S} \Phi_S \qquad (3.4)$$

- Les potentiels codent la distribution jointe P(V) du réseau par :

$$P(V) = \frac{\prod_i \Phi_{X_i}}{\prod_j \Phi_{S_j}} \tag{3.5}$$

Avec Φ_{X_i} et Φ_{Sj} sont les potentiels respectifs de clique et de séparateur.

Cette nouvelle structure a une importante propriété que pour chaque clique ou séparateur X, on a : $\Phi_X = P(X)$.

Dès lors, on peut calculer les probabilités marginales pour chaque variable V_i du réseau :

$$P(V_i) = \sum_{X|V_i} \Phi_X \tag{3.6}$$

f. Phase de propagation

Dans ce paragraphe nous allons décrire l'inférence avec observation ; c'est-à-dire comment calculer P(V|e) dans le contexte de l'observation *e* (évidence). Pour ce fait, nous proposons les définitions suivantes des notions de vraisemblance et de passage de message.

Définition 3.8 : Soit une variable *V*, la vraisemblance potentielle (Finding) de *V*, notée Λ_V est un potentiel sur $\{V\}$.

$\Lambda_V : \{V\} \rightarrow 0,1$
$v \rightarrow \Lambda_V(v)$

On peut coder un ensemble arbitraire d'observations (constituant l'évidence) *E* en utilisant Λ_V pour chaque variable *V* de la façon suivante :

1. Si $V \in E$, et si *V* est observable on a :

$$\Lambda_V(v) = \begin{cases} 1 \text{ si } v \text{ est une valeur observée de } V. \\ 0 \text{ autres.} \end{cases}$$

2. Si $V \notin E$, et si *V* n'est pas observée alors : $\Lambda_V(v) = 1 \ \forall v$.

Définition 3.9 : Soient *V* une clique, Φ_V son potentiel associé et S un séparateur voisin. Chaque séparateur *S* d'un arbre de regroupement fait passer deux messages dans les deux directions (convergente ou divergente) notés Ψ_S et Ψ^S. Soient $S_1,\ldots, S_k$ les autres séparateurs voisins de *V*. On suppose que chaque S_i reçoit un message Ψ_i de *V*. Ainsi *V* peut passer le message :

$\sum_{V|S} \Phi_V \Psi_1 \ldots \Psi_k$ à *S* et on dit que la direction *V*-*S* est activée.

La méthode de propagation consiste à répéter l'opération de passage de message à travers les directions activées. On ne doit pas arrêter le processus avant que les messages passent dans toutes les directions pour chaque arc. Dans ce cas on dit que l'arbre de jonction est *complet*.

Après avoir terminé la partie de passage de message on peut calculer la probabilité jointe de chaque clique dans le contexte de l'observation *e* en utilisant les formules suivantes :

Soient *T* un arbre de jonction représentant un Réseau Bayésien sur l'univers *U* et *e* l'observation. On suppose que *T* est complet. Soient *V* une clique, Φ_V son potentiel associé, $S_1,\ldots, S_k$ ses séparateurs voisins et $\Psi_1 \ldots \Psi_k$ les messages dirigés sur *V*, alors :

$$P(V, e) = \prod \Phi_V \prod \Psi_1 \ldots \Psi_k \tag{3.7}$$

Soit S un séparateur avec les Ψ_S, Ψ^S les messages de passage pour *S*. On a :

$$P(S,e) = \prod \Psi_S \prod \Psi^S \tag{3.8}$$

L'inférence avec observation se base sur les étapes suivantes :

1. Initialisation : se compose des étapes suivantes :

- Pour chaque clique et séparateur X, on met Φ_X à 1 : $\Phi_X \leftarrow 1$;
- Pour chaque variable V : on affecte V à une clique X qui contient F_V ; on multiplie Φ_X par $P(V|Pa(V))$: $\Phi_X \leftarrow \Phi_X.P(V|Pa(V))$;
- On met chaque vraisemblance $\Lambda_V(v)$ à 1 : $\Lambda_V(v) \leftarrow 1$

2. Entrée de l'observation : on fait rentrer les observations dans l'arbre de jonction de la façon suivante :

- Coder chaque observation $V = v$ comme une vraisemblance Λ_V^{new}.
- Identifier la clique contenant V (F_V).
- Mise à jour des Φ_X et Λ_V^{new} : $\Phi_X \leftarrow \Phi_X \Lambda_V^{new}$ et $\Lambda_V \leftarrow \Lambda_V^{new}$

3. Marginalisation et normalisation : après avoir propagé les informations dans l'arbre de jonction, on passé à la dernière étape. Pour chaque clique (ou séparateur) X, on a $\Phi_X = P(X, e)$ où e est l'observation. Quand on marginalise le potentiel de clique Φ_X d'une variable V, on obtient : $P(V, e) = \sum_{X|V} \Phi_X$.

Notre objectif est de calculer $P(V|e)$ probabilité de V étant donnée e. On obtient $P(V|e)$ à partir de $P(V, e)$ en normalisant $P(V, e)$:

$$P(V|e) = \frac{P(V,e)}{P(e)} = \frac{P(V,e)}{\sum_V P(V,e)} \tag{3.9}$$

La probabilité de l'observation $P(e)$ est la constante de normalisation.

1.5 Synthèse

Les Réseaux Bayésiens représentent un outil de choix dans la représentation de connaissances et dans l'exploitation de celles-ci. Par ailleurs, plusieurs domaines sont intéressés par ce type de représentation. En fait, l'inférence sur les réseaux bayésiens est un problème NP-difficile[15] [Cooper, 1990], c'est pourquoi il était convenable de le voir de façon complète pour des instances réalisables et incomplète dans les autres cas. Après cette approche statique, pour aller plus loin, il pourrait être intéressant de se pencher sur les réseaux bayésiens dynamiques. Ceux ci sont une répétition du réseau classique dans lesquels on rajoute un lien causal d'un pas de temps à l'autre. Ils contiennent chacun un certain nombre de variables aléatoires représentant les observations et les états cachés du processus. Le temps ici est discret et chaque unité de temps représente une nouvelle observation, l'unité de temps n'a donc pas toujours la même valeur en temps réel, la complexité inférencielle des réseaux bayésiens dynamiques est évidement bien plus élevée que celle vu précédemment.

Enfin, on retrouve les réseaux bayésiens dans beaucoup d'applications, sans même le savoir. Microsoft par exemple est un fervent utilisateur de cette structure (Answer Wizard, assistant Office par exemple), mais aussi Google et Mozilla via leurs filtres anti-spam. De nombreux travaux dans le domaine sont réalisés, preuve de l'intérêt porté par la communauté scientifique, et de la puissance de ces réseaux. Par ailleurs, nous nous intéressons

[15] Ceci parce que dans les réseaux généraux, il peut exister plusieurs chemins entre les paires de nœuds du graphe.

prticulièrement à l'application des réseaux bayésiens au domaine de la recherche d'information.

2. Modèle Bayésien de RI

Des travaux récents ont permis d'exploiter l'apport des Réseaux Bayésiens (RBs) pour définir des modèles de RI. L'avantage apporté par l'utilisation de ces réseaux a été principalement de pouvoir combiner des informations provenant de différentes sources pour restituer les documents qui seraient les plus pertinents étant donnée une requête.

2.1 Architecture générale du modèle Bayésien

La figure 3.7 présente l'architecture générale du modèle de RI basé sur les réseaux Bayésiens.

Les noeuds du réseau dans un modèle BNR (modèle RI basé sur les réseaux Bayésiens) [De Campos et al., 2002] [De Campos et al., 2003] ont été décomposés en deux ensembles de variables *T* et *D :*

- L'ensemble des termes $T = (T_1, T_2, \ldots\ldots, T_M)$, où M est le nombre de termes dans la collection ;
- L'ensemble des documents de la collection $D = (D_1, D_2, \ldots\ldots, D_N)$, où N est le nombre de documents dans la collection.

Les domaines des noeuds sont binaires {vrai, faux} signifiant que le noeud est instancié ou non.
T est l'ensemble des noeuds termes; une variable T_i associée à un terme prend ses valeurs dans le domaine dom(T_i) = $\{t_i, \overline{t_i}\}$, $\overline{t_i}$ désigne le fait que le terme T_i est non pertinent et t_i désigne le fait qu'il est pertinent. Un terme est considéré comme pertinent si tous les documents qui le contiennent sont jugés pertinents par l'utilisateur et non pertinent sinon.
$\mathcal{D}$ est l'ensemble des noeuds documents, une variable D_j prend ses valeurs dans le domaine dom(D_j) = $\{d_j, \overline{d_j}\}$, $\overline{d_j}$ signifie « le document D_j n'est pas pertinent » et d_j signifie « le document D_j est pertinent ». Un document est pertinent s'il répond au besoin utilisateur.

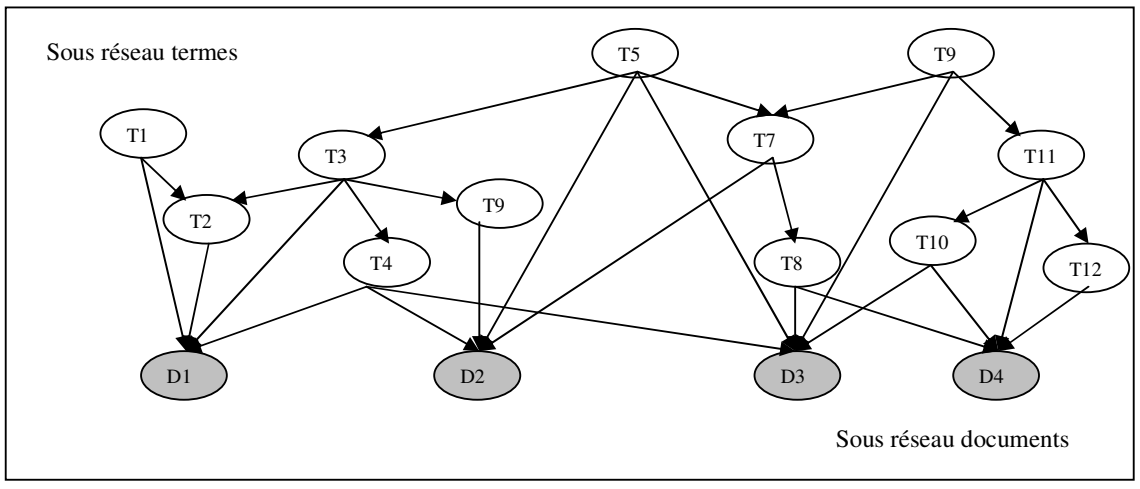


Figure 3.7 : Architecture générale du modèle Bayésien

2.2 Les modèles de RI basés sur les réseaux Bayésiens

Les Réseaux Bayésiens (RB) ont été utilisés en RI depuis les années 1990 avec [Pearl, 1988], [Buntine, 1994], [Jensen, 2000]. Ils fournissent un formalisme pour fusionner des informations provenant de différentes sources (requêtes passées, réinjection de pertinence), afin de restituer les documents, et ont permis de combiner différentes approches de RI [Ribeiro-Neto et al., 1996]. Les modèles les plus connus en RI utilisant les RBs sont les Réseaux d'Inférence [Turtle et Croft, 1990] et les Réseaux de Croyance.

2.2.1 Modèle à base de Réseaux Bayésiens d'Inférence

Un réseau d'inférence en RI est matérialisé par un graphe orienté sans cycle. Les nœuds du graphe correspondent à des concepts, à des groupes de mots ou à des documents (des variables propositionnelles). Un nœud particulier va représenter la requête. Les arcs du graphe représentent des relations sémantiques entre les nœuds ou les propositions. A ces nœuds sont associés des probabilités de croyance. Ce modèle repose sur le théorème de Bayes pour l'expression de la probabilité conditionnelle et sur la stratégie d'activation propagation (*spreading activation*).

La recherche peut être donc considérée comme un processus de raisonnement incertain pour estimer la probabilité qu'un document satisfasse la requête. La stratégie utilisée dans I3R (Intelligent Interface for Information Retrieval) est representative de ce modèle de recherche d'information. Elle se fait en suivant les étapes suivantes [Croft et Thompson, 1987] :

- Tout d'abord, on part d'un ensemble de nœuds qui représentent les termes de la requête ;
- Puis, on active tous les nœuds qui sont connectés à ces points d'entrée par un lien ;
- Ensuite, le processus de propagation continue, en respectant certaines contraintes, jusqu'à la vérification d'une certaine condition (un seuil pour l'activation). Le niveau d'activation décroît avec la longueur du chemin parcouru ;
- Finalément, les nœuds sont extraits et classés par ordre de leur niveau d'activation.

Des variantes de ce modèle ont été proposées pour tenir compte de la nature des liens entre les nœuds et de leur force, ce qui permet de gérer mieux les niveaux d'activation.

Dans le système GRANT [Cohen et Kjeldsen, 1987], les contraintes sont de trois types :

- Contrainte de distance : l'activation cesse au cinquième niveau ;
- Contrainte de branchement : l'activation est interrompue lorsque les nœuds atteints sont accessibles par un grand nombre d'arcs ;
- Contrainte de chemin : l'activation privilège des nœuds sensibles à certaines connaissances ou méta-connaissances par rapport à un domaine représenté.

Ces paramètres ne sont pas toujours faciles à déterminer. C'est le plus souvent en expérimentant qu'on les détermine parce qu'ils sont liés notamment au domaine traité.

Par ailleurs, les Réseaux d'Inférence sont utilisés aussi dans le système INQUERY [Turtle et Croft, 1990] [Turtle, 1991] [Turtle et Croft, 1991] et ses performances sont liées à sa capacité à représenter différentes approches de la RI et à les combiner dans un seul modèle. Le réseau d'inférence est composé de deux réseaux : le réseau document et le réseau requête. Le réseau document représente les documents de la collection et contient différents schémas de représentation (résumés, textes, etc.). Les noeuds du réseau requête représentent les concepts

de la requête et le besoin utilisateur. Les réseaux document et requête sont liés par l'intermédiaire des noeuds termes d'indexation. Les valeurs des noeuds sont binaires {vrai, faux} et les valeurs des arcs reliant les nœuds termes au noeud requête sont obtenues par l'utilisation d'un des schémas des modèles connus de la RI (booléen, vectoriel, etc.). Ce système évalue la pertinence du document étant donnée une requête, et le résultat est une liste de documents pondérés. Ces poids sont considérés comme des coefficients de similarité proportionnels à la fréquence des termes dans le document et inversement proportionnels à celle dans la collection. D'autres travaux basés sur ces réseaux ont été proposés pour les systèmes hypertextes [Savoy et al., 1991].

2.2.2 Modèle à base de Réseaux Bayésiens de croyance

Les Réseaux de Croyance (RC) [Ribeiro-Neto et al., 1996] [Silva et al., 2000] ont été utilisés pour extraire des connaissances des requêtes du passé et les combiner avec le modèle vectoriel [Salton et al., 1994]. La sélection d'un document s'appuie sur la similarité entre le document d_j et la requête Q, calculant la probabilité $P(d_j = 1|Q = 1)$. En effet, $Q = 1$ et $d_j = 1$ signifient respectivement Q activé et d_j activé.

Crestani et al. [Crestani et al., 2003], ont proposé un modèle pour la RI basé sur les réseaux Bayésiens pour les documents structurés. Un réseau à deux structures (BNR- 2) [De Campos et al., 2003] a été conçu et étendu à un réseau multi-structures. L'ensemble des variables dans le modèle BNR-2 est composé de deux ensembles distincts, l'ensemble des variables aléatoires binaires définissant les termes du dictionnaire et l'ensemble des variables aléatoires binaires représentant les documents de la collection. Chaque document est composé d'une structure hiérarchique comportant différents niveaux d'abstraction (titre, auteur, section, paragraphe, etc.). Le processus d'inférence calculé, étant donné une requête, les probabilités a *posteriori* de la pertinence de toutes les unités de structure. Les documents de score élevé sont restitués.

Certaines recherches récentes [De Campos et al., 2003] [Fernandez et al., 2003] ont proposé des modèles de Réseaux Bayésiens avec une topologie flexible qui peut tenir compte des relations de dépendance existant entre les termes ou les documents. Le sens des représentations des documents et du besoin utilisateur pour tous ces modèles est identique.

3. Reformulation de requêtes dans le modèle Bayésien

Nous nous intéressons dans cette section aux techniques de *Relevance Feedback* (RF) dans le modèle BNR.

Soit *b* le nombre de documents jugés par l'utilisateur. L'ensemble $\{ D_{k_1} = d_{k_1}, ..., D_{k_h} = d_{k_h} \}$ contient les documents pertinents et l'ensemble $\{ D_{k_{h+1}} = \bar{d}_{k_{h+1}}, ..., D_{k_b} = \bar{d}_{k_b} \}$ contient les documents non pertinents, alors la nouvelle requête sera:

$$Q1 = Q \wedge \{ D_{k_1} = d_{k_1}, ..., D_{k_h} = d_{k_h} , D_{k_{h+1}} = \bar{d}_{k_{h+1}}, ..., D_{k_b} = \bar{d}_{k_b} \} \quad (3.10)$$

Chaque noeud *X* non-instancié reçoit de tous ses noeuds parents un message sous forme de vecteur $\Pi_X(Z)$, il reçoit encore de tous ses noeuds fils *Y* un message sous forme vecteur $\lambda_Y(X)$.

Chaque noeud instancié *X* reçoit un message $\lambda_0(X)$ d'un noeud fils imaginaire avec :

$$\begin{cases} \lambda_0(X) = (1, 0) \text{ si } X = \bar{x} \\ \lambda_0(X) = (0, 1) \text{ si } X = x \end{cases} \quad (3.11)$$

Si l'évidence de X est partielle par rapport à une observation (*Obs*) alors :

$\lambda_0(X) = (P(Obs|\bar{x}), P(Obs|x))$ (3.12)

Dans ce cas le plus important est le rapport $P(Obs|\bar{x})/P(Obs|x)$ et on peut conclure que les deux expressions $\lambda_0(X) = (P(Obs|\bar{x}), P(Obs|x))$ et $\lambda_0(X) = (P(Obs|\bar{x})/P(Obs|x), 1)$ sont équivalentes.

Pour que tous les noeuds reçoivent λ_0, on utilise le vecteur $\lambda_0(X) = (1, 1)$ pour les nœuds non instancié.

Le tableau 3.1 représente la table de contingence des termes. Nous définissions quelques notions de base qui seront utilisées dans la suite de cette section.

	$T_i = \bar{t}_i$	$T_i = t_i$	Total
Non pertinent	$n_{\bar{r}\bar{t}_i}$	$n_{\bar{r}t_i}$	$n_{\bar{r}}$
pertinent	$n_{r\bar{t}_i}$	n_{rt_i}	n_r
	$n_{\bar{t}_i}$	n_{t_i}	$\|R_Q\|$

Tableau 3.1 : Table de contingence des termes

Avec :

R_Q : l'ensemble des documents restitués et évalués pour une requête Q ;

$|R_Q|$: cardinale de R_Q;

n_r : Nombre de documents pertinents ;

$n_{\bar{r}}$: Nombre de documents non pertinents ;

n_{t_i} : Nombre de documents restitués qui contiennent le terme t_i ;

$n_{\bar{t}_i}$: Nombre de documents restitués qui ne contiennent pas le terme t_i ;

n_{rt_i} : Nombre de documents pertinent qui contiennent le terme t_i ;

$n_{\bar{r}t_i}$: Nombre de documents non pertinent qui contiennent le terme t_i ;

$n_{r\bar{t}_i}$: Nombre de documents pertinent qui ne contiennent pas le terme t_i ;

$n_{\bar{r}\bar{t}_i}$: Nombre de documents non pertinent ne contiennent pas le terme t_i.

Par ailleurs, les termes indexant les documents restitués sont classés en trois catégories :

- Terme qui se trouve dans des documents pertinents seulement (termes positifs $\Im^+$).
- Terme qui se trouve dans des documents non pertinents seulement (termes négatifs $\Im^-$).
- Terme qui se trouve dans des documents pertinents et non pertinents (termes neutres $\Im^\pm$).

Il faut distinguer entre les termes indexant les documents trouvés et la requête et entre les autres qui indexent les documents trouvés et absents de la requête.

Soient encore:

$\Im^q$: L'ensemble de termes indexant les documents trouvés et la requête et qui sera utile pour répondérer les termes de la requête initiale ;

$\Im^e$: L'ensemble des termes indexant les documents trouvés et absents de la requête et ce dernier ensemble est utilisé pour représenter les termes à ajouter (Expansion de la requête).

3.1 Repondération de termes de la requête initiale Q

Les termes de l'ensemble $\Im^q$ qui étaient instanciés comme pertinents, reçoivent un message $\lambda_0(T) = (0,1)$. Les termes de la requête initiale qui occurrent seulement dans des documents non pertinents ne sont pas considérés. Par conséquence, ils devraient être pénalisés en diminuant leur pertinence. Les autres termes ($\lambda_0(T) = (1,1)$) sont considérés comme des termes n'appartenant pas à la requête (*nonquery term*) et il est plus valable d'utiliser le vecteur $\lambda_0(T_i) = (\gamma_{t_i},1)$ à la place de $\lambda_0(T_i) = (1,1)$ avec $0 \pi \gamma_{t_i} \pi 1$.

La méthode proposée par [De Campos et al., 2003] considère que γ_{t_i} est très sensible au nombre de documents non pertinents contenants T_i ($n_{\bar{r}t_i}$ définie ci-dessus) et a montré que la meilleur valeur de γ_{t_i} est celle qui tend vers le vecteur $\lambda_0(T) = (1,1)$ et a proposé une fonction qui satisfait cette condition avec $0.5 \pi \gamma_{t_i} \pi 1$.

$$\lambda_0(T_i) = (1 - \frac{1}{n_{\bar{r}t_i} + 1}, 1) \tag{3.13}$$

D'autre part, les termes appartenants à $\Im^q \cap \Im^+$ et ceux qui appartiennent à $\Im^q \cap \Im^\pm$ sont les plus important mais en principe on ne peut pas augmenter la pertinence de termes positifs ou neutres qui occurrent dans la requête initiale car ils sont déjà complètement pertinents. Ainsi, la première approche simple qui s'appelle ***tr-ins*** et qui traite ce genre de termes propose que chacun de ces termes reçoive le message $\lambda_0(T) = (0,1)$.

Une autre approche est proposée pour augmenter la pertinence de ce genre des termes qui s'appelle ***tr-rep.*** Cette approche est basée sur la duplication de ces noeuds termes dans le réseau. Le nombre de duplication de chacun de ces noeuds est égal au nombre de document pertinents contenant ce terme (n_{rt_i}) et après il faut instancier les noeuds dupliqués comme pertinents.

Pour changer la structure du réseau, il suffit de connecter les noeuds termes dupliqués comme fils de T_i et connecter les noeuds documents fils de T_i comme fils des noeuds dupliqués.

La figure 3.8 montre une duplication de trois fois le terme T_i :

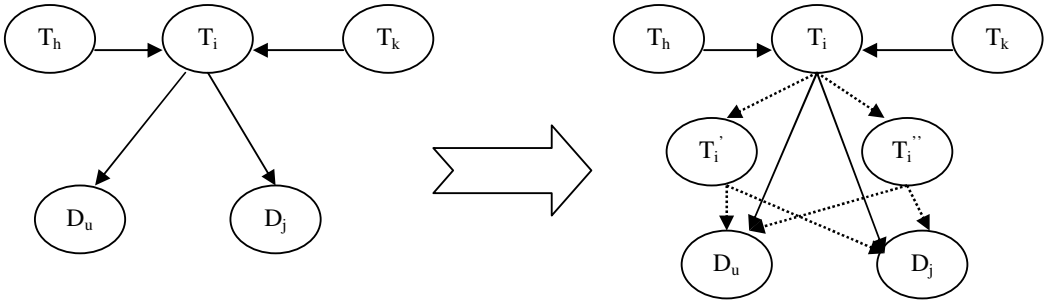


Figure 3.8 : Duplication trois fois du terme T_i

Par ailleurs, il est inutile de changer la structure du réseau. En effet, le changement se fait virtuellement et il suffit de multiplier par n_{rt_i} le facteur qui calcule le poids du terme dans la formule de probabilité générale et ceci pour chaque terme appartenant à $\Im^q \cap (\Im^+ \cap \Im^\pm)$. En conséquence, la pertinence de ces termes va augmenter n_{rt_i} fois automatiquement.

3.2 Expansion de la requête

L'expansion de la requête consiste à ajouter des nouveaux termes à la requête initiale Q. Comme nous l'avons indiqué dans la section 3.1, il existe trois classes de termes ($\Im^-, \Im^+, \Im^\pm$). Puisque $\Im^e$ est l'ensemble de termes duquel nous pouvons choisir les termes à ajouter à la requête initiale, les nouveaux ensembles des termes seront donc :

- Les termes négatifs qui appertiennent à $\Im^e \cap \Im^-$;
- Les termes positifs qui appertiennent à $\Im^e \cap \Im^+$;
- Les termes neutres qui appertiennent à $\Im^e \cap \Im^\pm$.

Tous les termes négatifs sont instanciés comme non pertinents et ont reçu le vecteur $\lambda_0(T_i) = (1,0)$. Les termes neutres ont reçu le vecteur $\lambda_0(T_i) = (1,1)$.

En générale la probabilité qu'un terme soit pertinent ou non est désignée par $P(r|t_i)$ (respectivement $P(r|\bar{t}_i)$). Ces deux probabilités sont utilisées pour calculer le vecteur $\lambda_0(T_i)$ et dans ce cas :

$$\lambda_0(T_i) = (P(r|\bar{t}_i) , P(r|t_i)) \tag{3.14}$$

$$\text{Où } \lambda_0(T_i) = (\frac{P(r|\bar{t}_i)}{P(r|t_i)},1) \tag{3.15}$$

Plusieurs méthodes ont été utilisées pour calculer ces probabilités, parmi lesquelles nous citons la méthode « *qe-gmle* » [De Campos et al., 2003] :

$$P(r|t_i) = \frac{n_{rt_i}}{n_{t_i}} \text{ et } P(r|\bar{t}_i) = \frac{n_{r\bar{t}_i}}{n_{\bar{t}_i}} \tag{3.16}$$

En conclusion, la reformulation de requête via l'application de la technique de RF permet d'augmenter substantiellement le niveau de précision par rapport à la requête initiale. Il s'agit donc d'un moyen efficace d'amélioration des performances du système de repérage d'information. Si plusieurs paramètres doivent être considérés pour une utilisation optimale des méthodes, il n'en demeure pas moins que les différentes variantes de la technique ont donné des résultats largement positifs. Le RF incrémental de par son interface utilisateur conviviale et sa formule unifiée et simplifiée devrait permettre de la vulgariser. Aujourd'hui, l'avantage procuré par cette technique est tel que plusieurs moteurs de recherche Web l'intègrent à leur mécanisme de recherche. L'impact direct est l'augmentation des requêtes soumises aux moteurs. Cette charge accrue sera d'autant plus limitée qu'il y aura convergence des résultats obtenus vers une satisfaction plus complète des usagers.

Cependant, le modèle proposé par Brini et al. [Brini et al., 2004abc], tente de fournir un autre sens possibiliste à ces représentations ainsi qu'à l'évaluation (comparaison de ces deux représentations). Une réponse à la problématique peut être apportée par l'utilisation des Réseaux Possibilistes (RP).

4. Les Réseaux Possibilistes

4.1 La théorie des possibilités

La théorie des possibilités introduite par Zadeh [Zadeh, 1978] et développée par Dubois et Prade [Dubois et Prade, 1988] [Dubois et Prade, 1998] traite l'incertitude sur l'intervalle [0,1], appelé échelle possibiliste, d'une manière qualitative ou quantitative. En fait, Lotfi Zadeh a formalisé la théorie des possibilités pour traiter l'incertitude permettant ainsi de traiter l'ignorance et de prendre en compte la pertinence d'une information incertaine. Dans cette théorie, l'information fournie par une source sur la valeur réelle d'une variable *x* est codée sous forme d'une distribution de possibilités dont les valeurs sont supposées être mutuellement exclusives, puisque *x* prend en définitive une seule valeur (sa vraie valeur), qui appartient à un ensemble Ω donné [Sandri, 1991]. La théorie des possibilités se base sur deux mesures de confiance : la mesure de possibilité et la mesure de nécessité [Fabiani, 1996].

4.1.1 Distribution de possibilité

La théorie des possibilités est basée sur les distributions de possibilité. Une distribution de possibilité, notée par π, est une application de Ω (l'univers de discours) vers l'échelle [0,1] traduisant une connaissance partielle sur le monde, noté ω. L'échelle possibiliste est définie de deux manières. Dans le cadre numérique les valeurs des possibilités traduisent souvent les bornes supérieures des probabilités. Dans le cadre qualitatif, les valeurs de possibilité peuvent être considérées comme un ordre de classement des états possibles. La combinaison des distributions de possibilité, exprimée à l'aide des normes triangulaires (t-normes) dépend du cadre. Les opérateurs « produit » et « minimum » peuvent être utilisés pour combiner des distributions de possibilité indépendantes dans les cadres quantitatif et qualitatif respectivement.

Normalisation : Une distribution de possibilité est dite α-normalisée, si son degré de normalisation, noté α(π), est égal à α. Ainsi :

$$\alpha = \alpha(\pi) = \max_{\omega} \pi(\omega) \tag{3.17}$$

Lorsque α = 1, π est dite normalisée.

Marginalisation : Soit une distribution de possibilité jointe, π sur Ω, une distribution marginale relative aux sous ensembles de variables peut être dérivée en utilisant l'opérateur *maximum*. Ainsi, $\forall X \subseteq V \; \forall x \in dom(X)$:

$$\pi(x) = \max_{\omega \in \Omega} \{\pi(\omega) : \omega[X] = x\} \tag{3.18}$$

Où V : ensemble de variables $\{A_1, A_2, ..., A_N\}$;

X : sous ensemble de V ;

$dom(X)$: domaine de X, produit cartésien des domaines des variables de X ;

x : une instance de X, si $X = \{A_1, A_2, \ldots, A_j\}$, alors $x = (\alpha_1, \alpha_2, ..., \alpha_j)$;

$\omega[X] = x$: configuration de X dans ω.

Une distribution de possibilité π sur permet de qualifier les évènements en terme de mesure de plausibilité et de certitude respectivement.

4.1.2 Mesures de possibilité et de nécessité

Dire qu'un évènement est non possible n'implique pas seulement que son événement contraire est possible mais qu'il est certain. Deux mesures duales sont utilisées : la mesure de possibilité $\Pi(\phi)$, et la mesure de nécessité $N(\phi)$.

- La possibilité d'un évènement A, notée $\Pi(A)$ est obtenue par $\Pi(A) = max_{x \in A}\pi(x)$ et décrit la situation la plus normale dans laquelle A est vraie ;
- La nécessité $N(A) = min_{x \notin A} 1 - \pi(x) = 1 - \Pi(\overline{A})$ d'un événement A reflète la situation la plus normale dans laquelle A est faux.

La distance entre $N(A)$ et $\Pi(A)$ évalue le niveau d'ignorance sur A. Rappelons que $N(A) > 0$ implique $\Pi(A) = 1$. Lorsque A est un ensemble flou, cette propriété n'est plus vérifiée et dans ce cas l'inégalité $N(A) \leq \Pi(A)$ est vérifiée.

4.1.3 Conditionnement possibiliste

En logique possibiliste, le conditionnement consiste à modifier la distribution de possibilité initiale π à l'arrivée d'une nouvelle information i. Soit ϕ, une sous classe de ω, $\phi = [i]$ l'ensemble des modèles de i. La distribution initiale π est remplacée par $\pi' = \pi(\bullet|\phi)$. Dans un cadre quantitatif, les éléments de ϕ sont proportionnellement modifiés :

$$\pi(\omega\Big|_p \phi) = \frac{\pi(\omega)}{\Pi(\phi)} si \omega \in \phi \ ; 0 \ Sinon \tag{3.19}$$

avec : $|_p$: conditionnement basé sur le produit dans un cadre qualitatif, le degré de possibilité maximal est affecté aux meilleurs éléments de ϕ :

$$\pi(\omega|_m \phi) = \left\{ \begin{array}{l} 1 si \pi(\omega) = \Pi(\omega) et \omega \in \phi \\ \pi(\omega) si \pi(\omega) \prec \Pi(\omega) et \omega \in \phi \\ 0 \quad sinon \end{array} \right\} \tag{3.20}$$

$|_m$: conditionnement basé sur le minimum.

4.1.4 L'indépendance possibiliste

La théorie des possibilités offre plusieurs définitions de l'indépendance [Ben Amor et al., 2002] [De Campos et al., 1999a] [De Campos et al., 1999b]. En particulier, deux définitions ont été utilisées pour le développement des réseaux possibilistes :

- *Relation de non-intéractivité* [Zadeh, 1978], cette relation est basée sur le conditionnement ordinal et elle est définie comme suit :

$$\Pi(x \wedge y|z) = \min(\Pi(x|z), \Pi(y|z)), \forall x, y, z. \tag{3.21}$$

- *Relation d'indépendance basée sur le produit*, cette relation est basée sur le conditionnement basé sur le produit et elle est définie comme suit : Cette forme d'indépendance est définie par :

$$\Pi(x \wedge y|z) = \Pi(x|z).\Pi(y|z), \forall x, y, z. \tag{3.22}$$

Ou d'une façon équivalente par :

$$\Pi(x|y \wedge z) = \Pi(x|z), \forall x, y, z. \tag{3.23}$$

4.1.5 Logique possibiliste

La logique possibiliste est une extension de la logique classique qui permet un raisonnement dans le cas d'évidence incomplète (incertitude) et de connaissances partiellement incohérentes. De point de vue syntaxique, un poids déterminant le niveau de priorité par rapport aux autres formules est associé à chaque formule. C'est un outil de raisonnement en présence d'informations incertaines, basé sur la relation de préférence entre les formules et non sur les valeurs numériques contrairement à la logique probabiliste.

L'ensemble de formules est appelé une base de connaissances possibilistes [Dubois et al., 1994]. De point de vue sémantique, la logique possibiliste permet d'ordonner les interprétations. Les modèles ont un degré 1 car ils sont complètement cohérents avec la base de connaissances. Les contre modèles dont les degrés de priorité sont faibles (poids) sont préférés aux contre modèles de degré de priorité plus important.

Dans la logique possibiliste, les règles sont modélisées par des clauses logiques :

$$p \rightarrow q = \neg p \vee q \tag{3.24}$$

Des valeurs sont attachées aux bornes inférieures des degrés de nécessité et de possibilité de p et q qui sont considérées comme des propositions booléennes. Les axiomes de la théorie des possibilités permettent de modéliser p implique q avec un poids $\alpha > 0$ par l'inégalité

$$N(p \rightarrow q) \geq \alpha \tag{3.25}$$

ou d'une manière équivalente par :

$$\Pi (p \wedge \neg q) \leq 1 - \alpha \tag{3.26}$$

pour signifier que $p \wedge \neg q$ est quelque peu impossible.

La distribution de possibilité exprimant cette information (connaissance) est π telle que :

$$\begin{aligned} \pi(x) &= 1 - \alpha \ \textit{si}\ p \wedge \neg q \text{ vraie à l'état } x \\ &= 1 \quad \textit{sinon} \end{aligned} \tag{3.27}$$

La distribution de possibilité induite par plusieurs propositions, mesurée par des nécessités, est obtenue par une intersection floue (utilisant le minimum) des distributions de possibilité induites par chaque proposition.

4.2 Réseaux Possibilistes (RP)

Les travaux existant sur les réseaux possibilistes sont soit des adaptations directes de l'approche probabiliste [Benferhat et al., 1999], ou des méthodes d'apprentissage à partir de données imprécises [Borgelt et al., 2000]. La théorie des possibilités offre deux définitions du conditionnement, ce qui conduit à deux définitions des réseaux causaux possibilistes. Les réseaux possibilistes basés sur le produit sont très similaires aux réseaux probabilistes.

4.2.1 Définitions

Un graphe possibiliste orienté sur un ensemble de variables $V = \{A_1, A_2, \ldots, A_N\}$ est caractérisé par une composante qualitative et une composante numérique. La première est un graphe acyclique orienté. La structure du graphe représente l'ensemble des variables ainsi que l'ensemble des relations d'indépendance. La seconde composante quantifie les liens du graphe en utilisant des distributions de possibilité conditionnelles de chaque noeud dans le contexte de ses parents. Ces distributions de possibilité doivent vérifier la contrainte de normalisation. Pour chaque variable A_i :

- Si A_i est un noeud racine et dom_{Ai} le domaine de A_i, la possibilité *a priori* de A_i doit satisfaire :

$$\max_{a_i} \Pi(a_i) = 1, \forall a_i \in dom_{A_i} \tag{3.28}$$

- Si A_i n'est pas un noeud racine, la distribution conditionnelle de A_i dans le contexte de ses parents doit satisfaire :

$$\max_{a_i} \Pi(a_i \left| \theta_{A_i}) = 1, \forall a_i \in dom_{A_i} \right. \tag{3.29}$$

Avec :

dom_{A_i} : Le domaine de A_i

θ_{A_i} : L'ensemble des configurations possibles des parents de A_i

4.2.2 Réseaux possibilistes basés sur le minimum

Un graphe possibiliste basé sur le minimum, noté par GP_M, est un graphe possibiliste où les possibilités conditionnelles sont obtenues par le conditionnement minimum (formule 3.30). La distribution de possibilité des réseaux possibilistes basée sur le minimum, notée par π_M, est obtenue par la règle de chaînage :

$$\pi_M(A_1, A_2, ..., A_N) = MIN_{i=1..N} \Pi(A_i \left| \theta_{A_i}) \right. \tag{3.30}$$

Avec : *MIN* est l'opérateur minimum.

4.2.3 Réseaux possibilistes basés sur le produit

Un graphe possibiliste basé sur le produit, noté par GP_P , est un graphe possibiliste où les possibilités conditionnelles sont obtenues par le conditionnement produit (formule 3.31). La distribution de possibilité des réseaux possibilistes basés sur le produit, notée par π_P , est obtenue par la règle de chaînage :

$$\pi_P(A_1, A_2, ..., A_N) = PROD_{i=1..N} \Pi(A_i \left| \theta_{A_i}) \right. \tag{3.31}$$

Avec : *PROD* est l'opérateur produit.

4.2.4 Exemple des Réseaux Possibilistes

La figure 3.9 représente un exemple des réseaux possibilistes. Les tableaux 3.2 et 3.3 fournissent les distributions de possibilité conditionnelles et a priori associées aux variables binaires A, B, C et D. En utilisant la règle de chaînage (3.30) basée sur l'opérateur minimum, la distribution jointe liée au réseau possibiliste est donnée par le tableau 3.4. En particulier [Ben Amor et al., 2006]:

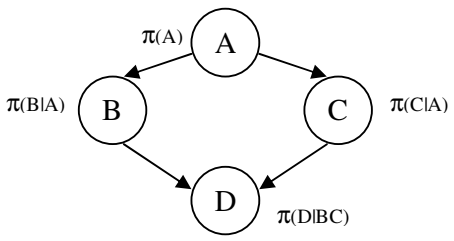


Figure 3.9 : Exemple de réseau causal possibiliste

a	**Π(a)**	**b**	**a**	**Π(b|a)**	**c**	**a**	**Π(c|a)**
a_1	1	b_1	a_1	1	c_1	a_1	0,3
a_2	0,9	b_1	a_2	0	c_1	a_2	1
		b_2	a_1	0,4	c_2	a_1	1
		b_2	a_2	1	c_2	a_2	0,2

Tableau 3.2 : Distribution de possibilité initiales (1)

$\pi(a_1b_2c_1d_2) = min\ (\pi(a_1), \pi(b_2|a_1), \pi(c_1|a_1), \pi(d_2|b_2c_1))$
$= min\ (1 ; 0,4 ; 0,3 ; 0,8) = 0,3.$

d	**b**	**c**	**Π(d|b ∧ c)**	**d**	**b**	**c**	**Π(d|b ∧ c)**
d_1	b_1	c_1	1	d_2	b_1	c_1	1
d_1	b_1	c_2	1	d_2	b_1	c_2	0
d_1	b_2	c_1	1	d_2	b_2	c_1	0,8
d_1	b_2	c_2	1	d_2	b_2	c_2	1

Tableau 3.3 : Distribution de possibilité initiales (2)

ω	π(ω)	ω	π(ω)	ω	π(ω)	ω	π(ω)
$a_1b_1c_1d_1$	0,3	$a_1b_2c_1d_1$	0,3	$a_2b_1c_1d_1$	0	$a_2b_2c_1d_1$	0,9
$a_1b_1c_1d_2$	0,3	$a_1b_2c_1d_2$	0,3	$a_2b_1c_1d_2$	0	$a_2b_2c_1d_2$	0,8
$a_1b_1c_2d_1$	1	$a_1b_2c_2d_1$	0,4	$a_2b_1c_2d_1$	0	$a_2b_2c_2d_1$	0,2
$a_1b_1c_2d_2$	0	$a_1b_2c_2d_2$	0,4	$a_2b_1c_2d_2$	0	$a_2b_2c_2d_2$	0,2

Tableau 3.4 : Distribution de possibilité jointe

4.2.5 Propagation dans les Réseaux Possibilistes

Un des traitements les plus intéressants que l'on peut appliquer sur les réseaux possibilistes est d'évaluer l'impact de la réalisation d'un certain événement sur le reste des variables. Ce traitement peut être réalisé à travers les algorithmes de propagation qui consistent à calculer les distributions de possibilité a posteriori pour chaque variable *A* sachant l'évidence *E* sur le reste des variables.

Dans les réseaux bayésiens probabilistes, ce problème est classé comme NP-difficile, sauf pour les polyarbres (graphes simplement connectés) où la propagation peut être réalisée en un temps polynomial [Cooper, 1990]. Plusieurs algorithmes de propagation dans les réseaux Bayésiens ont été développés. Ces algorithmes peuvent être classés en deux catégories :

- *méthodes exactes* tels que l'algorithme de Kim et Pearl [Pearl, 1988] (valable pour les graphes orientés acycliques sans boucles) et l'algorithme de propagation dans les arbres de jonction [Jensen, 1996] [Lauritzen et al., 1988] où les graphes initiaux avec boucles sont transformés en arbres de jonction sans boucles.
- *méthodes approximatives* qui fournissent une estimation des lois marginales et s'avèrent utiles avec certaines applications de grandes tailles. On peut citer à titre d'exemple la méthode Monte-Carlo [Chavez et Cooper, 1990].

Les algorithmes de propagation possibilistes qui ont été proposés dans la littérature sont, principalement, une adaptation directe des méthodes exactes [Fonck, 1994] [Borgelt et al., 1998], avec la même complexité algorithmique.

Par ailleurs, les algorithmes proposés pour les réseaux possibilistes basés sur l'opérateur produit sont très semblables aux algorithmes probabilistes puisqu'ils utilisent le même opérateur. Ceci n'est pas le cas si on utilise l'opérateur minimum. En effet, cet opérateur

possède des propriétés particulières, telles que l'idempotence, qui peuvent être exploitées afin d'éviter les adaptations directes. Ces propriétés ont motivés [Ben Amor et al., 2003] pour mieux étudier ces réseaux et proposer un nouveau algorithme de propagation pour les réseaux possibilistes basés sur le conditionnement ordinal.

4.3 Les interprétations de la théorie des possibilités

La théorie des possibilités permet de représenter tous les cas où il existe un ensemble convexe de mesure de probabilités admissibles sur Ω de cardinal fini ; cet ensemble étant défini comme l'ensemble des mesures de probabilités P vérifiant les contraintes [Fabiani, 1996] :

$$\forall\ A \in \mathcal{P}(\Omega),\ N(A) \leq P(A) \leq \Pi(A) \qquad (3.32)$$

Dans ce cadre Dubois et Prade suggèrent que [Fabiani, 1996] :

a. la théorie des possibilités est adaptée pour évaluer des degrés de vraisemblance sur des hypothèses d'état rivales et précises (les singletons par exemple).

b. la théorie des possibilités est plutôt adaptée pour représenter des degrés de confiance sur des propositions imprécises, c'est-à-dire pouvant chacune être partitionnée en sous hypothèses d'état mutuellement exclusives entre lesquelles on ne peut pas décider.

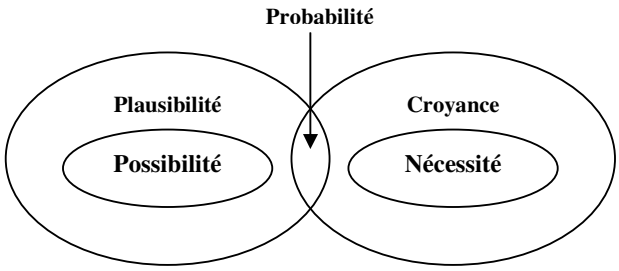


Figure 3.10 : Les limites des théories de traitement de l'incertitude [Gacôgne, 1997]

Par ailleurs, l'usage de la théorie des possibilités en RI avait déjà été suggéré par Prade et Testemale [Prade et Testemale, 1987] qui proposaient un nouveau modèle d'indexation sous forme de groupes de mot-clés, pondérés par des degrés de possibilité et de nécessité. De leur part, [Brini et Boughanem, 2003] ont proposé un modèle de reformulation de requête basé sur la technique de *Relevance Feedback.* Ensuite, ces mêmes auteurs ont proposé avec Didier Dubois un modèle possibiliste quantitatif de recherche d'information [Brini, 2005]. Nous mettons l'accent particulièrement, dans cet état de l'art, sur ce modèle en vue de proposer dans la suite une extension vers un cadre qualitatif possibiliste.

5. Modèle possibiliste quantitatif de RI

Le modèle proposé par Brini et al. [Brini et al., 2004abc] utilise d'une nouvelle manière les connaissances disponibles. Ces connaissances concernent les documents de la collection ainsi que la liste des termes d'indexation et de leur fréquence. Les documents de la collection ainsi que leurs termes d'indexation sont représentés par des réseaux naïfs possibilistes. Considérant un terme relatif à un document, une relation de dépendance quantifiable existe entre un terme et un document. La requête déclenche un processus de propagation entraînant le changement de croyance sur les nœuds documents.

Ce processus de recherche peut être analogue à une étape de diagnostic dans le domaine médical. La collection de documents est comme un ensemble de maladies possibles, les symptômes sont les termes. La requête est vue comme une observation. Le but étant de trouver la maladie (document) plausiblement développée par le patient (requête), étant donnés les symptômes qu'il présente. Dans le modèle proposé la pertinence est représentée dans le cadre quantitatif.

5.1 Architecture du modèle

Le modèle est représenté par un réseau possibiliste d'architecture définie sur la figure 3.11. Pour cette approche les relations de dépendance existant entre termes (terme-terme) et entre les documents (document-document) ne sont pas traitées [Brini et al., 2004abc].

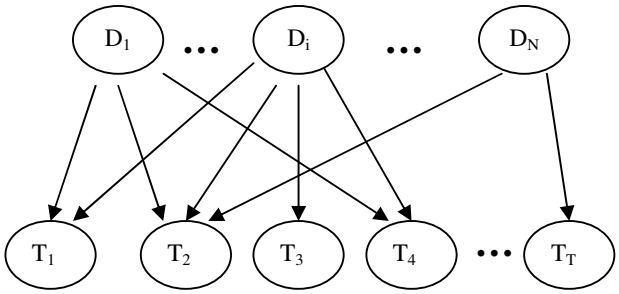


Figure 3.11 : Architecture générale du modèle possibiliste quantitatif

Avec :

Nœud D_j = nœud d'un document de la collection. Les variables D_j sont binaires. Le domaine de D_j est $\{d_j, \neg d_j\}$. L'instanciation $D_j = d_j$ signifie que le document D_j est pertinent pour la requête. $D_j = \neg d_j$, signifie que le document D_j est non pertinent.

Nœud T_i = nœud terme. C'est un terme d'indexation du document. Les variables T_i sont binaires. Le domaine d'un terme est $dom(T_i) = \{t_i, \neg t_i\}$. $T_i = t_i$ signifiera que le terme i est représentatif du document recherché, $T_i = \neg t_i$ signifie que le terme i est non représentatif de ce document. Ce domaine est lié au contexte du parent.

Arc : un arc orienté d'un noeud document D_j vers les noeuds termes d'indexation exprime une relation de dépendance entre le document et les termes qu'il contient. Un arc entre un nœud D_i et un noeud T_j traduit la possibilité et la nécessité que T_i soit représentatif (ou non) du document D_j et ceci en fonction de sa fréquence dans le document et de celle dans la collection.

5.2 Evaluation des poids du réseau

Pour évaluer la possibilité et la nécessité de pertinence, Brini et al. ont besoin de définir explicitement la pertinence représentée par des arcs dans le réseau. Une nouvelle interprétation de la pondération des termes est suggérée. L'approche proposée tente de distinguer entre les termes possiblement représentatifs des documents (ceux qui sont absents sont écartés) et ceux nécessairement représentatifs, c'est-à-dire les termes qui suffisent à caractériser les documents.

Hypothèse 1 **:** Un terme est d'autant moins représentatif d'un document qu'il apparaît peu fréquemment dans ce document ;

Hypothèse 2 **:** Un terme est d'autant plus nécessairement représentatif du document qu'il apparaît fréquemment dans ce document et peu fréquemment dans les autres documents de la collection.

Hypothèse 3 **:** A priori, un document possède une égale possibilité d'être pertinent ou non pour un utilisateur potentiel, soit

$$\Pi(d_j) = \Pi(\neg d_j) = 1, \forall\ j \tag{3.33}$$

D'après l'hypothèse 1, $\Pi(t_i \mid d_j)$ peut être estimée avec la fréquence tf_{ij} de t_i dans d_j :

$$\Pi(t_i \mid d_j) = nft_{ij} = tf_{ij} / \max(tf_{kj}) \tag{3.34}$$

Où nft_{ij} est la fréquence normalisée. Notons qu'avec l'hypothèse 3, on peut en déduire que :

$$\Pi(t_i \wedge d_j) = \Pi(t_i \mid d_j) \tag{3.35}$$

Un terme de poids 0 signifie que le terme n'est pas compatible avec le document. S'il est égal à 1, alors le terme est possiblement représentatif ou pertinent pour décrire (donc représenter) le document. Ici, le terme "représentatif" ne doit pas être considéré au sens large, mais comme "pertinent pour restituer le document". Si un terme est représentatif du document, dans le sens général, il n'aiderait pas forcément à restituer le document. Typiquement, pour un document traitant de la "logique floue", le terme "floue" est très représentatif, mais uniquement potentiellement, puisqu'il ne le caractérise pas sur une collection de documents traitant du même domaine. Notons que le degré de possibilité est normalisé (son maximum vaut 1). Ce degré évalue à quel point un terme est "typique" du document et donc à quel point il est possible qu'il contribue à sa restitution. S'il apparaît avec une fréquence maximale, alors il est considéré comme le meilleur candidat potentiel pour sa représentation.

En logique possibiliste, la mesure de possibilité possède une mesure duale : la nécessité. Celle-ci, dans ce contexte, exprime l'idée que s'il est certain qu'un terme ne représente pas un document, alors il est certain que la présence de ce terme rejette le document. Cette certitude est exprimée par :

$$N(t_i \rightarrow \neg d_j) \geq 1 - nft_{ij}, \tag{3.36}$$

où $\rightarrow$ l'implication matérielle.

Un terme discriminant dans une collection, est un terme qui apparaît fréquemment dans peu de documents de la collection. Un terme discriminant est un terme nécessairement représentatif du document, il contribue à sa sélection et donc à sa restitution en réponse à une requête. Brini et al. Définissent un degré de nécessaire pertinence ϕ_{ij}, du terme t_i pour représenter le document d_j, par :

$$N(t_i \rightarrow d_j) \geq \phi_{ij} \tag{3.37}$$

$$\text{Et } \phi_{ij} = \mu_1(nC/nd_i) * \mu_2(nft_{ij}) \tag{3.38}$$

Où

- nC = nombre de documents de la collection,
- nd_i = nombre de documents de la collection contenant le terme t_i ,
- μ_1 et μ_2 = fonctions de normalisation. Typiquement μ_1: fonction croissante de type logarithmique, μ_2 : la fonction identité.

Ce degré de nécessaire pertinence va donc permettre de limiter la possibilité que le terme soit compatible avec le rejet du document par :

$$\Pi(t_i \wedge \neg d_j) \leq 1 - \phi_{ij} \quad (3.39)$$

Le tableau 3.5 donne la distribution de possibilité la moins spécifique obéissant aux contraintes (3.36) et (3.37) définie sur $\{d_j, \neg d_j\} \times \{t_i, \neg t_i\}$.

	d_j	$\neg d_j$
t_i	nft_{ij}	$1 - \phi_{ij}$
$\neg t_i$	1	1

Tableau 3.5 : Distribution de possibilité

5.3 Un simple schéma de propagation

Dans le cadre numérique, les valeurs de possibilité et de nécessité, a priori et conditionnelles, ont un sens. L'idée est de répondre à des propositions du type :

- "d_i est pertinent pour Q" est possible ou non, quantifiée par $\Pi(d_i|Q)$,
- "d_i est pertinent pour Q" est certain ou non, quantifiée par $N(d_i|Q)$.

Pour le modèle de base de Brini et al. présenté ici, la requête est composée d'une simple liste de mots-clés. Lorsque la requête est connue, un processus de propagation est déclenché à travers le réseau, modifiant les valeurs des possibilités a priori des documents (ici possibilité 1 partout) en vertu de leurs liens avec les termes d'indexation. Dans ce modèle, la formule de propagation est identique à celle des réseaux Bayésiens naïfs [Ben Farhat et al., 2002]. Cependant, deux évaluations indépendantes sont réalisées : $\Pi(d_j|Q)$ et $\Pi(\neg d_j|Q)$ (car leur somme ne vaut pas 1). Soit une requête $Q = (t_i, \ldots, t_T)$ (interprétée conjonctivement), alors

$$\Pi(d_j|Q) = (\Pi(Q|d_j) * \Pi(d_j)) / \Pi(Q) \quad (3.40)$$

La possibilité de pertinence évalue à quel point $D_j = d_j$ est possiblement pertinent étant donnée une requête Q. Lorsque cette valeur vaut 0 le document est écarté. Le modèle suppose de plus l'indépendance conditionnelle des termes.

***Hypothèse 4* :** les termes de chaque document de la collection sont conditionnellement indépendants de ce document.

Si le document D_j est composé des termes T, l'hypothèse ci-dessus, jointe à l'hypothèse 3 d'absence de connaissance a priori sur la pertinence des documents, simplifie la formule (3.40) lorsque le document est instancié positivement ($D_j = d_j$) :

$\Pi(d_j|Q)$ est alors proportionnel à :

$$\Pi'(d_j|Q) = \Pi(t_1|d_j) * \ldots * \Pi(t_T|d_j) = nft_{1j} * \ldots * nft_{Tj} \quad (3.41)$$

Pour comparer les possibilités de pertinence des documents de la collection, uniquement ce numérateur est utile. Le numérateur (3.36) de la formule (3.35) mesure la pertinence potentielle relative d'un document pour une requête.

La certitude de restituer un document pertinent d_j pour une requête, notée $N(d_j|Q)$, est donnée par :

$$N(d_j|Q) = 1 - \Pi(\neg dj|Q) \quad (3.42)$$

Avec

$$\Pi(\neg d_j|Q) = (\Pi(Q|\neg d_j) * \Pi(\neg d_j)) / \Pi(Q) \quad (3.43)$$

Lorsque le document est instancié et d'après les hypothèses 3 et 4, $\Pi(\neg d_j| Q)$ est alors proportionnel à :

$$\Pi'(\neg d_j| Q) = \Pi(t_1| \neg d_j)* \ldots *\Pi(t_T| \neg d_j) \quad (3.44)$$

Ce numérateur peut être exprimé par :

$$\Pi'(\neg d_j| Q) = (1- \phi_{1j})*\ldots* (1- \phi_{Tj}) \quad (3.45)$$

Les documents préférés sont ceux qui ont une valeur $N(d_j|Q)$ élevée parmi ceux qui ont une valeur $\Pi(d_j| Q)$ élevée aussi. Si $N(d_j| Q)$ vaut zéro, les documents restitués sont (sans garantie d'adéquation totale), ceux qui ont une valeur $\Pi(d_j| Q)$ élevée. Notons que si la requête contient des mots-clés non souhaités t_k, on remplace $\Pi(t_k| d_j)$ par $\Pi(\neg t_k| d_j)$ (=1), et de même pour $\Pi(t_k| \neg d_j)$, dans les formules (3.41) et (3.44).

En conclusion, l'approche possibiliste quantitative présentée ci-dessus fournit un nouveau cadre pour l'évaluation de la pertinence aussi bien pour la représentation des documents et de la requête que pour la sélection des documents en réponse à un besoin utilisateur, et ceci en modélisant l'imprécision dans la définition de la pertinence. Les mesures de possibilité et de nécessité sont utilisées pour quantifier les relations de dépendance (ou indépendance) entre les termes et les documents qu'ils indexent et permettent de restituer les documents nécessairement ou possiblement pertinents étant donné une requête.

6. Reformulation de requêtes dans le modèle possibiliste

La problématique à laquelle s'intéresse [Chouaib, 2006] concerne la reformulation de requêtes par réinjection de pertinence possibiliste. Particulièrement, l'auteur a profité des informations concernant les termes, qui sont fournies par le modèle possibiliste de point de vue pertinence (Possible et nécessaire), pour trouver les meilleurs termes d'indexes dans les documents jugés pertinents par l'utilisateur pour pouvoir reconstruire une nouvelle requête. En fait, le modèle proposé se base sur la formule de Rocchio, donnée par l'équation suivante :

$$Q_1 = Q_0 + \frac{1}{n_1}\sum_{i=1}^{n_1} P_i - \frac{1}{n_2}\sum_{i=1}^{n_2} NP_i \quad (3.46)$$

Où n_1 est le nombre de documents pertinents et n_2 est le nombre de documents non pertinents.

En se basant sur cette formule, [Chouaib, 2006] a proposé d'y intégrer la possibilité et la nécessité de termes. Ceci nécessite un changement dans la formule (3.46). Ainsi, la formule proposée est de la forme suivant :

$$Q_1 = \alpha Q_0 + \beta F(P) - \gamma F(NP) \quad (3.47)$$

Avec :

Q : est le vecteur de la nouvelle requête ;

Q_0 : Est le vecteur de la requête initiale ;

P : Liste de documents pertinents restitués et évalués ;

NP : Liste de documents non pertinents restitués et évalués ;

F : Fonction qui combine les pondérations de chaque terme dans la liste des documents pertinents (respectivement Non pertinents) pour trouver un poids final où à partir de ce poids seront choisis les meilleurs termes.

α : Paramètre positif permet de pondérer les termes de la requête initiale ;

β: Paramètre positif permet de pondérer les termes des documents jugés pertinents par rapport aux documents non pertinents ;

γ : Paramètre positif permet de pondérer les termes des documents jugés non pertinent.

Par ailleurs, dans la liste de documents restitués (pertinent ou non pertinent), un terme peut exister dans plusieurs documents, mais son poids possibiliste et nécessaire change d'un document à un autre. Alors, il faut trouver le moyen pour agréger tous les poids d'un même terme dans la liste des documents.

[Chouaib, 2006] a proposé cinq formules, deux formules basées sur la nécessité, deux autres basées sur la possibilité et le cinquième est une combinaison des deux. Ces formules ont été définies dans le but de calculer les nouveaux poids pour les termes de la nouvelle requête lors du processus de réinjection de pertinence (*Relevance Feedback*).

La fonction *F* est alors une fonction qui applique l'une des cinq formules proposées sur l'une de deux listes de documents et qui trie le résultat final des poids des termes par ordre décroissant et renvoie les *n* premiers termes.

6.1 Formules basées sur la nécessité de termes

[Chouaib, 2006] a proposé deux formules qui sont basées sur la nécessité de termes ($N(t_i|D_j)$), à savoir la nécessité moyenne et la *Nécessité**(r/R). Nous détaillons dans la suite ces deux types de nécessité.

Le poids final de chaque terme est donné par l'équation (3.48), dans le cas de la *Nécessité moyenne*, et par l'équation (3.49) dans le cas de la *Nécessité ** (r/R) :

$$poidsfinal(t_i) = \frac{1}{R}\sum N(t_i|D_j) \tag{3.48}$$

$$poidsfinal(t_i) = \frac{r}{R}\sum N(t_i|D_j) \tag{3.49}$$

Avec : $N(t_i|D_j)$: la nécessité de t_i étant donné D_j ;

$D_j = \begin{cases} d_j \text{ s'il s'agit de la liste de documents pertinents ;} \\ \overline{d}_j \text{ s'il s'agit de la liste de documents non pertinents.} \end{cases}$

$R = \begin{cases} R_1 \text{ si } D_j = d_j \text{ ; avec } R_1 \text{ le nombre de documents pertinents ;} \\ R_2 \text{ si } D_j = \overline{d}_j \text{ ; avec } R_2 \text{ le nombre de documents non pertinents.} \end{cases}$

$r = \begin{cases} r_1 \text{ si } D_j = d_j \text{ ; avec } r_1 \text{ le nombre de documents pertinents contenant le terme } t_i \text{ ;} \\ r_2 \text{ si } D_j = \overline{d}_j \text{ ; avec } r_2 \text{ le nombre de documents non pertinents contenant le terme } t_i. \end{cases}$

6.2 Formules basées sur la possibilité de termes

Le même auteur [Chouaib, 2006] a proposé deux autres formules basées sur la possibilité de termes ($\Pi(t_i|D_j)$), à savoir la *possibilité moyenne* et la *possibilité**(r/R) :

Le poids final de chaque terme est donné par l'équation (3.50), dans le cas de la *Possibilité moyenne*, et par l'équation (3.51) dans le cas de la *Possibilité* * (*r*/*R*) :

$$poidsfinal(t_i) = \frac{1}{R}\sum \prod (t_i | D_j) \tag{3.50}$$

$$poidsfinal(t_i) = \frac{r}{R}\sum \prod (t_i | D_j) \tag{3.51}$$

Où Π $(t_i|D_j)$: la possibilité de t_i étant donné D_j ;

6.3 Formules basées sur la possibilité et la nécessité

Cette dernière formule est une combinaison de la possibilité et de la nécessité. La formule proposée est donnée par l'équation (3.52) :

$$poidsfinal(t_i) = \frac{1}{R}\sum \prod (t_i | D_j) * N(t_i | D_j) \tag{3.52}$$

Où Π $(t_i|D_j)$: la possibilité de t_i étant donné D_j ;

$N(t_i|D_j)$: la nécessité de t_i étant donné D_j.

Ainsi, [Chouaib, 2006] a proposé une nouvelle méthode possibiliste de reformulation de requête par réinjection de pertinence basé sur le jugement de l'utilisateur sur les documents restitués en intégrant la possibilité et la nécessité d'un terme. L'intégration de ces deux degrés de pertinence a aidé à préciser les termes à ajouter dans la nouvelle requête. Suivant les formules proposées, l'auteur a suggéré de choisir les *n* premier termes par ordre décroissant de leur pertinence finale (possible et nécessaire). Les résultats de ce processus ont effectivement amélioré les performances du moteur possibiliste de base dans la restitution de documents en réponse aux besoins d'utilisateurs. La précision moyenne a augmentée de plus de 53% pour les cinq formules proposées et elle atteint 121% pour la formule de nécessité normalisée et pour n = 10. Ces résultats montrent que l'introduction de la possibilité et de la nécessité est intéressante et fiable pour la reformulation par réinjection de pertinence.

7. Modèle Bayésien versus Modèle Possibiliste

Suite à cet état de l'art, nous distinguons deux principaux modèles basés sur les réseaux Bayésiens pour répondre aux besoins de la RI : le modèle de croyance instanciant la requête et le modèle inférentiel instanciant le document à la réception d'une requête. Une différence majeure dans la topologie de ces deux réseaux concerne le sens de la dépendance des termes d'indexation avec les documents. Dans le modèle de croyance la relation de dépendance est orientée des termes, qui constituent l'univers de discours, vers les documents et est quantifiable par $P(d_j|t_i)$. Pour le modèle inférentiel cette dépendance, quantifiée par $P(t_i|d_j)$, va des documents vers ses termes d'indexation.

Dans le modèle Bayésien, la notion de pertinence permet la généralisation des modèles de base, mais est difficilement *raffinable*. Par ailleurs, l'évaluation des documents par rapport à une requête ne prend en compte que les termes d'indexation présents à la fois dans les documents et la requête. En effet, l'absence des termes de la requête n'est pas traitée explicitement dans ces deux modèles, bien que dans le modèle de croyance les termes d'indexation de la requête constituent le point d'entrée du système (le processus de recherche est instancié par la réception de la requête).

Dans le modèle inférentiel, il existe une définition ambiguë de la probabilité *a priori* d'un document. Les documents de la collection sont représentés par des nœuds dans le réseau. Chaque nœud est de domaine binaire et la probabilité *a priori* d'un document devrait alors être égale à 1/2 et non pas à $1/N$ comme défini dans [Turtle, 1991]. Cette dernière définition ($P(d_j) = 1/N$) signifierait que tous les documents sont représentés dans un seul nœud représentant tous les documents de la collection et donc que $dom(D_j) = \{d_1, \ldots, d_N\}$.

Quant au modèle possibiliste de la RI, il traite l'incertitude d'une manière novatrice basée sur la théorie des possibilités et particulièrement les Réseaux possibilistes. Les nœuds dans ce réseau représentent les documents, les termes d'indexation ainsi que le besoin utilisateur. Les arcs reliant chaque couple de nœuds décrivent une relation de dépendance et sont quantifiés par deux mesures : la possibilité et la nécessité. Quel que soit le type de la relation décrite par un arc entre deux nœuds, sa quantification est engendrée par deux mesures. Alors que la première est utile pour écarter certaines informations, la seconde mesure renforce les informations restantes.

D'autre part, ce modèle considère que la restitution d'un document en réponse à une requête utilisateur peut être considérée dans un cadre d'inférence. En effet, la restitution d'un document est « causée » par la soumission d'une requête au système. Les données sur lesquelles se basent les modèles de la littérature pour restituer une liste de documents en réponse à un besoin utilisateur sont pauvres, incertains et imprécis. La logique possibiliste se prête naturellement à ce genre d'application. En fait, le modèle possibiliste a pu déterminer deux types de pertinence : la nécessaire et la plausible. Le première permet de renforcer « nos croyances » vis-à-vis des résultats de la recherche et la seconde permet d'éviter de restituer une liste de documents vides à une requête utilisateur et d'en écarter ceux qui ne sont pas intéressants. La combinaison de la représentation par réseaux et de l'utilisation de la théorie des possibilités, a permis de répondre à un tel type de pertinence. La requête introduit de l'information qui change nos croyances sur les nœuds termes d'indexation ainsi que leurs nœuds parents. La liste des documents restitués contient les documents nécessairement pertinents en haut de la liste, puis les documents plausiblement pertinents.

A notre sens, un cadre théorique intéressant, permettant à la fois d'exprimer l'ignorance et de tenir compte de l'imprécis et de l'incertain, est possible grâce à la théorie des possibilités. En fait, notre apport consiste à étendre l'approche possibiliste d'un cadre quantitatif à un cadre qualitatif. Cette extension consiste à rechercher les termes de la requête non pas dans la totalité d'un document, mais dans ses structures logiques. En effet, l'utilisateur devient capable de savoir les emplacements des informations recherchées dans les fragments des documents retrouvés par le SRI proposé. Autrement dit, il pourra demander au système des documents contenant des textes, des tableaux ou des figures à propos des mots-clés proposés. Il pourra aussi changer son profil d'une requête à une autre. En conséquence, la qualité des documents retournés change en terme de pertinence, en passant d'un profil à un autre. Cette nouvelle technique d'affinement de la recherche des documents permet entre autres d'engendrer de nouvelles définitions de la pertinence dans un SRI.

8. Conclusion

L'état de l'art que nous avons réalisé sur les SRI a montré que les modèles dits de première génération présentaient un intérêt par rapport à un contexte de recherche statique. D'une part, ces modèles étaient centrés sur la représentation de la requête de l'utilisateur et du document, et d'autre part, sur la mise en correspondance directe entre ces deux représentations pour déterminer les documents pertinents selon la vision du système. Nous citons dans ce cadre : le modèle booléen, le modèle vectoriel et le modèle probabiliste. Afin d'enrechir ces deux

représentations auxquelles sont associées deux types de connaissances : connaissances relatives aux documents et connaissances relatives à la requête, des extentions ont été proposées. Par ailleurs, ces extensions ont permis d'enrichir le niveau d'analyse des documents, notamment en introduisant l'indexation sémantique latente, les domaines sémantiques, les réseaux d'inférence bayésiens et les réseaux possibilistes. D'autre part, ces extensions ont concerné le niveau d'analyse de la requête, notamment le modèle booléen étendu, en introduisant des poids aux termes et des liens entre eux.

Ces derniers modèles, avec les extensions proposeés, avaient également tenté de prendre en considération d'autres types de connaissances. Ces connaissances sont liées aux domaines traités dans le corpus documentaire et au besoin d'information de l'utilisateur qui est en rapport avec l'utilisateur lui-même. Pour ce faire des techniques de reformulation de requêtes et de clustering ont été introduites dans le processus de recherche. Toutes ces techniques visaient à améliorer la recherche en ramenant des documents qui sont potentiellement pertinents mais qui ne sont pas retrouvés par une recherche directe. Cependant elles sont restées limitées à un cadre où les connaissances citées sont statiques.

Bien que ces modèles présentent des avantages liés aux points cités précédemment, ils présentent encore des limites. En fait, d'autres exigences non traités ou partielllement traités par tous ces modèles sont importants à prendre en considération dans un SRI :

- La proposition de différentes alternatives à l'utilisateur pour interroger et interagir avec le corpus et notamment par classification et par des vues thématiques.
- La gestion et la prise en compte, de manière plus efficace, de l'utilisateur dans le processus de recherche et notamment l'opération de mise en correspondance. En effet, le profil de l'utilisateur est une composante qui s'apprend par le système et qui évolue à travers les différentes sessions effectuées par ce dernier.

Notre objectif dans cette thèse est de proposer un modèle pour un SRI qui prend en compte ces nouvelles exigences et qui permet de les intégrer en se basant sur une forte composante classificatoire à base de Réseaux Petits Mondes Hiérarchiques (RPMH). D'une part, nous proposons également d'introduire dans le processus de reformulation sémantique de requêtes une phase de classification de termes de la requête qui permet d'explorer ces termes en fonction de leurs proximités sémantiques (proxémie de surface). En effet, l'utilisateur pourra identifier les classes des termes sémantiquement proches des termes de sa requête initiale pour construire sa requête reformulée. D'autre part, les documents retrouvés par le système seront aussi classifiés selon leurs proximités thématiques (proxémie en profondeur) afin de montrer leurs corrélations et faciliter leurs consultations. Nous présentons dans le chapitre suivant le modèle que nous proposons en mettant en avant les nouvelles fonctionnalités qu'il offre.

Deuxième Partie :

Conception et architecture d'un Système multi-Agent de Recherche Intelligente POssibiliste de Documents Web, SARIPOD

Chapitre 4

Modèle d'un SRI à base de Réseaux Petits Mondes Hiérarchiques et de Réseaux Possibilistes

Au terme de cette étude de l'état de l'art, nous avons remarqué que les exigences attendues d'un SRI dépassent celles qui étaient prévues au départ. C'est la raison pour laquelle de nombreuses approches ont été rajoutées aux approches de base. Les études en cours se sont donc orientées vers une recherche intelligente qui vise à satisfaire au mieux le besoin de l'utilisateur en le considérant comme membre actif dans le processus de recherche et en lui fournissant différentes manières d'accéder et d'explorer le corpus. Ces derniers sont en évolution contenue.

Notre problématique est donc de proposer un Système de Recherche d'Information (SRI) :

- qui intègre l'utilisateur dans le processus de recherche et s'adapte à ses besoins. Ce qui permet de construire des préférences (centres d'intérêts) constituant les profils utilisateurs. Ces préférences s'améliorent au fur et à mesure et permettent de guider le système et l'utilisateur dans le processus de recherche. En effet, un utilisateur peut être assisté, grâce à ses préférences, pour identifier ses besoins de manière plus précise et cerner ses préférences à partir des profils "similaires" d'autres utilisateurs. Le système peut se servir de ces préférences pour effectuer une recherche plus fine en reconnaissant un utilisateur à travers ses centres d'intérêt ;
- qui construit un premier modèle associé aux requêtes (profils requêtes) ainsi qu'un deuxième modèle associé aux résultats de recherche correspondants (profils documents). En effet, ces deux modèles sont à base de Réseaux Petits Mondes Hiérarchiques (RPMH) et sont utiles pour dégager les similarités sémantiques entre les termes de la requête, d'une part et entre les documents résultats de recherche, d'autre part. Ainsi, les profils requêtes sont exploitables pour la formulation et la reformulation de requêtes, alors que les profils documents sont utiles pour la classification des documents.
- qui traite l'appariement entre le modèle de requête et le modèle de document par un Réseau Possibiliste (RP) permettant de dégager les documents pertinents, au sens possibiliste, vis-à-vis une requête. En fait, cette phase de mise en correspondance est utile pour le raffinement, le filtrage et la purification des réponses aux requêtes.

L'originalité de notre démarche est qu'elle prend tout à la fois les trois dimensions sus-mentionnées pour aboutir à un SRI :

- *Coopératif* à travers le modèle associé aux documents et le modèle associé aux requêtes construites à partir de réseaux petits mondes hiérarchiques ;
- *Adaptatif* aux besoins des utilisateurs ;
- *Intelligent* car le système tient compte des profils dynamiques des ses utilisateurs ;

Ainsi, notre but est également d'offrir à l'utilisateur une interface interactive pour l'interrogation, l'affichage et l'évaluation des réponses proposées par le système en réponse à un besoin d'information.

Dans la première section nous définissons et nous détaillons les différents aspect introduits dans le modèle que nous proposons : modélisation de requêtes, modélisation de documents et mises en correspondance entre les deux. Nous mettons en exergue les choix pris pour les méthodes de classifications introduites pour la construction des différentes connaissances en les justifiant et en présentant les avantages. Il est à noter que les méthodes de classification choisies s'adaptent bien avec les systèmes à caractères coopératifs, adaptatifs et intelligents. Dans la deuxième section nous situons le modèle proposé ainsi que son originalité par rapport à d'autre travaux et particulièrement le modèle possibiliste quantitatif de RI proposé par [Brini et al., 2004abc] et le RPMH de dictionnaire proposé par [Gaume et al., 2004].

1. Modèle conceptuel du système SARIPOD

D'après l'étude effectuée dans le premier chapitre de l'état de l'art, nous avons pu distinguer les acteurs d'un SRI qui sont principalement l'utilisateur et le document. Autour de ces deux acteurs, différents types de connaissances peuvent être construites pour munir un SRI d'une base de connaissances lui permettant de bien agir pour arriver à satisfaire au mieux le besoin d'information de l'utilisateur. Ces connaissances peuvent être classées suivant qu'elles soient liées à l'utilisateur ou aux documents selon ces quatre classes :

- Des connaissances relatives à l'utilisateur ;
- Des connaissances relatives au besoin d'information de l'utilisateur ;
- Des connaissances relatives aux documents ;
- Des connaissances relatives aux concepts du domaine.

Les connaissances relatives à l'utilisateur peuvent être liées à une étape d'une session de recherche, à une ou plusieurs sessions. Il est donc possible de les définir selon trois classes :

- Les *connaissances à court terme* sont relatives à une étape d'une session de recherche ou à l'ensemble de la session de recherche. Elles sont déterminées en synthétisant le besoin de l'utilisateur ainsi qu'en le corrigeant d'une manière incrémentale ;
- Les *connaissances à moyen terme* sont basées sur la prise en compte du comportement de l'utilisateur lié à l'analyse de ses requêtes et de ses décisions vis-à-vis des documents fournis par le système. Cette forme de connaissance n'est pas couramment utilisée dans les SRI étant donné que le profil de l'utilisateur est souvent prédéfini avant la recherche ;
- Les *connaissances à long terme* sont soit relatives aux préférences des utilisateurs, soit issues d'une manière générale des classifications des documents ainsi que la correction incrémentale de l'indexation des documents qui permettent de produire des connaissances stables du contenu d'un fonds documentaire.

Par ailleurs, ces trois types de connaissances sont liés. En effet, les connaissances à court terme interviennent dans l'élaboration des connaissances à moyen et à longs termes.

Nous avons pu également étudier les différentes opérations concernées par un SRI qui sont principalement :

- La phase de représentation ou modélisation de l'utilisateur et de la requête ;
- La phase d'analyse qui permet d'aboutir à une représentation ou modélisation des documents ;
- La phase de mise en correspondance ou d'appariement ;
- La phase d'évaluation.

Outre ces opérations élémentaires et nécessaires, l'idée motrice du modèle est d'intégrer dans la stratégie de recherche d'un SRI des composantes classificatoires pour les documents et d'autres composantes classificatoires pour les requêtes. En effet, dans le système que nous proposons, une phase de reformulation sémantique de la requête est introduite et qui permet à l'utilisateur d'ajouter des termes sémantiquement proches à ses termes proposés au départ. D'autre part, les documents réponses à cette requête reformulée peuvent subir une classification thématique permettant de réajuster le résultat d'une requête en fonction du contenu du fonds documentaire.

L'objectif de diviser le processus de recherche en deux modèles (de requêtes et de documents) est d'offrir à l'utilisateur plusieurs alternatives de recherche qui ne peuvent que l'assister et enrichir son niveau par rapport à l'ensemble de connaissances gérées dans un SRI tout en cernant ses propres besoins.

La mise en correspondance entre le modèle de requête et le modèle de document est assurée par un réseau possibiliste. En effet, ce modèle présente une nouvelle approche possibiliste pour un système de Recherche d'Information. Ce système, qui voit la Recherche d'Information comme un problème de diagnostic, traduit à l'aide de réseaux possibilistes naïfs des relations de dépendance entre les documents et les termes de la requête. Ces relations sont quantifiables par deux mesures : la possibilité et la nécessité de pertinence. La mesure de possibilité est utile pour filtrer les documents et la mesure de nécessité pour renforcer la pertinence des documents restants. Le processus de recherche restitue les documents plausiblement ou nécessairement pertinents à un utilisateur. De plus, si l'approche de base tient compte ici de l'aspect quantitatif et ne tient pas compte de la dépendance entre les termes de la requête, notre système permet de l'étendre au cadre qualitatif possibiliste, en introduisant des préférences (pondérations) entre les termes de la requête.

Ainsi, l'architecture globale du système SARIPOD est illustrée par la figure 4.1.

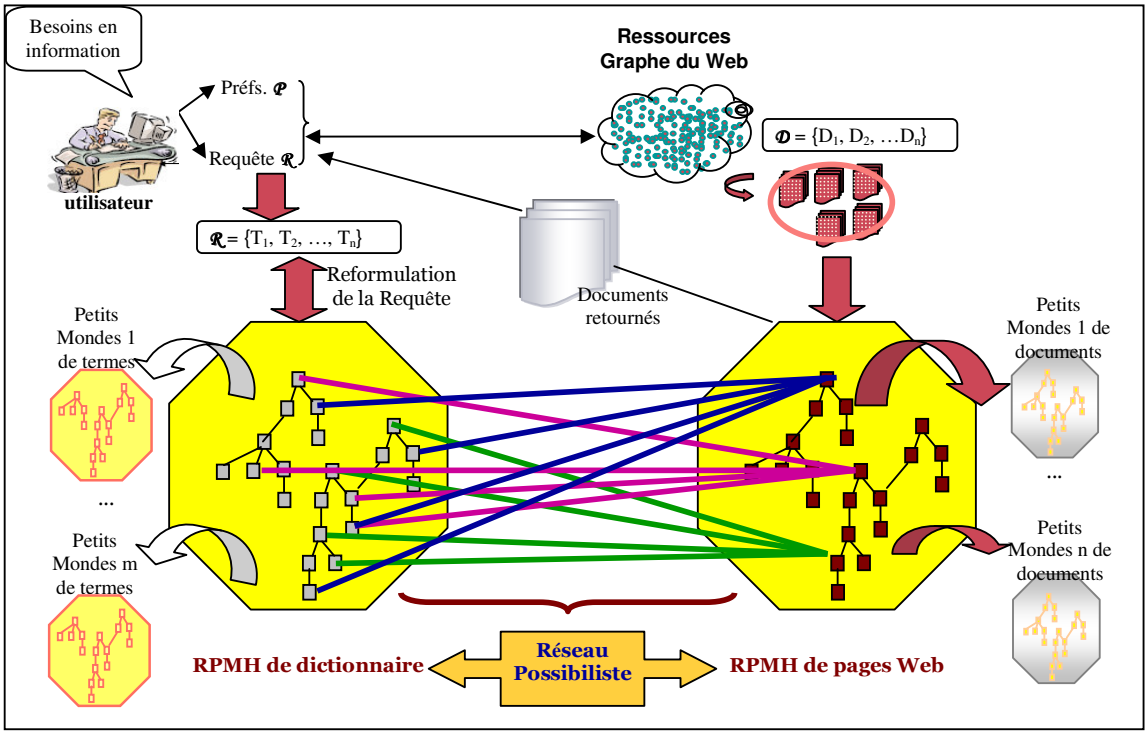


Figure 4.1 : Modèle conceptuel du système SARIPOD

En fait, nous distinguons deux usages très importants de ces deux RPMH (de dictionnaire et de pages Web) ainsi que leur combinaison dans le système SARIPOD [Elayeb et al., 2007a] :

Le premier RPMH est celui qui consiste à structurer les pages Web réponses à une requête en zones denses de pages Web thématiquement liées les unes aux autres. On fait ainsi apparaître des nuages denses de pages qui traitent d'un sujet et des sujets connexes (assez similaires sémantiquement) et qui répondent toutes fortement à une requête. Pour un autre nuage de pages Web fortement liées les unes aux autres il en va de même, elles répondent toutes à cette même requête. La différence essentielle est que chaque nuage de pages Web répond fortement d'une manière particulière à la requête.

Par exemple, la requête "vérifier", dans le RPMH des synonymes des mots du français, donne quatre nuages de verbes proches de vérifier : le premier nuage concerne A = {examiner, voir, éprouver, reconnaître,....}, le deuxième B = {essayer, contrôler, expérimenter, s'assurer,....} etc. pour les deux autres. Pour le Web il en va de même une requête (exprimée avec quelques mots-clés) renvoie un ensemble de pages Web (réponses à la Google, par exemple) qu'il faut organiser en RPMH de sorte à faire apparaître quelques grands nuages de pages Web parmi toutes ces réponses. Chaque nuage regroupe ainsi un lot de pages qui répondent toutes de façon pertinente et d'une certaine façon à la requête. Autrement dit, le premier nuage A répond pertinemment à la requête "vérifier" d'une certaine façon (celle qui s'intéresse à l'"examen"), alors que le second nuage B répond aussi pertinemment à la même requête "vérifier" mais cette fois d'une façon différente (celle qui s'intéresse au "contôle"), etc. Pour le Web chaque nuage de pages Web sera pertinent et, grâce à des mots-clés supplémentaires, il sera possible de sélectionner un nuage particulier ou une partie de ce nuage.

La qualité réside dans le fait que quand on regarde les pages Web d'un même nuage, toutes les pages sont pertinentes, mais si ce degré n'est pas encore suffisant, on peut faire des requêtes dans ce seul nuage (contrairement à Google, par exemple, qui n'organise jamais ses 300.000 réponses en nuages) pour obtenir un sous-ensemble de pages Web que l'on peut de nouveau (donc récursivement) organiser en sous-RPMHs et ainsi de suite. Au plus profond de cette entreprise de structuration on trouve des pages Web seules. L'ensemble des réponses a donc été organisé en RPMH et sous-RPMH de sorte à constituer une structure de classification des pages Web en fonction des mots-clés utilisés. Ce que ne fait pas Google qui sait seulement faire des recherches dans l'ensemble des réponses précédentes. En fait, Google est capable de renvoyer, suite à une sous-requête, des pages que notre système a mis dans des nuages différents (classes des thèmes) lors de la première requête.

Le deuxième usage très important des RPMH est celui qui consiste à ne pas prendre les mots-clés tels qu'ils sont mais à considérer une requête comme multiple en ce sens qu'on ne recherche pas seulement les mots-clés dans les pages Web mais aussi les substantifs qui lui sont sémantiquement "proches". Proche au sens du calcul de la proxémie définie par notre approche basée sur l'étude des circuits dans un RPMH de dictionnaire (détaillée dans la section 1.2). Les mots considérés comme proches incluent donc les synonymes de ce mot mais ne s'y restreignent pas (voir figure 4.2). On aura potentiellement (en pratique cela sera limité par une borne) tous les mots plus ou moins proches du mot de la requête. Ce nombre de mots est paramétrable (1, 5, 100, ...). Une requête est donc maintenant très flexible puisqu'elle tolère qu'une page Web soit une bonne réponse même si elle ne contient pas (à strictement parler) le mot-clé en question.

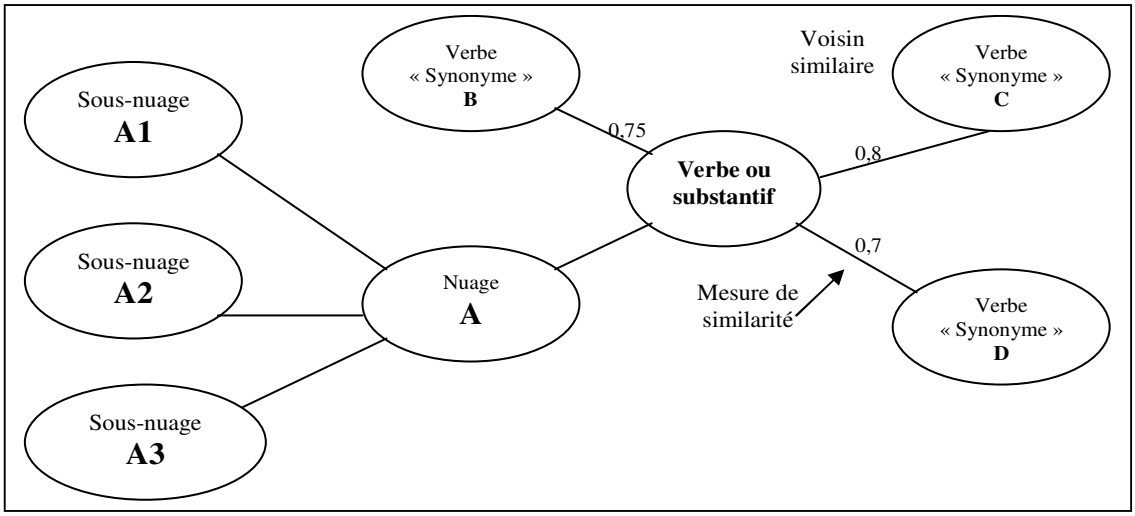


Figure 4.2 : Similarité sémantique entre les verbes

Or pour pouvoir disposer de cette flexibilité nous avons évidemment besoin d'un dictionnaire et surtout d'avoir structuré ce dictionnaire (l'ensemble des entrées de celui-ci) en RPMH justement pour savoir quel mot est proche de quel autre. Or il y a de nombreuses façons de faire émerger une structure de RPMH à partir d'un dictionnaire, celle de [Gaume et al., 2004][16] par exemple consiste à se servir des définitions : le mot M_1 est relié au mot M_2 si et seulement si M_2 appartient à la définition de M_1, à l'aide de cette définition de la relation entre deux mots il en déduit par proxémie la "proximité sémantique" de tout mot à tout autre. Le système SARIPOD reprend cette définition et s'appuie sur cette proxémie entre les mots pour rendre les requêtes plus flexibles. On peut à partir de là quantifier les pages Web obtenues suite à une requête utilisant certains mots-clés. Chaque page réponse sera caractérisée par un degré d'adéquation ou de pertinence qui résultera de la combinaison des degrés de proxémie aux mots-clés de la requête des mots effectivement présents dans cette page [Elayeb et al., 2007d].

Nous détaillons dans la suite les différentes étapes que nous proposons pour la modélisation de requêtes et de documents ainsi que les choix des méthodes de classification introduites. En fait, nous présentons une approche générique de recherche de composantes de sens dans un réseau d'information. Cette approche est valable dans le cas de mots d'un dictionnaire (RPMH de dictionnaire) ainsi que dans le cas de pages Web (RPMH de pages Web).

2. Les RPMH du système SARIPOD

2.1 Définition du RPMH

Des recherches récentes en théorie des graphes ont mis au jour un ensemble de caractéristiques statistiques que partagent la plupart des grands graphes de terrain ; ces caractéristiques définissent la classe des graphes appelée « Réseaux Petits Mondes Hiérarchiques » (RPMH) initialement proposés par [Watts et Strogatz, 1998] et dénommés « Small-World Networks » avant d'être repris par divers auteurs comme [Barabasi et al., 2000] [Ravasz et Barabàsi, 2003] [Newman, 2003] [Portrait, 2003] [Scharffe, 2004] [Gaume, 2004] [Gaume et al., 2004] [Gaume, 2006] [Gaume et al., 2006] [Gaume et al., 2007] [Gaume et Mathieu, 2007] (voir figure 4.3).

[16] Les auteurs se limitent uniquement aux mots de même catégorie gramaticale (les noms).

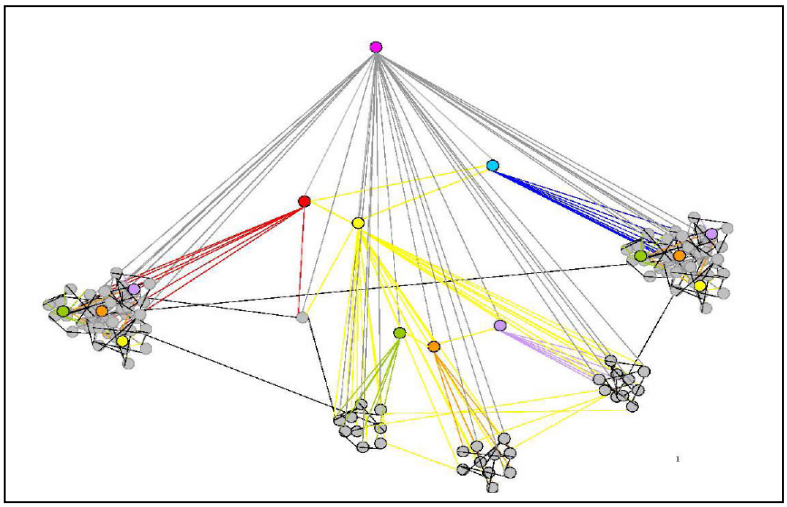


Figure 4.3 : Structure des graphes petits mondes hiérarchiques

Les RPMH sont caractérisés par quatre propriétés fondamentales :

D : ils sont peu denses, c'est-à-dire qu'ils ont relativement peu d'arêtes au regard du nombre de leurs sommets ;

L : la moyenne des plus courts chemins entre les sommets est petite ;

C : le taux de clustering ou d'agrégation, est défini de la manière suivante : C'est la valeur moyenne du rapport, pour chaque sommet, entre le nombre d'arcs entre ses voisins et le nombre total d'arcs possibles entre eux[17]. Le *C* d'un graphe est la moyenne des C_s sur ses sommets. Le *C* d'un graphe est donc toujours compris entre 0 et 1. Plus le *C* d'un graphe est proche de 1, plus il forme des agrégats ou clusters (des zones denses en arêtes). Dans un RPMH, le *C* est fort, les deux voisins d'un même sommet ont tendance à être connectés par une arête (« mes amis sont amis entre eux »). Par exemple, sur Internet[18], deux pages qui sont liées à une même page ont une probabilité relativement élevée d'inclure des liens l'une vers l'autre ;

I : la distribution des degrés d'incidence des sommets suit une loi de puissance (*power law*) : certains nœuds très peu nombreux ont beaucoup plus de voisins que d'autres plus nombreux, eux-mêmes ayant plus de voisins que d'autres qui eux-mêmes... La probabilité $P(k)$ qu'un sommet du graphe considéré ait k voisins décroît comme une loi de puissance $P(k) = k^{\lambda}$ (où $\lambda < 0$).

Le tableau 4.1 [Gaume et al., 2004] présente une comparaison des RPMH avec d'autres types de graphes pour ces différentes caractéristiques : des graphes aléatoires (construits en partant d'un ensemble de sommets isolés, puis en ajoutant aléatoirement un nombre déterminé d'arêtes entre ses sommets), et des graphes réguliers (des graphes classiquement étudiés en théorie des graphes, dont tous les sommets ont le même degré d'incidence) [Douglas et Houseman, 2002] [Sergi et Ricard, 2007] .

[17] Supposons qu'un sommet *S* ait K_s voisins, alors il y a $K_s(K_s\text{-}1)/2$ arêtes au maximum qui peuvent exister entre ces K_s voisins (ce qui arrive quand chacun des voisins de *S* est connecté à tous les autres voisins de *S*). Soit A_s le nombre d'arêtes qu'il y a entre les voisins de *S* (ce nombre est donc nécessairement plus petit ou égal à $K_s(K_s\text{-}1)/2$). Posons $C_s = A_s/(K_s(K_s\text{-}1)/2)$ qui est donc pour tout sommet *S* inférieur ou égal à un.

[18] Les sommets sont les 6 milliards de pages disponibles sur Internet, et une arête est tracée entre *A* et *B* si un lien hypertexte vers la page *B* apparaît dans la page *A* ou si un lien hypertexte vers la page *A* apparaît dans la page *B*.

à densité égale	**L : Moyenne des plus courts chemins**	**C : Taux de clustering**	**I : distribution des degrés d'incidences**
Graphes aléatoires	L petit (chemins courts)	C petit (pas d'agrégats)	loi de Poisson
Graphes de terrain (RPMH)	**L petit (chemins courts)**	**C grand (des agrégats)**	**loi de puissance**
Graphes réguliers	L grand (chemins longs)	C grand (des agrégats)	constante

Tableau 4.1 : Comparaison de trois graphes en fonction des paramètres L, C et I

La forte cractéristique classificatoire des RPMH par rapport aux autres types des graphes justifie davantage nos propositions de modéliser les termes de la requête par un premier RPMH de dictionnaire qui sera utile d'une part dans la classification de ces termes en plusieurs composantes sémantiques et d'autre part dans la reformulation sémantique de requête. Nous modélisons aussi les documents résultats de la recherche par un second RPMH de pages Web qui sera utile dans leurs classifications thématiques.

Ainsi, nous proposons dans la suite une nouvelle approche générique de génération de composantes de sens dans un réseau d'informations. Cette approche est applicable dans le cas d'un réseau de mots d'un dictionnaire ainsi que dans le cas d'un réseau de pages Web. Notons que l'approche de base a été initiée par [Awada, 2005] dans le cas de verbes d'un dictionnaire et développée encore plus par nous même dans le cadre de cette thèse afin de pouvoir l'utilisée dans la reformulation sémantique de la requête dans notre SRI SARIPOD. En fait, notre contribution consiste à commenter et améliorer les algorithmes existants afin de proposer des nouveaux algorithmes de classification en réponses à quelques limites et insuffisances non résolues par [Awada, 2005].

2.2 Approche générique de génération de composantes de sens dans un réseau d'informations

Cette section porte sur l'étude de la ressemblance de sens dans un réseau d'informations tout en traitant le problème de la polysémie de ces informations. Plus précisément, il s'agit de repartir des entités informatives similaires en groupes appelés composantes de sens correspondant chacune à un « sens » de cette entité. Ce modèle sera appliqué à deux types d'entités : les pages Web et les mots d'un dictionnaire.

Le Web, comme le dictionnaire, est un objet représenté par un graphe de type RPMH et le regroupement en familles de ressemblance des entités constitutives de cet objet se fait en étudiant les circuits dans ce graphe. En fait, nous nous sommes appuyé sur l'idée suivante : les entités se trouvant sur un circuit devraient appartenir à la même composante de sens. L'étude a donné lieu à l'implantation d'une interface graphique d'exploitation automatique du réseau (voir chapitre 6 de réalisation).

Nous proposons d'utiliser une structure susceptible de conserver suffisamment de sens pour notre propos : les graphes. Il semble évident qu'il existe différents types d'information, et par conséquent d'arcs, dans ces graphes tels que les rapports de synonymie[19] ou d'antonymie[20]

[19] La **synonymie** est un rapport de proximité sémantique entre des mots ou des expressions d'une même langue. La proximité sémantique indique qu'ils ont des significations très semblables. Des termes liés par synonymie sont des synonymes.

[20] Deux items lexicaux sont en relation d'**antonymie** si on peut exhiber une symétrie de leurs traits sémantiques par rapport à un axe. La symétrie peut se décliner de différentes manières, selon la nature de son support.

entre sommets, d'hyperonymie[21], de co-domaines d'activités. Par conséquent, l'étude des relations qu'entretiennent les entrées d'un dictionnaire entre elles se ramène à une étude sur les graphes cherchant à exploiter les réseaux ainsi établis entre les mots. D'autre part, tous les dictionnaires peuvent être représentés par des graphes dont les sommets et les arcs peuvent être définis de multiples façons. La manière la plus simple est de prendre pour sommets du graphe les entrées du dictionnaire et d'admettre l'existence d'un arc d'un sommet A vers un sommet B si et seulement si l'entrée B apparaît dans la définition de l'entrée A.

Les dictionnaires sont des sources de données pertinentes dans tout traitement automatique du langage naturel. En effet, ce sont des objets constitués, formels, comparables, existant dans presque toute langue, et surtout porteurs de sens. L'idée est la suivante : si les définitions d'un dictionnaire sont effectivement porteuses de sens, c'est nécessairement au moins par le réseau qu'elles établissent entre les mots qui en sont des entrées [Abdallah et al., 2003] [Awada, 2005].

D'autre part, la plupart des travaux sur les dictionnaires portent sur le rapport de synonymie. Il s'agit, très souvent, de détecter des composantes possédant des propriétés spécifiques en termes de graphe telles que les cliques [Ploux et Victorri, 1998] et les gangs [Venant, 2003] conduisant ainsi au regroupement de synonymes, l'ensemble des éléments appartenant à une même composante correspondant à un « sens élémentaire ». Dans une étude antérieure, [Awada, 2005] introduit la notion de « synonymétrie » pour quantifier la force de la synonymie entre deux mots. Cette étude avait pour but de détecter les composantes de sens dans un dictionnaire de verbes en se basant sur la N-connexité comme critère de regroupement et de classification de synonymes [Awada et Chebaro, 2004]. Toutefois, les différentes approches proposées souffrent de l'ambiguïté liée aux langues naturelles. En effet, cette ambiguïté se manifeste dans les dictionnaires par la présence d'entrées polysémiques confondues dans le graphe en un seul noeud. Ce problème provient en général d'utilisations de synonymes métaphoriques, la métaphorymie étant une notion proposée par [Duvignau et al., 2000] et [Gaume et al., 2002].

Nous présentons dans cette section une étude de composantes de sens à travers l'examen d'un réseau d'informations en essayant de traiter précisément le problème de la polysémie et d'y présenter quelques éléments concrets de solution. Nous définissons aussi un critère de regroupement basé sur la notion de circuit. Toutefois, ceci n'est cependant pas l'objectif principal de la thèse, mais c'est une étape préalable à nos travaux. En fait, ceci va s'avérer ensuite extrêmement utile vue que les travaux de [Gaume et al., 2004] n'ont pas apporter une solution optimale aux requêtes sur le Web qui soit flexible et peu ambiguë.

Par ailleurs, le problème de la classification des documents (clustering) est l'un des axes de recherche scientifique les plus importants dans le domaine de l'informatique documentaire. Plusieurs approches ont été proposées par la communauté scientifique qui a suggéré différentes methodes s'appuyant très souvent sur les techniques de Data Mining [Berry et Linof, 1997]. Notre approche de la classification d'entités documentaires consiste généralement à représenter ces entités (les pages Web ou les articles associés aux entrées d'un dictionnaire) par un graphe RPMH dont les sommets sont les entités et les arcs traduisent un lien (hypertextuel dans le cas de pages Web ou définitionnel dans le cas de mots d'un dictionnaire) direct entre deux sommets : il existe un arc d'un sommet *A* vers un sommet *B* si et seulement si l'entité *B* possède un lien avec l'entité *A*. Par conséquent, le problème de

[21] L'**hyperonymie** est la relation sémantique hiérarchique d'un lexème à un autre selon laquelle l'extension du premier terme, plus général, englobe l'extension du second, plus spécifique. Le premier terme est dit hyperonyme de l'autre, ou superordonné par rapport à l'autre. C'est le contraire de l'hyponymie.

classification (des pages Web ou des mots du dictionnaire) se ramène à une étude sur les graphes cherchant à exploiter les réseaux ainsi établis entre les entités. Il s'agit, très souvent, de détecter des composantes possédant des propriétés spécifiques des graphes telles que : présence de cliques ou de composantes N-connexes [Awada et Chebaro, 2004] conduisant ainsi au regroupement des entités.

2.2.1 Présentation de l'approche

Les deux sources de données alimentant les deux RPMH proposés dans notre modèle conceptuel sont deux fichiers au format XML dans lesquels les entités sont décrites par un ensemble de balises permettant chacune d'associer un lien (hypertextuel ou définitionnel) aux différents constituants (voir tableau 4.2).

La base de données des liens hypertextuels entre les pages Web	**La base de données des liens définitionnels entre les mots du dictionnaire**
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE site SYSTEM "F1.dtd"> <Site> <Page url = "URL($page_1$)"> <link>URL($page_{11}$)</link> <link>URL($page_{12}$)</link> <link>URL($page_{13}$)</link> …… <link>URL($page_{1n}$)</link> </Page> <Page url = "URL($page_{11}$)"> <link>URL($page_{111}$)</link> <link>URL($page_{112}$)</link> <link>URL($page_1$)</link> …… <link>URL($page_{11p}$)</link> </Page> </Site>	<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE site SYSTEM "F1.dtd"> <Dictionnaire> <mot m = "mot_1"> <def>mot_{11}</def> <def>mot_{12}</def> <def>mot_{13}</def> …… <def>mot_{1n}</def> </mot> <mot m = " mot_{11}"> <def>mot_{111}</def> <def>mot_{112}</def> <def>mot_1</def> …… <def>mot_{11p}</def> </mot> </Dictionnaire>

Tableau 4.2 : Les sources de données de deux RPMH

Ces bases de données n'étant pas utilisables sous leur forme XML, nous avons entrepris une transformation de ces deux fichiers en deux graphes (RPMH) afin de pouvoir représenter graphiquement les entités et leurs liens. Les nœuds sont alors considérés comme des entités liées par des arcs représentant la relation de lien entre elles.

La structure du réseau en tant que graphe se caractérise par une concentration de relations (arcs) entre toutes les entités (sommets) ayant le même « sens ». Ces entités entretiennent des relations qui en font parfois des circuits. En fait, deux familles d'entités ayant chacune un sens propre différent de celui de l'autre famille vont se traduire sur le graphe par deux ensembles de circuits disjoints. Nous en conclurons qu'il devrait y avoir équivalence entre la notion de sens et d'ensemble de circuits dans le graphe. Cette hypothèse devrait être vraie même en présence d'entités polysémiques[22] dans le réseau. En effet, en partant d'une entité

[22] ***Monosémie :*** une entité est dite monosémique si toutes les entités auxquelles elle est liée sont liées entre elles (l'entité appartient à une seule clique).
Homonymie : une entité est dite homonymique si l'ensemble des entités auxquelles elle est liée (autres qu'elle-même) est séparable en au moins deux sous-ensembles disjoints pour ce lien, c'est-à-dire que chacun des éléments de l'un des sous-ensembles n'est lié à aucun des éléments des autres sous-ensembles.
Polysémie : une entité est dite polysémique si elle n'est ni monosémique ni homonymique, c'est-à-dire si elle admet : (i) des entités auxquelles elle est liée qui ne sont pas toutes liées entre elles ; (ii) des entités non séparables: reliées entre elles par une chaîne d'entités, liées à l'entité considérée et différentes de cette entité.

"*E*" donnée à la recherche d'un circuit, l'existence d'une entité polysémique fait en sorte qu'il y a peu de chance que l'on revienne à l'entité de départ "*E*", et par suite l'entité polysémique est évidement éliminée.

L'idée consiste à regrouper deux entités E_2 et E_3 d'une entité donnée E_1 en un élément de sens S_1 de cette entité s'il existe au moins un certain nombre de circuits partant de E_1 et y aboutissant, passant par E_2 et E_3 en même temps.

Nous définissons la proximité sémantique entre deux entités E_1 et E_2 en terme du nombre de circuits passant par E_1 et E_2 de la manière suivante [Elayeb et al., 2008] :

$$Proximité\_Sémantique\ (E_1, E_2) = Nombre\ de\ circuits\ (E_1, E_2)/Nombre\ maximum\ de\ circuits\ détectés$$

Il est nécessaire de déterminer le nombre de circuits passant par chaque entité pour pouvoir évaluer les proximités entre les différentes entités formant les sommets du réseau d'informations. Ce nombre de circuits est utile pour la définition du paramètre appelé « le seuil d'acceptation ».

2.2.1.1 Choix du seuil d'acceptation

Le seuil d'acceptation joue le rôle du filtre qui empêchera de regrouper certaines entités sémantiquement proches d'une entité donnée dans une même composante de « sens », et permettra donc, par opposition, d'en regrouper d'autres. Une valeur faible de ce seuil ferait entrer dans la même composante de « sens » des entités qui ont peu ou pas assez de relations entre elles en tant qu'entités similaires à celle de départ car peu de circuits les réunissent. Alors qu'une valeur importante de ce seuil aurait pour effet d'empêcher le regroupement d'entités pouvant correspondre à une même signification, voire d'éliminer carrément certaines entités, qui seraient ainsi à tord considérées comme des entités similaires non acceptables de celle de départ.

Nous étudions dans la suite les effets de la variation du seuil d'acceptation et son influence sur la formulation de composantes de « sens ». Considérons l'exemple d'un réseau d'entités de la figure 4.4.

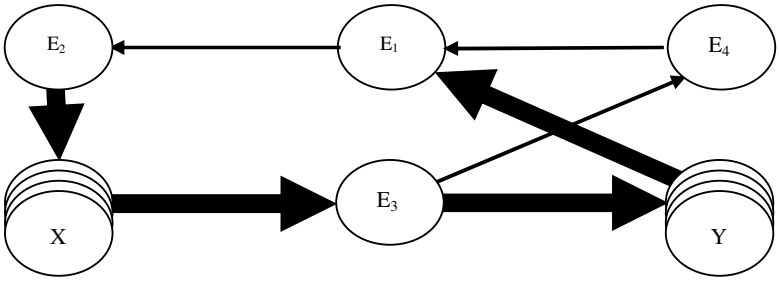


Figure 4.4 : Exemple du choix de seuil d'acceptation

Nous supposons que l'entité de départ est E_1 et que le nombre de circuits contenant à la fois E_1, E_2 et E_3 est N_1. Nous avons symbolisé les entités reliant E_2 à E_3 par l'entité *X*. Le nombre de circuits N_1 est obtenu en additionnant le nombre de circuits N_2 passant par l'entité E_4 d'un côté, et le nombre de circuits N_3 passant par *Y* (chacun des deux symboles *X* et *Y* représente plusieurs autres entités liées).

Supposons que N_1 soit supérieur au seuil d'acceptation. Ceci implique que E_2 et E_3 font partie du même « sens ». Concernant E_4, deux cas sont à prendre en considération :

- N_2 est supérieur au seuil d'acceptation, alors E_4 fait partie du même « sens » que E_2 et E_3.
- N_2 est inférieur au seuil d'acceptation, alors E_4 ne fait pas partie du « sens » précédemment évoqué. Deux cas peuvent alors encore se présenter :
    - E_4 figurera dans une autre composante (qui ne figure pas sur le schéma).
    - E_4 ne figurera dans aucune autre composante et donc ne sera plus considérée comme sémantiquement proche de E_1.

Il s'est avéré qu'il n'est pas du tout évident de choisir le seuil optimal d'acceptation. C'est pour cette raison que nous avons minimisé son rôle en le combinant à un autre facteur qui est la longueur du circuit.

Par ailleurs, le seuil d'acceptation est calculé automatiquement à partir de la moyenne du nombre des circuits dans la matrice des circuits communs (cf. section 2.2.1.3). Cette solution ne nécessite aucune expertise du côté de l'utilisateur et peut donc être exploitée automatiquement.

2.2.1.2 Importance de la longueur du circuit

La richesse d'un réseau d'informations se traduit par la complexité des liens entre entités qui le composent. En effet, la distance (en nombre d'arêtes) qui sépare ces entités, et donc la longueur du circuit les reliant est l'un des facteurs importants qui assure l'existence d'une proximité significative entre deux entités du réseau. De plus, il existe une inter-connectivité accrue entre les nœuds du graphe associé à une entité possédant un très grand nombre de liens. Par exemple dans le domaine lexical, plus il y a de sens associés à un mot, plus on trouve d'arcs connectés aux sommets le représentant dans le graphe.

D'autre part, il se peut qu'un circuit partant et aboutissant à une entité E_1 soit constitué de deux chemins : l'un partant de E_1 à E_2 et désignant un sens S_1, et l'autre partant de E_2 vers E_1 et désignant un autre sens S_2. Il s'agit en fait du problème de la « polysémie » des entités. Il s'est avéré que plusieurs anomalies peuvent être détectées lors du regroupement des entités dans les composantes de sens. Ces erreurs sont causées principalement par l'existence d'entités polysémiques dans un ou plusieurs circuits.

Vu la difficulté de résoudre définitivement le problème de la polysémie liée aux entités à caractère documentaire, nous proposons une contribution qui consiste à minimiser les effets néfastes à la compréhension en diminuant la longueur des circuits à traiter, et donc en restreignant le nombre d'entités y figurant. En effet, la prise en compte de circuits trop courts uniquement aurait pour effet de scinder une même composante de sens en plusieurs. Cependant, plus le circuit est long, plus il y a de chance d'y trouver des entités polysémiques et par conséquent de mélanger différentes composantes de sens.

Ainsi, le principe de regroupement que nous proposons est le suivant :

On regroupe deux entités E_2 et E_3 liées à une entité donnée E_1 en une composante de sens S_1 de cette entité s'il existe au moins un certain nombre de circuits de longueur inférieure ou égale à une longueur donnée partant de E_1 et y revenant, passant par E_2 et E_3 en même temps.

Nous appellerons la longueur maximale précédemment évoquée « la longueur limite ». Nous précisons que la longueur des circuits que nous avons pris en compte est de l'ordre de 4 arcs (dans les deux cas : mots du dictionnaire ou pages Web). Nous avons atteint ce chiffre après bien des tests sur la validité des résultats obtenus en fonction de la longueur des circuits étudiés (voir annexes 3 et 4). En effet, dans le cas du dictionnaire, l'ordre de mots sémantiquement proches récupérés pour un mot donné se stabilise à partir d'une longueur de circuit égale à 4. A partir d'une longueur de circuits égale à 5, le nombre de circuits récupérés

pour chaque sémantiquement mot proche devient très important. En conséquence, dans ce cas plusieurs mots ne font pas partie de la composante de sens du mot de départ.

2.2.1.3 Construction des classes de sens

Nous avons étudié trois méthodes permettant de grouper les entités en classes de sens (entités liées entre elles et partageant un même sens). Ces approches utilisent une matrice, dite des circuits communs, construite à partir de statistiques sur les circuits dans le graphe.

Par ailleurs, cette matrice constitue le matériel de base sur lequel s'effectuent les traitements permettant de regrouper les entités correspondant à une même composante de sens d'une entité donnée. Nous nous n'intéressons qu'aux circuits ayant une longueur inférieure ou égale à la longueur limite décrite dans la section précédente. Un compromis sur la longueur limite des circuits à prendre en compte s'avère donc nécessaire car cette longueur limite influe grandement sur les résultats. En effet, une valeur élevée de cette longueur présenterait l'avantage de diminuer le nombre de composantes mais y inclurait des entités indirectes ayant des sens éloignés de l'entité initiale. Par contre, une valeur basse de cette longueur permettrait d'éliminer les entités indirectes mais donnerait un grand nombre de composantes de sens vue qu'un sens sera associé à des petits groupes d'entités.

La matrice des circuits communs permet de générer les différentes relations existant entre les entités du réseau deux à deux. En effet, la construction de cette matrice carrée se fait de la manière suivante : Pour une entité donnée au départ, nous partons du graphe d'entités et nous parcourons la structure correspondante à la recherche de tous les circuits partant de l'entité de départ. Après avoir construit la liste des circuits, nous construisons la matrice des circuits communs dont les entrées sont les entités proches de l'entité de départ et où le contenu d'une cellule de coordonnées (E_i, E_j) correspond au nombre de circuits partant de l'entité de départ et contenant à la fois E_i et E_j.

Cette matrice est utile dans l'extraction des couples d'entités ayant une relation significative en comparant le contenu de chaque cellule avec le seuil d'acceptation représentant le nombre moyen de circuits figurant dans la matrice.

La génération des groupes d'acceptations potentiels se fait entité par entité. En effet, nous commencons par construire des groupes contenant chacun deux éléments, puis on réitère comme suit:

Une relation *R* existe entre deux entités E_i et E_j si la valeur correspondant à la ligne *i* et la colonne *j* dans la matrice est supérieure au seuil d'acceptation. Ces deux entités forment alors un couple comme le montre la figure 4.5.

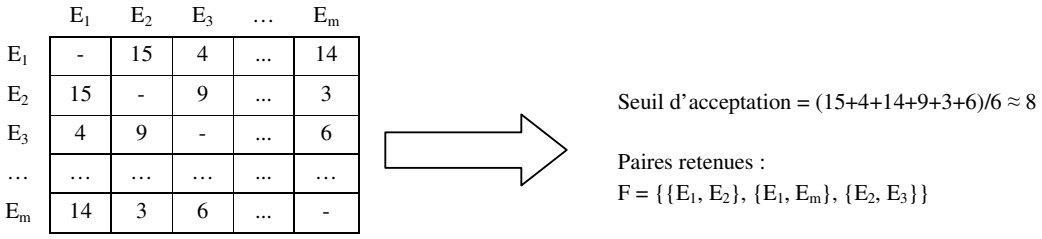


Figure 4.5 : Couples des entités issus d'une matrice des circuits communs

Une fois élaboré l'ensemble F de ces couples, nous transformons F en un ensemble de triplets en essayant d'y inclure une entité significative (correspondant au même sens que les deux entités du couple), puis en un ensemble de quadruplets, etc. Finalement, lorsque l'ensemble F

se stabilise, nous obtenons dans F les composantes potentielles de sens représentant les classes de sens finales.

Nous présentons ci-après trois méthodes de regroupement permettant d'étendre l'ensemble F. Ces méthodes ont été initiées par [Awada, 2005] dans le cas des verbes d'un dictionnaire. Nous commentons d'avantage ces trois méthodes tout en proposant un nouveau cadre générique de leur application et nous proposons des extensions vers d'autres algorithmes génériques de regroupement des composantes de sens (cf. section 2.2.2).

Ces trois méthodes utilisent les conventions suivantes :

F : l'ensemble de couples d'entités obtenus à partir de la matrice des circuits communs.
G_j : le $j^{ème}$ groupe de F.
n : le nombre des groupes de F (cardinalité de F), donc F = {G_1, G_2, …, G_n}.
E_k : le $k^{ème}$ entité obtenue à partir de celle du départ.
m : le nombre d'entités dans la matrice des circuits communs.

(i) Première méthode : Regroupement par allongement de circuits

Considérons le groupe d'entités $\{E_i, E_{i+1}, \ldots, E_j\}$, où j > i, on inclut l'entité E_k, $k \notin [i, j]$, dans ce groupe si et seulement si E_k entretient une relation significative avec tous les éléments de ce groupe simultanément. Ceci se traduit par le fait que le nombre de circuits qui contiennent E_k et tous les éléments de ce groupe sont supérieurs au seuil d'acceptation[23]. Cet algorithme est présenté par la figure 4.6.

```
Répéter
        stabilité = Vrai; /*C'est une variable booléenne indiquant que la construction de F est non encore achevée*/
        Pour j allant de 1 à n Faire
                Pour k allant de 1 à m Faire
                        Si (Ek ∉ Gj) Alors
                                Si le nombre de circuits contenant Ek &
                                        tous les éléments de Gj sont > seuil d'acceptation Alors
                                            {Gj ← Gj ∪ {Ek}; stabilité = Faux ;}
                                FinSi
                        FinSi
                FinPour /*fin pour k*/
         FinPour /*fin pour j*/
Jusqu'à (stabilité == Faux);
```

Figure 4.6 : Algorithme de regroupement par allongement de circuits

Le but étant d'obtenir des composantes de sens grâce à la condition de regroupement consistant à inclure une entité dans une composante uniquement si le nombre de circuits réunissant cette entité à toutes celles de la composante est supérieur au seuil d'acceptation. En effet, cette méthode est trop contraignante car si relier E à $G = \{E_1, \ldots, E_m\}$ se traduisait graphiquement par une arête entre E et chaque E_i de G (pour i = 1,…,m) dans un graphe qu'on appellera H(G), mais comme la dernière entité entrée dans G (supposons que ce soit E_m) y est entrée pour la même raison et est donc reliée à chaque E_j de G (pour j = 1, m-1), et ainsi de suite pour tous les précédents alors la représentation graphique de H(G) est une clique sur G (tout E_i est relié par une arête à tout E_j).

La condition est donc forte puisque n'entre dans H (de cardinal n) qu'une entité E à condition que $H \cup \{E\}$ reste une clique et que le nombre de circuits, utilisant tous les éléments de

[23] Si R(x, Y) signifie que x entretient une relation significative avec tous les éléments de Y, alors $R(S_k, \{S_i, S_{i+1}, \ldots, S_j\}) \Leftrightarrow$ Nbre_circuits(S_k, $\{S_i, S_{i+1}, \ldots, S_j\}$) > seuil.

$H_n \cup \{E\}$ où H_n est un sous-ensemble à n éléments de H, soit supérieur à un seuil. Si $Nbre(H_n \cup \{E\})$ représente ce nombre de circuits et s représente le seuil, la condition est $Nbre(H_n \cup \{E\}) \geq s$.

(ii) Deuxième méthode : Regroupement par associations séparées

Considérons le groupe d'entités $\{E_i, E_{i+1}, \ldots, E_j\}$ augmenté d'une entité E_k si et seulement si E_k entretient une relation significative[24] avec chaque élément de ce groupe séparément. Ceci se traduit dans la matrice par des valeurs de (E_i, E_k), (E_{i+1}, E_k),…, (E_j, E_k) toutes supérieures au seuil d'acceptation. Cet algorithme est présenté par la figure 4.7.

Bien que cette deuxième méthode soit plus souple que la première, elle est encore contraignante car elle consiste encore à préserver la nature de clique de H(G) mais cette fois avec la condition que le nombre de circuits, utilisant tous les éléments de $H_1 \cup \{E\}$ où H_n est un sous-ensemble à n éléments de H(G), soit supérieur à un seuil. Soit : $\forall H_1 \subseteq H(G)$, $Nbre(H_1 \cup \{E\}) \geq s$.

```
Répéter
        stabilité = Vrai;
        Pour j allant de 1 à n Faire
                Pour k allant de 1 à m Faire
                        Si (Ek ∉ Gj) Alors
                                S'il existe une relation R entre Ek & chaque élément de Gj Alors
                                    {Gj ← Gj ∪{Ek}; stabilité = Faux ;}
                                FinSi
                        FinSi
                FinPour /*fin pour k*/
        FinPour /*fin pour j*/
Jusqu'à (stabilité == Faux);
```

Figure 4.7 : Algorithme de regroupement par associations séparées

Il est à signalé que nous avons proposé des méthodes intermédiaires entre la première et la deuxième méthode. Il suffisait encore une fois de préserver la nature de clique de H(G) avec la condition (paramétrable sur k) suivante : le nombre de circuits, utilisant tous les éléments de $H_k \cup \{E\}$ où $1 \leq k \leq n$ pour tous les H_k, est supérieur à un seuil. Soit : $\forall H_k \subseteq H(G)$, $Nbre(H_k \cup \{E\}) \geq s$.

(iii) Troisième méthode : Regroupement par contrainte minimale

Considérons le groupe de *n* entités $\{E_i, E_{i+1}, \ldots, E_j\}$, on augmente ce groupe d'entités E_k si et seulement s'il existe un circuit de longueur n+1 (où n = Card(G)) contenant les éléments du groupe et E_k. Cet algorithme est présenté par la figure 4.8.

Chaque élément de F doit correspondre à un groupe d'entités ayant un « sens » spécifique. Toutefois, cette solution présente un certain nombre de lacunes. En effet, le sens de parcours des entités influe sur le résultat obtenu. Prenons l'exemple d'un groupe $G = \{E_1, E_2, \ldots, E_p\}$ et deux candidats E_k et E_l tels qu'il existe un circuit de longueur p+1 contenant $E_1, E_2, \ldots, E_p$ et E_k et un autre circuit de longueur p+1 (où p = Card(G)) contenant $E_1, E_2, \ldots, E_p$ et E_l mais il n'existe pas de circuit de longueur p+2 contenant $E_1, E_2, \ldots, E_p, E_k$ et E_l. L'entité à inclure dans le groupe G est le premier examiné, l'autre n'y entrera jamais. Ceci fait que le nombre de

[24] Si R(x, y) signifie que x entretient une relation significative avec y, alors $\forall p \in [i, j]$ $R(S_k, S_p)$; c-à-d : $R(S_k, S_p) \Leftrightarrow Nbre\_circuits(S_k, S_p) > seuil$.

groupes obtenus reste supérieur au nombre d'acceptions possibles. Nous avons ainsi prévu une étape de fusion consistant à réunir les groupes correspondant au même « sens » à l'intérieur de la même composante de « sens ». Nous tenons quand même à signaler que nous avons adopté cette dernière méthode qui nous semble meilleure que les deux premières du point de vue des résultats obtenus.

```
Répéter
        stabilité = Vrai;
        Pour j allant de 1 à n Faire
                Pour k allant de 1 à m Faire
                        Si (Ek ∉ Gj) Alors
                            S'il existe un circuit qui contient seulement Ek &
                                tous les éléments de Gj Alors
                                {Gj ← Gj ∪ {Ek}; stabilité = Faux ;}
                            FinSi
                        FinSi
                FinPour /*fin pour k*/
         FinPour /*fin pour j*/
Jusqu'à (stabilité == Faux);
```

Figure 4.8 : Algorithme de regroupement par contrainte minimale

En fait, cette méthode est beaucoup moins contraignante que les deux autres car elle consiste encore à préserver la nature de clique de H(G) avec la condition (plus faible) suivante :

Le nombre de circuits, utilisant tous les éléments de $H_n \cup \{E\}$ est supérieur à 1. Soit $Nbre(H_n \cup \{E\}) \geq 1$. Si l'on résume cela par le tableau 4.3 :

	H(G) = clique			H(G) ≠ clique
x = Nbre d'éléments pris dans G Seuil ≥ y	x =1	x = k, 1 < k < n	x = n	?
y = 1	?	Méthodes intermédiaires	?	
y = 2	Méthode 2	?	Méthode 1	

Tableau 4.3 : Récapitulation de méthodes de regroupement des entités

Le symbole (?) dans le tableau 4.3 montre qu'il y a encore d'autres méthodes à tester. En fait, nous avons complètement occulté le cas H(G) non clique. En effet si l'on reprend la toute première définition en la modifiant comme suit :

Etant donné un groupe d'entités $\{E_i, E_{i+1}, \ldots, E_j\}$ de cardinalité (j-i+1). Nous augmentons cet ensemble d'une entité E_k si et seulement si E_k entretient une relation suffisamment significative avec chaque élément de ce groupe. Ceci se traduit par le fait que le nombre de circuits, qui contiennent E_k et suffisamment d'éléments de ce groupe, est supérieur au seuil d'acceptation.

Le mot « suffisamment » pourrait signifier par exemple qu'il passe un nombre de circuits (supérieur au seuil y) avec un nombre suffisamment grand d'éléments de G (≥ 90%*card(G), par exemple), mais pas tous les éléments de G que ce soit séparément (i.e. : pour n'importe quel groupe d'un élément) ou simultanément (i.e. : pour n'importe quel groupe de *n* éléments) ou partiellement (i.e. : pour n'importe quel groupe de *k* éléments). Il se pourrait alors qu'une entité (ou peut-être plusieurs) ne fasse jamais partie d'aucun circuit contenant E_k, auquel cas il ne serait pas relié dans H(G) à E_k et H(G) ne serait plus une clique.

Ensuite nous pouvons faire de nouveau varier x et y dans le cas H(G) non clique. Quant à l'influence de l'ordre d'entrée des entités dans G, elle était déjà présente pour la méthode 1 comme pour la méthode 2. Dans l'exemple cité, si E_k entre dans G il faudrait que E_l entre aussi (bien que l'entité E_k ne soit pas reliée à E_l puisque aucun circuit ne les contient). Cela milite pour une méthode qui s'applique quand H(G) est non clique.

2.2.1.4 Fusion des groupes potentiels en composantes de sens

L'étape de regroupement produit un ensemble F de groupes G_i contenant chacun des entités ayant le même sens. Cependant, il se peut que deux groupes puissent correspondre à un même sens. Ceci découle, entre autres choses, du problème évoqué dans le paragraphe précédent. Une fusion de ces deux groupes est nécessaire pour obtenir une unique composante de sens. Le principe de fusion des groupes potentiels en composantes de sens est le suivant :

Deux groupes G_i et G_j ($card(G_i) = n_i$; $card(G_j) = n_j$ avec $n_j \leq n_i$) doivent être fusionnés si :

1. G_i contient $(n_j - 1)$ mots de G_j.
2. Il existe un arc entre les entités E_1 et E_2 tels que $E_1 \in G_j - G_i$ et $E_2 \in G_i - G_j$.

En effet, G_i contient $(n_i - n_j)$ entités qui ne sont pas dans G_j. Soit Q l'ensemble de ces entités et $q = card$(Q). Nous avons envisagé l'étude de différentes possibilités de relation entre E_1 ($E_1 \in G_j$ et $E_1 \notin G_i$) et un certain nombre d'éléments de Q. Nous avons constaté qu'imposer à E_1 d'avoir une relation (arc) avec chaque élément de Q ne permet pas de réduire les sens intermédiaires de façon significative et laisserait des groupes non fusionnés ayant des sens proches. Après une étude approfondie du problème, [Awada, 2005] a abouti à la conclusion suivante : pour inclure E_1 dans G_i, il suffit qu'il y ait un arc entre E_1 et un des éléments de Q. La figure 4.9 présente cet algorithme.

```
Répéter
        arrêt = Vrai;
        Pour i allant de 1 à n Faire
                Pour j allant de 1 à n Faire
                    Si (Gi ≠ Gj) Alors
                        ni = card(Gi); nj = card(Gj);
                            Si (nj > ni) Alors
                               échanger Gi et Gj ; /* Gi plus petit que Gj */
                                   Si (card (Gi ∩ Gj) >= nj – 1) Alors
                                       E1 = Gi \ (Gi ∩ Gj);
                                       G = Gi \ (Gi ∩ Gj);
                                         S'il existe une relation R  entre E1 & un élément de G Alors
                                            Gi ← Gi ∪ Gj;
                                            arrêt = Faux ;
                                           Supprimer Gj; /*Fusionner Gi et Gj */
                                         FinSi
                                   FinSi
                            FinSi
                    FinSi
                FinPour /*fin Pour j*/
        FinPour /*fin Pour i*/
Jusqu'à (arrêt == Faux) ;
```

Figure 4.9 : Algorithme de fusion des groupes potentiels en composantes de sens

Nous remarquons ici que la fusion de groupes ressemble à l'agrégation des entités en une non-clique. Dans l'exemple cité précédemment on aurait pu obtenir : $G_1 = \{E_1, \ldots, E_m, E_k\}$ et

$G_2 = \{E_1,\ldots, E_m, E_l\}$. A l'évidence $G = G_1 \cup G_2 = \{E_1,\ldots, E_m, E_k, E_l\}$ pourrait très bien vérifier $Nbre(H(G)) \geq s$ si l'on n'impose pas $Nbre(H_n) \geq s$ pour $\forall H_n \subseteq H(G)$ et pour $\forall n \in [1, m+2]$. Si on relaxe ces deux quantificateurs, la méthode pourrait ne pas nécessiter la fusion de groupes.

2.2.2 Extension à d'autres algorithmes de classification

Nous présentons dans cette section d'autres algorithmes de classification qui semblent suceptibles de résoudre notre problématique de recherche des composantes de sens dans un RPMH d'entités. En fait, ces algorithmes traitent le cas où H est un graphe non clique (i.e. $H(G) \neq$ clique dans le tableau 4.3).

Algorithme 1 :

L'entité E intègre le groupe G si et seulement si il existe un nombre Nbre(E) de circuits, notés $C_1,\ldots, C_{Nbre(E)}$, tel que $Nbre(E) \geq s$, (s fonction de Card(G)), de longueurs $L(C_i)$ tels que $\forall i$, $L(C_i) \leq l$, (l fonction de Card(G)) passant par E et utilisant tous les entités de G.

Algorithme 2 :

L'entité E intègre le groupe G si et seulement si il existe un nombre Nbre(E) de circuits, notés $C_1,\ldots, C_{Nbre(E)}$, tel que $Nbre(E) \geq s$, (s fonction de Card(G)), de longueurs $L(C_i)$ tels que $\forall i$, $L(C_i) \leq l$, (l fonction de Card(G)) passant par E et utilisant suffisamment d'entités de G.

L'algorithme 2 assouplit un peu l'algorithme 1 trop contraignant. Par contre E peut être relié à G par un arc ou par un chemin court (empruntant des sommets non encore dans G).

Algorithme 3 :

L'entité E intègre le groupe G si et seulement si il existe un nombre Nbre(E) de circuits, notés $C_1, \ldots, C_{Nbre(E)}$, tel que $Nbre(E) \geq s$, (s fonction de Card(G)), de longueurs $L(C_i)$ tels que $\forall i$, $L(C_i) \leq l$, (l fonction de Card(G)) passant par E et utilisant un ensemble d'entités $F = \cup_{i=1,Nbre(E)} C_i$ tel que $Card(F - G) \leq n$ et $Card(F \cap G) = \lambda * Card(G)$ (où n fonction de Card(G) et $\lambda \approx 1$).

Pourquoi faut-il garder F dans l'algorithme et ne pas restreindre à G ? Tout simplement parce qu'au moment d'intégrer E_{m+1} à $G = \{E_1,..., E_m\}$, il faut se rappeler que G n'existe que grâce à F (et en particulier aux quelques éléments de F - G qui ont permis de dénombrer suffisamment de circuits de longueurs acceptables pour autoriser tous les éléments de G à se regrouper).

Si E_m est le dernier noeud à être entré dans G et cela grâce à un noeud E_k appartenant à F - G, E_k a donc permis l'existence de circuits justifiant G. Si l'on supprime E_k et que l'on cherche maintenant à intégrer E_{m+1} dans $G = \{E_1,..., E_m\}$, il est déjà probable que le nombre de circuits a diminué et que E_m ne devrait peut-être déjà plus être dans G. Il n'est pas non plus certain que l'introduction de E_m avec la disparition de E_k permette d'intégrer E_{m+1}. E_m pourrait par exemple s'avérer inutile à l'intégration de E_{m+1} (il pourrait ne faire apparaître aucun circuit contenant E_{m+1}) tandis que E_k aurait par contre été utile. Ce serait le cas par exemple si E_{m+1} était lié à E_k (donc à E_m) sans pour autant être directement lié à E_m.

F est-il donc condamné à croître sans cesse? F oui! Mais F - G non, et pour deux raisons:

- il faut que Card(F – G) reste petit ;
- il faut préférentiellement tenter d'intégrer à G les éléments de F – G (ce qui fait diminuer Card(F – G) en cas de succès).

Soit G un ensemble non vide d'entités proches entre eux. Soit E un nouveau entité que l'on cherche à intégrer à G. Soit F l'ensemble d'entités qui appartiennent aux circuits qui ont permis de regrouper les entités de G de telle sorte que F - G soit petit.

D'autre part, si les entités étaient des maisons, la structure de RPMH donnerait une répartition des maisons formant de grandes métropoles (denses, zones en marron) (voir figure 4.10) et des banlieues de villes proches mais un peu moins denses (zones orangées) et puis rapidement la campagne avec quelques villages clairsemés avec peu d'habitations (zones jaunes) et encore plus rare ensuite quelques lieux dits de quelques maisons (zones crèmes) et puis presque rien sous forme de maisons isolées (zone grise).

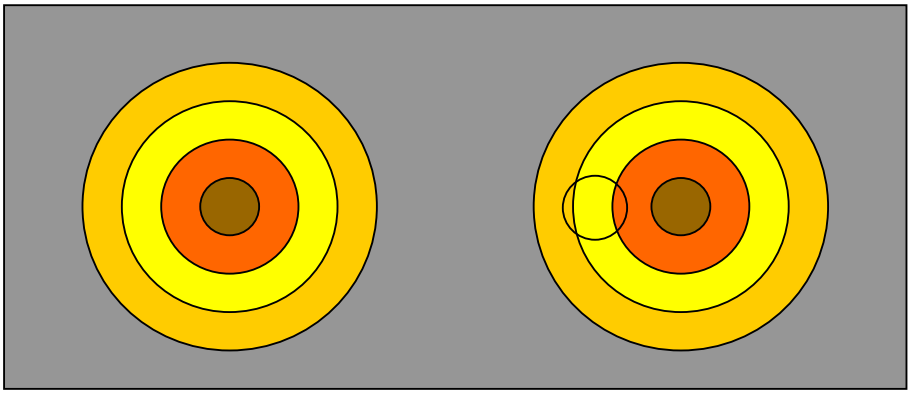


Figure 4.10 : Répartition des zones denses dans une zone urbaine

Il en va un peu de même pour les entités. En effet, les éléments de la zone marron entretiennent de nombreux circuits courts entre eux, mais certains (ceux à la périphérie de la zone marron) grâce à des éléments de la zone orangée. Ainsi de suite. Quand on est sur un élément de la zone crème, on profite des éléments de la zone jaune mais pas des éléments de la zone grise car ces derniers sont trop peu nombreux et/ou nécessitent des circuits trop longs.

En fait l'algorithme devrait encore pouvoir s'améliorer. Ici nous avons proposé Card(F- G)≤ n. Nous pourrons cependant penser qu'il ne serait pas forcement très gênant que Card(F – G) >> n. Ce qui importe est que les éléments de F - G ne soient pas "trop loin" de G. Un autre critère de restriction alternatif pourrait donc être $dist(E_k, G) < d$ où E_k, appartenant à F - G, est une entité permettant l'existence de suffisamment de circuits courts autorisant l'intégration de l'entité E_{m+1} à G. Plusieurs définitions de *dist*(E, G) sont envisageables, mais il faut encore approfondir cette notion et bien choisir *d* (noté plus loin d_G).

Peut-être même que l'éloignement de E à G n'est pas le meilleur critère possible. Sur la figure ci-dessus on peut imaginer un cercle vide (disons de la taille du cercle de la zone en marron) que l'on promènerait sur cette figure. L'idée serait alors de compter le nombre de circuits courts dans cette zone circulaire. On la déplacerait pour permettre ainsi l'agrégation des éléments à ceux dont ils sont les plus proches (cela pourrait constituer les zones de différentes couleurs de la figure ci-dessus). G résulterait ainsi de cette agrégation "en pelure d'oignon" (incluant les éléments de la zone marron jusqu'à la zone crème, mais pas au delà). En résumé: on agrège E à G s'il existe suffisamment de circuits courts passant par E et par certains éléments de G pour autant que E et ces éléments soient à une distance courte les uns des autres (correspondant à la taille maximale de la zone circulaire).

Pour déterminer au mieux cette distance *dist*(E, G), il suffit de s'imaginer le résultat final qui est une liste de groupes d'entités constituant chacun une « acception ». Dans une « acception » il y a un certain nombre d'entités sémantiquement proches qui entretiennent entre elles des liens. Il s'ensuit que, dans une « acception », la distance qui autorise les éléments à être ensemble (en plus du nombre suffisant de circuits courts entre eux) est la distance maximale qu'il y a entre deux de ces éléments. Donc à chaque étape de l'agrégation d'un nouvel élément dans G, nous calculons cette distance maximale (elle sera donc

dynamique). Nous choisissons donc de définir *dist*(E, G) et la contrainte sur celle-ci de la façon suivante:

dist(E, G) = Inf $_{E' \in G}\{d(E, E')\}$ et *dist*(E, G) $\leq d_G$ où d_G = diam(G) = $Sup_{E1 \in G, E2 \in G}\{d(E_1, E_2)\}$ avec plusieurs variantes pour le calcul de $d(E_1, E_2)$ entre deux entités.

- $d_1(E_1, E_2)$ = longueur en nombre d'arcs du plus court chemin entre E_1 et E_2 ;
- $d_2(E_1, E_2)$ = Prox[25](D, t, E_1, E_2) la probabilité en partant de E_1 d'arriver sur E_2 au terme d'un parcours de t arcs dans D. Prox(D, t, E_1, E_2) = $[\hat{D}^t]_{E1,E2}$;
- $d_3(E_1, E_2)$ = longueur du plus court chemin entre E_1 et E_2 où chaque arc $<E_i, E_j>$ est valué par Prox(D, t, E_1, E_2) ;
- $d_4(E_1, E_2)$ = distance euclidienne des entités E disposes dans IR^3 suite à une Analyse en Composante Principale (ACP) appliquée aux vecteurs $\hat{E} = ([\hat{D}^t]_{E,Ei})_{i=1, Card(D)}$

Il nous reste à fixer les trois paramètres qui ont été introduit précédemment "s", "l" et "λ". Autrement dit, nous répondons aux questions de type : Quel nombre minimal de circuits ? Quelle longueur maximale pour un circuit ? Quelle proportion d'éléments de G ?

Nous svons que si Card(G) = m, il ne peut y avoir plus de 2^m - m - 1 circuits entre les éléments de G (les arcs sont assimilés à des arêtes). Il serait donc très étonnant que E (entité à intégrer dans G) participe à autant de circuits. Il doit néanmoins y avoir au moins un circuit (se rappeler que G ne contiendra au début qu'un seul élément). Il faudrait donc prendre un nombre s tel que $1 \leq s \leq 2^m$ - m -1 (pour tout m > 1). Pourquoi pas une sorte de moyenne entre ces deux cas extrêmes: par exemple $s \approx 2^{m-1}$.

Nous svons que dans G, les éléments forment des circuits. Le plus long d'entre eux contient au plus tous les éléments de G donc est de longueur Card(G) = m. Si E doit former de nombreux circuits avec les éléments de G (ou avec des éléments extérieurs à G, néanmoins proches de G) alors il ne devra pas être loin du plus éloigné d'entre eux (qui se trouve au maximum à diam(G) + 1).

Ceci nous a permis de proposer l'algorithme suivant :

Nouvel Algorithme proposé :

L'entité E intègre les m éléments du groupe G si et seulement si il existe Nbre(E) circuits C_i passant par E tels que Nbre(E) $\geq 2^{m-1}$, $\forall$ i $\in$ I=[1, Nbre(E)], L(C_i) $\leq$ m, $\forall$ M $\in$ F=$\cup_{i=1,Nbre(E)}$ C_i, dist(M,G) $\leq$ 1+diam(G).

Appliquons ce nouvel algorithme à un petit graphe d'entités de la figure 4.11. Les résultats sont récapitulés dans le tableau 4.4.

[25] La méthode *Prox* est une méthode stochastique pour l'étude de la structure des RPMH. En fait, nous nous sommes inspirés de cette méthode, proposée par [Gaume et al., 2004] dans le cas d'un RPMH de mots d'un dictionnaire. Cette méthode consiste à transformer un graphe RPMH de entités en une chaîne de Markov dont les états sont les sommets du graphe en question et ses arêtes les transitions possibles : une particule en partant à l'instant t = 0 d'une entité e_0, se déplace en un pas sur une autre entité e_1 l'un des voisins de e_0 sélectionné aléatoirement ; la particule se déplace alors à nouveau en un pas sur e_2, l'un des voisins de e_1 sélectionné aléatoirement etc. Si au t-ième pas la particule est sur l'entité e_t elle se déplace alors en un pas sur l'entité e_{t+1} qui est sélectionné aléatoirement parmi les voisins de e_t avec des probabilités variables. Une trajectoire e_1, e_2, ..., e_t, ... ainsi sélectionnée est une «balade» aléatoire sur le graphe, et ce sont les dynamiques de ces trajectoires qui donnent des propriétés structurelles aux graphes étudiés [Gaume et Ferré, 2004] [Gaume et Mathieu, 2007].
Par définition Prox(G, i, e_r, e_s) est la probabilité qu'en partant à l'instant t = 0 d'une entité e_r la particule soit à l'instant $t = i$ sur l'entité e_s.

$G_0 = \{E_1\}$ m = 1 1+diam(G_0) = 1 E=E_2 car <E_1,E_2> existe $2^{m-1} \approx 1$ Nb(E) ≥ 1 car C_1={E_2,E_1} L(C_1) = 2 ≤ m = 1 F = {E_1, E_2} si M∈F alors dist(M, G_0) ≤ 1	$G_1 = \{E_1, E_2\}$ m = 2 1+diam(G_1) = 2 E=E5 car <E_1,E_5> existe $2^{m-1} = 2$ Nb(E) ≥ 2 car $C_1 = \{E_5, E_1\}$, $C_2 = \{E_5, E_1, E'_5\}$ L(C_1) = 2 ≤ m = 2 mais L(C_2) = 3 E_5 n'integre par G_1	$G_1 = \{E_1, E_2\}$ m = 2 1+diam(G_1) = 2 E=E_3 car <E_1,E_3> existe $2^{m-1} = 2$ Nb(E) ≥ 2 car $C_1 = \{E_3, E_1\}$, $C_2 = \{E_3, E_2\}$ L(C_1) = 2 ≤ m = 2, L(C_2) = 2 ≤ 2, F = {E_1, E_2, E_3} si M∈F alors dist(M, G_1) ≤ 1 ≤ 1+diam(G_1) = 2	$G_2 = \{E_1, E_2, E_3\}$ m = 3 1+diam(G_2) = 2 E=E_4 car <E_1,E_4> existe $2^{m-1} \approx 3$ Nb(E) ≥ 3 car $C_1 = \{E_4, E_1\}$, $C_2 = \{E_4, E_2\}$, $C_3 = \{E_4, E_3\}$ L(C_1) = 2 ≤ m = 3, L(C_2) = 2 ≤ 3, L(C_3) = 2 ≤ 3, F = {E_1, E_2, E_3, E_4} si M∈F alors dist(M, G_2) ≤ 1 ≤ 1+diam(G_2) = 2
$G_3 = \{E_1, E_2, E_3, E_4\}$ m = 4 1+diam(G_3) = 2 E=E'_5 car <E_1,E'_5> existe $2^{m-1} = 4$ Nb(E) ≥ 4 car $C_1 = \{E'_5, E_1\}$, $C_2 = \{E'_5, E_4, E_1\}$, $C_3 = \{E'_5, E_5, E_1\}$, $C_4 = \{E'_5, E''_5, E_1\}$ L(C_1) = 2 ≤ m = 4, L(C_2) = 3 ≤ 4, L(C_3) = 3 ≤ 4, L(C_4) = 3 ≤ 4. F ={E_1,E_2,E_3,E_4,E'_5,E''_5} si M∈F alors dist(M, G_3) ≤ 1 ≤ 1+diam(E_3) = 2	G_4 ={E_1, E_2, E_3, E_4, E'_5} ETC		

Tableau 4.4 : Récapitulation des résultats du nouvel algorithme

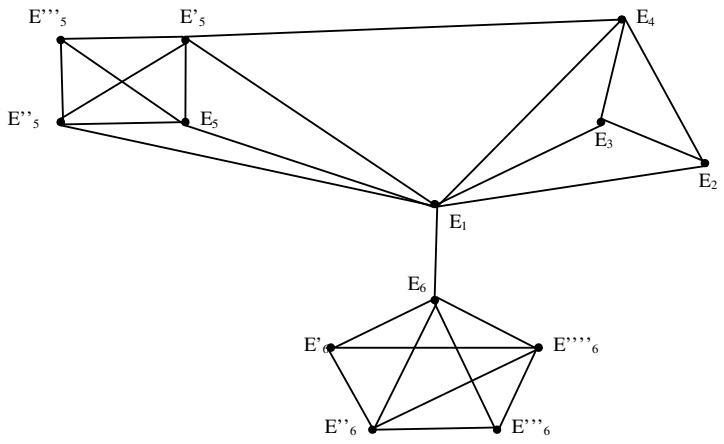


Figure 4.11 : Application du nouvel algorithme à un graphe RPMH

L'algorithme suggère qu'il n'y a pas 2 composantes pour E_1 mais une seule car {E_1, E_2, E_3, E_4, E_5, E'_5, E''_5,...} vont se regrouper. Le dessin de la figure 4.12 aurait donc dû être celui de la figure 4.11.

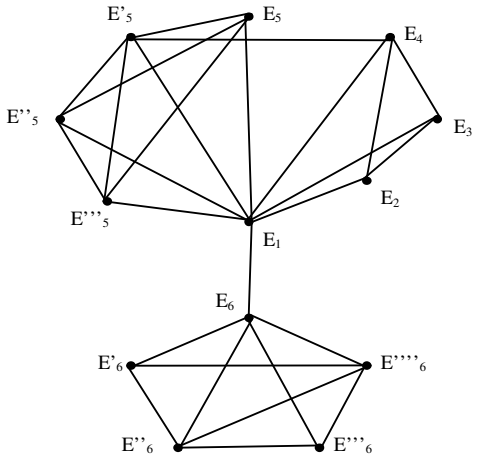


Figure 4.12 : Résultat du groupement dans le RPMH de l'exemple

Par contre, il est clair que la composante "E_1" ne sera jamais regroupée avec la composante "E_6" car il n'y a pas de cycle possible entre elles.

Soit D la matrice d'adjacence de ce graphe de 13 entités et soit DD la matrice markovienne de D, explicitées comme suit :

D = [0,1,1,1,1,1,1,1,1,0,0,0,0;
1,0,1,1,0,0,0,0,0,0,0,0,0;
1,1,0,1,0,0,0,0,0,0,0,0,0;
1,1,1,0,0,1,0,0,0,0,0,0,0;
1,0,0,0,0,1,1,1,0,0,0,0,0;
1,0,0,1,1,0,1,1,0,0,0,0,0;
1,0,0,0,1,1,0,1,0,0,0,0,0;
1,0,0,0,1,1,1,0,0,0,0,0,0;
1,0,0,0,0,0,0,0,0,1,1,1,1;
0,0,0,0,0,0,0,0,1,0,1,1,1;
0,0,0,0,0,0,0,0,1,1,0,1,0;
0,0,0,0,0,0,0,0,1,1,1,0,1;
0,0,0,0,0,0,0,0,1,1,0,1,0]

DD = [0,1/8,1/8,1/8,1/8,1/8,1/8,1/8,1/8,0,0,0,0;
1/3,0,1/3,1/3,0,0,0,0,0,0,0,0,0;
1/3,1/3,0,1/3,0,0,0,0,0,0,0,0,0;
1/4,1/4,1/4,0,0,1/4,0,0,0,0,0,0,0;
1/4,0,0,0,0,1/4,1/4,1/4,0,0,0,0,0;
1/5,0,0,1/5,1/5,0,1/5,1/5,0,0,0,0,0;
1/4,0,0,0,1/4,1/4,0,1/4,0,0,0,0,0;
1/4,0,0,0,1/4,1/4,1/4,0,0,0,0,0,0;
1/5,0,0,0,0,0,0,0,0,1/5,1/5,1/5,1/5;
0,0,0,0,0,0,0,0,1/4,0,1/4,1/4,1/4;
0,0,0,0,0,0,0,0,1/3,1/3,0,1/3,0;
0,0,0,0,0,0,0,0,1/4,1/4,1/4,0,1/4;
0,0,0,0,0,0,0,0,1/3,1/3,0,1/3,0]

Nous calculons DD^7 (car en 7 arcs/arêtes nous avons le temps de parcourir tous les noeuds de la composante "E_1" au moins une fois, et un peu plus d'une fois pour ceux de la composante "E_6").

DD^7 =

! 0.1785334 0.0711515 0.0711515 0.0950763 0.0946581 0.1187209 0.0946581 0.0946581 0.0563950 0.0358364 0.0266622 0.0358364 0.0266622 !
! 0.1897373 0.0808388 0.0812960 0.1070578 0.0957972 0.1223823 0.0957972 0.0957972 0.0450367 0.0242291 0.0189006 0.0242291 0.0189006 !
! 0.1897373 0.0812960 0.0808388 0.1070578 0.0957972 0.1223823 0.0957972 0.0957972 0.0450367 0.0242291 0.0189006 0.0242291 0.0189006 !
! 0.1901527 0.0802933 0.0802933 0.1045816 0.0978215 0.1258854 0.0978215 0.0978215 0.0438645 0.0229295 0.0178028 0.0229295 0.0178028 !
! 0.1893161 0.0718479 0.0718479 0.0978215 0.1062312 0.1314688 0.1062923 0.1062923 0.0421404 0.0215447 0.0168261 0.0215447 0.0168261 !
! 0.1899535 0.0734294 0.0734294 0.1007083 0.1051750 0.1296367 0.1051750 0.1051750 0.0419412 0.0212009 0.0164873 0.0212009 0.0164873 !
! 0.1893161 0.0718479 0.0718479 0.0978215 0.1062923 0.1314688 0.1062312 0.1062923 0.0421404 0.0215447 0.0168261 0.0215447 0.0168261 !
! 0.1893161 0.0718479 0.0718479 0.0978215 0.1062923 0.1314688 0.1062923 0.1062312 0.0421404 0.0215447 0.0168261 0.0215447 0.0168261 !
! 0.0902320 0.0270220 0.0270220 0.0350916 0.0337123 0.0419412 0.0337123 0.0337123 0.1617770 0.1468807 0.1110079 0.1468807 0.1110079 !
! 0.0716727 0.0181718 0.0181718 0.0229295 0.0215447 0.0265011 0.0215447 0.0215447 0.1836008 0.1692541 0.1278745 0.1693151 0.1278745 !
! 0.0710993 0.0189006 0.0189006 0.0237371 0.0224348 0.0274789 0.0224348 0.0224348 0.1850132 0.1704993 0.1232837 0.1704993 0.1232837 !
! 0.0716727 0.0181718 0.0181718 0.0229295 0.0215447 0.0265011 0.0215447 0.0215447 0.1836008 0.1693151 0.1278745 0.1692541 0.1278745 !
! 0.0710993 0.0189006 0.0189006 0.0237371 0.0224348 0.0274789 0.0224348 0.0224348 0.1850132 0.1704993 0.1232837 0.1704993 0.1232837 !

Nous remarquons très bien les deux composantes (gris pour "E_1" et jaune pour "E_6"). Autrement dit il n'est pas vraiment nécessaire d'utiliser les algorithmes basés sur la recherche de circuits car la matrice DD^k permet d'effectuer les regroupements recherchés.

DD^{35} =

```
! 0.1527434 0.0577788 0.0577788 0.0771212 0.0772165 0.0965451 0.0772165 0.0772165 0.0871037 0.0683319 0.0513078 0.0683319 0.0513078 !
! 0.1540767 0.0584230 0.0584230 0.0780054 0.0781284 0.0781284 0.0781284 0.0957972 0.0855110 0.0666657 0.0500752 0.0666657 0.0500752 !
! 0.1540767 0.0584230 0.0584230 0.0780054 0.0781284 0.0976920 0.0781284 0.0781284 0.0855110 0.0666657 0.0500752 0.0666657 0.0500752 !
! 0.1542424 0.0585040 0.0585040 0.0781152 0.0782416 0.0978345 0.0782416 0.0782416 0.0853131 0.0664587 0.0499221 0.0664587 0.0499221 !
! 0.1544331 0.0585063 0.0585063 0.0782416 0.0783721 0.0979985 0.0783721 0.0783721 0.0850854 0.0662205 0.0497459 0.0662205 0.0497459 !
! 0.1544722 0.0586152 0.0586152 0.0782676 0.0783988 0.0980322 0.0783988 0.0783988 0.0850386 0.0661716 0.0497097 0.0661716 0.0497097 !
! 0.1544331 0.0585963 0.0585963 0.0782416 0.0783721 0.0979985 0.0783721 0.0783721 0.0850854 0.0662205 0.0497459 0.0662205 0.0497459 !
! 0.1544331 0.0585963 0.0585963 0.0782416 0.0783721 0.0979985 0.0783721 0.0783721 0.0850854 0.0662205 0.0497459 0.0662205 0.0497459 !
! 0.1393659 0.0513066 0.0513066 0.0682505 0.0680683 0.0850386 0.0680683 0.0680683 0.1030828 0.0850483 0.0636737 0.0850483 0.0636737 !
! 0.1366638 0.0499993 0.0499993 0.0664587 0.0662205 0.0827145 0.0662205 0.0662205 0.1063104 0.0884248 0.0661715 0.0884248 0.0661715 !
! 0.1368208 0.0500752 0.0500752 0.0665628 0.0663278 0.0828495 0.0663278 0.0663278 0.1061228 0.0882286 0.0660264 0.0882286 0.0660264 !
! 0.1366638 0.0499993 0.0499993 0.0664587 0.0662205 0.0827145 0.0662205 0.0662205 0.1063104 0.0884248 0.0661715 0.0884248 0.0661715 !
! 0.1368208 0.0500752 0.0500752 0.0665628 0.0663278 0.0828495 0.0663278 0.0663278 0.1061228 0.0882286 0.0660264 0.0882286 0.0660264 !
```

DD^{100} =

```
! 0.1482038 0.0555825 0.0555825 0.0741110 0.0741121 0.0926404 0.0741121 0.0741121 0.0925262 0.0740046 0.0555041 0.0740046 0.0555041 !
! 0.1482199 0.0555903 0.0555903 0.0741217 0.0741231 0.0926543 0.0741231 0.0741231 0.0925069 0.0739844 0.0554892 0.0739844 0.0554892 !
! 0.1482199 0.0555903 0.0555903 0.0741217 0.0741231 0.0926543 0.0741231 0.0741231 0.0925069 0.0739844 0.0554892 0.0739844 0.0554892 !
! 0.1482219 0.0555912 0.0555912 0.0741230 0.0741245 0.0926560 0.0741245 0.0741245 0.0925045 0.0739819 0.0554874 0.0739819 0.0554874 !
! 0.1482242 0.0555924 0.0555924 0.0741245 0.0741261 0.0926580 0.0741261 0.0741261 0.0925017 0.0739790 0.0554852 0.0739790 0.0554852 !
! 0.1482247 0.0555926 0.0555926 0.0741248 0.0741264 0.0926584 0.0741264 0.0741264 0.0925012 0.0739784 0.0554848 0.0739784 0.0554848 !
! 0.1482242 0.0555924 0.0555924 0.0741245 0.0741261 0.0926580 0.0741261 0.0741261 0.0925017 0.0739790 0.0554852 0.0739790 0.0554852 !
! 0.1482242 0.0555924 0.0555924 0.0741245 0.0741261 0.0926580 0.0741261 0.0741261 0.0925017 0.0739790 0.0554852 0.0739790 0.0554852 !
! 0.1480418 0.0555041 0.0555041 0.0740036 0.0740014 0.0925012 0.0740014 0.0740014 0.0927196 0.0742069 0.0556538 0.0742069 0.0556538 !
! 0.1480091 0.0554883 0.0554883 0.0739819 0.0739790 0.0924730 0.0739790 0.0739790 0.0927586 0.0742478 0.0556841 0.0742478 0.0556841 !
! 0.1480110 0.0554892 0.0554892 0.0739832 0.0739803 0.0924747 0.0739803 0.0739803 0.0927564 0.0742454 0.0556823 0.0742454 0.0556823 !
! 0.1480091 0.0554883 0.0554883 0.0739819 0.0739790 0.0924730 0.0739790 0.0739790 0.0927586 0.0742478 0.0556841 0.0742478 0.0556841 !
! 0.1480110 0.0554892 0.0554892 0.0739832 0.0739803 0.0924747 0.0739803 0.0739803 0.0927564 0.0742454 0.0556823 0.0742454 0.0556823 !
```

```
0.1480091 0.0554883 0.0554883 0.0739819 0.0739790 0.0924730 0.0739790  0.0739790 0.0925012 0.0739784 0.0554848 0.0739784 0.0554852
    A         B         B         C         C         D         C          C         D         C         B         C         B
```

Soient les groupes d'entités suivants déduits de la matrice DD^{100} :

$A = \{E_1\}$
$B = \{E_2, E_3, E''_6, E''''_6\}$
$C = \{E_4, E_5, E''_5, E'''_5, E'_6, E'''_6\}$
$D = \{E'_5, E_6\}$

En étant parti de n'importe quel noeud et en naviguant assez longtemps dans le graphe on obtient une probabilité de l'ordre de 0.1480091 pour arriver à E_1. Nous pourrons dire que E_1 ne peut pas être "illuminé"/"activé" davantage que 14,8% (un seul exemplaire du groupe d'entités A : 1 x 14,8% = 14,8%). Pour les éléments de B cela vaut 5,5% (4 exemplaires du groupe d'entités B : 4 x 5,55% = 22,2%). Pour ceux du groupe d'entités C, cela vaut 7,4% (6 x 7,4% = 44,4%). Pour le groupe D cela vaut 9,25% (2 x 9,25% = 18,5%). Le total fait 99,9%.

Pour une puissance de DD moindre, ces valeurs peuvent être soit plus élevées soit quasiment nulles. Les classes les plus importantes (en pourcentage individuel) sont A (14,8%), puis D (9,25%), puis C (7,4%) et enfin B (5,55%). On peut interpréter cela en disant que E_1 est un noeud d'articulation important (qui concentre les chemins: un "*hub*"), viennent ensuite plus modestement E'_5 et E_6. Ces classes de valeurs caractérisent la nature de "*hub*" d'un noeud et absolument pas son appartenance à une même composante que ceux de sa classe.

Notons que dans un 1-graphe complet d'arêtes sur m sommets, il y a $C_m^0 = 1$ seul cycle à 0 arête c'est le cycle vide {}. L'ensemble des cycles à une arête, c'est l'ensemble de toutes les boucles de chaque sommet sur lui même, il y en a $C_m^1 = m$ $\{E_1\},...,\{E_m\}$. Il y a $C_m^2 = m(m-1)/2$ cycles à 2 arêtes est $\{E_1, E_2\}$, $\{E_1, E_3\}$, ... , $\{E_1, E_m\}$, $\{E_2, E_3\}$, ... , $\{E_{m-1}, E_m\}$, etc. et il y a $C_m^m = 1$ seul cycle à m arêtes. Si l'on enlève les cycles à 0 arête et ceux à 1 arête, il y a au plus $2^m - m - 1$ cycles dans un graphe à m sommets. Un cycle se caractérise par son nombre d'arêtes, c'est-à-dire de sommets différents le constituant. Par exemple, dans un cycle 1 seul sommet apparaît 2 fois, les autres n'apparaissent qu'une seule fois. E_1-E_2-E_3-E_1 noté $\{E_1, E_2,$

E_3} est de longueur 3 mais E_3-E_1-E_3-E_4-E_5-E_3 ne sera pas considéré comme un cycle de longueur 5.

2.3 Conclusion

Nous avons présenté une approche générique permettant une exploitation automatique d'un réseau d'informations afin d'extraire les composantes de sens associées à une entité donnée en se basant sur l'étude des circuits dans le graphe associé à une large collection d'entité de même espèce (mot d'un dictionnaire ou page Web).

Dans le cas d'un dictionnaire, ce graphe est structuré sous la forme d'un RPMH, où les groupements de sens ainsi que leurs fusions représentent respectivement les sous petits mondes et les petits mondes de sens associés à un mot donnée. Cette étude a donné naissance à une interface utilisateur permettant à ce dernier d'entrer un mot (initiale) puis d'effectuer toutes les étapes décrites précédemment pour renvoyer les différentes composantes de sens associées à ce mot. En fait, cette étape nous a été très utile dans le processus de la reformulation sémantique de la requête dans le système SARIPOD qui sera détaillée dans le chapitre suivant.

Les résultats obtenus lors de la phase de test (voir annexe 3 pour le cas de dictionnaire) nous permettent d'affirmer qu'une même composante de sens contient rarement des mots ayant des sens différents [Elayeb et al., 2007c]. Cependant, un même sens peut couramment se retrouver dans deux composantes différentes. Chaque composante correspond ainsi à une nuance de l'acception du mot initial. C'est le cas du verbe garder par exemple auquel correspondent les quatre composantes suivantes : {<préserver, épargner, éviter, sauver, garantir, protéger, conserver>, <conserver, maintenir, préserver>, <conserver, maintenir, retenir>, <retenir, éviter, empêcher>}. Nous remarquons que la composante la plus fournie est celle correspondant à l'acception la plus courante du mot initial. Par ailleurs, un même mot peut se retrouver dans deux composantes différentes désignant chacune une nuance. L'exemple du verbe "peser" illustre nos dires de façon plus claire puisque l'essai lui associe les composantes suivantes : {<examiner, juger, considérer, apprécier, étudier, calculer, approfondir, estimer>, <conserver, maintenir, préserver>, <importuner, presser, harceler>, <importuner, fatiguer, ennuyer>, <évaluer, valoir, examiner>}.

D'autre part, cette approche offre à l'utilisateur la possibilité de paramétrer sa recherche de composantes. Ainsi, peut-il choisir lui-même la valeur du seuil d'acceptation et la longueur limite des circuits à prendre en compte. Bien évidemment, ceci requiert de l'utilisateur une expertise aussi bien en informatique qu'en linguistique. Ceci nous a poussés à envisager une solution dans laquelle le seuil d'acceptation est calculé automatiquement à partir de la matrice des circuits communs. Cette solution ne nécessite aucune expertise du côté de l'utilisateur et peut donc être exploitée par n'importe qui. Toutefois, la valeur calculée ne produit pas toujours les meilleurs résultats à cause de la variation de la répartition de la densité d'arcs dans le graphe. Les résultats obtenus dans [Elayeb et al., 2007bc] semblent encourageants et correspondent souvent aux différentes acceptions du mot à étudier. Cependant, la notion de "sens" est assez complexe et ambiguë en linguistique et certaines nuances de sens semblent très difficiles à cerner.

Par ailleurs, la construction du dictionnaire que nous avons utilisé pose quelques problèmes pour certains mots. Il est clair qu'un verbe comme faire (ou prendre, etc.) ne porte pas lui-même le sens mais c'est plutôt le rôle du groupe nominal qui le suit (faire le malin, faire mal, faire semblant, etc.). Or, les différentes acceptions sont associées à *faire* sans tenir compte du groupe nominal qui suit. Par conséquent, *faire* devient un verbe polysémique par excellence et se retrouve aussi bien comme synonyme de violenter (*faire mal*) que de procréer (*faire un*

enfant). Nous avons adopté la solution radicale consistant à éliminer ce type de verbes de notre étude afin de minimiser les erreurs résultant de leur usage. D'autre part, le même type de problème a été évoqué avec les noms. Par exemple, le mot *prise* est un mot polysémique désignant les trois sens différents : « prise de bec », « prise de judo » et « prise électrique », etc.

Dans le domaine des pages Web, nous obtenons aussi des résultats encourageants (voir les tests en annexe 4). En fait, cette approche générique nous a permi de créer des groupements des pages Web sous la forme de sous petits modes et de petits mondes des thèmes ou « sens commun ». L'utilisateur peut naviguer à partir de n'importe quelle page de ce réseau tout en visitant les autres pages hypertextuellement liées et thématiquement proches de sa page Web de départ.

Enfin, nous estimons que le choix du seuil d'acceptation est crucial et qu'une attention particulière doit lui être prêtée. En effet, les résultats sont étroitement liés à la valeur de ce seuil et en dépendent donc grandement. Il semble primordial de trouver une méthode robuste permettant de déterminer une valeur optimale du seuil d'acceptation en fonction du mot étudié et de ses connexions. Une étude statistique de la variation du seuil et ses effets sur les résultats sont envisageables.

3. Le Réseau Possibiliste du système SARIPOD

La mise en correspondance entre les deux RPMH du système SARIPOD est effectuée par le biais d'un réseau possibiliste dont les nœuds sont, d'une part les termes du RPMH de dictionnaire et d'autre part les documents du RPMH de pages Web. Notre objectif consiste à gérer une approche basée sur les mesures de nécessité et de possibilité dans un modèle de Recherche d'Information (RI). En effet, l'appariement de ces deux RPMH via un réseau possibiliste permet de calculer les degrés de pertinence possibilistes des documents suivant deux critères, l'un quantitatif et l'autre qualitatif. En fait, nous avons appliqué l'approche quantitative de [Brini et al., 2004abc] [Brini et al., 2005ab] présentée dans le chapitre 3, non pas à la totalité d'un document, mais à ses entités logiques, obtenues suite au processus d'analyse de document permettant de générer les fragments logiques de chaque page Web retrouvée (voir détail dans le chapitre suivant).

Les fragments logiques retenus dans le tableau 4.5 sont obtenus suite à une phase d'apprentissage réalisée sur la base de test contenant 974 documents HTML (voir détail dans le dernier chapitre de la réalisation). En fait, nous avons remarqué que la majorité de ces documents possèdent une ou plusieurs de ces fragements logiques retenus. La qualité d'un document réside dans le poids de chaque fragment logique par rapport aux désires de l'utilisateur.

Pour cela, nous attribuons un coefficient de pertinence possibiliste à chaque entité (ou fragment) logique selon son importance dans le document Web. Ces coefficients représentent la première partie des préférences de l'utilisateur et sont calculés de la manière suivante :

$$\alpha_{NM} = NM + Max(\alpha_{Légendes}, \alpha_{Paragraphe}) \quad (4.1)$$

$$\alpha_{Ni} = NM - N_i + Max(\alpha_{Légendes}, \alpha_{Paragraphe}) \quad (4.2)$$

Où NM est le niveau maximal détecté dans le document et N_i est le niveau de la $i^{ème}$ entité logique.

Par ailleurs, les préférences de l'utilisateur du système SARIPOD sont définies comme étant la qualité du document qu'il recherche; c'est-à-dire ses préférences pour certains fragments

logiques dans les documents recherchés : des informations situées soit dans le titre principal du document, soit dans les sous-titres, soit dans les paragraphes, etc. ainsi que ses préférences pour certains types d'informations : informations dans des figures, dans des tableaux ou sous forme de séquences multimédia (voir tableau 4.5).

Entité logique du document Web	**coefficient de pertinence possibiliste (α_j)**
NiveauMax (NM=5)	5 +5 = 10
NM - 1	5 -1+5 = 9
NM - 2	5-2+5 = 8
NM - 3	5-3+5 = 7
NM - 4	5-4+5 = 6
Légende Figure (LF)	5
Légende Tableau (LT)	4
Légende Séquence Multimédia (LSM)	3
Paragraphe (P)	2

Tableau 4.5 : Coefficient de pertinence possibiliste de chaque entité logique

La pertinence quantitative de chaque entité logique d'un document (ELd_j) de la collection, sachant que la requête est $Q = (t_1, t_2, \ldots, t_T)$, est calculée de la manière suivante:

D'après la formule (3.36) du chapitre 3, $\Pi(ELd_j|Q)$ est alors proportionnel à :

$$\Pi'(ELd_j|Q) = \Pi(t_1|\ ELd_j)*\ldots* \Pi(t_T|\ ELd_j) = nft_{1j} *\ldots* nft_{Tj} \quad (4.3)$$

Avec $nft_{ij} = tf_{ij} /max(tf_{kj})$: fréquence normalisée des termes de la requête dans l'entité logique.

La certitude de restituer une entité logique d'un document pertinent d_j (ELd_j) pour une requête, notée $N(ELd_j|Q)$, est donnée par :

$$N(ELd_j|Q) = 1- \Pi\ (\neg ELd_j|Q) \quad (4.4)$$

Avec :

$$\Pi(\neg ELd_j|Q) = (\Pi(Q|\neg ELd_j)*\ \Pi(\neg ELd_j))/\Pi(Q) \quad (4.5)$$

De même $\Pi(\neg ELd_j|Q)$ est alors proportionnel à :

$$\Pi'(\neg ELd_j|Q) = \Pi(t_1|\ \neg ELd_j)*\ \ldots*\Pi(t_T|\neg ELd_j) \quad (4.6)$$

Ce numérateur peut être exprimé par :

$$\Pi'(\neg ELd_j|Q) = (1- \phi EL_{1j})*\ldots* (1- \phi EL_{Tj}) \quad (4.7)$$

Avec :

$$\phi EL_{ij} = Log_{10}(nCEL/nELd_i)*(nft_{ij}) \quad (4.8)$$

Où :

nCEL = nombre d'entités logiques des documents de la collection,

$nELd_i$ = nombre d'entités logiques des documents de la collection contenant le terme t_i,

Nous définissons le degré de pertinence possibiliste mixte de chaque entité logique d'un document d_i (ELd_i) par :

$$DPMEL(d_i) = \Pi(ELd_i|Q) + N(ELd_i|Q) \quad (4.9)$$

Enfin, nous définissons le degré de pertinence possibiliste mixte du document d_i par:

$$DPM(d_i) = \sum_j (\alpha_j * DPMEL_j(d_i)) \quad (4.10)$$

Les documents préférés sont ceux qui ont une valeur $DPM(d_i)$ élevée. En fait, les coefficients α_j de pertinence possibiliste sont paramétrés dans SARIPOD et peuvent être modifiés selon les préférences de l'utilisateur. Par exemple, si nous cherchons des documents ayant des figures contenant le mot « M », il suffit de donner la plus grande importance au coefficient de pertinence possibiliste correspondant à l'entité logique légende figure (α_{LF}). En conséquence les $DPM(d_i)$ de ces documents seront les plus importants et seront affichés en tête de la liste triée des documents recherchés [Elayeb et al., 2006].

3.1 Apport de l'approche qualitative du système SARIPOD

Considérons une mini-collection de 3 documents contenant des termes t_1, t_2, t_3 et t_4 :

$d_1 = \{t_1, t_1, t_1, t_2, t_2, t_3\}$,

$d_2 = \{t_1, t_1, t_2, t_2, t_2, t_2\}$,

$d_3 = \{t_1, t_3, t_3, t_3, t_3, t_4, t_4\}$

Ces termes sont répartis sur les entités logiques de ces trois documents comme l'indique le tableau 4.6. Notons le degré de pertinence possibiliste mixte (quantitative et qualitative) de chaque document d_i par $DPM(d_i)$. Par ailleurs, l'approche quantitative ne tient pas compte des emplacements des termes de la requête Q dans les entités logiques des documents de la collection. Soit $DPP(d_i)$, le degré de pertinence possibiliste de chaque document d_i calculé par cette approche [Elayeb et al., 2009].

L'évaluation des documents d_1, d_2 et d_3 pour la requête $Q = (t_1, t_2, t_3, t_4)$ donne (nous ne donnons que le calcul non trivial de notre approche pour les préférences 1) :

$\forall EL_j \in \{NM, (NM-1), (NM-2), (NM-3), (NM-4), LF, LT, LSM, P\}, \forall i=1, 2, 3$

$\Pi(EL_j d_i|Q) = 0$, $N(NMd_1|Q) = N(Pd_1|Q) = 0.18$,

$N((NM-1)d_1|Q) = N((NM-4)d_1|Q) = N(LTd_1|Q) = N((NM-3)d_2|Q) = N(LSMd_2|Q)$

$= N((NM-2)d_3|Q) = N(LFd_3|Q) = N(NMd_3|Q) = 0.48$,

$N(NMd_2|Q) = N(Pd_2|Q) = 0.58$, $N((NM-1)d_3|Q) = N(Pd_3|Q) = 0.73$.

Entité logique du document	**d_1**	**d_2**	**d_3**
Niveau maximal (NM)	t_1	t_1, t_2	t_4
NM-1	t_2		t_1, t_3
NM-2			t_3
NM-3		t_2	
NM-4	t_3		
Légende Figure (LF)			t_3
Légende Tableau (LT)	t_2		
Légende Séquence Multimédia (LSM)		t_2	
paragraphe (P)	t_1, t_1	t_1, t_2	t_3, t_4

Tableau 4.6 : Répartition des termes dans les entités logiques des trois documents

Préférences de l'utilisateur / **Entité logique du document**	Coefficients α_j Préférences 1 (P1)	Coefficients α_j Préférences 2 (P2)	Coefficients α_j Préférences 3 (P3)
Niveau maximal (NM)	10	2	2
NM-1	9	5	8
NM-2	8	6	6
NM-3	7	9	4
NM-4	6	7	9
Légende Figure (LF)	5	3	7
Légende Tableau (LT)	4	4	10
Légende Séquence Multimédia (LSM)	3	10	5
paragraphe (P)	2	8	3

Tableau 4.7 : Les trois préférences de l'utilisateur du système SARIPOD

	Préférences 1 (P1)			**Préférences 2 (P2)**			**Préférences 3 (P3)**		
	d_1	**d_2**	**d_3**	**d_1**	**d_2**	**d_3**	**d_1**	**d_2**	**d_3**
[Brini et al., 2005a] : DPP(d_i)	0,16	0,18	0,24	0,16	0,18	0,24	0,16	0,18	0,24
Ordre de pertinence de documents	**3**	**2**	**1**	**3**	**2**	**1**	**3**	**2**	**1**
SARIPOD : DPM(d_i)	11,28	11,76	19,07	9,48	14,92	14,77	13,86	7,22	15,23
Ordre de pertinence de documents	**3**	**2**	**1**	**3**	**1**	**2**	**2**	**3**	**1**

Tableau 4.8 : Résultats de l'approche qualitative du système SARIPOD

La requête Q, interprétée comme une conjonction de termes serait trop restrictive, puisque aucun document de la collection ne contient les quatre termes à la fois. La nécessité et la possibilité d'avoir un des documents de cette collection comme résultat sont nulles. Pour éviter d'obtenir une liste vide de documents résultats, nous cherchons les documents qui contiennent au moins deux termes de la requête puis au moins un terme (si aucun document de la collection ne contient deux termes); ici, avec un seul terme, la possibilité de tous les documents vaut 1 et leur nécessité vaudra 0. Nous cherchons alors les documents qui traitent des ensembles $\{t_1, t_2\}$ ou $\{t_1, t_4\}$, ou $\{t_2, t_4\}$. Nous voyons à travers cet exemple, la nécessité de permettre à l'utilisateur d'exprimer des préférences entre les termes de la requête (cf. section 3.2).

D'autre part, nous remarquons que notre approche est plus fine que l'approche quantitative dans le calcul des pertinences possibilistes des documents de la collection car nous avons contribué à augmenter les scores des pertinences des documents contenant ces termes dans le but de pénaliser les scores de pertinence des documents ne les contenant pas.

Notons aussi que les scores des pertinences possibilistes des trois documents, calculés par l'approche quantitative sont très faibles par rapport à ceux calculés par notre approche et ceci grâce aux coefficients de pertinence α_j, facteurs primordiaux dans notre approche qualitative. En effet, et pour les préférences 1 de l'utilisateur, les différences de scores sont faibles dans la première approche (0,02 ; 0,08 et 0,06) à cause d'une faible différence dans le nombre de termes de chaque document (6 ; 6 ; et 7) alors que dans le cas de notre approche, ils sont

beaucoup plus remarquables (0,48 ; 7,79 et 7,31), ce qui montre bien la différence entre un document pertinent par rapport à ceux qui sont moins pertinents dans la collection.

Pour l'exemple de cette requête Q et pour certaines préférences, l'ordre de pertinence de documents change en changeant les préférences de l'utilisateur. En effet, dans le cas de préférences 1 (P1), le document d_3 est préféré aux documents d_2 et d_1 dans les deux approches. Ceci est dû au nombre de termes figurant dans d_3 d'une part (pour les deux approches), et au terme t_4 figurant dans une entité logique de poids important (pour notre approche). Alors que pour les deux autres préférences (P2 et P3), l'ordre de pertinence de documents change par rapport aux premières préférences (P1). En fait, et selon notre approche, le document le plus pertinent est celui dont les termes de la requête existent dans ses entités logiques possédant des coefficients de pertinence α_j importants tels que le niveau maximal (NM) et (NM-1) pour les préférences 1, LSM et (NM-3) pour les préférences 2, LT et (NM-4) pour les préférences 3, etc (voir tableau 4.7).

Suite à notre nouvelle approche, nous avons remarqué que même si les termes choisis tendent à sélectionner ce document, ces termes ne sont pas les plus fréquents dans le document (le terme t_4 n'est pas le plus fréquent dans d_3 alors qu'il a fortement contribué dans l'augmentation du score de d_3), ce qui montre l'atout de l'approche qualitative du système SARIPOD dans la sélection des documents pertinents [Elayeb et al., 2008, 2009].

3.2 Pondération des termes de la requête dans le système SARIPOD

Lors de la reformulation de sa requête, l'utilisateur choisi, pour chaque terme de sa requête initiale, un nombre de termes sémantiquement proches à ajouter pour la contruction de sa requête reformulée. Ces termes sont extraits des classes de « sens » construites dans le RPMH de dictionnaire. En fait, ces préférences entre les termes de la requête représentent la seconde partie des préférences proposées par l'utilisateur au système.

Considérons une requête $Q(t_1, t_2, t_3)$ composée de trois termes. Elle deviendra, après reformulation, la requête $Q'(t_1, t_{11}, t_{12}, t_{13}, t_2, t_3, t_{31}, t_{32})$, où t_{11}, t_{12}, t_{13} sont les trois termes les plus proches de t_1 et t_{31}, t_{32} sont les deux termes les plus proches de t_3. En fait, ces termes proches sont insérés dans Q' chaque fois que l'utilisateur saisit un nombre de termes proches pour un terme donné de la requête Q.

Nous définissons le degré de préférence (pondération) de l'utilisateur d'un terme t_i par rapport aux autres termes de la requête par :

$$Préf(t_i) = [Nbre\ termes\ proches\ choisis\ pour\ t_i\ dans\ Q' / Nbre\ termes\ de\ Q] + 1 \qquad (4.11)$$

Ici nous ajoutons le facteur 1 pour éviter que les préférences des termes pour lesquels nous n'avons pas choisi de termes proches soient nulles. Pour l'exemple du paragraphe précédent nous avons :

$Préf(t_1) = 3/3 + 1 = 2$; $Préf(t_{11}) = 0 + 1 = 1$; $Préf(t_{12}) = 0 + 1 = 1$
$Préf(t_{13}) = 0 + 1 = 1$; $Préf(t_2) = 0 + 1 = 1$; $Préf(t_3) = 2/3 + 1 = 5/3$
$Préf(t_{31}) = 0 + 1 = 1$; $Préf(t_{32}) = 0 + 1 = 1$

Il est clair ici que le terme t_1 est plus préférable que t_3 et t_2 ; parce que l'utilisateur a choisi un nombre plus important de mots sémantiquement proches de t_1, ce qui prouve bien qu'il s'agit d'un terme d'appui à sa requête. Le terme t_3 est aussi préférable au terme t_2 car l'utilisateur n'a pas demandé de mots proches de t_2 pour en préciser le sens.

Ainsi, les préférences calculées ici sont bien conformes avec le profil de l'utilisateur, parce que pour ce dernier le terme le plus important est celui dont il cherche le maximum de termes

proches. De cette manière, nous introduisons ces préférences entre les termes de la requête dans notre modèle possibiliste de la manière suivante [Elayeb et al., 2008] :

La pertinence quantitative de chaque entité logique d'un document (ELd_j) de la collection, sachant que la requête est $Q' = (t_1, t_2, \ldots, t_T)$, est calculée de la manière suivante:

La formule (4.3) de la section précédente devient [Elayeb et al., 2009]:

$$\Pi'(ELd_j|Q') = \Pi(t_1|\ ELd_j)*Préf(t_1)*\ldots* \Pi(t_T|\ ELd_j)*\ Préf(t_T)$$
$$= nft_{1j} * Préf(t_1)*\ldots* nft_{Tj}* Préf(t_T) \quad (4.12)$$

Avec $nft_{ij} = tf_{ij}/\max(tf_{kj})$: fréquence normalisée des termes de la requête dans l'entité logique.

La certitude de restituer une entité logique d'un document pertinent d_j (ELd_j) pour une requête, notée $N(ELd_j|Q')$, est donnée de façon analogue à celle présentée dans la section précédente, sauf que la formule (4.7) devient :

$$\Pi'(\neg ELd_j|Q') = [(1- \phi EL_{1j})/Préf(t_1)]*\ldots* [(1- \phi EL_{Tj})/Préf(t_T)] \quad (4.13)$$

En fait, nous avons bien introduit le facteur $Préf(t_i)$ dans le calcul de la *possibilité* ainsi que de la *nécessité*, parce que ce facteur est bien lié aux fréquences normalisées des termes (nft_{ij}) dans le document recherché.

Exemple.

Considérons une mini-collection de 3 documents d_1, d_2 et d_3 :

$d_1 = \{t_1, t_1, t_1, t_{11}, t_{11}, t_{12}, t_{12}, t_{12}, t_{13}, t_2, t_2, t_3, t_{31}\}$,

$d_2 = \{t_1, t_1, t_1, t_{11}, t_{11}, t_{12}, t_{12}, t_{12}, t_{13}, t_2, t_2, t_3, t_{32}\}$,

$d_3 = \{t_1, t_{11}, t_{11}, t_{12}, t_{12}, t_2, t_2, t_3, t_{31}, t_{32}, t_{32}\}$

Ces termes sont répartis sur les entités logiques de ces trois documents comme l'indique le tableau 4.9. L'évaluation des documents d_1, d_2 et d_3 pour la requête $Q'(t_1, t_{11}, t_{12}, t_{13}, t_2, t_3, t_{31}, t_{32})$ donne (nous ne donnons que le calcul non trivial pour les préférences 1) :

$\forall\ EL_j \in \{NM, (NM-1), (NM-2), (NM-3), (NM-4), LF, LT, LSM, P\}, \forall\ i=1, 2, 3$
$\Pi(EL_jd_i|Q') = \Pi_{Préf}(EL_jd_i|Q') = 0$, $N(NMd_1|Q') = 0.48$, $N_{Préf}(NMd_1|Q') = 0.73$,
$N(Pd_1|Q') = 0.94$, $N_{Préf}(Pd_1|Q') = 0.96$, $N(Pd_2|Q') = 0.82$, $N_{Préf}(Pd_2|Q') = 0.94$,
$N(LSMd_3|Q') = 0.73$, $N_{Préf}(LSMd_3|Q') = 0.84$, $N(Pd_3|Q') = 0.18$, $N_{Préf}(Pd_3|Q') = 0.5$,

Entité logique du document	**d_1**	**d_2**	**d_3**
Niveau Maximal (NM)	t_1, t_2	t_1, t_3	t_1, t_{32}
NM-1			t_2
NM-2	t_{12}		
NM-3		t_{12}	
NM-4			t_{31}
Légende Figure (LF)	t_{11}		
Légende Tableau (LT)		t_{11}	
Légende Séquence Multimédia (LSM)	t_1, t_{12}	t_{13}, t_{12}	t_3, t_{32}
paragraphe (P)	$t_1, t_2, t_{13}, t_{31}, t_{12}, t_3, t_{11}$	$t_1, t_1, t_2, t_2, t_{32}, t_{11}, t_{12}$	$t_{11}, t_{11}, t_{12}, t_{12}, t_2$

Tableau 4.9 : Répartition des termes dans les entités logiques des trois documents

Préférences de l'utilisateur / **Entité logique du document**	Coefficients α_j Préférences 1 (P1)	Coefficients α_j Préférences 2 (P2)	Coefficients α_j Préférences 3 (P3)
Niveau maximal (NM)	10	2	2
NM-1	9	6	10
NM-2	8	5	4
NM-3	7	10	7
NM-4	6	4	9
Légende Figure (LF)	5	3	3
Légende Tableau (LT)	4	9	6
Légende Séquence Multimédia (LSM)	3	7	5
paragraphe (P)	2	8	8

Tableau 4.10 : Les trois préférences de l'utilisateur du système SARIPOD

	Préférences 1 (P1)			**Préférences 2 (P2)**			**Préférences 3 (P3)**		
	d_1	**d_2**	**d_3**	**d_1**	**d_2**	**d_3**	**d_1**	**d_2**	**d_3**
Sans préférences entre termes de la requête	14,66	13,46	14,55	16,38	20,7	12,31	14,74	16,66	15,17
Ordre de pertinence de documents	**1**	**3**	**2**	**2**	**1**	**3**	**3**	**1**	**2**
Avec préférences entre termes de la requête	17,8	17,3	18,02	18,44	26,38	16,14	16,4	22,34	18,78
Ordre de pertinence de documents	**2**	**3**	**1**	**2**	**1**	**3**	**3**	**1**	**2**

Tableau 4.11 : Résultats de l'effet de l'ajout de préférences entre termes de la requête

Le système SARIPOD enregistre les préférences (pondérations) entre les termes de la requête lors de l'interaction de l'utilisateur avec le système. En fait, ces préférences entrent bien dans le cadre de la définition de son profil au système.

Les résultats collectés dans le tableau 4.11 montrent bien l'importance de la définition des préférences entre les termes de la requête utilisateur pour le cas de préférences 1 (P1). En effet, ce facteur a été introduit comme un facteur multiplicatif dans le calcul de la possibilité et comme un quotient dans le calcul de la nécessité ; ce qui permet en conséquence d'augmenter les deux scores de la possibilité et de la nécessité à la fois.

En cas du non prise en compte de pondérations des termes et pour les trois préférences du tableau 4.10, l'ordre de pertinence de documents change en passant de préférences à des autres. Alors qu'en cas de la prise en compte de ces pondérations, uniquement les préférences 1 s'avèrent significatives et contribuent au changement de l'ordre de pertinence de documents. Ceci grâce au terme t_1 (de préférence 2 et existant dans une entité logique de poids 10) et au terme t_2 (existant dans une entité logique de poids 9) qui ont contribué à l'augmentation du score de d_3 par rapport aux autres. Pour les deux autres préférences P2 et P3, le terme le plus préférable (t_1) existe dans une entité logique de poids 2 ; c'est pour cette raison le facteur *Préf*(t_1) n'a pas fait les différences dans les scores des documents. Il s'agit,

en fait, d'un facteur qui dépend de coefficients de pertinence possibiliste pour définir le profil de l'utilisateur du système.

Globalement, l'insertion des facteurs $Préf(t_i)$ dans les calculs des possibilités et des nécessités, consiste à augmenter les scores de pertinences possibilistes des documents contenant ces termes dans le but de pénaliser les scores de pertinences des documents ne les contenant pas. La pénalisation et l'augmentation des scores sont proportionnelles au pouvoir des termes à discriminer entre les documents de la collection.

D'autre part, ces pondérations permettent de restituer des documents classés par préférence de pertinence. Il est possible dans ce cas d'évaluer à quel point un document d_1 est préféré au document d_2 ou de mesurer la préférence du document d_1 par rapport à un ensemble de documents $\{d_3, d_4\}$.

En fait, ces facteurs *Préf* sont plus efficaces que le facteur *idf*, puisque la distribution des termes dans la collection de documents ne dépend pas seulement de la présence ou de l'absence des termes dans les documents de la collection (comme *idf*), mais de la distribution de leur densité dans les documents de la collection. Ainsi, comparé à *idf*, ces mesures sont plus performantes pour la discrimination négative.

4. Travaux similaires à notre approche

L'idée de base de la méthode de [Gaume et al., 2004] est de considérer qu'un dictionnaire est un graphe non orienté dont les mots sont les sommets et tel qu'il existe un arc entre deux sommets si l'un apparaît dans la définition de l'autre. Plus précisément, le graphe du dictionnaire encode deux types d'informations lexicographiques : les définitions qui décrivent les différentes acceptions de chaque vedette au moyen de séquences langagières ; la structure des articles qui organise ces sous sens.

Selon [Gaume et al., 2004], la nature hiérarchique des dictionnaires (distribution des degrés d'incidence des sommets en loi de puissance) est une conséquence du rôle de l'hyperonymie associée à la polysémie de certains sommets, alors que le fort *C* (existence de zones denses en arêtes) reflète le rôle de la cohyponymie [Duvignau, 2002], [Duvignau, 2003], [Gaume et al., 2002]. Par exemple, le mot *corps* se trouve dans de nombreux définissants (*tête*, *chimie*, *peau*, *division*). De ce fait, le sommet *corps* a une forte incidence. D'autre part, les auteurs constatent qu'il existe de nombreux triangles par exemple : {*écorce*, *enveloppe*}, {*écorce*, *peau*}, {*peau*, *enveloppe*}, ce qui favorise les zones denses en arêtes et plus précisément un fort taux de clustering *C*.

Par ailleurs, les auteurs ont présenté une méthode pour désambiguïser une entrée de dictionnaire en utilisant la notion de distance sémantique introduite par [Veronis et Ide, 1990] [Ide et Véronis, 1998] [Resnik et Yarowsky, 2000]. Ils ont défini la tâche comme suit : soit un lemme α qui apparaît dans la définition de l'un des sens d'un mot, β considéré comme un nœud du graphe. Le but étant donc d'associer α avec le sens le plus probable qu'il a dans ce contexte. Chaque entrée du dictionnaire est codée par un arbre de sous-sens dans le graphe du dictionnaire, avec une liste de nombres correspondants à chaque niveau de sous-sens caractéristique.

Soit un graphe non orienté $G = (V, E)$ définit par la donnée d'un ensemble non vide fini V de sommets, et d'un ensemble E de paires de sommets formant des arêtes. Si l'arête $\{r, s\} \in E$ on dit que les sommets r et s sont voisins, le nombre de voisins d'un sommet r est $d(r)$ son degré d'incidence.

Soit $[\hat{G}]$ la matrice $n \times n$ de transition de la chaîne de Markov homogène dont les états sont les sommets du graphe en question telle que la probabilité de passer d'un sommet $r \in V$ à l'instant i vers un sommet $s \in V$ à l'instant $i+1$ est égale à :

$[\hat{G}]_{r,s} = 0$ si $\{r, s\} \notin E$ (s n'est pas un voisin de r) ; $[\hat{G}]_{r,s} = 1/d(r)$ si $\{r, s\} \in E$ (s est un des $d(r)$ voisins de r qui sont tous équiprobables).

Gaume et al. ont appliqué l'algorithme suivant :

1. On supprime les voisins de β dans G de sorte que $\forall x \in V$, $[G]_{\beta,x} = [G]_{x,\beta} = 0$;
2. On calcule $[\hat{G}]^i$; pour un i bien défini (par exemple $i = 6$) ;
3. Soit L, le vecteur ligne de β alors $\forall k$, $L[k] = [\hat{G}]^i_{\beta,k}$;
4. Soit $F = \{x_1, x_2, \ldots, x_n\}$ les nœuds correspondant à tous les sous-sens de la définition de α. On prend alors $x_k = argmax_{x \in F}(L[x])$

x_k est alors le sous-sens le plus « proche » du nœud β, par rapport à la mesure Prox. Deux étapes demandent un peu plus d'explication :

1. Les voisins sont supprimés pour ne pas laisser un biais favorable aux sous-sens de β, qui formeraient alors une sorte de cluster artificiel par rapport à la tâche donnée. Ainsi la « marche aléatoire » dans le graphe peut vraiment avoir lieu dans le graphe plus général des autres sens.
2. Choisir une bonne valeur pour la longueur de la marche aléatoire n'est pas simple, et est le facteur essentiel de la réussite de la procédure. Si elle est trop petite, seules les relations locales vont apparaître (synonymes proches, etc.) et ils peuvent ne pas apparaître dans les contextes à désambiguïser (c'est notamment le problème de la méthode de [Lesk, 1986]) ; si la valeur de i est trop grande par contre, les « distances » entre tous les mots tendent à converger vers une constante, faisant disparaître les différences. Cette valeur doit donc être reliée d'une façon ou d'une autre à la distance moyenne entre deux sens quelconques du graphe. Une hypothèse raisonnable est donc de rester proche de cette valeur, et les auteurs ont pris le nombre 6, la moyenne calculée étant de 5,21 (sur le graphe contenant tous les sous-sens, pas sur celui ne contenant que les entrées, pour lequel L = 3,3).

Ainsi, l'approche présente une méthode de désambiguïsation dans laquelle le sens est déterminé en utilisant un dictionnaire. La méthode est basée sur un algorithme qui calcule une distance « sémantique » entre les mots du dictionnaire en prenant en compte la topologie complète du dictionnaire, vu comme un graphe sur ses entrées. La méthode, ne nécessitant pas de corpus annoté, est testée sur la désambiguïsation des définitions du dictionnaire elles-mêmes.

A notre connaissance, les travaux qui concernent la prise en compte des proximités sémamtiques entre les mots nœuds d'un graphe de dictionnaire pour la reformulation sémantique de requêtes sont limités. Cet aspect est important à considérer puisqu'il peut apporter un gain dans la finalisation de la requête reformulée dans un SRI.

Le modèle de SRI à base de deux RPMH que nous proposons est bien adapté pour représenter les requêtes et les documents, pour construire l'ensemble des connaissances et pour définir une stratégie de recherche plus fine et plus pertinente. La stratégie proposée se base sur une mise en correspondance par le biais de Réseaux Possibilistes. En effet, nous choisissons de mixer principalement deux approches possibilistes l'une quantitative proposée par [Brini et al., 2004abc] et l'autre qualitative.

L'approche qualitative que nous proposons est basée sur la fragmentation logique des documents. En fait, le système ne se limite pas uniquement à l'existence ou non des termes de la requête dans les documents, mais il s'intéresse aussi à ses emplacements dans les fragments logiques des documents. Cette approche introduit l'utilisateur dans le processus du choix de la qualité de ses documents recherchés. Dans ce cas le résultat de la recherche change de préférences utilisateurs à des autres.

L'approche quantitative est plus adaptée pour la représentation des documents dont les poids des leurs fragments logiques sont identiques et particulièrement quand il s'agit des utilisateurs ne possédant pas des préférences dans la qualité des leurs documents recherchés. Autrement dit, le système se limite dans ce cas à la vérification de l'existence ou non des termes de la requête dans les documents recherchés. En conséquence, le résultat de la recherche ne change pas lors du passage de préférences à des autres.

L'idée que nous voulons développer dans le système proposé est de faire combiner le modèle de requête et le modèle de document par le biais d'un réseau possibiliste mixant les deux approches possibilistes quantitative et qualitative pour tirer profit des avantages et des points forts de chacun par rapport au contexte utilisé :

- Reformulation sémantique de requêtes,
- Recherche dans les fragments logiques des documents,
- Recherche intelligente possibiliste,
- Recherche précise,
- Recherche exploratoire.

Nous développons davantage ces idées dans le chapitre suivant (Chapitre 5).

5. Conclusion

Nous avons présenté dans ce chapitre les choix, en les argumentants, que nous avons effectués pour satisfaire les objectifs fixés. Ces derniers se résument dans la proposition d'un SRI intelligent, adaptative, flexibilite et dynamique. En effet, l'originalité du modèle proposé se décline selon les trois volets suivants qui synthétisent nos contributions :

Le premier volet s'intéresse au processus itératif de la reformulation sémantique de requêtes. Cette technique est à base de relations de dépendance entre les termes de la requête. Nous évaluons notamment les proximités des mots du dictionnaire français « Le Grand Robert » par rapport aux termes de la requête. Ces proximités sont calculées par le biais de notre approche de recherche des composantes de sens dans un RPMH de dictionnaire de mots par application d'une méthode basée sur le dénombrement des circuits dans le réseau. En fait, l'utilisateur du système proposé choisit le nombre de mots sémantiquement proches qu'il désire ajouter à chaque terme de sa requête originelle pour construire sa requête reformulée sémantiquement. Cette dernière représente la première partie de son profil qu'il propose au système. La seconde partie de son profil est constituée des choix des coefficients de pertinence possibilistes affectés aux entités logiques des documents de la collection. Ainsi, notre système tient compte des profils dynamiques des utilisateurs au fur et à mesure que ces derniers utilisent le système. Ce dernier est caractérisé par son intelligence, son adaptativité, sa flexibilité et sa dynamicité.

Le second volet consiste à proposer des relations de dépendance entre les documents recherchés dans un cadre ordinal. Ces relations de dépendance entre ces documents traduisent les liens sémantiques ou statistiques évaluant les distributions des termes communs à des paires ou ensembles de documents. Afin de quantifier ces relations, nous nous sommes basés

sur les calculs des proximités entres ces documents par application d'une méthode de dénombrement de circuits dans le RPMH de pages Web. En effet, les documents peuvent ainsi être regroupés dans des classes communes (groupes de documents thématiquement proches).

Le troisième volet concerne la définition des relations de dépendance, entre les termes de la requête et les documents recherchés, dans un cadre qualitatif. Les valeurs affectées à ces relations traduisent des ordres partiels de préférence. En fait, la théorie des possibilités offre deux cadres de travail : le cadre qualitatif ou ordinal et le cadre quantitatif. Nous avons proposé notre modèle dans un cadre ordinal. Ainsi, des préférences entre les termes de la requête se sont ajoutées à notre modèle de base. Ces préférences permettent de restituer des documents classés par préférence de pertinence. Nous avons mesuré aussi l'apport de ces facteurs de préférence dans l'augmentation des scores de pertinence des documents contenant ces termes dans le but de pénaliser les scores de pertinence des documents ne les contenant pas.

Nous présentons dans le chapitre suivant la spécification et la conception du système proposé. Nous expliquerons davantage le rôle de chaque composante du système et son apport par rapport à la recherche.

Chapitre 5

Spécification et conception du système SARIPOD

La satisfaction d'une demande d'information est devenue à la fois plus facile et plus compliquée. Elle est devenue plus facile dans la mesure où grâce à l'émergence de nouvelles sources de données, comme le réseau international appelé Internet, chacun, en principe, peut avoir accès à une source d'informations inépuisable. Cependant, la masse énorme d'informations disponibles sur Internet, même sur un intranet ou un Data Warehouse, qui semble à première vue être sa force majeure, est en même temps l'une de ses faiblesses. La quantité d'informations à la disposition de l'utilisateur, généralement un décideur, est trop grande : l'information recherchée est probablement disponible quelque part, mais il arrive souvent qu'une seule partie soit retrouvée, et parfois même rien du tout. Les méthodes conventionnelles de recherche d'information se sont avérées incapables de résoudre ces problèmes. Ces méthodes supposent que nous connaissons d'avance quelle information est valable et où exactement elle peut être trouvée. De telles méthodes sont utilisées de la manière suivante : les systèmes d'informations, comme les bases de données, sont approvisionnés avec des indices qui fournissent ces informations aux usagers. Grâce à ces indices, l'utilisateur peut, à tout moment, vérifier si certaines informations sont offertes par la base de données, si elles sont disponibles, et où il peut les trouver. Avec les nouvelles technologies notamment Internet, mais aussi Intranet/Extranet et Data Warehouse, ces stratégies ne sont plus applicables. Les raisons à cela sont les suivantes :

- *La nature dynamique d'Internet* : aucune supervision centrale ne s'applique quant au développement d'Internet. Toute personne qui désire l'utiliser et/ou offrir des informations ou des services est libre de le faire. Ceci a crée une situation où il est devenu très difficile d'avoir une idée claire sur la taille réelle d'Internet ;
- *La nature dynamique des informations* : les informations qui ne sont pas disponibles aujourd'hui peuvent être disponibles demain et le contraire s'applique aussi ;
- *L'information est hétérogène* : l'information est offerte sous plusieurs formats et de plusieurs façons. Ceci complique la recherche automatique d'une information donnée, puisque chaque format et chaque service nécessitent une approche particulière.

Plusieurs solutions existent pour résoudre les problèmes identifiés précédemment. La plupart sont des solutions ad hoc. C'est ainsi qu'en utilisant des programmes qui circulent sur Internet, nous pourrons gérer des méta-informations concernant tous les documents disponibles. L'information collectée, caractérisée par un ensemble de mots-clés, est sauvegardée dans des bases de données de grande taille. Toute personne qui désire chercher des informations peut les localiser en donnant un ou plusieurs mots-clés à ce moteur de recherche. Bien que les moteurs de recherche fournissent des services plus ou moins bons, ils possèdent plusieurs inconvénients.

Nous présentons dans ce chapitre les choix utilisés pour la mise en œuvre du modèle proposé pour une Recherche d'Information coopérative, adaptative et intelligente. Pour ce faire nous choisissons une architecture dôtée d'une capacité d'adaptation à un environnement

dynamique, tel est notre cas. Le choix d'une telle architecture est motivé par la facilité de la décomposition de problèmes, et par la richesse de combiner et faire coopérer plusieurs méthodes ; dans ce cadre il s'agira principalement de méthodes de classification de documents dans le RPMH de pages Web et de classification de mots dans le RPMH de dictionnaire ainsi que l'approche combinaisant ces deux RPMH via un Réseau Possibiliste.

Dans la première section de ce chapitre, nous proposons une spécification du système SARIPOD. Dans la deuxième section une conception détaillée du système proposé est présentée.

1. Spécification du système SARIPOD

Comme d'autres technologies, l'évolution d'Internet est continue. Le volume des données sera trop grand et trop varié de sorte qu'il sera impossible pour l'être humain de suivre ce qui se passe. Le pire, c'est que prochainement les logiciels conventionnels ne seront plus capables de maîtriser la situation, par conséquent une nouvelle structure pour la recherche d'informations s'avère dès aujourd'hui nécessaire. Une telle structure facilitera la tâche et fera abstraction des différentes techniques. Ce type d'abstraction est comparable à celui avec lequel les langages de programmation de haut-niveau ont débarrassé les programmeurs de tous les problèmes de bas-niveau.

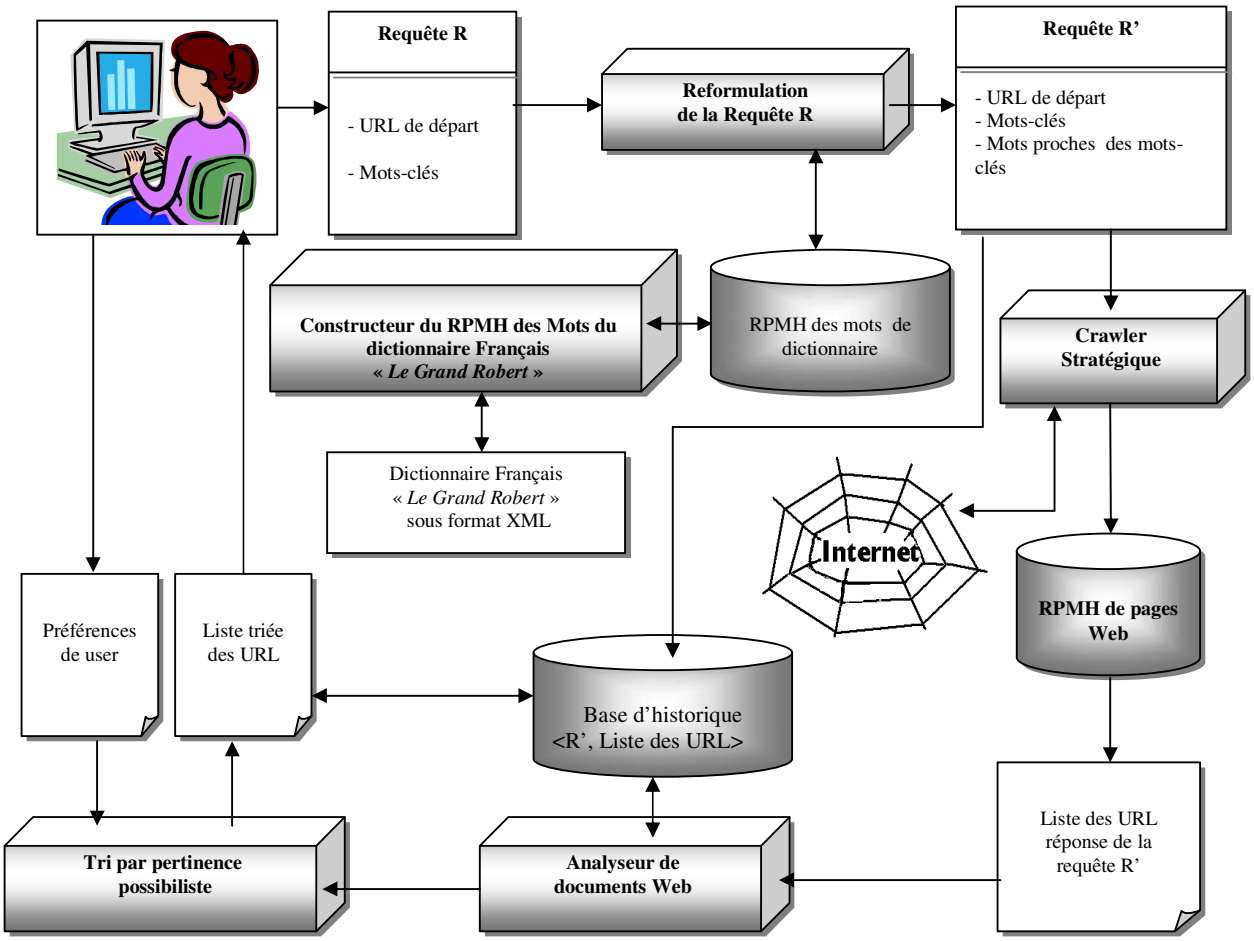


Figure 5.1 : Architecture générale du système SARIPOD

Pour favoriser la réutilisation, nous avons opté pour la modularité pour notre système. Ce dernier sera composé de plusieurs modules dont chacun est responsable de l'une des tâches du processus de recherche des documents sur Internet. La nécessité de coopération des différents modules permet de concevoir l'architecture générale du système SARIPOD composée des sept modules suivants (voir figure 5.1) [Elayeb et al., 2006] :

1. Module de construction du RPMH de dictionnaire ;
2. Module de reformulation de la requête utilisateur ;
3. Module de crawlage stratégique ;
4. Module de construction du RPMH de pages Web ;
5. Module d'analyse de documents Web ;
6. Module de tri des documents par leurs pertinences possibilistes ;
7. Module d'optimisation du système.

Nous détaillons dans la suite la fonctionnalité de chacun de ses modules et nous présentons dans le chapitre suivant quelques interfaces extraites de la réalisation de ce système.

1.1 Module de construction du RPMH de dictionnaire

Dans le cadre de la reformulation de la requête utilisateur, le système interroge le RPMH du dictionnaire de mots en vue de déterminer les mots sémantiquement proches des mots-clés proposés par l'utilisateur du système.

Ce module accepte en entrée le dictionnaire français « *Le Grand Robert* » sous format XML et engendre l'ensemble des circuits dans ce RPMH des mots de ce dictionnaire. En effet, l'objectif ultime de ce module réside dans la génération des mots sémantiquement les plus proches d'un mot donné dans la requête utilisateur. Cette proximité entre deux mots quelconques M_1 et M_2 du dictionnaire est calculée par la formule suivante :

$$Proximité\_Dictionnaire\ (M_1, M_2) = Nbre\ de\ circuits\ (M_1, M_2) / Nbre\ maximum\ de\ circuits\ détectés$$

L'architecture logicielle de ce module est présentée par la figure 5.2. En fait, dans notre prototype, ce module interagit avec le module d'analyse de documents Web pour permettre au module de tri de trier les pages Web sélectionnées selon leurs degrés de pertinences possibilistes (les tâches des ces modules seront détaillées dans la suite).

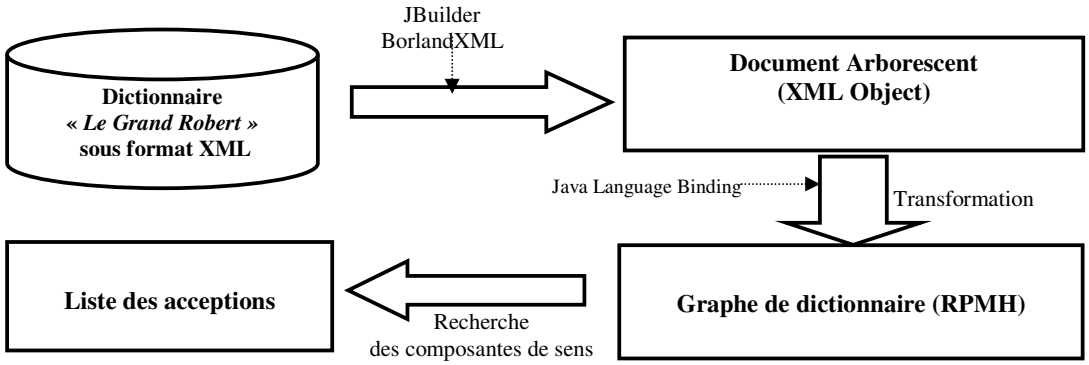


Figure 5.2 : Architecture interne de module de construction du RPMH de dictionnaire

En utilisant le graphe de dictionnaire comme source de données au format XML, ce module génère la liste des acceptions d'un mot donné. Il commence, en fait, par étudier certaines

propriétés des circuits collectés à partir du graphe afin de déduire les composantes de sens cherchées.

La construction du RPMH de dictionnaire est faite une fois pour toutes par ce module dès le démarrage du système SARIPOD. De plus elle est purement liée à la source de données XML dictionnaire français « *Le Grand Robert* ». En effet, pour chaque requête utilisateur, ce module accède aux parties de ce RPMH, correspondantes aux mots-clés de la requête, et ce en vue de faire les groupes de sens intermédiaires ainsi que leur fusion afin d'obtenir les composantes de sens. La figure 5.3 présente une description fonctionnelle de la recherche des composantes de sens.

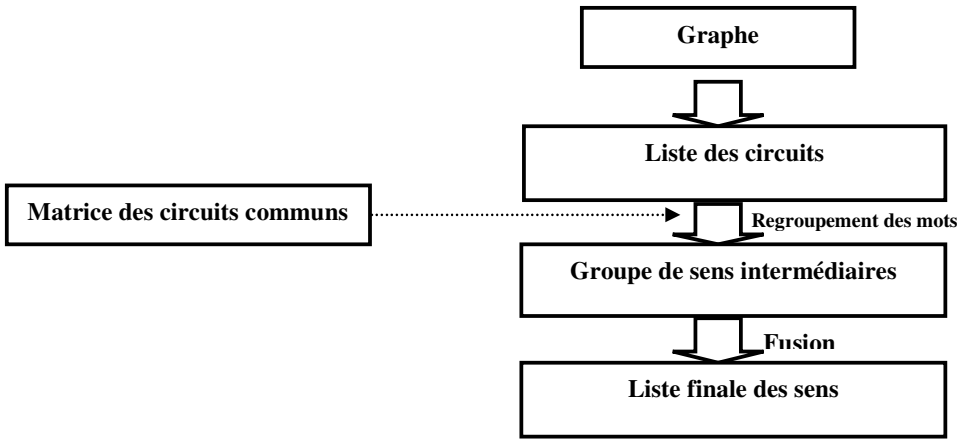


Figure 5.3 : Description fonctionnelle de la recherche des composantes de sens

Par ailleurs, le traitement fait par le module de construction du RPMH de dictionnaire passe par les 8 étapes suivantes afin d'arriver à la fin à la liste finale des composantes de sens :

(i) Phase préliminaire

Cette phase préliminaire consiste à nettoyer la source de données utilisée au format XML. Dans ce fichier, chaque entrée étant décrite par un ensemble de balises traduisant des informations de natures syntaxiques (voir figures 5.4 et 5.5).

```
<!-- Premier NIVEAU de l'arbre -->
        <!ELEMENT DICO (DEF+)>
<!-- Deuxième NIVEAU de l'arbre -->
        <!ELEMENT DEF
    (ENTRANT?,(STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU|AUTREFORME|PRONOMINAL)*,(CONTRAIRE|DERI
    VATIF|COMPARATIF|HOMONYME)*)>
        <!ATTLIST DEF
            mot             CDATA #REQUIRED
            phonetique      CDATA #REQUIRED
            cat             CDATA #REQUIRED>
        <!ELEMENT CONTRAIRE     (#PCDATA)*>
        <!ELEMENT DERIVATIF     (#PCDATA)*>
        <!ELEMENT COMPARATIF    (#PCDATA)*>
        <!ELEMENT HOMONYME      (#PCDATA)*>
        <!ELEMENT AUTREFORME (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
        <!ELEMENT PRONOMINAL (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
        <!ELEMENT NIVEAU (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
        <!ATTLIST NIVEAU type (1|2|3|4) #REQUIRED>
<!-- Troisième NIVEAU -->
        <!ELEMENT META          (STANDARD|ITALIQUE)*>
        <!ELEMENT MOTSLIES      (STANDARD|ITALIQUE)*>
        <!ELEMENT STANDARD      (#PCDATA)*>
        <!ELEMENT ITALIQUE      (#PCDATA)*>
```

Figure 5.4 : La DTD initiale du dictionnaire

```
<DEF mot="a" phonetique="[a, à]" cat="n. m. ">
<STANDARD>
<MOT>premier</MOT><TAG>ADJ:num:ord</TAG>
<MOT>lettre</MOT><TAG>NOM</TAG>
<MOT>et</MOT><TAG>CON:coo</TAG>
<MOT>premier</MOT><TAG>ADJ:num:ord</TAG>
<MOT>voyelle</MOT><TAG>NOM</TAG>
<MOT>de</MOT><TAG>PRE</TAG>
<MOT>le</MOT><TAG>DET:def</TAG>
<MOT>alphabet</MOT><TAG>NOM</TAG>
<MOT>A</MOT><TAG>PRE</TAG>
</STANDARD>
<ITALIQUE>
<MOT>majuscule</MOT><TAG>ADJ</TAG>
</ITALIQUE>
<STANDARD>
<MOT>avoir</MOT><TAG>VER:pres</TAG>
</STANDARD>
<ITALIQUE>
<MOT>minuscule</MOT><TAG>ADJ</TAG>
</ITALIQUE>
…
```

Figure 5.5 : La source de données initiale de dictionnaire

En effet, nous avons commencé par un nettoyage automatique en éliminant les redondances à l'intérieur de chaque définition de mot du dictionnaire. Puis, nous avons procédé à un nettoyage manuel tout en laissant dans la définition d'un mot uniquement les balises des mots sémantiquement proches de ce dernier (voir figures 5.6 et 5.7). Cette phase de nettoyage permet de préparer le terrain aux phases suivantes dans le traitement afin d'améliorer les performances de ce module.

```
<?xml version="1.0" encoding="ISO8859_1" ?>
<!DOCTYPE dictionnaire SYSTEM "F1.dtd">
<dictionnaire>
<mot texte="abaca">
        <traduction>bananier</traduction><TAG>NOM</TAG>
        <traduction>philippin</traduction><TAG>ADJ</TAG>
        <traduction>pétiole</traduction><TAG>NOM</TAG>
        <traduction>fournir</traduction><TAG>VER:pres</TAG>
        <traduction>matière</traduction><TAG>NOM</TAG>
        <traduction>textile</traduction><TAG>ADJ</TAG>
        <traduction>matière</traduction><TAG>NOM</TAG>
        <traduction>appeler</traduction><TAG>VER:pper</TAG>
        <traduction>chanvre</traduction><TAG>NOM</TAG>
        <traduction>manille</traduction><TAG>NOM</TAG>
        <traduction>tagal</traduction><TAG>NOM</TAG>
        <traduction>tirer</traduction><TAG>VER:pper</TAG>
        <traduction>bananier</traduction><TAG>NOM</TAG>
        <traduction>cordage</traduction><TAG>NOM</TAG>
        <traduction>natte</traduction><TAG>NOM</TAG>
        <traduction>paillasson</traduction><TAG>NOM</TAG>
</mot>
<mot texte="abacule">
        <traduction>cube</traduction><TAG>ADJ</TAG>
        <traduction>élément</traduction><TAG>NOM</TAG>
        <traduction>mosaïque</traduction><TAG>NOM</TAG>
</mot>
…
</dictionnaire>
```

Figure 5.6 : la source de données finale de dictionnaire sous format XML

La DTD finale du dictionnaire sous forme de fichier XML est donnée par la figure 5.7.

```
<!-- Premier NIVEAU de l'arbre -->
  <!ELEMENT DICO (COMMENTAIRES,INFO_DICO,DEF+)>

  <!-- Deuxième NIVEAU de l'arbre -->
  <!ELEMENT COMMENTAIRES (#PCDATA)>
  <!ELEMENT INFO_DICO
(nbr_sommets?,nbr_verbes?,nbr_noms?,nbr_adjectifs?,nbr_adverbes?,nbr_prepositions?,nbr_conjonctions?,nbr_articles?,nbr_p
ronoms?)>
  <!ELEMENT DEF
(ENTRANT?,(STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU|AUTREFORME|PRONOMINAL)*,(CONTRAIRE|DE
RIVATIF|COMPARATIF|HOMONYME)*)>
  <!ATTLIST DEF
          mot       CDATA #REQUIRED
          phonetique CDATA #REQUIRED
          cat       CDATA #REQUIRED
          ishomo    CDATA #IMPLIED>
  <!-- Troisième NIVEAU de l'arbre -->
  <!ELEMENT nbr_sommets                EMPTY>
  <!ELEMENT nbr_verbes                 EMPTY>
  <!ELEMENT nbr_noms                   EMPTY>
  <!ELEMENT nbr_adjectifs              EMPTY>
  <!ELEMENT nbr_adverbes               EMPTY>
  <!ELEMENT nbr_prepositions           EMPTY>
  <!ELEMENT nbr_conjonctions           EMPTY>
  <!ELEMENT nbr_articles               EMPTY>
  <!ELEMENT nbr_pronoms                EMPTY>
  <!ATTLIST nbr_sommets n     CDATA #REQUIRED>
  <!ATTLIST nbr_verbes n      CDATA #REQUIRED>
  <!ATTLIST nbr_noms n        CDATA #REQUIRED>
  <!ATTLIST nbr_adjectifs n   CDATA #REQUIRED>
  <!ATTLIST nbr_adverbes n    CDATA #REQUIRED>
  <!ATTLIST nbr_prepositions n CDATA #REQUIRED>
  <!ATTLIST nbr_conjonctions n CDATA #REQUIRED>
  <!ATTLIST nbr_articles n    CDATA #REQUIRED>
  <!ATTLIST nbr_pronoms n     CDATA #REQUIRED>
  <!ELEMENT ENTRANT          (#PCDATA)>
  <!ELEMENT CONTRAIRE        (MOT,TAG?)*>
  <!ELEMENT DERIVATIF        (MOT,TAG?)*>
  <!ELEMENT COMPARATIF       (MOT,TAG?)*>
  <!ELEMENT HOMONYME         (MOT,TAG?)*>
  <!ELEMENT AUTREFORME (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
  <!ELEMENT PRONOMINAL (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
  <!ELEMENT NIVEAU (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
  <!ATTLIST NIVEAU type (1|2|3|4) #REQUIRED>
  <!-- Quatrième NIVEAU -->
  <!ELEMENT META        (STANDARD|ITALIQUE)*>
  <!ELEMENT MOTSLIES    (STANDARD|ITALIQUE)*>
  <!ELEMENT STANDARD    (MOT,TAG?)*>
  <!ELEMENT ITALIQUE         (MOT,TAG?)*>
  <!ELEMENT MOT              (#PCDATA)>
  <!ELEMENT TAG              (#PCDATA)>
```

Figure 5.7 : La DTD finale du dictionnaire sous format XML

(ii) La première phase : transformation XML → DOM

Dans cette première phase, la source de données sous format XML sera transformée en un arbre DOM. En fait, le DOM (Document Object Model) est une interface de programmation (API) qui consiste à décomposer le contenu d'un document HTML ou XML en une arborescence de noeuds (chaque élément du document est un noeud).

D'autre part, DOM est une recommandation du W3C[26] (consortium qui gère les standards liés à Internet). Son principe consiste à représenter en mémoire le contenu d'un document HTML ou XML sous la forme d'une arborescence d'objets.

Les développeurs d'applications qui désirent manipuler le contenu d'un document HTML ou XML utilisent un parseur logiciel compatible DOM[27]. Ils ont alors un ensemble d'API leur

[26] http://www.w3.org/

permettant de parcourir l'arborescence des objets afin d'affecter des opérations de lecture, ajout, modification, suppression de données.

(iii) La deuxième phase : transformation DOM →Graphe

Les services de "JAVA XML Binding" nous permettent de transformer l'arbre DOM, obtenu suite à la première phase du traitement, en un graphe. En effet, la manipulation de ce dernier est plus facile pour générer l'ensemble de circuits existants entre les nœuds de ce graphe dans la phase suivante.

(iv) La troisième phase : recherche de circuits

Cette phase est consacrée au dénombrement des circuits à partir du graphe RPMH de dictionnaire résultant de la phase précédente. Rappelons qu'un circuit correspondant à un mot donné est un enchaînement de plusieurs mots en partant de ce mot donné et en y revenant à ce dernier. De plus l'interface utilisateur permet un contrôle du paramétrage des circuits (longueur, nombre, etc.) afin que le paramétrage de ce module soit optimisé.

(v) La quatrième phase : recherche de mots sémantiquement proches

Le module de construction du RPMH de dictionnaire s'intéresse dans cette phase à la recherche des mots sémantiquement proches d'un mot donné à partir du nombre de circuits collectés pour ce dernier tout en précisant les proximités sémantiques entre ces mots. Le poids d'un mot sémantiquement proche est proportionnel au nombre de circuits le reliant au mot de départ (voir exemples en annexe 3).

(vi) La cinquième phase : construction de la matrice des circuits communs

La construction de cette matrice permet de récapituler les relations existantes entre les mots, sémantiquement proches du mot de départ, collectés dans la phase précédente. Ces relations sont comptabilisées par les nombres des circuits contenant à la fois les deux mots, entrées de chaque cellule de la matrice.

(vii) La sixième phase : construction de groupes de sens intermédiaires

En utilisant la matrice des circuits de la cinquième phase ainsi que la liste de mots proches pondérés, le module de construction du RPMH de dictionnaire construit, durant cette phase, les groupes de sens intermédiaires (possédant chacun un sens) correspondant au mot de départ. Nous obtenons en conséquence plusieurs groupes sémantiquement proches, une dernière phase de fusion de ces groupes s'avère utile afin d'obtenir les composantes (ou classes) de sens finales.

(viii) La dernière phase : fusion de groupes de sens intermédiaires → composantes de sens

C'est la dernière phase dans laquelle le module de construction du RPMH de dictionnaire fusionne les groupes de sens sémantiquement proches pour obtenir les composantes de sens correspondantes au mot de départ qui pourrait être l'un de termes de la requête utilisateur.

1.2 Module de reformulation de la requête utilisateur

Dans les approches présentées dans le premier chapitre de l'état de l'art (section 6), l'expansion de requête consiste à ajouter des termes reliés à ceux de la requête initiale à partir

[27] L'API DOM est composé d'un ensemble d'interface. Un parser logiciel compatible DOM implémente ces interfaces dans le langage de la plate-forme de développement (C++, Java, JavaScript, .NET). Notons aussi que cette transformation se fait en plusieurs étapes en utilisant la bibliothèque "BorlandXML" facilitant la manipulation directe de l'arbre d'éléments.

d'un thesaurus, d'un document jugé pertinent par l'utilisateur ou par le système, ou à l'issue d'une phase préalable de classification. Au sein de notre système, nous proposons une reformulation sémantique de la requête en fonction des proximités sémantiques existants entre les termes dans le RPMH de dictionnaire.

Ce module accepte en entrée la requête initiale de l'utilisateur. Ce dernier choisit un nombre de termes sémantiquement proches dont le système l'ajoute à la requête initiale à partir du RPMH de dictionnaire :

$Q^{old} = (t_1, t_2, \ldots, t_n)$

$Q^{new} = (\beta_1 t_1, \beta_2 t_2, \ldots, \beta_n t_n, \beta_{n+1} t_{n+1}, \beta_{n+2} t_{n+2}, \ldots, \beta_m t_m)$

Avec :

$\beta_j = Préf(t_j)$: La pondération (préférence) du terme t_j dans la requête reformulée ;

$t_1, t_2, \ldots, t_n$: Les termes initiales choisi par l'utilisateur ;

$t_{n+1}, t_{n+2}, \ldots, t_m$: Les termes sémantiquement proches (au sens du RPMH) de termes initiales.

En fait, l'utilisateur pourra choisir un seuil de proximité sémantique pour construire sa requête reformulée. La figure 5.8 présente un exemple de répartition des mots sémantiquement proches du verbe « vérifier » dans le RPMH de dictionnaire selon leur seuil de proximité.

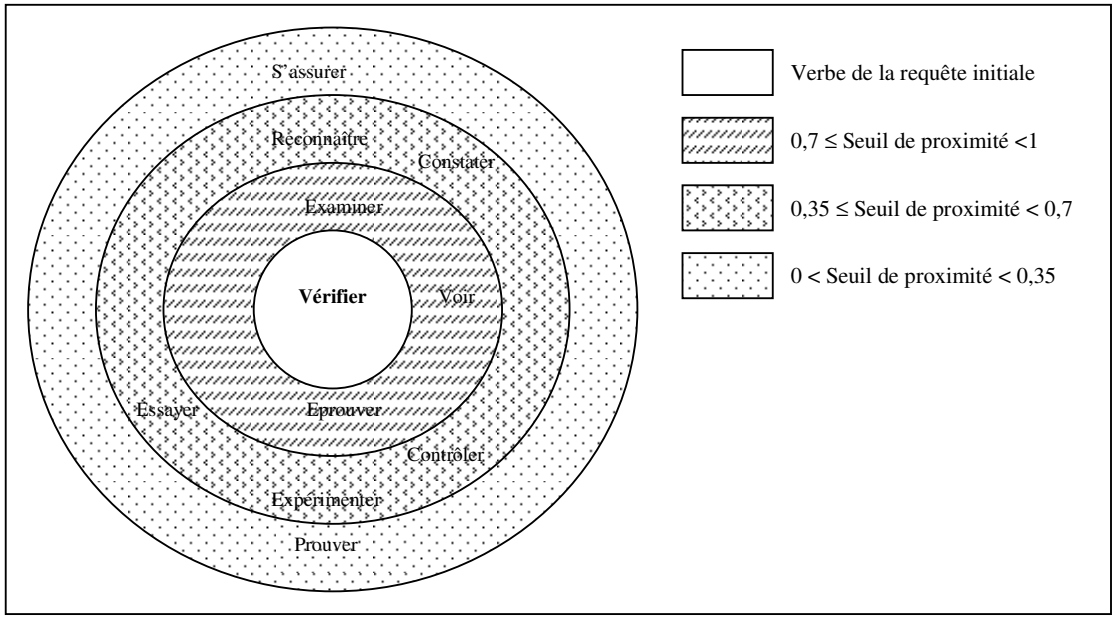


Figure 5.8 : Exemple du choix du seuil de proximité sémantique

1.3 Module de "*Crawlage*" stratégique

Selon [Miller et Bharat, 1998], un *Web crawler* est un mot anglais (en français : fouineur Internet) désignant un programme qui traverse automatiquement le Web en téléchargeant, de page en page, les URLs des documents (parfois les documents). Son point de départ est l'URL d'une page Web racine et sa profondeur de propagation est généralement paramétrable.

Après avoir obtenu une nouvelle requête reformulée Q^{new}, le module de *crawlage* explore le Web en partant de l'URL de départ choisie par l'utilisateur. Il obtient, en conséquence, plusieurs autres liens dont certaines pages peuvent contenir les mots-clés recherchés et

d'autres non. Dans ce cas, ce module ne s'intéresse qu'aux liens dont les pages, contiennent l'un de ces mots ou bien, si elles ne le contiennent pas, aux pages des liens inclus à celles-ci qui le contiennent. Les pages sémantiquement les plus proches, en terme de contenu, d'une page quelconque de ce réseau sont localisées dans les zones les plus denses du RPMH de pages Web.

Nous proposons dans ce cadre une exploration (*crawlage*) systématique dont le principe est donné par les algorithmes suivants :

Algorithme 1 :

1. Tant qu'une page sur *N* pages successives (parcourues grâce aux liens hypertextes) contient le mot *M* recherché, le crawler continue à visiter les pages sortantes de cette page ; quelle que soit la profondeur (car ces pages peuvent contenir le mot *M*) ;
2. Quand *N* pages de suite ne contiennent pas le mot *M* recherché (quelle que soit la profondeur), on stoppe la recherche dans cette branche.

En effet, on abandonne toute exploration d'une page ne contenant pas le mot recherché si aucune de ses pages filles ne le contient pas non plus. Tandis que dans tous les autres cas on continue.

Nous appelons par la suite cet algorithme : $Strat_N(R, M)$ = l'ensemble des pages ainsi récupérées pour le mot *M* en partant de $R(M)$ = l'ensemble des liens des pages affichées réellement par un moteur de recherche (par exemple Google).

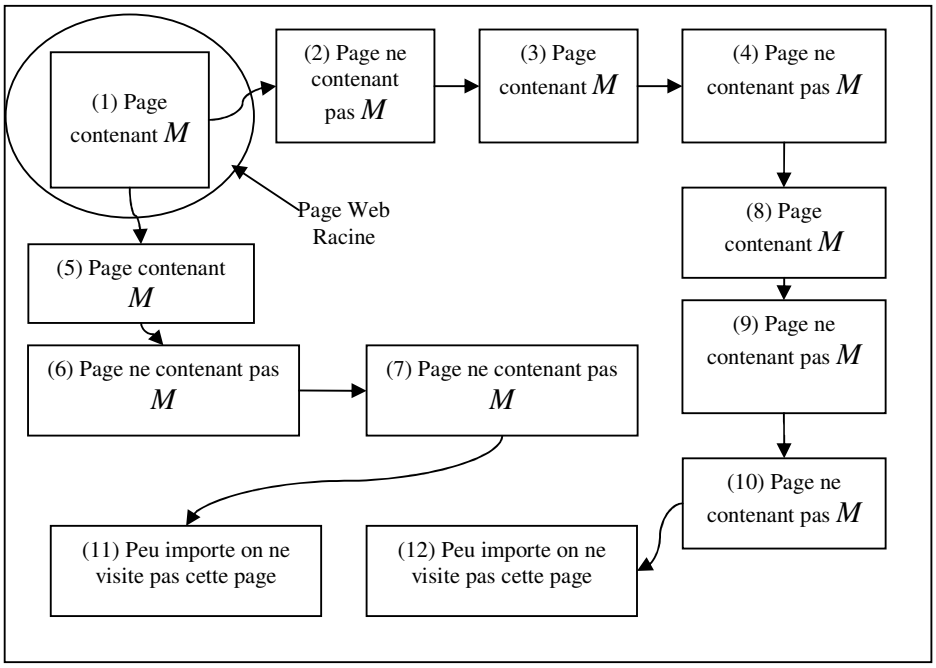


Figure 5.9 : Exemple de l'algorithme $Strat_2$

Nous remarquons, à partir de l'exemple de la figure 5.9, que $Strat_2(1, M)$ contient les pages à au plus 2 arcs d'une page contenant le mot *M* à partir de la page 1, soit {1, 2, 3, 4, 8, 9, 10, 5, 6, 7} alors que les pages à au plus 2 arcs de la page 1 sont {1, 2, 3, 5, 6}.

Par ailleurs, pour des valeurs faibles du pas de crawlage *N*, l'algorithme s'arrête très vite et ne charge qu'un nombre limité de pages, alors que pour des valeurs élevées de *N*, l'algorithme charge un nombre très important de pages dont plusieurs risquent d'être non pertinentes. Afin

de résoudre certaines limites de ce premier algorithme, nous proposons une extension comme suit.

Algorithme 2 :

Ce deuxième algorithme tient compte de la rencontre ou non du mot-clé *M* recherché dans les pages liées mais aussi d'un mot sémantiquement proche de *M*. Quand une page ne contiendra plus le mot *M*, ni un de ses mots proches sémantiquement, le saut se fera si l'on trouve dans cette page un mot *V* pas trop éloigné de *M* (bien que ne faisant pas partie de ce que nous avons appelé les mots proches de *M*). Ainsi $Strat_{\delta,N}(R, M)$ serait définie par :

1. Tant qu'une page, parcourue grâce aux liens hypertextes, contient le mot *M* recherché ou un de ses proches, au sens de Proximité du Dictionnaire détaillée dans la section 1.1, il faut garder l'URL de cette page et continuer à visiter les pages sortantes de cette page.
2. Si une page ne contient pas le mot *M* ni l'un de ses proches, il serait quand même dommage de ne pas la garder si elle contient néanmoins des mots *V* pas trop éloignés de *M* (d'une proximité $\geq \delta$ au sens de Proximité du Dictionnaire), il faut garder l'URL de cette page et continuer à visiter ses pages sortantes (qui ont une très forte probabilité de contenir de nouveau le mot *M* ou un de ses proches).
3. Si *N* pages de suite ne contiennent pas le mot *M*, ni un de ses proches et ne contiennent que des mots *V* d'une proximité $< \delta$, on stoppe la recherche dans cette branche et on ne garde aucune de ces pages.

Cet algorithme est plus performant que le premier algorithme $Strat_N(R, M)$ car il n'a pas le caractère ad-hoc pour la valeur de *N* qui ne pouvait être ni $N = 1$, ni $N > 2$.

Par ailleurs, cet algorithme semble plus conforme à ce que nous faisons dans la réalité. En effet, l'utilisateur continu à explorer la page et les liens sortants d'une page Web donnée s'elles contiennent des informations sémantiquement proches de ce qu'il cherche. Par contre, si les informations existantes sur ces pages Web sont loins de ce qu'il désir, il abandonne plus ou moins rapidement la navigation dans ce branche : c'est exactement le rôle du paramètre δ.

D'autre part, nous pouvons même faire de *N* une fonction de l'écart avec ce que l'on recherche. En effet, si les pages *P* ne vérifient pas $\exists V \in$ Mots tel que $V \in P$ et *Proximité_Dictionnaire* $(M, V) \geq \delta$, autrement dit si les mots *V* de *P* ne réalisent qu'une valeur de *Proximité_Dictionnaire* $(M, V) = \delta - \varepsilon$ alors si ε est petit on peut continuer encore plus mais si ε est grand on s'arrêtera assez rapidement. Ainsi, *N* est une fonction décroissante de ε.

Nous avons encore amélioré et simplifié cet algorithme 2 en proposant un troisième algorithme.

Algorithme 3 :

Nous définissons $Strat_\delta(R, M)$ par :

1. On garde toute page *P* qui contient un mot *V* suffisamment proche de *M*, au sens où $\pi =$ *Proximité_Dictionnaire* $(M, V) \geq \delta$.
2. On explore les pages sortantes de *P* jusqu'à une profondeur limite *N* (fonction croissante de *Proximité_Dictionnaire* (M, V) pour les *V* de cette page, mais fonction néanmoins majorée).
3. La limite d'exploration est mise à jour par celle de la page la plus profonde de la branche explorée.

La première étape (1) permet de garder les pages qui contiennent *M* ou un synonyme de *M* si $\pi \geq \delta$.

La seconde (2) assure que si l'on est en deçà de δ, c'est-à-dire $\pi < \delta$, on continue l'exploration d'autant moins loin que π est petit et on ne continue de toute façon pas indéfiniment.

La troisième (3) permet de repartir de plus belle si au terme de pages pas très intéressantes on retombe un jour sur une page concernée fortement par *M* (ou ses proches).

1.4 Module de construction du RPMH de pages Web

Ce module accepte en entrée l'ensemble de pages Web chargé par le Crawler et génère en sortie le RPMH correspondant. Nous calculons une proximité sémantique entre ces pages afin de préparer le terrain à leur classification par la suite.

Nous définissons la proximité entre deux pages P_i et P_j en terme du nombre de circuits passant par P_i et P_j et revenant à P_i de la manière suivante :

$$Proximité\ (P_i, P_j) = Nombre\ de\ circuits\ (P_i, P_j)/Nombre\ maximum\ de\ circuits\ détectés$$

Par ailleurs, le RPMH des pages Web est construit selon les trois phases suivantes:

(i) Phase de transformation HTML $\rightarrow$ DOM

Chaque document Web est transformé, grâce à l'API SAX de Java, en un graphe de noeuds caractérisé chacun par son type et son contenu. Il s'agit de la phase de transformation HTML-DOM au terme de laquelle un document DOM sera engendré pour chaque page HTML. Ce document DOM contient les mêmes informations contenues dans la page Web mais sous forme d'un arbre d'éléments.

La manipulation du document HTML initial sera la manipulation d'une structure de données sous forme d'un arbre d'éléments, et cette technique, offerte par l'API SAX de Java, va nous faciliter la tâche de lecture des documents Web ; puisqu'on procèdera par la suite à un parcours direct de l'arbre DOM et à une lecture des champs de données de chaque élément de cet arbre.

(ii) Phase de recherche des chaînes de caractères

Chaque élément de l'arbre DOM présente une structure de données dont les deux attributs les plus importants sont le type du noeud et sa valeur. Le premier attribut indique que le noeud représente l'une des structures logiques contenues dans la page HTML du document, ou bien qu'il contient un lien hypertexte vers une autre page. Dans le premier cas, l'attribut valeur contient le champ d'information porté par le noeud qui est affiché quelque part dans la page Web. Dans le second cas il contient le chemin vers la page cible du lien hypertexte.

La recherche des mots sera alors une recherche dans les champs valeurs des noeuds de l'arbre DOM du document. En effet, si un champ valeur contient un lien hypertexte, la recherche se fera d'une manière récursive dans la page cible de ce lien après construction de son arbre DOM et ainsi de suite. La condition d'arrêt est contrôlée par le paramètre « *pas de crawlage* » introduit par l'utilisateur.

(iii) Phase de contrôle du pas de crawlage

Le pas de crawlage est un paramètre entier introduit au système par l'utilisateur. Il indique le nombre de pages successives, ne contenant pas les mots recherchés, qu'il ne faut pas dépasser dans l'ensemble des pages liées entre elles par des liens hypertextes. Si ce pas est dépassé, la

recherche dans le sens des liens hypertextuels sera stoppée et réinitialisée au niveau de la dernière page contenant les mots recherchés.

1.5 Module d'analyse de documents Web

Ce module utilise des connaissances sur la structure logique de chacun des documents retrouvés par le crawler. En effet, il tient compte de la liste des termes de la requête et leurs fréquences respectives dans chacune des entités logiques du document. De plus, ce module permet d'extraire la structure logique d'un document Web en termes de titres, de paragraphes et de légendes suivant l'architecture de la figure 5.10.

Nous proposons une décomposition de ce module en trois étapes successives :

1. la génération de l'arbre DOM ;
2. la segmentation de document ;
3. l'identification des fragments logiques du document.

Nous détaillons dans la suite chacun de ces trois étapes.

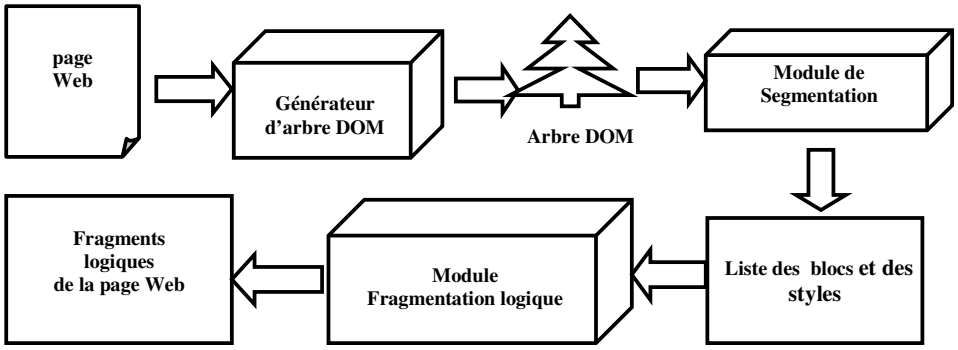


Figure 5.10 : Architecture interne du module d'analyse de page Web

1.5.1 Segmentation du document

La segmentation du document HTML[28] en une liste de blocs se base sur le changement de style et sur les séparateurs visuels. En effet, nous avons opté pour l'utilisation de l'arbre DOM. Ce dernier fournit plusieurs informations sur les nœuds du document et facilite son parcours. Une fois le document transformé en un arbre DOM, il s'agit de parcourir cet arbre pour engendrer la liste des blocs physiques et déterminer le style de chaque bloc sachant que le style est représenté par les différents attributs de style. La segmentation tient compte aussi des blocs non textuels. En effet, la liste des blocs engendrée par cette étape contient aussi bien des blocs textuels que des images, des tableaux, des séquences multimédia, des listes et des liens. Pour chacun de ces types de blocs, il faut calculer un certain nombre d'attributs. Par exemple, pour les images nous identifions l'attribut «src» qui indique le chemin au fichier source de l'image. Pour distinguer les différents types de blocs et leurs attributs, nous utilisons l'étiquetage basé sur les balises et les attributs HTML. En effet, nous traitons tous les types

[28] Dans cette version du système SARIPOD, nous traitons les documents HTML, mais un document XML facilitera encore plus les choses car on n'aura plus besoin d'interpréter les composants de la structure physique (aspects visuels, styles,…) et il suffira d'examiner la feuille de style attachée au document ou encore mieux le nom ou le rôle (la signification) même des balises (ex: titre, note, légende, énumération,…).

d'informations en même temps. Dans cette étape nous engendrons aussi la liste des styles rencontrés dans le document Web comme dans les travaux de [Bounhas, 2006].

1.5.2 Identification des titres et des légendes

Il s'agit d'utiliser la méthode d'étiquetage pour retrouver le plan du document. A ce stade, nous utilisons deux types d'étiquetage à savoir :

L'étiquetage basé sur le style. Après avoir calculé le niveau de chaque style de la liste des styles construite dans l'étape précédente, nous attribuons à chaque bloc une étiquette qui traduit son niveau dans la hiérarchie du document.

L'étiquetage sémantique. Il s'agit d'identifier les titres des légendes telles que les légendes des figures, des tableaux et des séquences multimédia. Nous attribuons à chaque bloc, dont le texte correspond à un titre de légende, une étiquette qui indique le type de la légende.

Identification des titres potentiels. L'objectif de ce traitement est d'identifier les blocs qui « peuvent être des titres ». Ainsi, nous calculons, pour chaque bloc, un attribut indiquant s'il « peut être un titre ». Pour ce faire, nous nous basons sur les attributs précédemment calculés. Nous considérons qu'un bloc peut être un titre s'il satisfait aux conditions suivantes :

- Le séparateur avant est non nul ;
- Le nombre de mots est inférieur à un seuil donné ;
- Le bloc n'est pas un lien ;
- Le bloc n'est ni un tableau, ni une image, ni une liste ;
- Le nombre de caractères alphanumériques est supérieur à zéro ;
- Le séparateur arrière est non nul ou il est nul mais le nombre de caractères du bloc est supérieur à un seuil donné.

En effet, pour tout bloc qui « peut être un titre », nous calculons un attribut « niveau » qui traduit son niveau dans la hiérarchie des titres du document. En fait, chaque bloc qui « peut être un titre » hérite le niveau de son style calculé en combinant trois critères à savoir le poids du style, sa régularité ainsi que sa fréquence dans le document.

Le calcul des poids des styles. Nous calculons, pour chaque style, un poids en fonction de ses attributs en utilisant la formule suivante :

$$Poids(S_i) = \sum_{j=1}^{6} p^j a_i^j \qquad (5.1)$$

En effet, le poids d'un style est la somme des valeurs de ses attributs pondérés par des poids. Dans cette formule a_i^j indique la valeur de l'attribut de style numéro j pour le style S_i. Le calcul de cette valeur dépend de la nature de l'attribut. En effet:

a_i^1 : prend la valeur de l'attribut taille.

a_i^2 , a_i^3 , a_i^4 : prennent la valeur 1 si le style est gras, italique ou souligné, 0 sinon.

a_i^5 : fréquence de la police du style en nombre de mots dans le document.

a_i^6 : prend la valeur 2 si le style est centré, 1 si le style est aligné à droite, 0 sinon.

p^j indique le poids que nous attribuons à l'attribut de style numéro j. Ces poids ont été fixés par apprentissage. En effet, d'après notre étude des documents HTML, la taille de la police puis l'alignement sont les attributs les plus utilisés pour distinguer les titres. Le gras, l'italique

et le soulignement viennent dans un deuxième niveau. En effet, nous considérons que le poids d'un style peut être calculé en se limitant à ces attributs. Nous attribuons, ensuite, des niveaux préliminaires aux blocs. Le style ayant le poids le plus élevé se verra attribué du niveau 1. Il est à noter que deux styles différents peuvent avoir le même niveau préliminaire puisque nous attribuons le même poids au soulignement, au gras et à l'italique. Pour résoudre ce problème, nous procédons au calcul de la régularité des titres.

Le calcul de la régularité des titres. Ce calcul vise à différencier les styles qui ont le même niveau préliminaire. Il s'agit de parcourir les blocs qui ont l'étiquette "PeutEtreTitre" pour attribuer des scores de régularité aux différents styles. Le score d'un style est fonction des styles qui le précèdent. En effet, l'algorithme de régularité, que nous avons défini, utilise une pile P pour stocker les styles déjà rencontrés. Le score d'un style S_i est incrémenté s'il est précédé d'un style ayant un poids plus important. Il est décrémenté s'il est précédé d'un style de même niveau préliminaire.

Pour illustrer le fonctionnement de l'algorithme, nous présentons un exemple de document où la notion de régularité permet de rectifier les niveaux préliminaires attribués aux blocs.

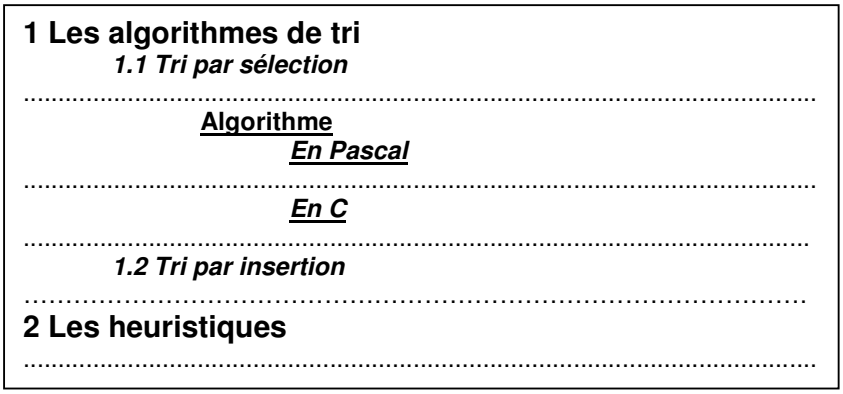


Figure 5.11 : Exemple de document où la notion de régularité peut être appliquée

Lors de la segmentation du document présenté par la figure 5.11, quatre styles sont identifiés. Cependant, trois d'entre eux auront le même niveau dans le calcul préliminaire car ils ont la même taille de la police (Il s'agit des styles des titres du deuxième, du troisième et du quatrième niveau) et le même alignement (pour les styles des titres de même niveau).

Lorsqu'on calcule la régularité, le score de S_2 est incrémenté lors de la rencontre du titre "1.1Tri par sélection" car le style qui le précède (S_1) a un niveau inférieur. Le score de S_3 est décrémenté lors de la rencontre du titre "Algorithme" car le style qui le précède (S_2) a le même niveau. Le score de S_4 est décrémenté de deux unités lors de la rencontre du titre "En pascal" car il est précédé par deux styles de même niveau à savoir S_2 et S_3. Les scores de régularité ainsi calculés permettent de distinguer les niveaux des styles. Etant donnés deux styles ayant le même niveau préliminaire, celui qui a le score de régularité le plus élevé aura le niveau le plus bas.

En fait, le calcul de régularité permet de rectifier les niveaux des titres dans les documents ayant une hiérarchie de sections à plusieurs niveaux. Sans ce calcul, plusieurs titres ayant des niveaux différents se verront attribués le même niveau.

Tri des styles par fréquence. Le troisième critère de tri des styles est la fréquence en nombre de mots dans le document. En effet, nous considérons que le style le moins fréquent est le style le plus important. Ainsi, une fois les scores de régularité calculés, les styles sont triés en

utilisant le poids, le score de régularité puis la fréquence en nombre de mots. Ensuite nous attribuons un niveau définitif à chaque style. Une fois le calcul des niveaux des styles achevé, il s'agit de calculer l'attribut "niveau" des blocs de la manière suivante:

- Tout bloc qui « peut être un titre » hérite le niveau de son style. Ainsi il est marqué comme titre.
- Les autres blocs auront un niveau fictif égal à -1.

1.6 Module de tri de documents par leurs pertinences possibilistes

Ce module calcule la pertinence possibiliste de chaque page Web en fonction de ses structures logiques générées par le module précédent. Le modèle possibiliste proposé pour le calcul des scores possibilistes est déjà présenté dans la section 3.1 du chapitre 4.

Rappelons que le degré de pertinence possibiliste mixte du document d_i est donné par :

$$DPM(d_i) = \sum_j (\alpha_j * DPMEL_j(d_i)) \quad (5.2)$$

Où $DPMEL(d_i)$ est le degré de pertinence possibiliste mixte de chaque entité logique d'un document d_i (ELd_i). Il est calculer par la formule suivante :

$$DPMEL(d_i) = \Pi(ELd_i|Q) + N(ELd_i|Q) \quad (5.3)$$

Enfin, ce module assure le tri décroissant des ces scores ($DPM(d_i)$) correspondants aux préférences proposées par l'utilisateur au système.

1.7 Module d'optimisation du système SARIPOD

Le module d'optimisation permet aux utilisateurs du système un gain important en terme de temps de réponse. En effet, ce module permet de construire une base d'historique des requêtes, traitées par le système, ainsi que leurs réponses. A la réception d'une nouvelle requête, le module d'optimisation consulte cette base d'historique, cherche la plus proche requête dans cette dernière et enfin il l'actualise tout en éliminant les URLs désormais non disponibles sur le Web et en ajoutant des nouvelles URLs inexistantes dans l'ancienne recherche. Cette dernière étape nécessite le renvoi de la nouvelle requête au crawler. Cette nouvelle réponse obtenue servira de même comme historique pour des requêtes ultérieures.

Le processus de recherche de la plus proche requête dans la base d'historique repose sur une technique de Data Mining intitulée le Raisonnement à Base de Cas ou de Mémoire (RBC ou RBM) [Berry et Linof, 1997]. En effet, cette technique utilise une fonction de distance entre la nouvelle requête et celles de l'historique et sélectionne la plus petite distance qui correspond à la requête la plus proche. Cette distance est déterminée en fonction du nombre de termes en commun entre la nouvelle requête et celle de l'historique, elle est calculée de la façon suivante :

1. Si les termes de la nouvelle requête sont identiques à ceux de la requête de l'historique, l'utilisateur pourra utiliser directement le résultat de l'historique ou changer ses préférences tout en lançant un nouveau processus de tri des documents selon de nouveaux paramètres ;
2. Si les termes de la nouvelle requête sont supérieurs à ceux de la requête de l'historique, l'utilisateur profitera de la partie de sa recherche existante dans l'historique et le système lui ajoute des nouvelles pages correspondantes au reste de

termes de sa requête. L'utilisateur pourra aussi changer ses préférences par rapport au profil existant dans l'historique.

3. Si les termes de la nouvelle requête sont inférieurs à ceux de la requête de l'historique, l'utilisateur profitera uniquement de la partie de l'historique qui correspond aux termes de sa requête. Il pourra aussi proposer d'autres préférences différentes de celles de l'historique.

En fait, la tâche principale de ce module consiste à ajouter au système une aptitude à l'apprentissage lui permettant ainsi de profiter des requêtes déjà jouées, pour des classes d'utilisateurs donnés. Autrement dit, le système pourra affiner le profil de son utilisateur au fur et à mesure que ce dernier utilise le système.

2. Conception du système SARIPOD

UML est un langage de modélisation fondé sur les concepts orientés objet qui sont nés depuis plus de trente ans ; UML n'est donc pas à l'origine des objets ; néanmoins, il constitue une étape majeure, dans le sens où il unifie les différentes approches (BOOCH, OMT, OOSE et ROO d'IBM) et en donne une définition plus formelle [Kettani, 1998].

Dans l'approche orientée objet, un système est vu comme étant une société d'objets qui coopèrent pour réaliser un certain objet global. Cette approche se base sur la représentation des éléments d'un système sous la forme des objets et dans laquelle un objet est défini par :

- *Une identité* qui constitue le moyen de l'identifier par rapport aux autres objets,
- *Un comportement* qui définit la manière dont l'objet agit et réagit aux divers messages qui lui parviennent de son environnement,
- *Un ensemble d'états* qui définit en fait les différentes possibilités dans lesquelles un objet peut se trouver à un instant donné de sa vie.

En outre, dans un problème réel, il est difficile de décrire tous les objets d'un domaine et il est plus réaliste de les classer dans des groupes d'objets appelés classes. Une classe sert donc d'abstraction qui décrit plusieurs objets partageant un ensemble de propriétés et/ou d'associations avec d'autres classes d'objets. Les objets forment les instances (occurrences réelles) d'une classe. Des associations entre les différentes classes d'une modélisation orientée objet d'un système peuvent exister et elles représentent en fait les liens existants entre les éléments du système réel.

Le langage UML propose plusieurs moyens de description et modélisation d'un système utilisant des diagrammes ou des graphes. De la modélisation UML nous adoptons dans ce rapport uniquement deux genres de diagramme : le diagramme des classes et le diagramme des séquences.

i. Modèle statique

Le diagramme de classes est une collection d'éléments du modèle statique, il montre uniquement les aspects statiques du système. Les différents modules du système SARIPOD sont représentés par ce diagramme de classes, des groupes et des rôles que peuvent jouer ces modules dans les différents groupes.

ii. Modèle dynamique

Le modèle dynamique est une vision microscopique du fonctionnement du système. Il sert à mettre en évidence les relations temporelles inter-objets et la représentation sous forme d'un

automate du comportement de chaque objet. Il intervient aprés la définition du modèle statique.

Un scénario est une séquence spécifique d'actions illustrant des comportements (effets observables d'une opération ou d'un évènement). Un scénario peut être utilisé pour illustrer une intéraction (spécification comportementale comprenant un ensemble de messages échangés entre des objets, dans un contexte particulier pour atteindre un but spécifique). En effet, une intéraction peut être illustrée par un ou plusieurs scénarios.

D'une manière générale, un scénario utilise deux types de concepts [Roques, 2001]:

- ✓ Des objets : ces sont les concepts faisant partie du système la plupart du temps, ainsi que des objets externes au système et en interaction avec celui-ci. En fait, les objets intervenant dans les scénarios sont des instances et il est donc nécessaire de spécifier leur nom et leur classe. Ils sont représentés par des barres verticales ;
- ✓ Des événements : ces sont les concepts émis et reçus par les objets impliqués dans le scénario. En fait, les interactions entre ces objets sont des événements précis et spécifiques. Un événement est représenté par une flèche horizontale reliant l'objet émetteur à l'objet récepteur. Les scénarii sont des suites d'événements séquencés dans le temps, la lecture séquentielle s'effectue de haut en bas [Lopez et al., 1998].

Nous présentons dans la suite une conception détaillée des principaux modules du système SARIPOD. Nous nous limitons aux présentations des diagrammes les plus importants dans le processus de recherche.

2.1 Conception et mise en œuvre du RPMH de dictionnaire

La figure 5.12 montre le détail du diagramme de classes pour le module de construction du RPMH de dictionnaire. Nous ne présentons ici que les classes principales et nous ignorons les autres classes d'importance moindre telles que les classes utilitaires engendrées par Java et la classe *frame* utilisée comme interface utilisateur.

D'autre part, le modèle de ce diagramme de classes représente une superposition de deux modèles : un premier modèle de dictionnaire contenant des informations syntaxiques et sémantiques, et un second modèle d'un graphe où l'on y trouve des notions attachées à la théorie des graphes telles que sommet, circuit, arc, etc. Dans cette superposition la relation "*correspond à*" entre *circuit* et *sens* représente le fait qu'un sens est obtenu à partir d'un groupe de circuits, et cette association permet d'obtenir un ensemble de mots proches (synonymes) qui sont des sommets-mots.

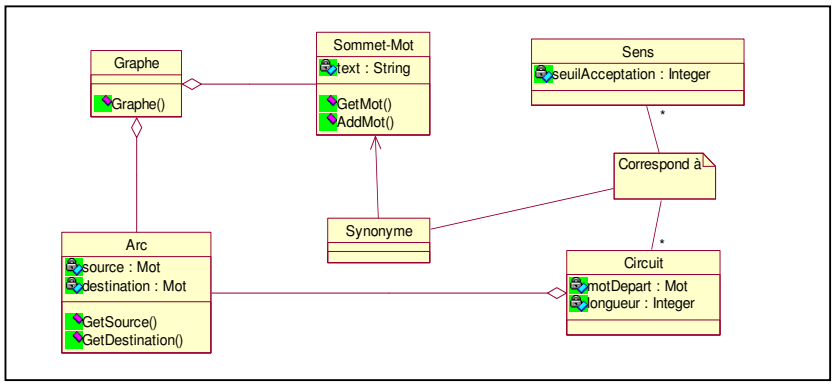


Figure 5.12: Diagramme de classes de la construction du RPMH de dictionnaire

Nous distinguons deux scénarii dans lesquels l'objet *frame* joue le rôle de l'interface utilisateur. Le premier scénario consiste en la récupération des mots proches d'un mot donné, alors que le second modélise le regroupement de ces mots proches en composantes de sens.

ChercherSynonymes(*mot , lgCircuit***)**

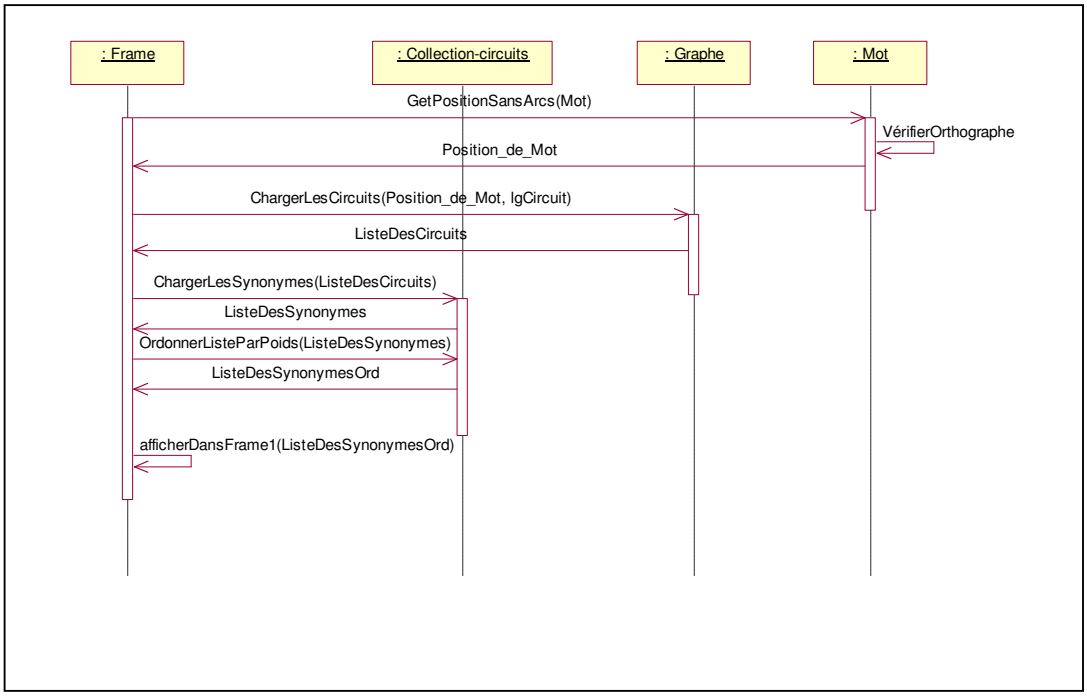


Figure 5.13 : Diagramme de séquences de la recherche des mots proches d'un mot

L'objet *frame* récupère la position du mot à étudier dans le dictionnaire puis cherche les circuits qui lui sont associés. Il s'agit ensuite de structurer ces circuits pour en déduire les mots proches du mot de départ.

GrouperLesSynonymes (*Seuil***)**

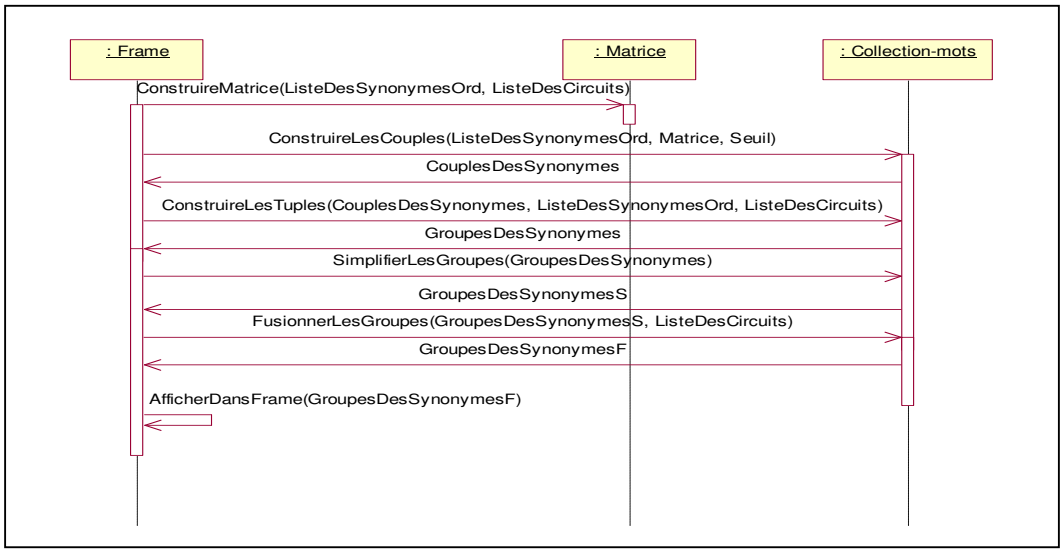


Figure 5.14 : Diagramme de séquences du groupement des mots proches d'un mot

Eventuellement, les groupes de mots obtenus contiennent des mots ayant des sens proches. Ces mots peuvent encore être regroupés en une même composante de sens. Il s'agit de la dernière phase de fusion de ces composantes de sens. En fait, les éléments de chacune des groupes fusionnés possèdent un sens spécifique par rapport aux autres.

2.2 Conception et mise en œuvre du crawlage stratégique

La figure 5.15 montre le détail du diagramme de classes d'ordre général pour les deux modules de crawlage stratégique et de tri des documents Web par leurs pertinences possibilistes.

Notons ici que le paquetage SDA correspond à l'analyse d'un document Web et le paquetage RPMH dictionnaire correspond au module de détermination des synonymes des mots-clés. En fait, nous avons développé les paquetages suivants :

- Le paquetage « Crawlage » renferme tous les objets contribuant à la réalisation de la tâche de crawlage des documents Web contenant une information représentée par des mots-clés de l'utilisateur qui sont enrichis par des synonymes grâce au module *«RPMH Dictionnaire»*.
- Le paquetage « Possibiliste » renferme toutes les classes assurant le tri des documents Web analysés par le paquetage « SDA ». En effet, ces documents sont collectés par le module de crawlage stratégique.
- Le paquetage « Historique » renferme toutes les classes contribuant à la réalisation de la tâche de gestion de la base d'historique du système en tant que lieu de conservation et de réutilisation des résultats déjà trouvés.

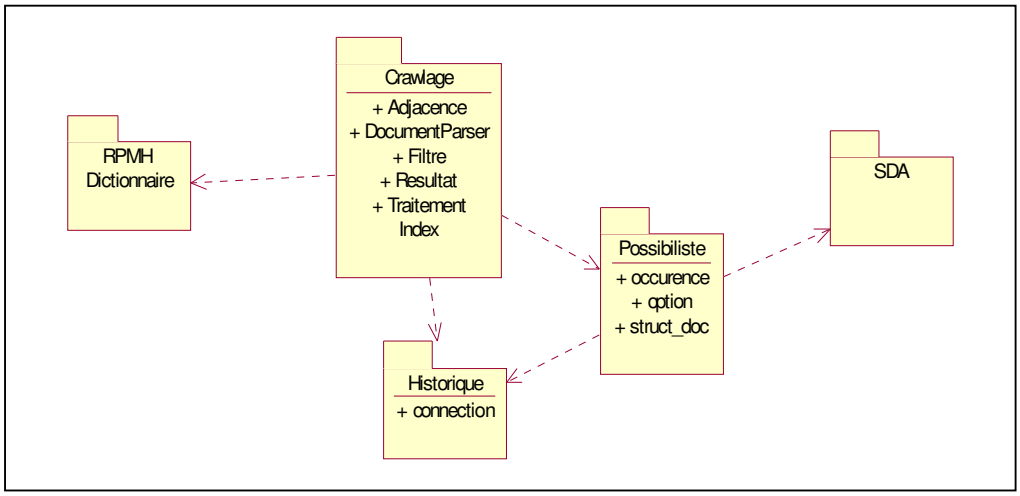


Figure 5.15 : Diagramme de classes générale de deux modules de crawlage et de tri

2.2.1 Diagramme de classes du module de crawlage stratégique

Le diagramme de classes du module de crawlage stratégique est présenté par la figure 5.16. Ce diagramme renferme les différentes classes nécessaires pour concevoir et pour mettre en œuvre toutes les opérations nécessaires pour lire des documents Web, suivre les liens hypertextes entre eux et créer le RPMH de pages Web.

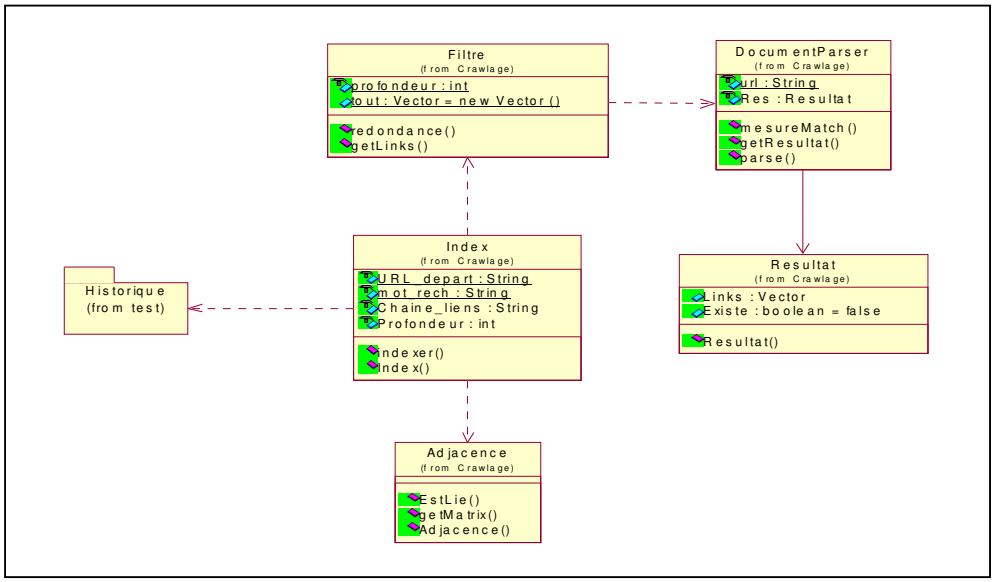


Figure 5.16 : Diagramme de classes du module de crawlage stratégique

La classe *"Index"* joue le rôle d'un chef d'orchestre dans ce module, elle fournit les paramètres initiaux pour les autres classes et médiatise la communication entre elles pour retourner enfin la collection des documents résultats de la recherche ainsi que la matrice d'*Adjacence* qui représente les proximités entre les pages. Dans le cas où le résultat du crawlage figure dans la base d'historique du système, la classe *"Index"* retourne le résultat stocké, sans recours à une nouvelle opération de crawlage.

La recherche des mots-clés ainsi que des liens hypertextes dans les pages Web sont faites par les méthodes de la classe *"DocumentParser"*. En effet, la méthode *"parse"* sert à construire l'arbre DOM d'un document à partir de son URL. En conséquence, le document Web sera représenté par un ensemble de nœuds de l'ardre DOM. En partant de la racine de cet arbre, la méthode *"getResultat"* examine le contenu de chaque nœud. En effet, la méthode s'intéresse aux nœuds contenant des mots-clés ou des liens hypertextes. Dans le premier cas c'est la méthode *"mesureMatch"* qui teste l'existence des ces mots recherchés. Dans le second cas, comme première itération, une nouvelle URL liée au document en question est ajoutée pour être traitée d'une manière récursive.

Ces paramètres retournés par la lecture d'un document sont conservés dans une structure de données propre pour chaque document. La classe *"Resultat"* renferme les deux attributs *"Existe"* et *"Links"* : le premier indique si les mots recherchés existent dans le document, le deuxième renferme la liste des URLs construite à partir des liens hypertextuels trouvés dans le document.

Après avoir traité le document Web par la classe *"DocumentParser"*, ce dernier est alors représenté par une structure *"Resultat"*. La classe "*Filtre*" renferme les méthodes servant à construire la liste des documents pertinents après avoir filtré les structures *"Resultats"* des documents lus.

La méthode *"getLinks"* ajoute les URLs des documents contenant les mots recherchés à la liste finale et relance le processus de recherche dans les documents liés. Elle exclut les liens internes qui renvoient vers le même document grâce à la méthode *"redondance"*.

La relance du processus de recherche dans les URLs du vecteur "*Links*" est contrôlée par la méthode *"getLinks"* grâce à l'attribut "*profondeur*" servant à compter le nombre de pages successives ne contenant pas les mots à rechercher.

2.2.2 Diagramme de séquences du module de crawlage stratégique

La figure 5.17 décrit les différentes interactions entre les classes participant au module de crawlage stratégique.

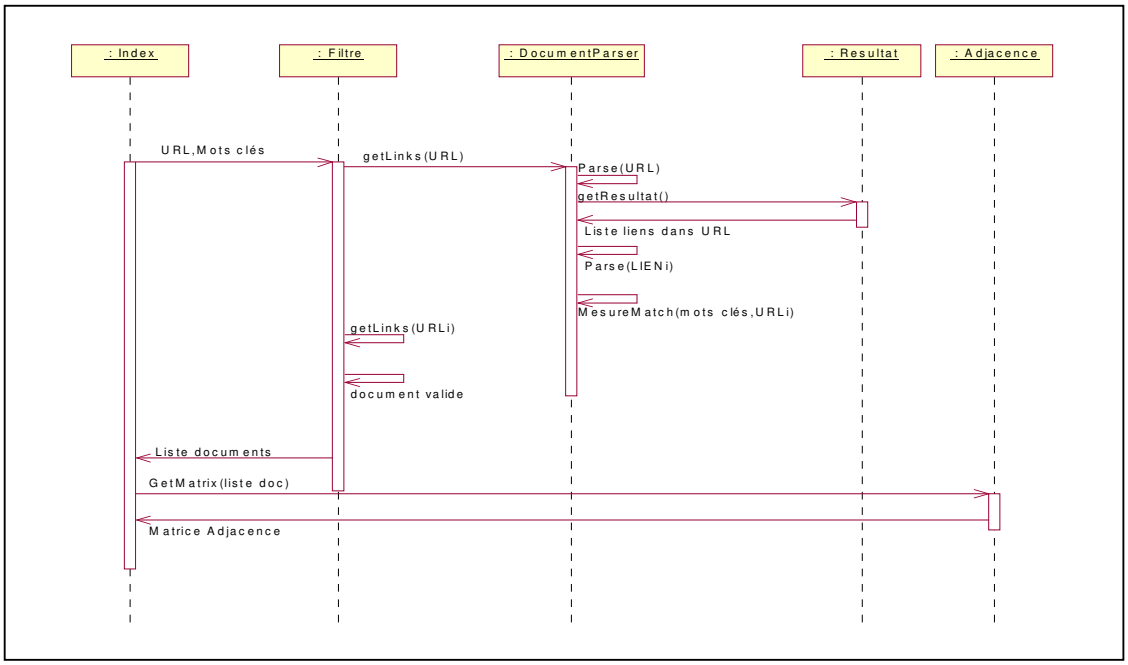


Figure 5.17 : Diagramme de séquences du module de crawlage stratégique

2.3 Conception et mise en œuvre de l'analyse de document Web

La conception UML du module d'analyse de document Web est faite via un package Java composé de plusieurs sous-packages dont les classes coopèrent tout au long du processus d'analyse. Pour illustrer l'architecture de ce module, nous présentons le diagramme de classes général (voir figure 5.18). Le fonctionnement de ce module est illustré par le diagramme des séquences de la figure 5.23.

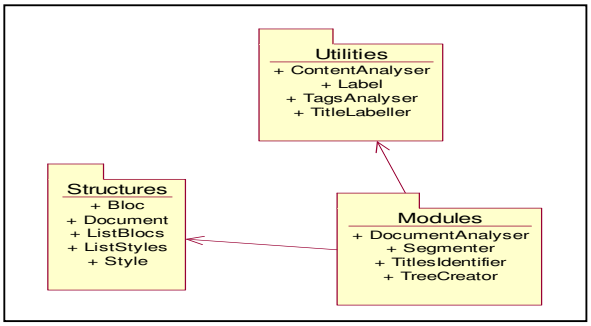


Figure 5.18 : Diagramme de classes général du module d'analyse de document Web

2.3.1 Diagramme de classes du processus de segmentation

Dans une première itération, nous avons examiné la structure de l'arbre DOM. En se basant sur ce résultat, nous avons développé la classe "*Segmenter*" qui correspond au processus de segmentation. La méthode "*getSegments*" de cette classe permet d'engendrer la liste des blocs et la liste des styles du document dans un seul parcours de l'arbre DOM. Nous avons aussi créé les classes du sous-package "*Structures*" et la classe "*TagsAnalyser*" du sous-package "*Utilities*" qui permettent d'analyser les balises et les attributs HTML.

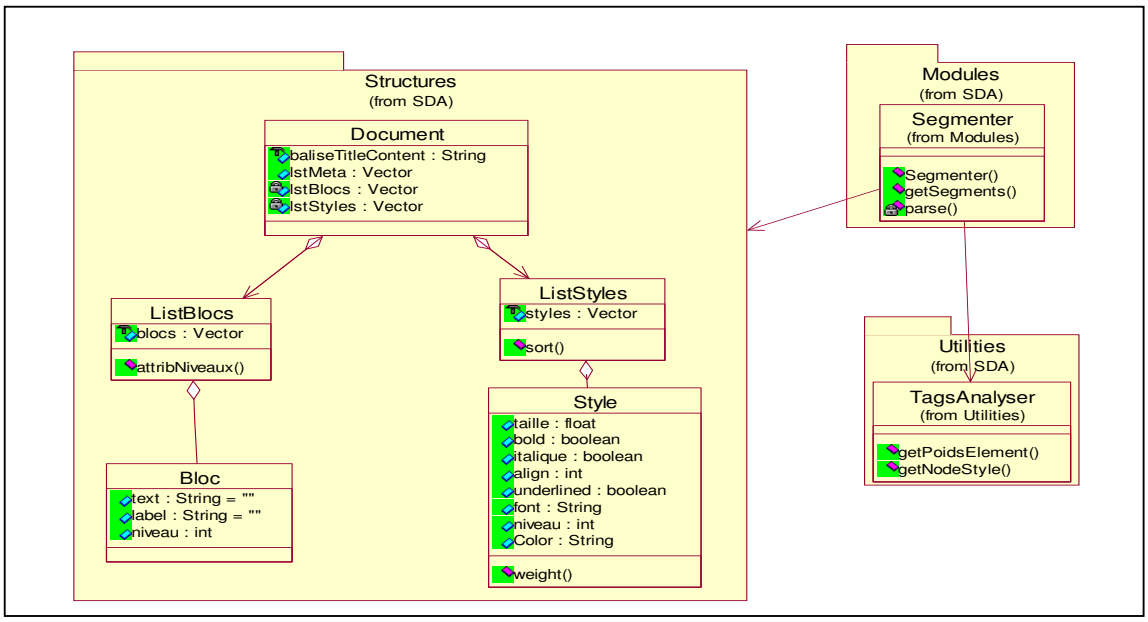


Figure 5.19 : Digramme de classes du processus de segmentation

Lors de la segmentation, les méta-données et le contenu de la balise "*title*" sont extraits. Nous avons analysé les résultats obtenus en effectuant des statistiques sur le contenu des balises "*meta*" et "*title*".

La figure 4.19 schématise le diagramme de classes de ce processus. La méthode "*parse*" parcourt récursivement l'arbre DOM en identifiant le type de chaque bloc et son style (il s'agit de l'étape d'étiquetage présentée dans la spécification de ce processus). Pour ce faire, elle fait appel aux méthodes "*getNodeStyle*" et "*getPoidsElement*" de la classe "*TagsAalyser*". La première permet de calculer le style d'un noeud et la deuxième identifie le poids du séparateur vertical engendré par une balise.

La classe "*Document*" est une structure de données qui contient la liste des blocs, la liste des styles, les méta-données et le contenu de la balise "*title*" du document. Toutes ces informations sont récupérées dans un seul parcours de l'arbre DOM.

2.3.2 Diagramme de classes du calcul des niveaux des styles

Nous nous sommes focalisés, dans la deuxième itération, sur le calcul des niveaux des styles puisqu'il s'agit d'un traitement essentiel pour l'identification des titres. L'idée de départ consistait à attribuer à chacun des attributs de style un poids et de calculer pour chaque style la somme pondérée des valeurs de ses attributs. Ayant testé plusieurs combinaisons de poids, nous avons découvert qu'il faut tenir compte de la régularité des titres et la fréquence des styles. Enfin, nous avons abouti à la solution présentée dans la spécification ci-dessus.

Dans cette itération, la méthode "*sort*" de la classe "*ListStyles*" permet d'attribuer des niveaux aux styles. Nous avons aussi créé la classe "*ContentsAnalyser*" dont la méthode "*canBeTitle*" permet d'attribuer l'étiquette "*PeutEtreTitre*" aux blocs en analysant leurs contenus. La figure 5.20 illustre le diagramme partiel de classes de cette itération.

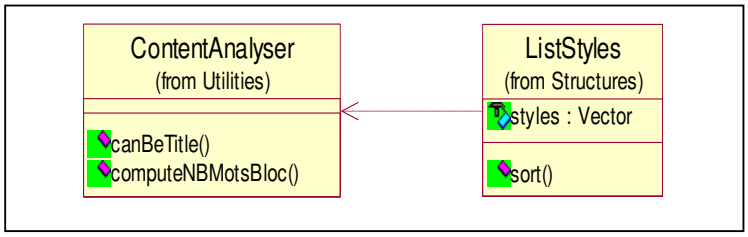


Figure 5.20 : Diagramme de classes du calcul des niveaux des styles

2.3.3 Diagramme de classes de l'étiquetage sémantique des blocs

L'objectif de cette itération est d'identifier les titres des sections "*non corps*" et les légendes. En effet, des étiquettes sont attribuées à ces blocs en se basant sur des expressions régulières. Pour chaque étiquette, nous avons défini une expression régulière apprise à partir des exemples de documents.

Ayant un bloc B et la liste des expressions régulières, nous calculons le taux de correspondance entre B et chaque étiquette. L'étiquette ayant le taux de correspondance le plus élevé sera sélectionnée. Nous attribuons cette étiquette à B si le taux de correspondance est supérieur à un seuil donné. Le taux de correspondance est calculé en identifiant la sous-chaîne du texte du bloc qui correspond à l'expression régulière. Le taux est égal au rapport entre la longueur de cette sous-chaîne et la longueur du texte du bloc.

L'étiquetage sémantique relève de la classe "*TitleLabeller*" du package "*Utilities*" qui définit les différents labels et expressions régulières. Il implémente la méthode "*mesure-Match*" permettant de comparer une chaîne de caractères à une expression régulière en se basant sur la classe "*RegularExpression*" appartenant à l'environnement de développement. Tel que présenté par la figure 5.21, la classe "*TitleLabeller*" définit plusieurs instances de la classe "*Label*" qui ont chacune un nom et une expression régulière associée.

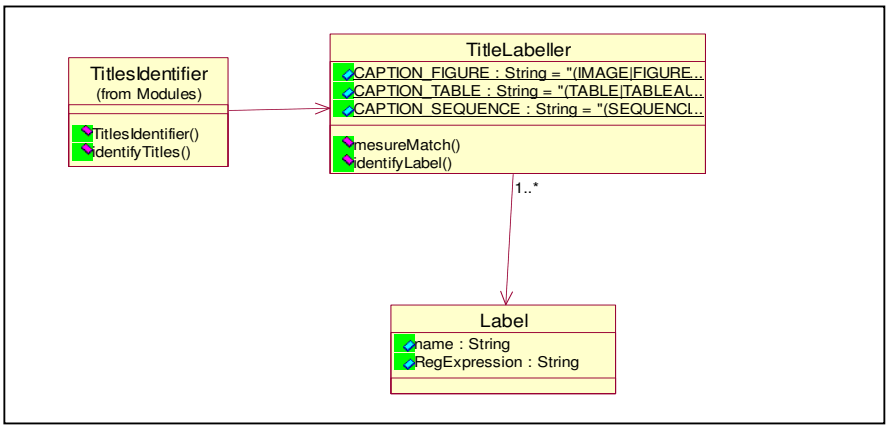


Figure 5.21 : Diagramme de classes de l'étiquetage sémantique des blocs

2.3.4 Diagramme de séquences du module d'analyse d'un document Web

Le diagramme de séquences que nous présentons dans la figure 5.22 illustre les grandes étapes du processus d'analyse d'un document Web. Il s'agit de mettre l'accent sur l'enchaînement des traitements et son partage entre les principaux modules sans présenter tous les messages échangés entre les classes.

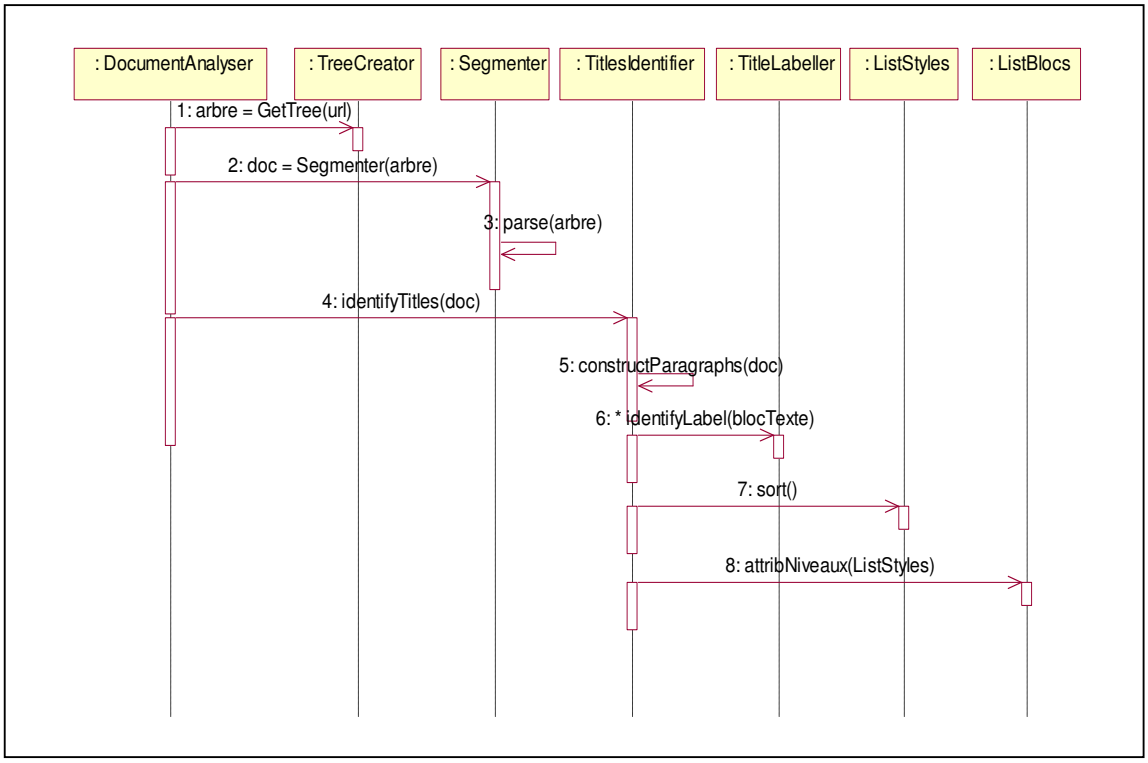


Figure 5.22 : Diagramme de séquences du module d'analyse d'un document Web

2.4 Conception et mise en œuvre du tri de documents par pertinence possibiliste

2.4.1 Diagramme de classes du module de tri par pertinence possibiliste

La figure 5.23 présente le diagramme de classes du module de tri de documents par pertinence possibiliste qui renferme les classes nécessaires pour le calcul des pertinences, en termes de possibilité et de nécessité, des documents déjà retournés par le module de crawlage.

A chaque document de la liste retournée par le module de crawlage, nous faisons correspondre une structure de donnée représentée par la classe *"struct_doc"* qui renferme toutes les structures logiques de ce document (paragraphe, titre principal, sous-titre, légende, figure, etc.) obtenues suite à un processus d'analyse, ainsi que les attributs stockant les calculs nécessaires pour les pertinences des documents en question.

La classe *"CalculPossibiliste"* renferme les méthodes contribuant à la construction de la nouvelle liste triée partant d'une collection de documents retournés dans un ordre quelconque. En effet, la méthode *"Construire_Tab_doc"* a pour tâche de lire chaque document de la collection et remplir le tableau *"Tab_doc"* dont chaque entrée est une structure *"struct_doc"*. La méthode *"Calcul_Occurrences"* permet de calculer pour chaque mot-clé ses occurrences dans chacune des structures du document, en utilisant la classe *"occurrence"*. Les calculs des

degrés de pertinences possibilistes de chaque document ainsi que les tris selon ces degrés sont réalisés par la méthode *"Trier_Tab_doc"*. Les choix des coefficients de pertinences possibilistes feront l'objet de la classe *"option"*.

Communiquant avec le module d'optimisation, la classe *"CalculPossibiliste"* retourne le résultat du tri directement s'il figure dans la base d'historique du système sans lancer de nouveau un processus de *crawlage*, mais il pourra faire un nouveau tri selon d'autres préférences.

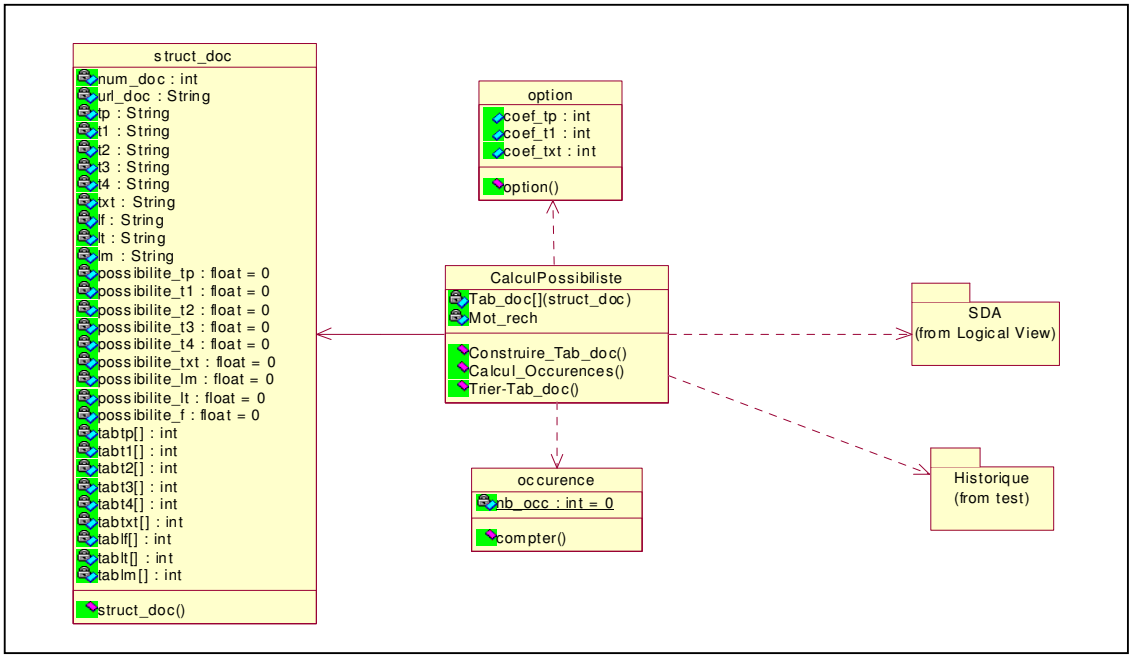


Figure 5.23 : Diagramme de classes du module de tri par pertinence possibiliste

2.4.2 Diagramme de séquences du module de tri par pertinence possibiliste

La figure 5.24 représente le diagramme de séquences du module de tri, des documents collectés, selon leurs degrés de pertinences possibilistes.

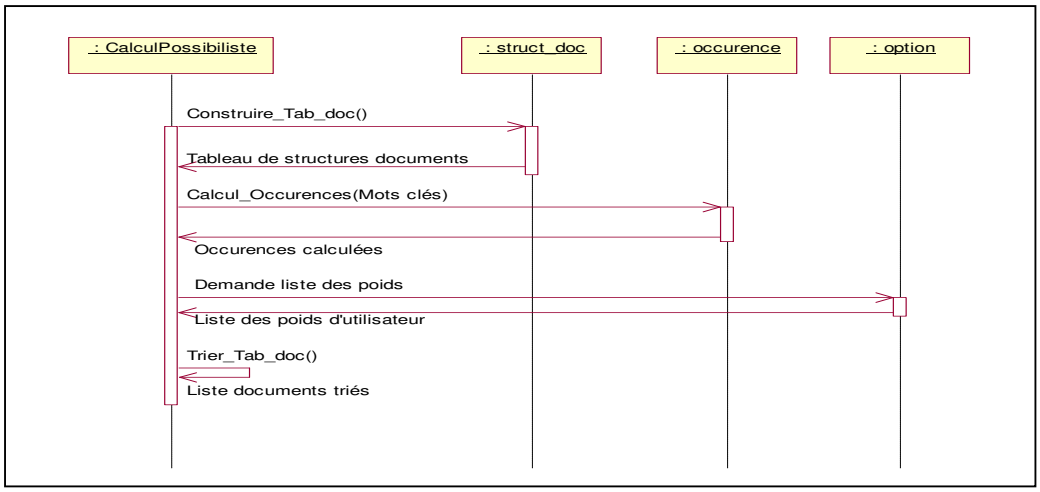


Figure 5.24 : Diagramme de séquences du module de tri par pertinence possibiliste

2.5 Conception et mise en œuvre du module d'optimisation

Comme le montre la figure 5.25, le module d'optimisation du système contient la classe *"Connection"* renfermant toutes les méthodes nécessaires à la gestion de la base de données et les opérations de mise à jour et de manipulation de la table Historique.

Les deux méthodes *"Insertion"* et *"Suppression"* sont responsables de la mise à jour des entrées de la table Historique. La méthode *"find"* cherche une entrée de la table et la méthode *"find_path"* a pour tâche l'extraction des informations stockées dans le fichier dont le chemin figure dans une entrée trouvée dans la table. L'authentification de l'utilisateur se connectant à la base, pour l'opération de suppression des enregistrements de la table, est contrôlée par la méthode *"isValidUser"*.

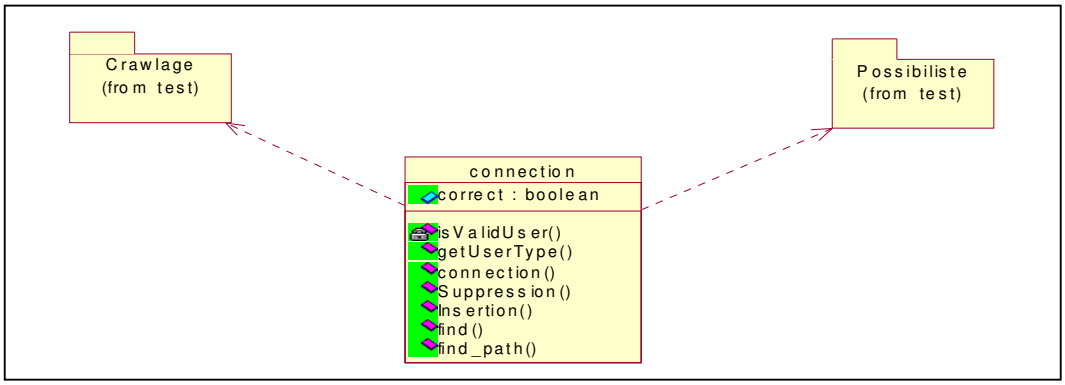


Figure 5.25 : Diagramme de classes du module d'optimisation

3. Conclusion

La spécification et la conception de notre système de recherche d'information que nous présentons dans ce chapitre répondent bien à notre problématique de départ présentée au début de cette thèse. En effet, le fait qu'on a affaire à des sources d'informations collectées à partir du réseau Internet, nous a fait opter pour le développement d'un *crawler* capable d'accéder à l'Internet grâce au nouvel algorithme de *crawlage* stratégique proposé. Il nous a paru également intuitif d'interfacer l'utilisateur au moyen d'interface permettant l'entrée et la sortir des informations au système. Enfin, le fait qu'on a affaire à des environnements ouverts et dynamiques nous a fait opter pour le développement de modules intermédiaires assurant les traitements sur les documents recherchés, à savoir l'analyse des documents Web, le calcul de leurs pertinences possibilistes et enfin leur tri selon les préférences de l'utilisateur du système.

Afin de réaliser l'architecture proposée pour le système SARIPOD, nous choisissons les Systèmes Multi-Agents (SMA) qui s'adaptent bien à des systèmes complexes et ouverts où il est difficile de tout prévoir à l'avance. Par ailleurs, notre étude des SRI a bien montré la complexité d'un tel système faisant intervenir des acteurs, des connaissances et des interactions multiples. Pour mettre ces connaissances en synergie dans un SRI, les SMA sont également adaptés pour la modélisation du comportement d'un SRI, ce dernier étant complexe.

Nous présentons dans le chapitre suivant la réalisation, l'expérimentation ainsi que l'évaluation du système SARIPOD tout en proposant des extraits de sa réalisation.

Chapitre 6

Réalisation et expérimentation du système SARIPOD

L'approche méthodologique suivie lors du développement du système SARIPOD est le prototypage. Il s'agit, en fait, d'un développement qui a pour objectif de démontrer la faisabilité de ce projet et de mettre en exergue l'importance et la convivialité de l'interface graphique et de la technique de *crawlage* stratégique dans la recherche d'informations sur Internet. De plus, ce développement doit permettre une certaine flexibilité pour constituer la pierre angulaire d'un grand projet de système multi-agent de recherche de documents qui serait hebergé par un serveur.

Pour toutes ces différentes considérations et compte tenu des informations manipulées par ce genre de système et de l'aspect parallèle du traitement, le développement du système SARIPOD doit permettre une certaine rapidité de traitement, une flexibilité et une portabilité. Par conséquent, le choix du langage Java s'est imposé étant donnée sa grande portabilité.

Une solution efficace pour satisfaire tous ces objectifs est l'utilisation des agents logiciels. Maes Patie [Maes, 1994] définit un agent logiciel comme étant un programme informatique autonome qui assiste l'utilisateur dans l'exécution de ses tâches et qui communique avec d'autres agents. Si en plus de ces caractéristiques l'agent peut manipuler des symboles ou des abstractions, s'il peut agir en temps réel, peut apprendre et peut s'adapter aux préférences de l'usager, nous parlons alors dans ce cas d'agents logiciels intelligents.

L'utilisation des agents logiciels pour la recherche d'informations offre certains avantages par rapport aux méthodes courantes telles que les moteurs de recherche. Le Tableau 6.1 récapitule ces avantages [Hermans, 1997].

Les expérimentations que nous avons réalisées concernent la phase de reformulation sémantique de requêtes et la phase de classification thématique des documents résultat d'une requête de recherche. Pour la première phase nous avons utilisé un dictionnaire de verbes extrait du dictionnaire français « Le Grand Robert ». Nous avons prouvé l'utilité de l'usage du RPMH de dictionnaire dans la reformulation de requête. Pour la deuxième phase de classification des documents nous avons utilisé comme base de test l'encyclopédie informatique libre nommée « *CommentCaMarche* » qui répond bien à nos besoins de test. Nous proposons également un cadre comparatif entre la classification proposée par le système et celle proposée par l'expert réalisateur de l'encyclopédie.

Nous commençons dans la première section par présenter le cadre de notre travail en terme d'environnement logiciel et la plate-forme multi-agent choisie. La deuxième section est consacrée à la définition des rôles et des intéractions des agents du système SARIPOD. La troisième section permet d'exposer l'implémentation du système proposé en présentant quelques extraits de sa réalisation. La quatrième section propose les résultats des expérimentations en ce qui concerne les deux phases de reformulation de requêtes et de classification des documents résultat de recherche.

	Les moteurs de recherche	Les agents
Critères de recherche	La recherche d'informations est faite en se basant sur un ou plusieurs mots-clés. Ceci suppose que l'usager est capable de formuler exactement ses mots-clés. Dans le cas contraire, plusieurs informations non pertinentes seront retournées et des informations pertinentes ne seront jamais retrouvées.	Les agents sont capables de chercher l'information d'une façon plus intelligente, par exemple en cherchant des concepts plutôt que des mots-clés. Les agents sont également capables de corriger les requêtes de l'usager, en se basant sur le modèle de ce dernier ou sur d'autres informations.
Indexation	L'indexation d'information est faite par collection de méta-informations sur les informations et sur les documents disponibles sur le Web. C'est une méthode coûteuse (en temps et en ressources), inefficace et qui ne correspond pas bien à la nature dynamique de l'Internet.	Les agents peuvent créer leurs propres bases de connaissances qui sont mises à jour après chaque recherche. Si l'information change de site, les agents sont capables de la trouver et, par la suite, s'adapter à ce changement. En plus, les agents sont capables de communiquer et coopérer entre eux (et c'est là leur vraie force), ce qui accélère et facilite la recherche.
Interface usager	La recherche d'information est souvent limitée à quelques services (WWW). Trouver l'information offerte par d'autres services (des bases de données) oblige souvent l'usager à se débrouiller seul.	Les agents peuvent débarrasser l'utilisateur de certains détails, comme la façon avec laquelle un service doit être manipulé. L'usager se concentre seulement sur ce qu'il cherche, l'agent s'occupe du reste.
Accessibilité	Les moteurs de recherche ne sont pas toujours accessibles, faute de connexion ou de congestion. L'usager sera alors obligé d'utiliser un ou plusieurs autres moteurs de recherche ce qui nécessitera probablement une autre façon de procéder.	Etant donné que l'agent réside sur la machine de l'utilisateur, il est toujours à la disposition de ce dernier. Un agent peut exécuter plusieurs tâches jour et nuit, et parfois même il pourra les exécuter en parallèle. L'avantage d'un tel agent réside aussi dans le fait qu'il est intelligent et qu'il peut par conséquent essayer d'éviter les heures de pointe.
Adaptabilité	L'information sur le réseau est très dynamique, souvent les moteurs de recherche font références à des informations dont la localité a changé. Les moteurs de recherche n'apprennent pas et ne s'adaptent pas aux usagers. En plus, l'utilisateur ne peut pas recevoir les mises à jour des informations. Faire de la recherche d'informations d'une telle façon est très coûteux.	Les agents s'adaptent aux préférences et aux souhaits de chaque usager. Ils peuvent ainsi apprendre de leurs recherches précédentes et par la suite comprendre mieux les besoins des utilisateurs.

Tableau 6.1 : Comparaison entre les moteurs de recherche et les agents logiciels

1. Cadre du travail

1.1 Environnement Logiciel

La facilité d'intégration et la réutilisation sont les principales caractéristiques de notre système. Le développement d'un paquetage Java intégrable dans toute plate-forme ou application (Windows, Linux, Web) était l'objectif principal de la tâche d'implémentation. Pour ce faire, nous avons choisi l'environnement Borland JBuilder Entreprise 10.0.176.120. Outre les avantages de la technologie orientée objet, ce dernier possède plusieurs qualités relativement à nos besoins. En effet, l'intégration de composants logiciels dans les applications développées dans cet environnement est simplifiée. D'une part, JBuilder nous a permis d'intégrer deux composants logiciels fondamentaux pour notre architecture. Il s'agit des cinq paquetages : un pour le RPMH de dictionnaire, un pour le RPMH de pages Web, un pour le traitement possibiliste, un pour l'analyse des documents Web collectés et un pour

l'applet graphique 3D. Une fois intégrés, nous avons pu structurer et adapter ces paquetages proprement aux besoins de notre système. D'autre part, cet environnement permet le développement d'un paquetage JAR intégrable que ce soit dans des applications monoposte ou des applications Web. Nous rappelons que l'un des objectifs du présent travail est de contribuer à l'amélioration des performances des systèmes de recherche d'information sur Internet.

La conception des diagrammes de classes et de séquences a été réalisée avec le langage de modélisation UML via le software Rational Rose Enterprise Edition 2003.

1.2 La plate-forme multi-agent Jade

Afin de profiter des travaux d'autres chercheurs dans le domaine et pour la rapidité de prototypage nous avons eu recours aux plates-formes existantes de développement de systèmes multi-agents. L'étude réalisée au sein du laboratoire RIADI par [Ben Mena et al., 2005], nous a servi dans le choix d'une plate-forme multi-agent convenable pour le développement du système SARIPOD. En effet, le choix de la plate-forme JADE s'avère raisonnable. De plus, JADE sera aussi adaptée pour un déploiement sur serveur.

Ce choix est le résultat d'une comparaison entre la plate-forme JADE et d'autres plates-formes, essentiellement : DECAF, AGENTBUILDER, ZEUS, JAFMAS/JIVE, JACK, AGENTTOOL, MADKIT, SWARM et STARLOGO. Les critères de comparaison retenus sont :

1. *Méthodologie associée à l'outil :* La plate-forme doit associer une méthodologie couvrant les différentes étapes du cycle de vie du développement d'un SMA.
2. *Facilité d'implémentation et de déploiement :* Pour réaliser un système de recherche d'information sur Internet, il faut utiliser un langage de programmation de haut-niveau supportant la programmation Orientée-Objet. Dans ce cas la programmation des « threads » et leur synchronisation où l'échange de message est aussi indispensable. D'autre part l'implémentation des communications doit être transparente.
3. *Interface graphique pratique et multifonction :* La plate-forme doit posséder une interface utilisateur qui facilite le développement. La plate-forme doit permettre la visualisation des agents ainsi que leur gestion et celle des interactions avec le système.
4. *Réutilisation simple :* Parmi les objectifs fixés, nous avons mentionné l'extension du système pour d'autres domaines proches ce qui nécessite de prendre en compte ce critère.
5. *Possibilité de suivi et de déboguage :* Nous avons réalisé un système multi-agent de recherche d'information sur Internet avec une forte coopération entre les acteurs de ce système. De plus, le nombre de composants est très important. Des outils de suivi et de déboguage sont ainsi nécessaires.
6. *Connexion à d'autres composants :* Notre système est connecté à une base de données enregistrant les requêtes déjà jouées par ce système afin de faciliter la gestion de l'historique et d'améliorer le temps de réponse de notre système.
7. *Possibilité de distribution :* Parmi les besoins identifiés pour l'extension de notre système on s'intéresse à la distribution de la recherche d'information sur des machines distantes afin de rendre la recherche plus efficace et permettre la coopération entre les hommes et les machines.

8. *Disponibilité de documentation :* La documentation est disponible dans JADE non seulement pour l'étape de développement mais aussi pour l'étape de déploiement, pour pouvoir remédier au problème de maintenance et d'exécution.
9. *Standard :* La plate-forme JADE est conforme à un standard pour une interopérabilité avec des agents hétérogènes et interactifs et des systèmes multi-agents.
10. *Accès au code source :* La plate-forme JADE est accessible ainsi que son code source. Ce qui lui permet une flexibilité d'extension.
11. *Portabilité :* L'outil JADE est portable sur différents environnements et permet aussi une exécution simple du système indépendamment de l'environnement.

En effet, à partir du résultat de la comparaison, la plate-forme JADE a montré une nette supériorité pour l'ensemble de ces critères.

Par ailleurs, Jade est un outil qui répond aux normes FIPA 97. Il fourni des classes qui implémentent « JESS[29] » pour la définition du comportement des agents. L'outil possède trois modules principaux (nécessaire à la norme FIPA[30]). Le DF (Director Facilitator) fourni un service de pages jaunes à la plate-forme. Le ACC (Agent Communication Chanel) gère la communication entre les agents. Le AMS (Agent Management System) supervise l'enregistrement de chaque agent, son authentification, son accès au système et son utilisation. Les agents communiquent par le langage FIPA ACL. Un éditeur est disponible pour l'enregistrement et la gestion des agents. Aucune autre interface n'est disponible pour le développement ou l'implémentation ce qui nécessite une bonne connaissance des classes et des différents services offerts [Bellifemine et al., 2003].

2. Les agents du système SARIPOD

L'idée de départ pour les SRI est de distribuer les connaissances pour parer aux problèmes de capacité de stockage, de cohérence de la masse d'information traitée et de complexité de résolution due à cette masse d'information. Les systèmes multi-agents (SMA) sont les plus représentatifs de cette catégorie.

Par ailleurs, et devant la diversité des connaissances et la complexité du processus de recherche, il est fort intéressant d'adopter la représentation sous forme d'agents qui vont être soit représentatifs des opérations de mises en œuvre, soit détentir les connaissances appropriées. Nous avons donc choisi un certain nombre d'agents qui participent tous, chacun selon sa compétence propre, à la concrétisation de l'objectif global qui est de satisfaire le besoin d'information de l'utilisateur. Ce but peut être considéré suivant plusieurs angles pour constituer des sous-buts, et ce, selon la nature de besoin. Pour chaque sous-but, une stratégie de recherche appropriée, qui doit s'adapter à la nature de besoin, peut être suivie. La définition d'une stratégie de recherche consiste à choisir quels sont les agents qui seront activés au cours de la résolution du problème.

Pour introduire les différentes connaissances, nous avons choisi d'utiliser trois couches d'agents, dont chacune est réservée à certains agents soit pour récupérer ou pour y stocker des connaissances. Les agents que nous proposons interagisssent et coopèrent selon le rôle qui leur est confié pour atteindre l'objectif commun.

[29] http://www.jessrules.com/
[30] http://www.fipa.org

L'organisation du système proposé (affectation des rôles des agents, interactions et coopérations) ainsi que les différentes connaissances sont détaillées et illustrées par des schémas dans les sections suivantes.

2.1 Les couches d'agents du SARIPOD

Le fait qu'on a affaire à des sources d'informations collectées à partir du réseau Internet, nous a fait opter pour le développement d'agent crawler capable d'accéder à l'Internet. Il nous a paru également intuitif d'interfacer l'utilisateur au moyen d'agents d'interface. Finalement, et comme nous l'avons souligné plus haut, le fait qu'on a affaire à des environnements ouverts et dynamiques nous a fait opter pour le développement d'une couche d'agents intermédiaires. On voit donc apparaître trois niveaux d'abstraction au niveau de l'architecture multi-agent abstraite du système SARIPOD :

- La couche de communication avec l'utilisateur ;
- La couche de traitement d'informations ;
- La couche d'extraction d'informations.

Dans le système SARIPOD, les trois couches font référence à la technologie agent et à la médiation entre ces mêmes agents (voir figure 6.1).

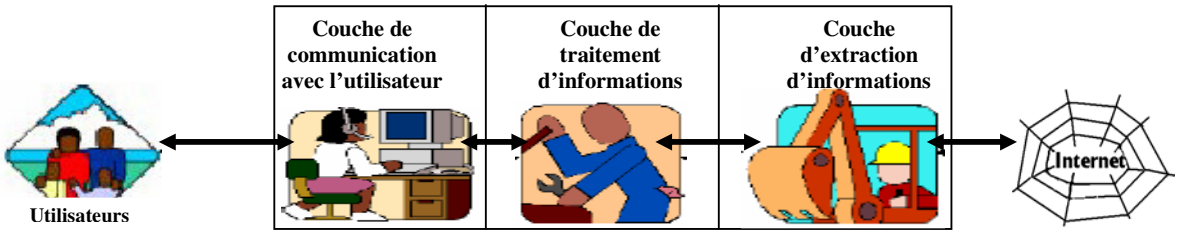


Figure 6.1 : Les couches abstraites du système SARIPOD

Au niveau du contenu précisément, les trois couches se définissent comme suit :

2.1.1 Couche de communication avec l'utilisateur

Cette couche est chargée des communications entre SARIPOD et l'usager. Elle comprend des agents d'interface interagissant avec l'utilisateur pour l'aider à réaliser une tâche bien précise. Cette interaction se traduit par une transformation des requêtes de l'usager afin de faciliter la communication avec les agents de la couche de traitement. Cette couche vérifie également la consistance des données fournies par l'utilisateur.

2.1.2 Couche de traitement d'informations

Cette couche de traitement d'informations reçoit de la couche de communication les requêtes à reformuler ainsi que les préférences de l'utilisateur. Elle détermine, à partir du RPMH de dictionnaire, les mots les plus proches des mots-clés de l'utilisateur et permet, en conséquence, de reformuler ses requêtes via un agent lexicographique interagissant avec le RPMH du dictionnaire. Cette couche fournit également la définition d'un agent page Web donnant la structure logique de chacune de pages Web recherchées, celle d'un agent d'historique enregistrant toutes les requêtes et leurs réponses dans une base d'historique, celle d'un agent de mesures possibilistes (mesure de possibilité, mesure de nécessité, mesure de pertinence possibiliste) ainsi que celle d'un agent sélectionneur permettant l'organisation des pages Web, retournés par la couche d'extraction d'informations, selon les préférences de l'utilisateur. En fait, l'assistance globale dans cette couche est assurée par des agents superviseurs (décideur, médiateur et contrôleur d'erreur).

2.1.3 Couche d'extraction d'informations

Cette couche est composée uniquement d'un agent crawler assurant l'exploration (crawlage) du Web pour sélectionner les pages Web contenants les mots-clés recherchées. En effet, cet agent forme une interface entre la source d'informations (le réseau Internet) et la couche de traitement d'informations.

Nous pouvons travailler avec plusieurs agents crawler. En effet, comme pour les fourmis, s'il y a des zones du Web peu intéressantes, un seul agent peut suffire, par contre si on tombe sur un ensemble riche en pages potentiellement pertinentes, plusieurs agents pourraient travailler en parallèle. En fait, les pages Web pertinentes sont comme la nourriture, elles devraient attirer beaucoup d'agents. Si on vise des millions de pages Web, il n'est pas crédible qu'un seul agent soit dévolu à cette tâche. Mais l'inconvénient majeur de ce type de système est de constituer un goulot d'étranglement qui peut diminuer considérablement les performances du système dès que le nombre des agents et des demandes augmente [Ferber, 1995].

2.2 Rôle des différents agents

La coopération entre les différents agents du système SARIPOD est représentée par la figure 6.2. Ces agents sont répartis sur les trois couches ci-dessus dans des concentrations variables. Ces différents agents sont : agent utilisateur ; agents d'interface ; agent lexicographique ; agent de mesures possibilistes ; agent sélectionneur ; agent page Web ; agent crawler ; agent d'historique; agents superviseurs [Elayeb et al., 2007b].

Agents utilisateurs

L'agent utilisateur est la porte d'entrée des requêtes externes au système. Il facilite à l'utilisateur la formulation de sa requête proposée au système. L'agent utilisateur est capable de garder les préférences de l'utilisateur au fur et à mesure que celui-ci utilise le système. En effet, il enregistre dans une base d'historique les requêtes déjà jouées par un utilisateur ainsi que les préférences correspondantes dans le but de pouvoir les utilisées ultérieurement. Il est capable aussi de stocker de l'information pour l'utilisateur et d'agir comme un agent ressource. Évidemment il y a autant d'agents utilisateurs qu'il y a d'utilisateurs. Chaque agent s'occupe de l'utilisateur auquel il est rattaché.

Agents d'interface

Ils ont pour rôle d'assurer la communication entre le système et ses utilisateurs. Ils sont de deux types :

a. Agent d'entrée

L'agent d'entrée analyse la requête utilisateur et transmet par la suite les mots-clés recherchés à l'agent lexicographique qui détermine leurs mots proches à partir du RPMH de dictionnaire de mots.

b. Agent de sortie

L'agent de sortie est chargé de présenter les résultats de la recherche à l'utilisateur. En effet, il est capable de confronter les résultats de sortie aux préférences de l'utilisateur. Cette confrontation nécessite la présentation des pages Web recherchées selon les préférences proposées, dans le cas où le résultat fourni par l'agent sélectionneur est différent de celui préféré par l'utilisateur.

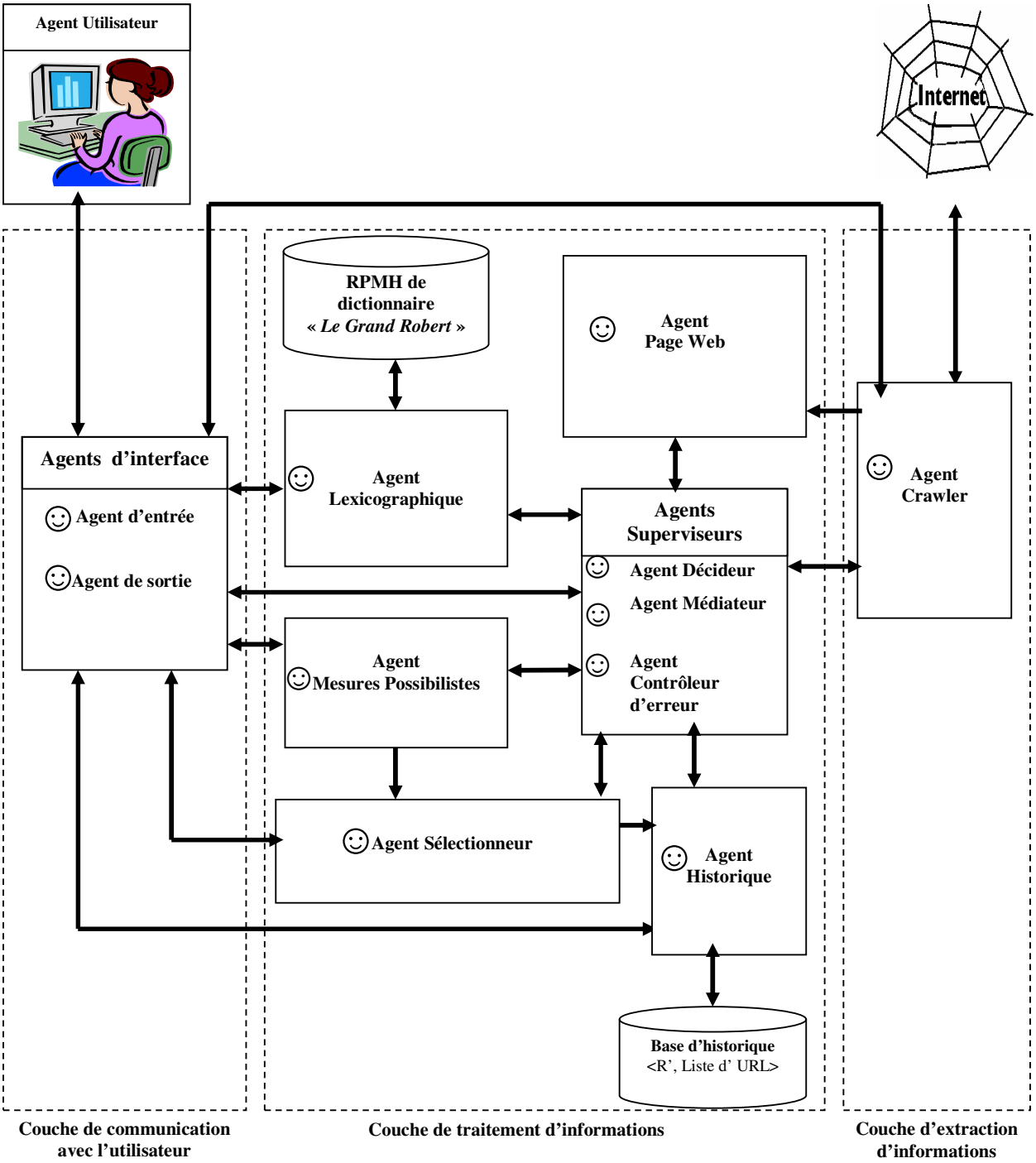


Figure 6.2 : La coopération entre les agents de SARIPOD

Agents superviseurs

Ils veillent au bon fonctionnement du système, tous les autres agents doivent être à leur service et sous leur responsabilité. Ils sont chargés d'affecter les tâches aux différents agents intervenant dans le processus de recherche d'informations (agent médiateur), de décider en cas d'une multitude de choix (agent décideur) et de contrôler les erreurs possibles lors d'une

session de sélection des documents Web les plus pertinents (agent contrôleur d'erreur). Nous détaillons dans la suite la tâche de chacun de ces trois agents.

a. Agent médiateur

Il a pour rôle d'affecter les différentes tâches de recherche aux agents appropriés. Un problème de recherche d'information peut être décomposé en plusieurs tâches à savoir :

1. L'entrée et l'analyse de la requête utilisateur de recherche d'information ;
2. L'interrogation du RPMH du dictionnaire via l'agent lexicographique qui détermine les mots les plus proches (synonymes) des mots-clés de l'utilisateur du système par l'application de l'approche à base de circuits existants entre les nœuds termes ;
3. La transformation des URLs crawlées en des pages Web et la détermination de leurs structures logiques par l'agent page Web;
4. La détermination de la pertinence de chaque document via l'agent de mesures possibilistes ;
5. L'organisation de ces documents dans un ordre décroissant de pertinence via un agent sélectionneur;
6. La sortie du résultat final de la recherche dans une représentation conforme aux préférences de l'utilisateur ;
7. Le stockage d'une copie du résultat final dans une base d'historique via un agent d'historique.

Toutes ses tâches sont réalisées par les différents agents du système. En effet, l'agent crawler explore le Web pour extraire les URLs des pages Web recherchées. L'analyse des pages Web correspondantes à ses URLs est faite par l'agent page Web.

L'agent médiateur permet de planifier ces différentes tâches et les affecter aux différents agents du système, c'est un rôle moteur qui peut facilement être limité dans le cas où le système devient complètement distribué ; c'est-à-dire le nombre d'agents médiateurs est inversement proportionnel au degré de cognition des autres agents du système. Dans cette version du système SARIPOD, l'agent médiateur-facilitateur joue le rôle d'un facilitateur.

L'agent médiateur (ou facilitateur) permet l'allocation des tâches, il dispose de l'information à propos des compétences d'autres agents. En fait, l'intérêt principal de cette architecture est de favoriser la cohérence du système. De plus le besoin d'optimisation est plus facilement satisfait. Connaissant l'ensemble des agents disponibles, il est plus facile à l'agent médiateur de choisir le "meilleur" des agents par rapport à une demande de tâche donnée. Mais l'inconvénient majeur de ce type de système est de constituer un goulot d'étranglement [Zaghdoud, 2003] qui peut diminuer considérablement les performances du système dès que le nombre des agents et des demandes augmente [Ferber, 1995].

En effet, pour le cas de la présente application, il est préférable d'utiliser un agent superviseur (ou médiateur), d'ailleurs le nombre d'agents est limité et le risque d'avoir un goulot d'étranglement est minime. Par contre, pour un développement réel du système SARIPOD l'aspect totalement distribué devient nécessaire.

b. Agent décideur

Cet agent décideur a un rôle fondamental dans le système SARIPOD. Dans un premier temps, il est chargé de faire une sélection post-traitement après avoir mené à terme les différentes pages Web sélectionnées par les agents sélectionneurs pour que les agents de sortie sachent organiser ce résultat dans l'ordre préféré par l'utilisateur. D'autre part, cet agent décideur sera doté d'une intelligence pour faire un prétraitement des documents Web pertinents, lui permettant ainsi de faire gagner au système un temps considérable.

b. Agent contrôleur d'erreur

Il est chargé de contrôler le bon fonctionnement du système en exécutant les directives de contrôle des erreurs communiquées par chaque agent du système. Il informe le décideur de ce qui se passe dans le système qui à son tour décide d'arrêter ou non un agent. Souvent, il analyse la cause d'erreur de chaque agent en difficulté, s'il s'agit par exemple d'un manque d'information, il essaye de résoudre ce problème en demandant plus d'information auprès de l'agent source d'erreur. Dans le pire des cas, il décide d'arrêter le fonctionnement d'un agent.

Agent lexicographique

Dans le cadre de la reformulation de la requête utilisateur, l'agent lexicographique veille sur la construction et l'interrogation du RPMH du dictionnaire de mots en vue de déterminer les mots sémantiquement proches des mots-clés proposés par l'utilisateur du système.

Cet agent interagit avec les agents d'interface pour décider la requête finale à proposer au crawler à travers les agents superviseurs.

Agent crawler

L'objectif de cet agent est de pouvoir *crawler* le Web selon la stratégie décrite dans le chapitre précédent. Il obtient un ensemble des URLs des pages Web, dont chacune contient un ou plusieurs mots-clés de la requête reformulée. Par ailleurs, cet agent se charge aussi de la création du RPMH de pages Web ainsi que leur classification sous forme de petits mondes. En fait, dans notre prototype, l'agent crawler interagit avec l'agent page Web via l'agent de mesures possibilistes pour permettre à l'agent sélectionneur de trier les pages Web sélectionnées selon leurs degrés de pertinences possibilistes (les tâches des ces agents seront détaillées dans la suite).

Agent page Web

L'agent page Web est chargé de l'analyse de pages Web collectées par l'agent crawler. La structure logique de chaque page Web est envoyée par cet agent vers l'agent de mesures possibilistes.

Agent de mesures possibilistes

Cet agent s'occupe du réseau possibiliste du système SARIPOD. Il calcul le scores de pertinence possibiliste de chaque page Web sélectionnée par l'agent crawler, en se basant sur la structure logique de chacune de ses pages. Ces scores seront acheminés vers l'agent sélectionneur qui décide leur organisation selon les préférences de l'utilisateur.

Agent sélectionneur

L'agent sélectionneur est capable de répondre à des propositions du type : le document d_1 est préférable au document d_2 ou l'ensemble $\{d_1, d_2\}$ est préférable à l'ensemble $\{d_3, d_4\}$. En effet, cette proposition montre bien que la liste ordonnée des documents en réponse à un besoin utilisateur est traitée d'une manière qualitative, et que notre approche qualitative ordinale est utilisée dans la représentation des documents et des requêtes.

Par ailleurs, cet agent trie les documents Web dans un ordre décroissant de leurs degrés de pertinences possibilistes ; le document répondant le plus aux préférences de l'utilisateur sera affiché en tête de la liste triée des documents, retourné à l'agent de sortie qui vérifie sa conformité aux préférences de l'utilisateur.

Agent d'historique

Cet agent stocke une copie de chaque requête jouée par le système ainsi que sa réponse dans une base d'historique. En cas où l'utilisateur réjoue une requête, cet agent lui propose celle de l'historique et lui donne aussi la possibilité de changer ses préférences et relancer des nouveaux calculs des scores des pertinences possibilistes des documents Web.

Ainsi, le système SARIPOD bénéficie des avantages apportés par la richesse de la modélisation multi-agent, faisant coopérer les différentes tâches, et par les méthodes utilisées par les agents et particulièrement les agents de création des deux RPMH et du Réseau Possibiliste. D'autre part, nous avons proposé un système de communication entre les différents agents permettant de synchroniser leurs comportements et leurs actions sur l'ensemble des connaissances. Ces communications sont assurées via des messages qui rendent les agents plus indépendants. La figure 6.3 montre quelques communications par messages échangés entre les différents agents de la plate-forme SARIPOD.

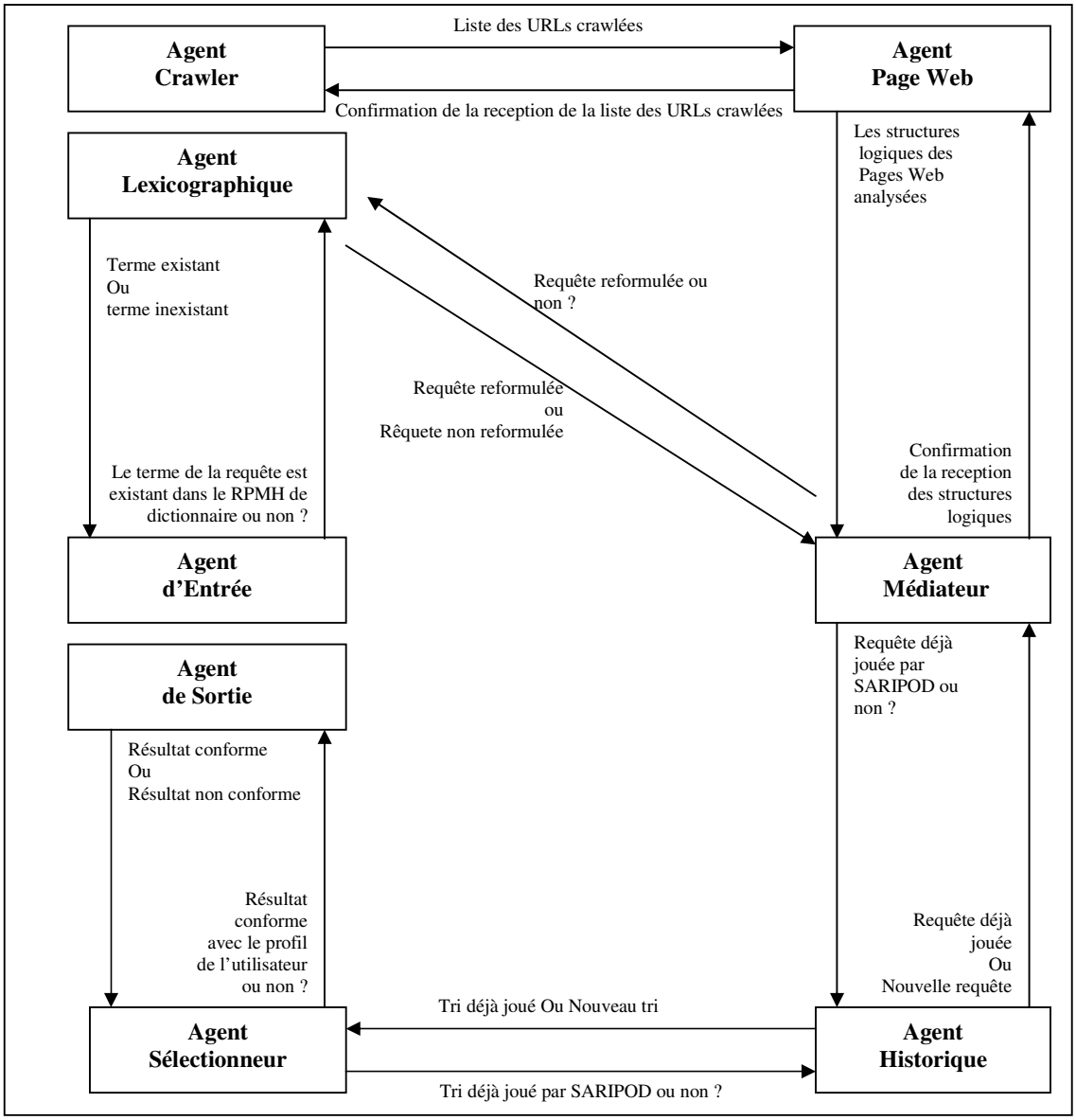


Figure 6.3 : Communications par messages échangés entre les agents de SARIPOD

3. Implémentation du système SARIPOD

Comme toute autre application le système SARIPOD est intégré sous la forme d'un package Java dans la plate-forme Jade. Les classes agents héritent leurs propriétés et leurs méthodes des classes de base. Nous présentons dans la suite l'implémentation de chacun des modules du système et nous présentons quelques extraits de la réalisation. La figure 6.4 présente l'interface Jade du système SARIPOD.

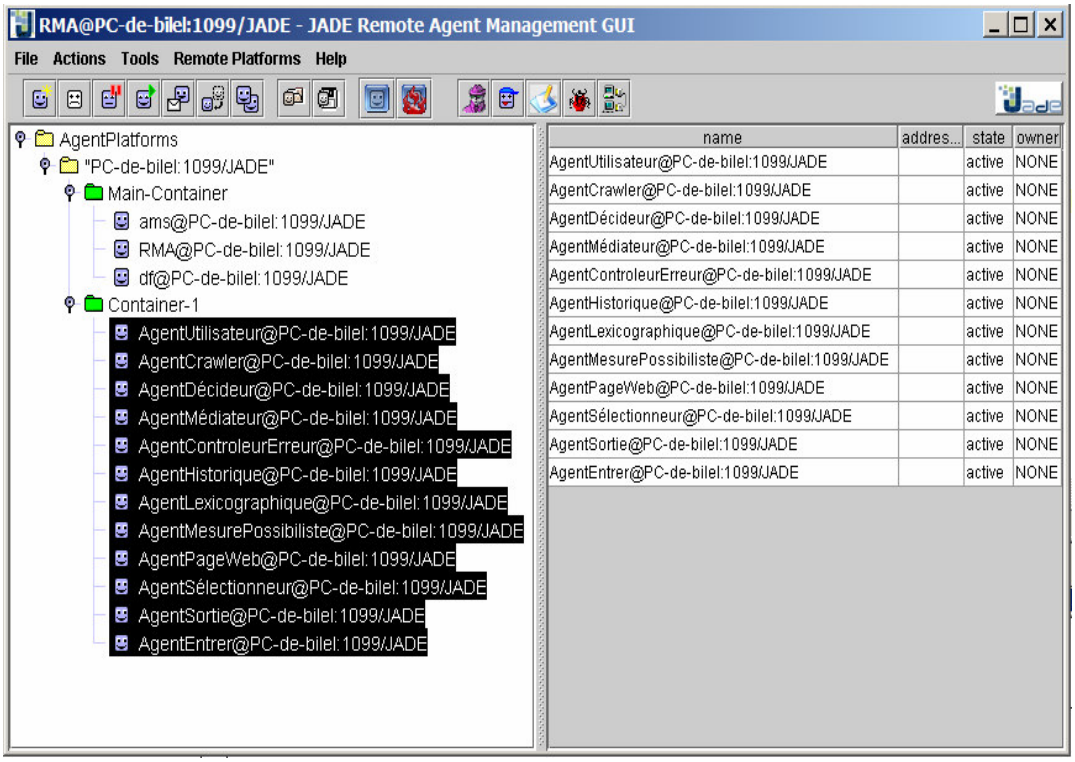


Figure 6.4 : Interface Jade du système SARIPOD

3.1 Interfaces principales du SARIPOD

Nous avons regroupé toutes les fonctionnalités utiles de notre système SARIPOD dans une seule interface graphique conviviale et interactive. Ainsi, l'interface générale comprend les cinq onglets suivants (voir figure 6.5).

1- Onglet « RPMH de pages Web » : cet onglet sert à afficher les URLs contenant les mots-clés recherchés suivant la technique de *crawlage* déjà spécifiée. En effet, l'utilisateur insère son URL de départ ainsi qu'un ensemble de mots-clés. Il demande la reformulation de sa requête via le bouton « Ajouter Synonyme ». Le système demande à chaque itération le nombre de synonymes désirés par l'utilisateur pour chaque mot-clé. Cette technique est utile et permet au système de détecter les préférences entre les mots-clés de l'utilisateur. Le processus de recherche est lancé par le bouton « Rechercher ». Le bouton « Vider la Base » permet de vider la base d'historique. Le bouton « Arrêter » permet de suspendre le système. Le bouton « initialiser » permet d'initialiser le système. Le bouton « RPMH des pages Web » permet la construction du RPMH des pages Web par l'approche des circuits. Une visualisation 3D de cet RPMH est assurée par le bouton « RPMH 3D ». Le bouton

« Quitter » permet de fermer le système. Finalement le système permet l'affichage des informations à propos du temps de réponse ainsi que du nombre total des documents retrouvés.

2- Onglet « Pertinence Possibiliste» : cet onglet permet de trier les documents collectés selon leurs pertinences possibilistes. En effet, l'utilisateur validera son profil par le bouton « Préférences » de l'interface de la figure 6.6. Il pourra enfin, enregistrer le résultat final (voir figure 6.7) sous n'importe quel format (.txt ; .doc ; .pdf ; .html ; etc.) en appuyant sur le bouton « Enregistrer ».

3- Onglet « Matrice Index» : cet onglet sert à afficher la matrice Index du RPMH de pages Web. En effet, l'utilisateur peut afficher la matrice index nécessaire à la construction du RPMH de pages Web via l'interface de la figure 6.8. Il pourra, en outre, enregistrer cette matrice sous n'importe qu'elle format (.txt ; .doc ; .pdf ; .html ; etc.). Une copie de ce fichier sera automatiquement stockée dans le dossier d'historique du système.

4- Onglet « Matrice Adjacence» : cet onglet sert à afficher les proximités entre les pages Web selon la méthode présentée dans le chapitre 4. En effet, cette matrice ne pourra être engendrée qu'à partir de la matrice d'index. De la même façon que la matrice Index, l'utilisateur pourra l'enregistrer sous n'importe quel format et une copie de ce fichier sera automatiquement stockée dans le dossier d'historique du système (voir figure 6.9).

5- Onglet « RPMH de dictionnaire » : cet onglet permet de déterminer les synonymes d'un mot donné en paramètre via le graphe du RPMH de dictionnaire.

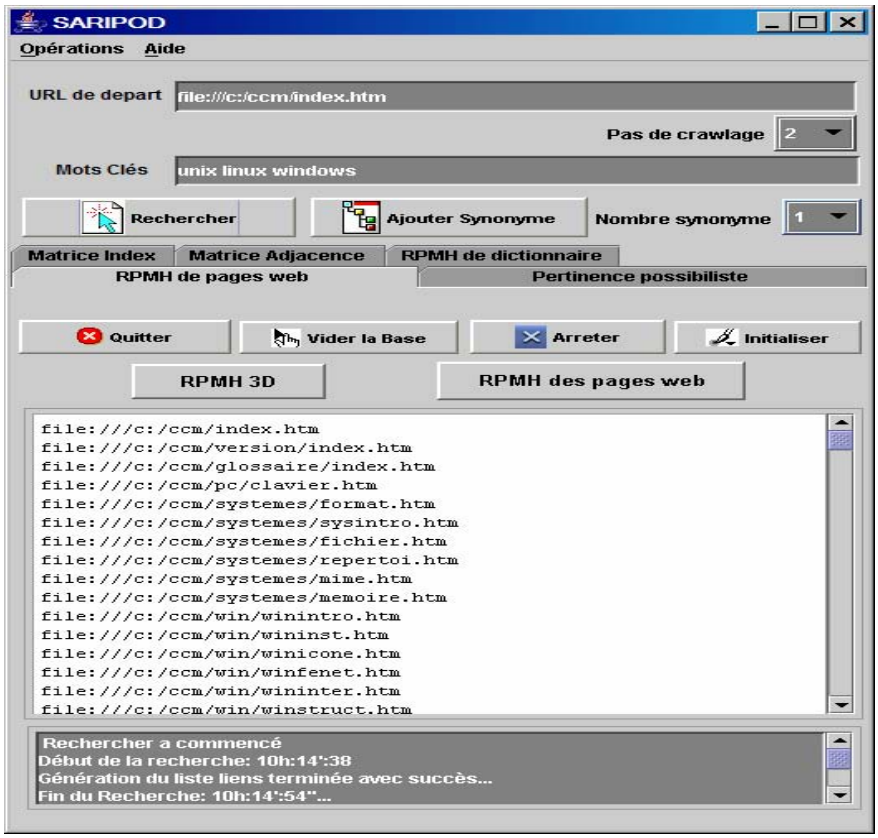


Figure 6.5 : Interface générale du système SARIPOD

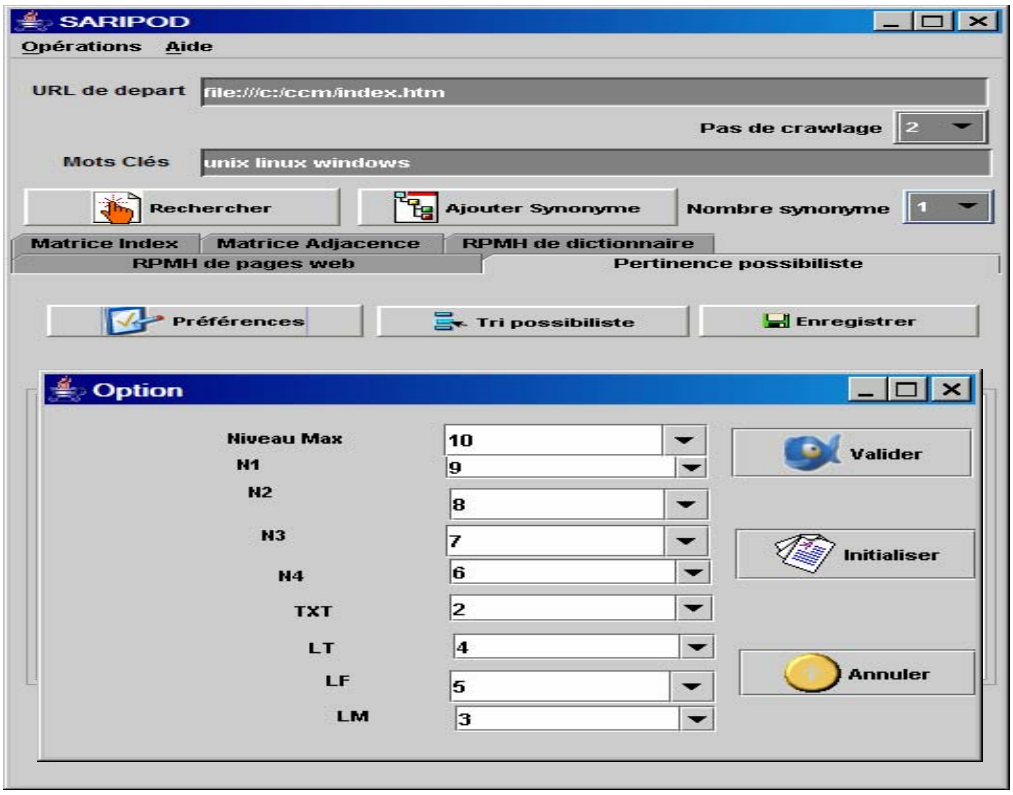


Figure 6.6 : Interface de paramétrage des coefficients de pertinence possibiliste

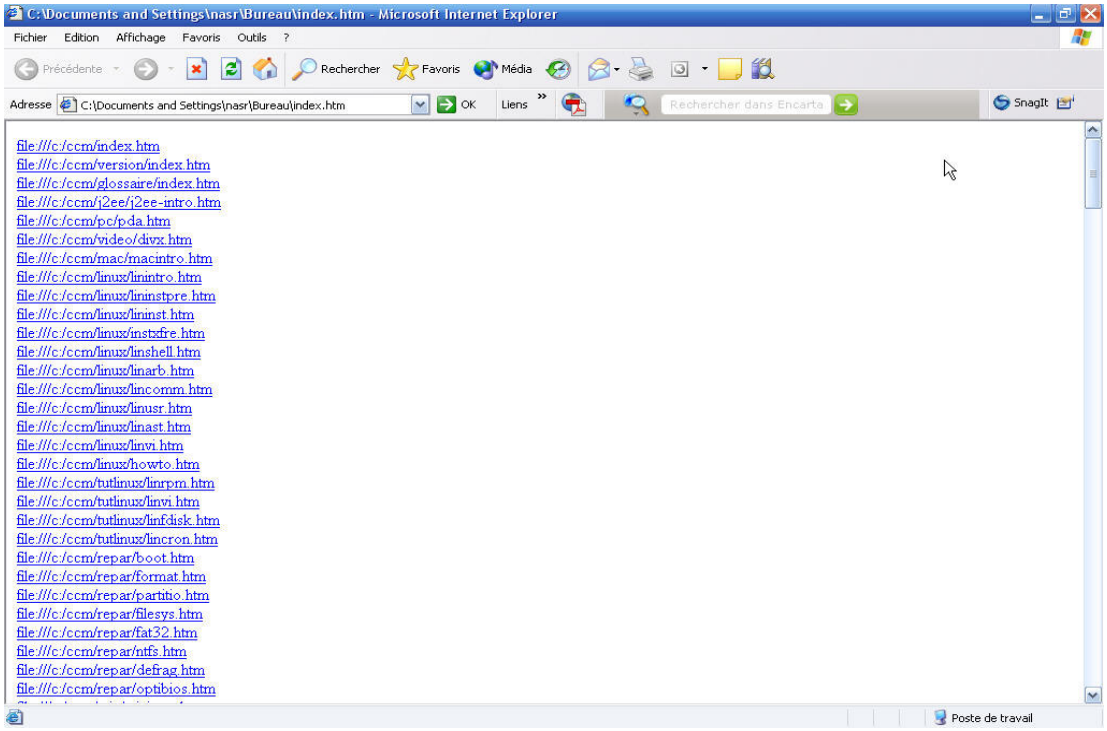


Figure 6.7 : Interface du fichier résultat du système SARIPOD

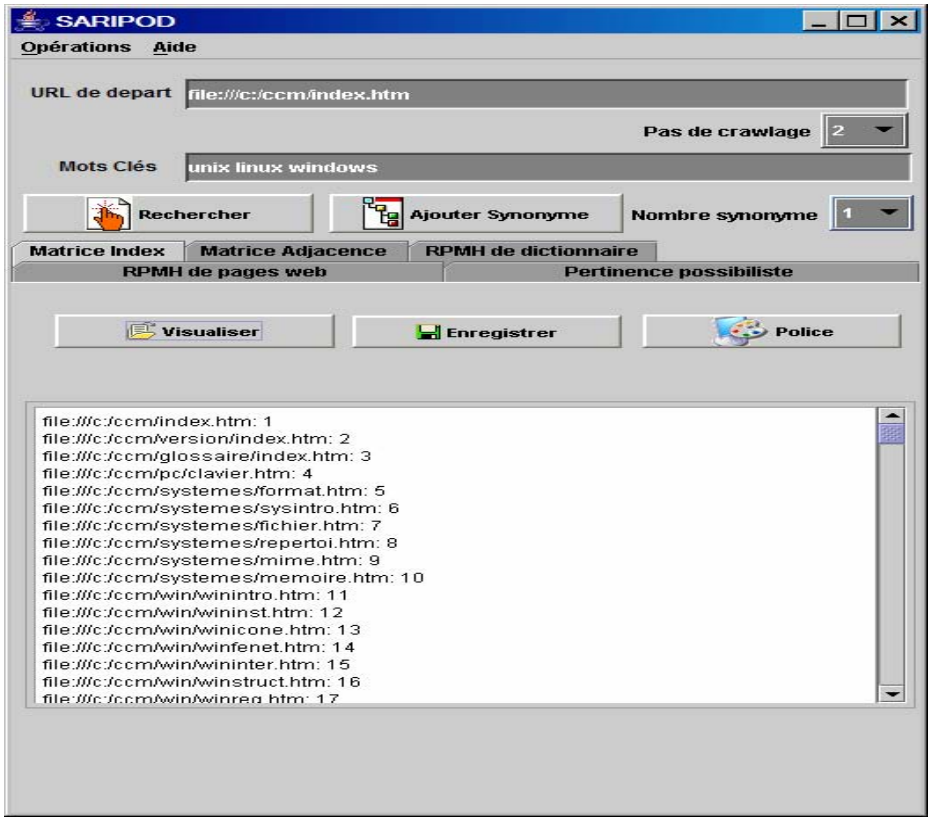


Figure 6.8 : Interface des URLs collectées par le crawler

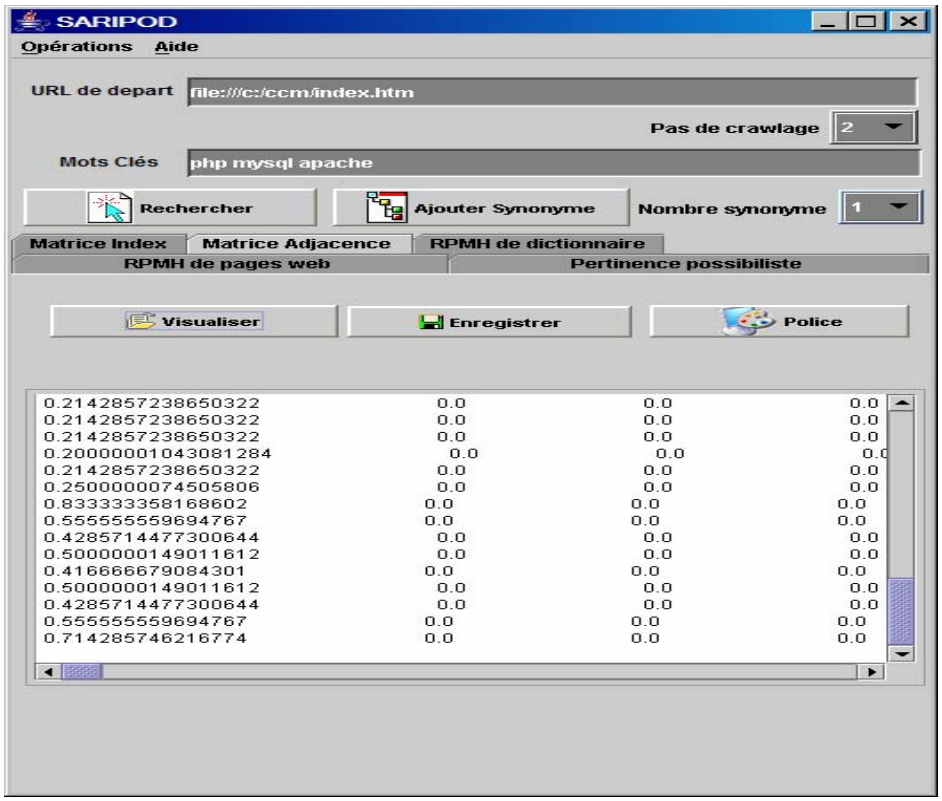


Figure 6.9 : Interface de proximité entre les pages Web

3.2 Interfaces du RPMH de pages Web

Nous avons réalisé une interface du RPMH de pages Web servant pour la détermination des pages contextuellement proches de la page racine. En fait, l'interface du RPMH de pages Web, présenté par de la figure 6.10, est formée d'un cadre principal (*Frame*) contenant un champ texte servant à saisir l'URL de la page Web racine, trois boutons « *chercher* » « *Graphe* » et « *Groupes* » et une liste (*JcomboBox*) « *Lg Circuit* » désignant la longueur de circuit entre la page racine et les pages contextuellement proches.

Le bouton « *chercher* » dans cette interface déclenche la recherche des URLs des pages Web recherchées et affiche trois colonnes : la première contient tous les URLs des pages proches triés par ordre de priorité, la deuxième indique la proximité de chaque page par rapport à la page racine et la troisième indique le nombre de circuits qui passent par chaque page.

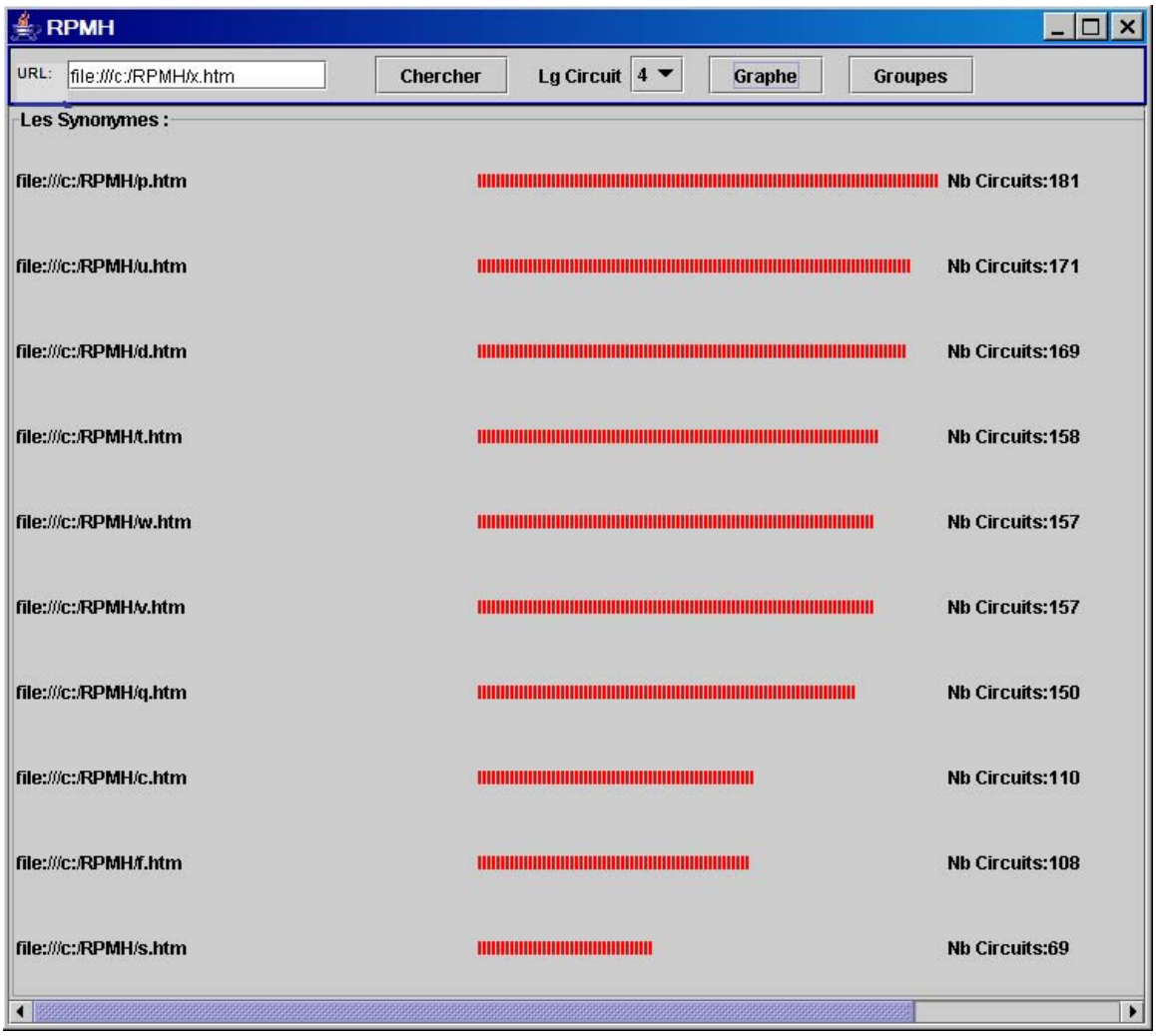


Figure 6.10 : Interface de calcul du nombre de circuits sélectionnés entre les pages Web

L'appui sur le bouton «*Graphe*» de la figure 6.10 nous affiche une nouvelle fenêtre contenant les circuits passant par la page Web racine et présentant les branches du RPMH de pages Web (voir figure 6.11).

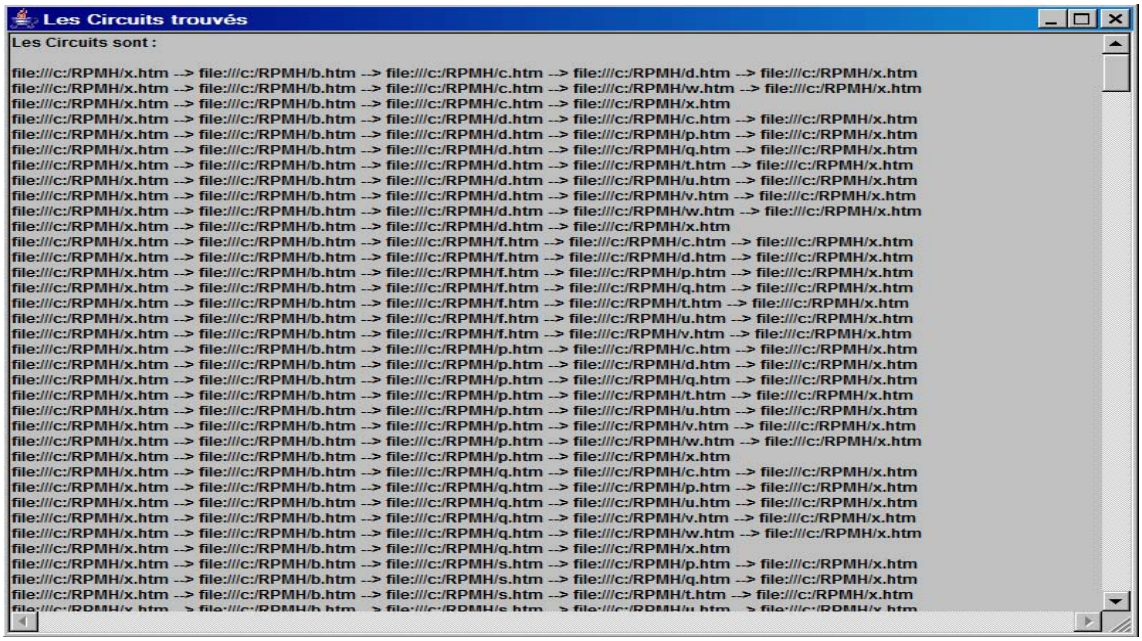


Figure 6.11 : Interface des branches de RPMH de pages Web

L'appui sur le bouton «*Groupes*» de la figure 6.10 nous affiche un nouveau cadre (*Frame*) contenant un bouton « *Grouper les pages* » et une liste (*JcomboBox*) «*seuil* ».

Le choix du *seuil* et l'appui sur le bouton « *Grouper les pages* » déclenche la recherche des groupes de pages (voir figure 6.12) avec leur fusion et l'affichage de ces groupes dans des panneaux (voir figure 6.13). En fait, ces groupes fusionnés représentent les petits mondes de "sens" dans le RPMH de pages Web.

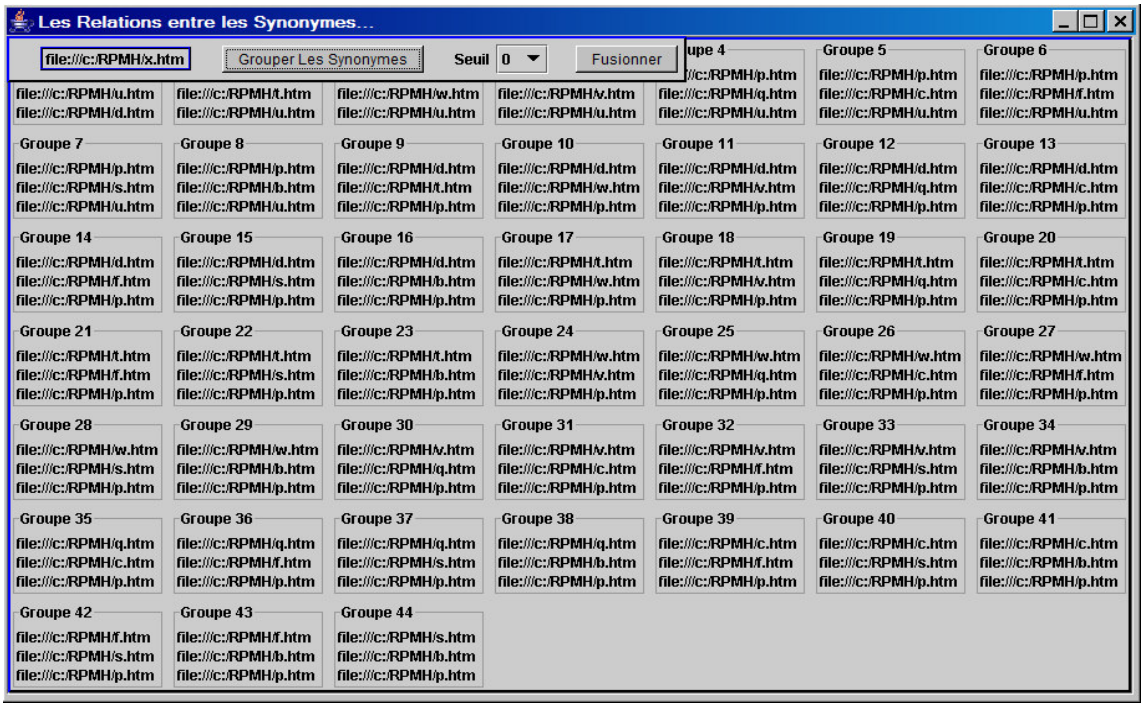


Figure 6.12 : Interface de groupement des pages dans le RPMH de pages Web

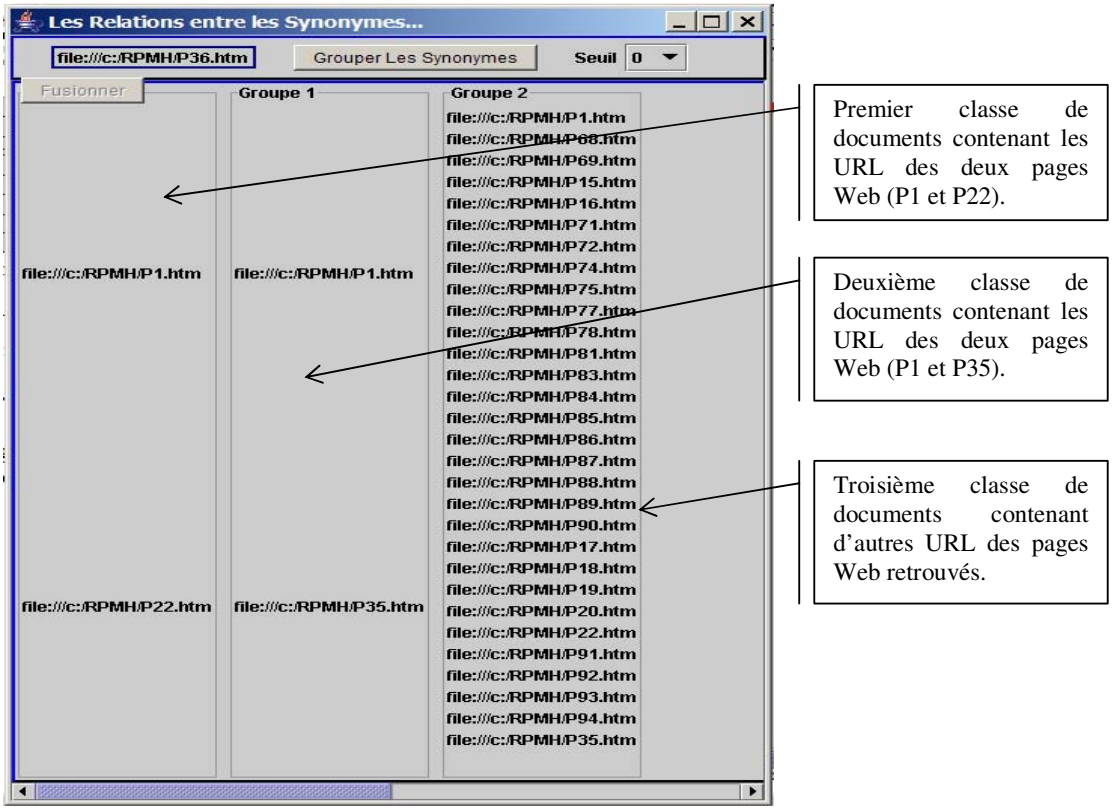


Figure 6.13 : Interface de fusion des groupes de pages dans le RPMH de pages Web

La visualisation graphique 3D du RPMH des pages Web est présentée par la figure 6.14. En effet, l'utilisateur de SARIPOD pourra naviguer dans le graphe RPMH des pages Web résultat de la recherche tout en faisant des zoom + ou – ainsi que des rotations et des déplacements du graphe dans les quatre directions (droite, gauche, haut et bas).

En fait, ce graphique correspond à un échantillon des résultats présentés dans les tableaux donnés en annexe 4. Le premier petit monde de pages Web s'intéresse au thème « réseaux et protocoles ». Ces pages sont caractérisées par un ensemble de mots ou expressions faisant partie de ce thème tels que : Protocoles, Transmission de données, équipements réseaux, Internet, Technologies, réseaux sans fil, WiFi (802.11), BleuTooth (802.15), Courants porteurs (CPL), etc. Alors que le second petit monde de ces pages Web s'intéresse au thème « systèmes d'exploitation». Ces pages sont caractérisées par un ensemble de mots ou expressions faisant partie de ce thème tels que : Unix, Linux, Mac/MacOS, Windows 95/98/Me, Windows NT/2000/XP, MS-DOS, AS/400 – OS/400, etc. Ainsi, toutes ces pages Web répondent fortement à cette description alors que toutes les autres pages crawlées y répondent bien moins telle que la page Web visualisée unique dans ce graphe.

Dans ce graphe 3D, les pages Web de chacun de ces deux petits mondes sont thématiquement proches. En effet, ces pages sont obtenues suite à un processus de *crawlage* stratégique (on ne garde des pages que si elles contiennent un même mot-clé ou, plus généralement, des mots proches de ces mots clés de départ) où les arêtes entre les nœuds pages représentent les proximités thématiques entre ces pages obtenues grâce à la matrice d'adjacence élevée à une certaine puissance (voir chapitre 4). En outre, nous avons bien démontré dans la figure 6.21 et le tableau 6.11 que toutes les pages, obtenues suite à cette démarche, sont des RPMH (L petit, C grand, loi de puissance).

D'autre part, l'intérêt de la visualisation 3D du RPMH de pages Web, dans le système SARIPOD, est de montrer à l'utilisateur du système une justification claire de ce qu'il a obtenu dans sa matrice de proximités entres les pages Web. Car si cette matrice est de très grande taille, il est difficile de détecter quelles sont les pages qui font partie du même petit monde, alors qu'à l'aide de cette visualisation 3D l'affichage est plus clair et il pourra consulter les pages directement à partir du graphe par le simple clic sur le nœud pages Web.

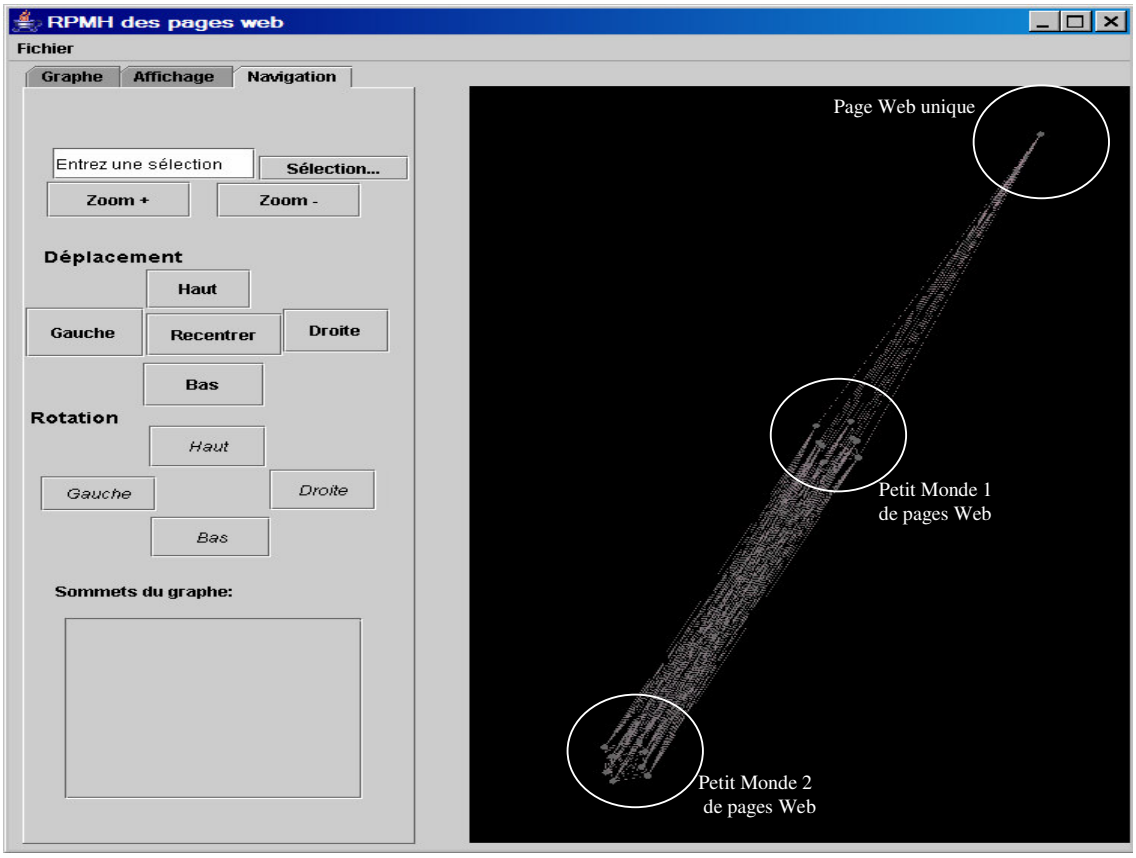


Figure 6.14 : Interface 3D du RPMH de pages Web

Nous pouvons travailler autrement en utilisant une Analyse en Composantes Principales (ACP) pour la visualisation 3D de ce RPMH de pages Web (comme le cas du logiciel Prox[31]). En effet, grâce à une certaine puissance de la matrice d'adjacence on obtient une matrice de vecteurs lignes où chaque page est plus ou moins proche des autres. Chaque page est représentée par un vecteur sur l'ensemble de toutes les pages. On est donc dans un espace de dimension D (D = le nombre total de pages Web "crawlées") équivalent à IR^D. Chaque page est un point de IR^D et on peut calculer la distance euclidienne de chaque page avec chaque autre. On postule que si deux pages se comportent à peu près de la même façon (c'est-à-dire si ces deux vecteurs V_1 et V_2 pointent à peu de chose près dans la même direction et sont à peu près de même longueur : $Cosinus(V_1, V_2)$ et $\|V_1\|/\|V_2\|$ sont voisins de 1) alors ces deux pages parlent à peu près de la même chose et elles seront dans un même petit monde (mais en termes du seul lien hypertextuel, elles ne sont peut-être pas si proches l'une de l'autre que cela).

[31] http://prox.irit.fr/

L'intérêt de l'espace de dimension 3 est qu'il est visualisable et dans ce cas ramener l'espace IR^D à IR^3 est ce qui est habituellement fait par une Analyse en Composantes Principales (ACP). Dans ce cas on peut avoir des clairs graphiques où l'impression de proximité géométrique traduit la proximité thématique des pages Web. Notons que cette ACP n'est pas nécessaire si l'on ne veut pas visualiser. La distance euclidienne dans IR^D suffit à comparer deux pages.

3.3 Interfaces du RPMH de Dictionnaire

Nous avons réalisé une interface de connexion avec l'interface du RPMH de dictionnaire. Cette interface a été initiée par [Shibly et al., 2004] et améliorée par nous même dans le cadre de la réalisation du système SARIPOD. En fait, l'appui sur le bouton « *Ajouter Synonyme* » de la figure 6.5, déclenche une manipulation automatique de cette interface pour la détermination de l'ensemble de mots sémantiquement proches des mots-clés de la requête.

L'interface de manipulation du RPMH de dictionnaire de mots, présenté par l'interface de la figure 6.15, est formé d'un cadre principal (*Frame*) contenant un champ texte servant à saisir le mot à étudier, trois boutons « *chercher* », « *Graphe* » et « *Groupes* » et une liste (*JcomboBox*) « *Lg Circuit* » désignant la longueur de circuit.

D'autre part, cette interface peut être manipulée à part d'une manière indépendante du système SARIPOD. En effet, si l'on veut chercher les mots proches d'un mot quelconque, ce dernier doit être insérée dans le champ de texte «*Mot* ». Le bouton « *chercher* » déclenche la recherche des mots proches du mot en question et affiche trois colonnes : la première contient la liste de mots proches triés par ordre de priorité, la deuxième indique le poids de chacun de ces mots proches et la troisième indique le nombre de circuits qui passent par chaque mot proche (voir figure 6.16). En fait, cette interface est très semblable à celle du RPMH de pages Web de la figure 6.10.

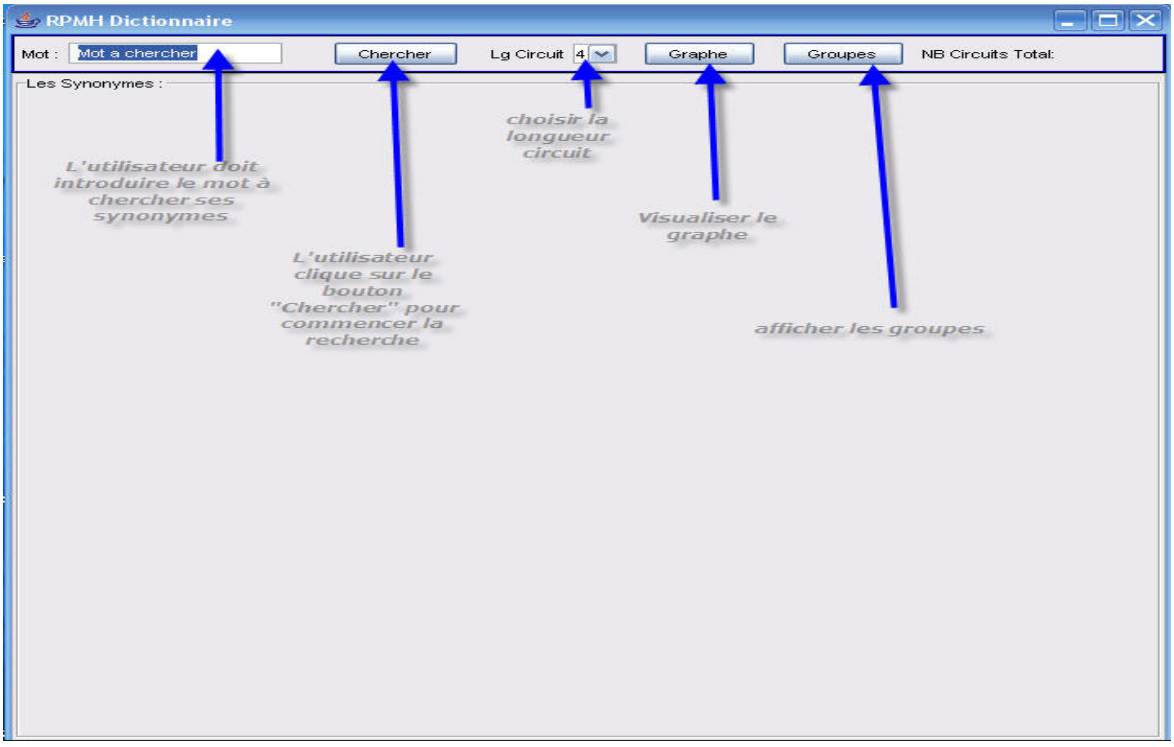


Figure 6.15 : Interface du RPMH de dictionnaire

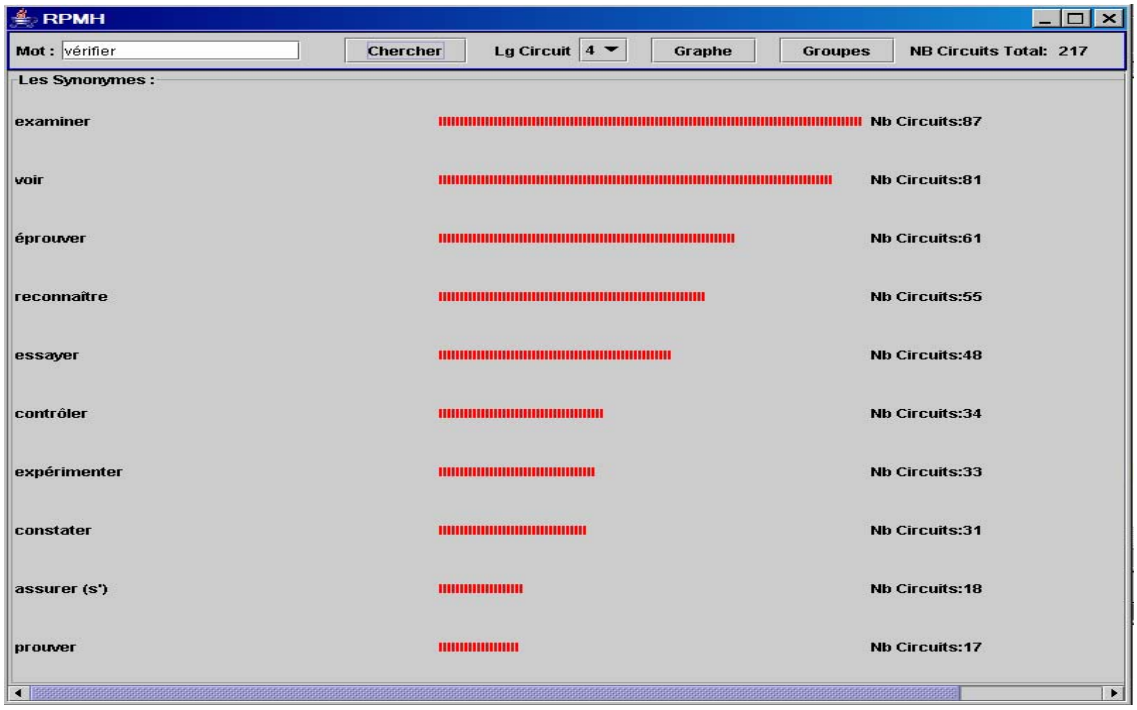


Figure 6.16 : Interface de calcul du nombre de circuits sélectionnés entre les mots de dictionnaire

L'appui sur le bouton «*Graphe*» de la figure 6.15 nous affiche une nouvelle fenêtre contenant les circuits passant par le mot de départ et présentant les branches du RPMH de mots de dictionnaire (voir figure 6.16).

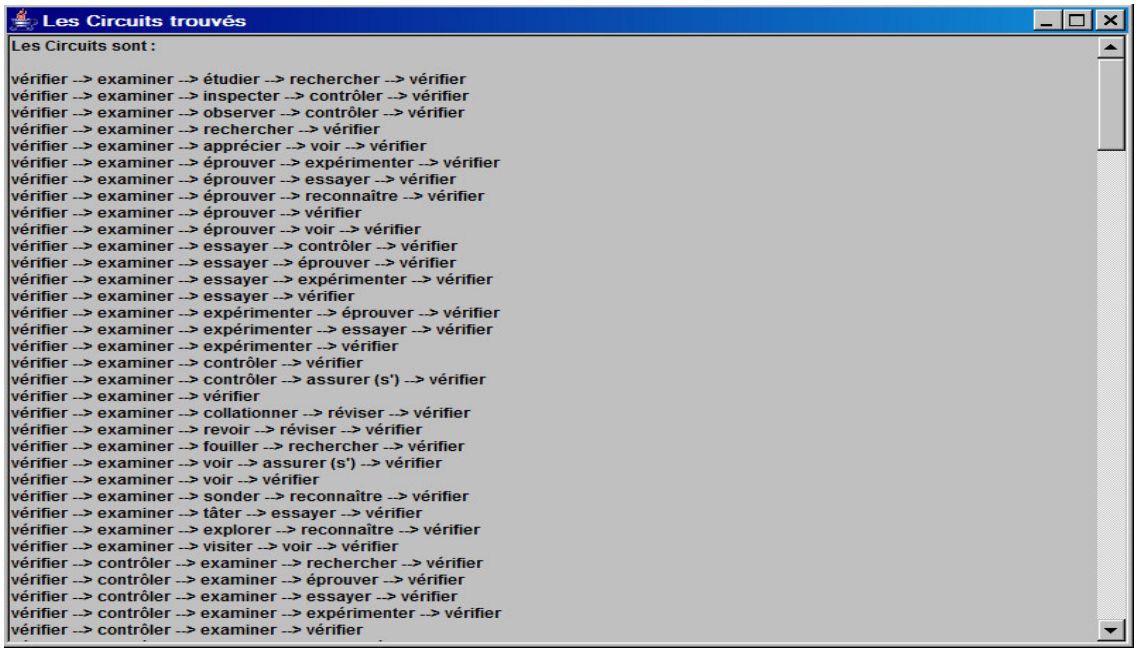


Figure 6.16 : Interface des branches de RPMH de mots de dictionnaire

L'appui sur le bouton «*Groupes*» de la figure 6.15 affiche un nouveau cadre (*Frame*) contenant un bouton « *Grouper les Synonymes* » et une liste (*JcomboBox*) «*seuil* ». En effet, le choix du *seuil* et l'appui sur le bouton « *Grouper les synonymes* » déclenche la recherche

des groupes de mots proches (voir figure 6.18) avec leur fusion et l'affichage de ces groupes dans des panneaux (voir figure 6.19). Ces groupes de mots fusionnés représentent les petits mondes de sens correspondant au mot de départ.

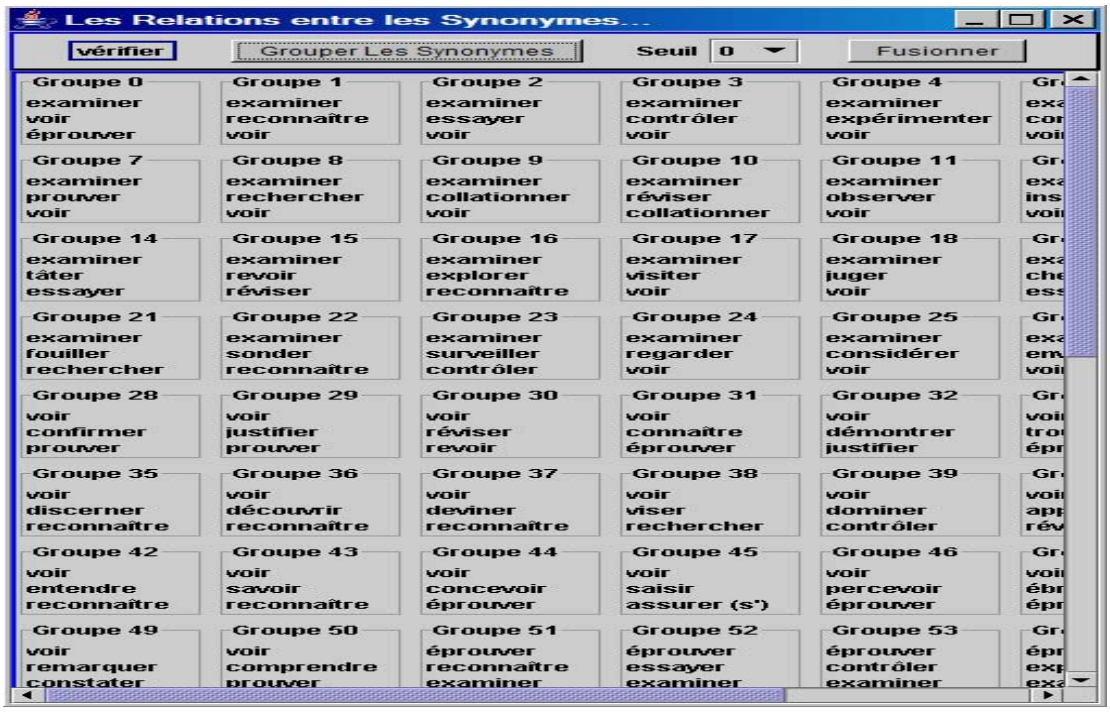


Figure 6.18 : Interface de groupement des mots proches dans le RPMH de dictionnaire

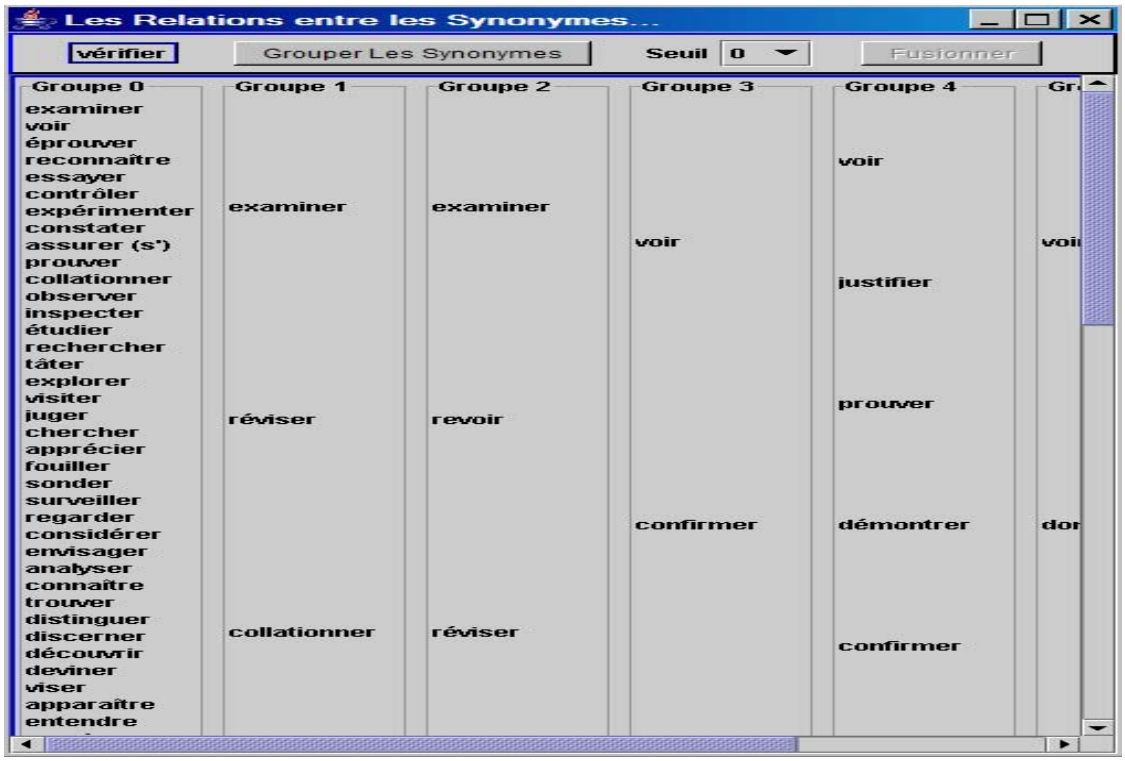


Figure 6.19 : Interface de Fusion des mots proches dans le RPMH de dictionnaire

4. Expérimentations et résultats

L'expérimentation d'un système informatique est l'étape la plus importante pour l'amélioration de ses performances. En effet, nous avons choisi d'expérimenter notre système à travers plusieurs axes pour en déduire, au titre de conséquence, le paramétrage optimal recommandé pour tout utilisateur de notre système.

Pour ce faire, nous nous sommes intéressés à tester les phases de reformulation sémantique de requêtes, de calcul des scores des pertinences possibilistes et de la classification des documents résultats d'une requête de recherche. Notre objectif est d'évaluer l'apport de telles phases au comportement du système et des agents impliqués.

4.1 Reformulation sémantique de requêtes

Pour tester la reformulation sémantique de requêtes, nous avons choisi d'utiliser un dictionnaire de verbes, extrait du dictionnaire français « *Le Grand Robert* », comme source de données alimentant le RPMH de dictionnaire. Ce dictionnaire contient 11000 entrées de taille globale 971 KOctets. Nous nous intéressons particulièrement aux groupes des sens intermédiaires ainsi que leurs fusions afin de construire les composantes de sens finales pour chaque verbe proposé à l'agent lexicographique. Nous calculons la moyenne des plus courts chemins (L) ainsi que le taux de clustering ou d'agrégation (C) pour chaque expérience afin de monter que la structure de dictionnaire est un RPMH dans les cinq expériences de test.

	Mot-clé (Verbe)	**Nombre de mots proches**	**Nombre de groupes de sens intermédiaires**	**Nombre de groupes de sens fusionnés**	**La moyenne des plus courts chemins : L**	**Le taux de clustering ou d'agrégation : C**
Expérience 1	Vérifier	68	115	14	1,1108	0,3875
Expérience 2	Nettoyer	106	285	32	1,0283	0,3556
Expérience 3	Analyser	40	43	13	1,1714	0,4567
Expérience 4	Jouer	216	504	19	1,1062	0,3888
Expérience 5	Préserver	57	133	4	1,0870	0,3691
				Moyenne	**1,10074**	**0,39154**

Tableau 6.2 : Récapitulation des résultats des cinq expériences sur le RPMH de dictionnaire

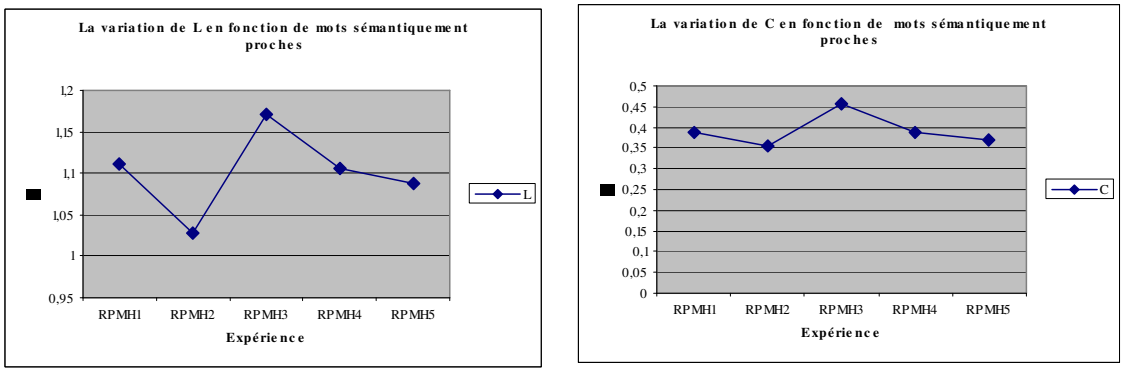


Figure 6.20 : Les variations de L et C en fonction du nombre de mots sémantiquement proches

A partir de l'analyse de ces deux dernières courbes de la figure 6.20 nous pouvons conclure que l'ensemble de mots proches collectés pour chaque requête est un RPMH. Ce résultat justifie notre hypothèse du modèle conceptuel proposé et qui consiste à structurer la requête de l'utilisateur sous forme d'un RPMH afin de pouvoir extraitre les termes sémantiquement proches d'un terme donné. En effet, à densité D égale, la moyenne des plus couts chemins entre les pages Web (L) est petit et le taux de clustering ou d'agrégation (C) est grand. Par ailleurs, les variations de L et C sont faibles en fonction du nombre de mots sémantiquement proches.

4.2 Comparaison avec les travaux de [Gaume et al., 2004]

Gaume et al. [Gaume et al., 2004] ont montré que les graphes d'origine linguistique et notamment ceux qui sont construits à partir de dictionnaires sont de type RPMH. Par exemple le graphe G_1 des noms construit à partir du dictionnaire *Le Robert* (les sommets sont les entrées qui sont des noms, et il existe une arrête entre deux sommets si l'un est dans la définition de l'autre – les auteurs ne tiennent pas compte ici de la structure hiérarchique des définitions) est un RPMH typique. Dans le graphe G_2 du tableau 6.3, chaque sommet est remplacé par l'arbre reflétant la structure hiérarchique de l'entrée qui lui correspond, ce qui a pour conséquence d'affaiblir le *C* et d'allonger le *L*. Dans le tableau 6.3, le symbole * indique que les mesures sont calculées sur la plus grande partie connexe.

Graphe	Nb. sommets	Nb. arcs	Nb. sommets*	Nb. Arcs*	Diamètre*	C*	L*
G1	51 559	392 161	51 511	392 142	7	0,1829	3,32
G2	140 080	399 969	140 026	399 941	11	0,0081	5,21

Tableau 6.3 : Quelques caractéristiques des graphes G1 et G2

Selon Gaume et al. [Gaume et al., 2004], la nature hiérarchique des dictionnaires (distribution des degrés d'incidence des sommets en loi de puissance) est une conséquence du rôle de l'hyperonymie associée à la polysémie de certains sommets, alors que le fort *C* (existence de zones denses en arêtes) reflète le rôle de la cohyponymie [Duvignau, 2002], [Duvignau, 2003], [Gaume et al., 2002]. Par exemple, le mot *corps* se trouve dans de nombreux définissants (*tête*, *chimie*, *peau*, *division*). De ce fait, le sommet *corps* a une forte incidence. D'autre part, les auteurs constatent qu'il existe de nombreux triangles par exemple : {*écorce*, *enveloppe*}, {*écorce*, *peau*}, {*peau*, *enveloppe*}, ce qui favorise les zones denses en arêtes et plus précisément un fort taux de clustering *C*. Ce sont ces zones denses en arêtes, qui orientant la dynamique des trajectoires de la particule, vont permettre la désambiguïsation.

Ainsi, nos contributions par rapport aux travaux de [Gaume et al., 2004] consistent principalement à l'exploitation du RPMH de dictionnaire dans le processus de reformulation sémantique de requêtes dans notre SRI intelligent. D'autre part nous avons pu introduit une composante classificatoire à base de dénombrement des circuits existant entre les nœuds verbe de notre RPMH de dictionnaire. Cette composante nous a servi davantage dans l'identification des composantes des sens correspondant aux termes de la requête utilisateur. D'ailleurs, nous remarquons que notre moyenne de plus court chemin $L = 1{,}10074 < L^*$ et notre moyenne du taux de clustering ou d'agrégation $C = 0{,}39154 > C^*$.

De plus, notre approche contribue à la résolution du problème de la polysémie ce qui permet d'afiner la requête reformulée. Les documents retrouvés seront en conséquence dispensés de cet effet néafaste de la langue.

4.3 Classification des documents

Nous avons testé SARIPOD sur des pages Web constituants l'encyclopédie informatique libre « CommentCaMarche »[32] dans sa version 2.0.5, où les circuits entre les pages sont fréquentes et le nombre de mots-clés sont bien réparties sur plusieurs thèmes organisés sur les divers classes de thèmes créés par les experts créateurs de l'encyclopédie. Il s'agit en fait d'une base documentaire de référence dans le domaine informatique et dont le contenu est récapitulé dans le tableau 6.4. Cette base contient 976 documents HTML, répartis sur 20 classes des thèmes et de taille globale 33 Méga Octets.

Classe du thème	**Mots-clés les plus fréquents**	**Nombre de documents dans la classe**
Ordinateur	Matériel informatique Assemblage Optimisation – Réparation Drivers (pilotes) Câbles et connecteurs	79
Bases de l'informatique	Binaire, hexadécimal Logique combinatoire Analogique / Numérique Informatique Son numérique Image et vidéo numériques	51
Sécurité informatique	Introduction à la sécurité informatique Virus et codes cachés Attaques et arnaques Sécurité sur Internet Cryptographie Protection Prévention / Détection Processus Windows	80
Systèmes d'exploitation	Notions fondamentales UNIX Linux Mac/MacOS Windows 95/98/Me Windows NT/2000/XP MS-Dos AS/400 - OS/400	90
Bureautique	Introduction à la bureautique Tableur (Excel/StarOffice)	23
Développement	Programmation Algorithmique Programmation orientée objet Programmation réseau UML CVS J2EE .NET Framework	60
Histoire	Histoire de l'informatique Petites histoires	20
Loi et droits	La législation Informatique morale	20
Pratique	Utilisation de logiciels Fiches pratiques	25

[32] http://www.commentcamarche.net/

	Internet utile	
Réseaux et protocoles	Initiation aux réseaux Transmission de données Equipements réseau Protocoles (Internet) Technologies	92
Réseaux sans fil	Introduction au wireless WiFi (802.11) Bluetooth (802.15) Courants porteurs (CPL)	32
Développement Web	Webmastering ASP CGI DHTML HTML Feuilles de styles (CSS) Javascript JSP PHP Servlets VBScript WAP XML	145
Bases de données	Initiation aus bases de données Conception - MERISE Modèle relationnel Langage SQL Langage PL/SQL JDBC ODBC Annuaires LDAP Active Directory	52
Langages	Assembleur Langage C Langage C++ Java LaTeX Perl Pascal / Delphi Visual Basic	78
Organisation	Gestion de projet Informatique d'entreprise Qualité	43
Internet	World Wide Web Outils réseaux	18
Client-Serveur	Introduction au client-serveur Sockets, RMI - IIOP	17
Tutorial réseau	Linux en réseau Création d'un intranet	27
Tutorial Linux	Utiliser Linux	9
Tutorial Web	MySQL, Oracle, PostgreSQL, Apache	19

Tableau 6.4 : Répartition des documents Web de la base du test

L'objectif des expériences que nous allons décrire ci-dessous est de montrer l'intérêt d'utiliser les RPMH et les Réseaux Possibilistes (RP) pour réduire la dimension de l'espace de recherche de documents et d'exploration ainsi que pour proposer une vue générale sur l'ensemble des thématiques traitées dans un fond documentaire. Ces aspects permettent donc, à un SRI de mener une recherche plus pertinente et de proposer à l'utilisateur des résultats synthétiques facilement interprétables.

Les classifications de documents pertinents que nous avons menées, sont construites en utilisant notre approche détaillée dans le chapitre 4. Toutefois nous comparons les classes crée par notre système par rapport aux classes proposer par l'expert créateur du base de test.

Par ailleurs, l'évaluation de nos résultats se base sur les critères standards d'évaluation des SRI présentés dans la cinquième section du premier chapitre de l'état de l'art. En effet, le calcul du Rappel et de la Précision se fait en considérant, d'une part les documents retrouvés pour une requête donnée et l'ensemble des documents pertinents associés à cette dernière.

Les documents jugés non pertinents par le système sont ceux qui possèdent des scores des pertinences possibilistes négatifs ou nuls (voir annexe 5). En fait, un document est titulaire d'un score de pertinence possibiliste négatif ou nul si les termes de la requête existent dans des structures logiques non pertinentes pour le système telles que les liens publicitaires, les légendes des logos, etc.

Enfin, nous prenons le paramètre β, de pondération de la précision ou du rappel dans le calcul de la fonction *F-mesure*, égal à la valeur 1.

Expérience 1

Dans cette première expérience (voir tableau 6.5), les termes de la requête font partie d'une seule classe de documents (la classe du thème « ordinateur »). Il est possible de remarquer que les documents retrouvés par le système représentent 30,37% de la totalité des documents formant la classe proposée par l'expert (79 documents), alors que 100% de ces documents retrouvés sont qualifiés pertinents. Par ailleurs, le système propose une seule classe de documents pour cette requête, ce qui est conforme avec le choix de l'expert. Cette conformité est dûe aux termes de la requête qui sont au cœur du thème « ordinateur » et faiblement existants dans le reste de documents du corpus.

Les termes de la requête (4 termes)	Réparation, drivers, câbles, connecteurs
Nombre de documents retrouvés	24
Nombre de documents pertinents dans tout le corpus	24
Nombre de classes de documents proposés par l'expert	1
Nombre de classes de documents de SARIPOD	1
Rappel **Précision** **F-mesure**	0,303 1,00 0,465

Tableau 6.5 : Données et résultats de la première expérience

Expérience 2

Dans cette deuxième expérience (voir tableau 6.6), les termes de la requête sont enrichis par d'autres termes faisant partie de la classe du thème « système d'exploitation ». Les documents retrouvés par le système représentent 101,77% de la totalité des documents formant les deux classes proposées par l'expert (169 documents). 84,88% de ces documents retrouvés sont jugés pertinents. Par ailleurs, le système a créé quatre classes de documents pour cette requête, alors que l'expert n'a proposé que deux classes. Cette augmentation dans la classification du système est causée par les termes Unix, Linux et Windows qui sont fréquents dans la majorité des documents de l'encyclopédie de test.

Les termes de la requête (7 termes)	Réparation, drivers, câbles, connecteurs, Unix, Linux, Windows
Nombre de documents retrouvés	172
Nombre de documents pertinents dans tout le corpus	146
Nombre de classes de documents proposés par l'expert	2
Nombre de classes de documents de SARIPOD	4
Rappel **Précision** **F-mesure**	0,863 0,848 0,855

Tableau 6.6 : Données et résultats de la deuxième expérience

Expérience 3

Dans la troisième expérience (voir tableau 6.7), les termes de la requête sont enrichis encore plus, par rapport à l'expérience précédente, par d'autres termes faisant partie de la classe du thème « Sécurité informatique» ainsi que la classe du thème « Base de l'informatique ». Les documents retrouvés par le système représentent 116,33% de la totalité des documents formant les quatre classes proposées par l'expert (300 documents). D'autre part, 90,83% des documents retrouvés sont jugés pertinents.

Par ailleurs, le système a créé cinq classes de documents pour cette requête, ce qui est légèrement augmenté par rapport à la classification de la deuxième expérience, malgré l'ajout à la requête des autres termes faisant partie de deux autres classes différentes. En fait, le système a fusionné les documents réponses aux trois termes image, vidéo et son avec la première classe du thème « ordinateur », ce qui est conforme avec la réalité vue que ces trois termes existent aussi dans cette classe. Alors que pour les deux autres nouveaux termes (cryptographie et protection), une nouvelle classe a été créée par le système, vue l'indépendance thématique de cette classe par rapport aux quatre classes créées dans la deuxième expérience.

Les termes de la requête (12 termes)	Réparation, drivers, câbles, connecteurs, Unix, Linux, Windows, cryptographie, protection, image, vidéo, son
Nombre de documents retrouvés	349
Nombre de documents pertinents dans tout le corpus	317
Nombre de classes de documents proposés par l'expert	4
Nombre de classes de documents de SARIPOD	5
Rappel **Précision** **F-mesure**	0,943 0,908 0,924

Tableau 6.7 : Données et résultats de la troisième expérience

Expérience 4

Dans cette quatrième expérience (voir tableau 6.8), les documents retrouvés par le système, représentent 159,67% de la totalité des documents formant les cinq classes proposées par l'expert (186 documents). D'autre part, 88,21% de ces documents retrouvés sont jugés pertinents. Les autres documents sont qualifiés non pertinents à cause de l'existence de plusieurs pages Web « polythématique » ; c'est-à-dire des pages existants dans plusieurs classes des thèmes différents, ce qui affaibli en conséquence leurs scores des pertinences possibilistes.

D'autre part, les termes de la requête sont extraits des cinq classes thématiquement proches. En fait, les classes des thèmes « Réseaux et protocoles », « Réseaux sans fils » et « Tutorial réseau » sont thématiquement très proches, vue qu'elles possèdent plusieurs mots-clés en commun. En outre, les deux autres classes des thèmes « Internet » et « Client-Serveur » sont aussi thématiquement proches entre eux et proches aussi de trois premières classes. C'est pour cette raison que le système SARIPOD a proposé une classification à base de deux classes uniquement. Une première classe du thème fusionnant les trois premières classes ci-dessus proposées par l'expert, et une deuxième classe du thème fusionnant les deux autres classes des thèmes proposés par l'expert.

Les termes de la requête (16 termes)	Réseaux, transmission, protocoles, Internet, intranet Wireless, WiFi, Bluetooth, World, Wide, Web, Client, serveur, Sockets, RMI, IIOP
Nombre de documents retrouvés	297
Nombre de documents pertinents dans tout le corpus	262
Nombre de classes de documents proposés par l'expert	5
Nombre de classes de documents de SARIPOD	2
Rappel	0,803
Précision	0,882
F-mesure	0,840

Tableau 6.8 : Données et résultats de la quatrième expérience

Expérience 5

Dans cette cinquième expérience (voir tableau 6.9), les documents retrouvés par le système, représentent 61,79% de la totalité des documents formant les quatre classes proposées par l'expert (335 documents). D'autre part, 91,3 % des documents retrouvés sont jugés pertinents.

En outre, les termes de la requête sont extraits des quatre classes thématiquement indépendantes selon la décision de l'expert. En conséquence, la classification proposée par le système propose trois classes en considérant les deux classes des thèmes « Développement Web » et « Langage » comme une seule classe contenant les langages de programmation indifférement de son type Web ou autre.

Les termes de la requête (15 termes)	prévention, protection, cryptographie, HTML, Javascript, VBscript, PHP, XML, assembleur, perl, delphi, pascal, Wireless, Wifi, Blootooth
Nombre de documents retrouvés	207
Nombre de documents pertinents dans tout le corpus	189
Nombre de classes de documents proposés par l'expert	4
Nombre de classes de documents de SARIPOD	3
Rappel	0,564
Précision	0,913
F-mesure	0,696

Tableau 6.9 : Données et résultats de la cinquième expérience

Synthèse des résultats

Nous présentons dans le tableau 6.10 les moyennes des valeurs obtenues dans les cinq expériences ci-dessus. Les documents retrouvés par le système, représentent en moyenne 93,98% de la totalité des documents formant la moyenne des classes proposées par l'expert dans les cinq expériences (213,8 documents en moyenne). D'autre part, le taux moyen de documents pertinents par rapport aux documents retrouvés est 91,04%.

A partir de l'analyse de ces deux taux moyens nous pouvons conclure que le système SARIPOD, et grâce à son algorithme de crawlage, ne se limite pas uniquement aux documents existants dans les classes des thèmes proposés par l'expert, mais il prouve l'existence d'autres documents pertinents dans d'autres classes thématiquement proches des classes de l'expert. Ainsi, une reclassification des documents resultats de la recherche s'avère très intéressante pour l'utilisateur et contribue fortement à l'amélioration de la performance de notre SRI. D'autre part, le système est doté d'une haute efficacité dans la sélection des documents pertinents parmi les documents retrouvés. Ceci est un facteur pertinent dans l'évaluation de l'approche possibiliste proposée.

Nous remarquons aussi que le système SARIPOD a atteint 93,75% du taux moyen de réussite de classification des documents résultats des cinq requêtes des tests, par rapport à la classification des documents proposée par l'expert réalisateur de l'encyclopédie. En effet, le système augmente le nombre de classes des thèmes, par rapport à la classification proposée par l'expert, si les termes de la requête existent dans plusieurs classes thématiquement indépendantes. Par contre, il diminue le nombre de classes des thèmes si les termes de la requête existent dans des classes thématiquement proches. Ainsi, toute corrélation sémantique entre les termes de la requête provoque une corrélation thématique entre les documents résultat de la recherche. En conséquence, le système propose une réclassification des pages Web thématiquement proches. Cette classification est plus optimale que celle proposée par l'expert créateur de l'encyclopédie.

Par ailleurs, la F-mesure constitue une mesure intermédiaire entre le Rappel et la Précision mesurés à partir des documents pertinents parmi les documents à classer. Nous avons considéré également l'effet du choix de plusieurs documents pertinents pour chaque résultat

de recherche à classifier. Nous avons observé que, dans ce cas, le Rappel augmente et que la Précision et la F-mesure moyenne diminuent.

Le nombre moyen de termes de la requête	10,8
Nombre moyen de documents retrouvés	209,8
Nombre moyen de documents pertinents dans tout le corpus	187,6
Nombre moyen de classes de documents proposés par l'expert	3,2
Nombre moyen de classes de documents de SARIPOD	3
Taux moyen de documents retrouvés par rapport aux documents formants les classes de l'expert	93,98%
Taux moyen de documents pertinents par rapport aux documents retrouvés	91,04%
Rappel moyen **Précision moyenne** **F-mesure moyenne**	0,695 0,910 0,756

Tableau 6.10 : Synthèse des résultats des expériences

D'un autre côté, le tableau 6.11 récapitule les résultats des ces cinq expériences mais en traitant cette fois les deux axes primordiaux dans le système SARIPOD : les deux RPMH et le Réseau Possibiliste (RP).

Expérience	Nombre de Mots-clés de la requête	Nombre de pages Web retrouvées	Nombre de pages Web pertinentes	La pertinence la plus élevée $DPM(d_1)$	La pertinence la plus faible $DPM(d_N)$
Expérience 1	4	24	24	20,22	4,07
Expérience 2	7	172	146	53,65	0,13
Expérience 3	12	349	317	58,81	0,0734
Expérience 4	16	297	262	102,85	0,0739
Expérience 5	15	207	189	66,36	0,609

Tableau 6.11 : Résultats des expérimentations

Par ailleurs, au cours de la phase de reformulation sémantique de la requête, la détermination de mots sémantiquement proches du mot-clé de départ est très dépendante du degré de nettoyage du dictionnaire français utilisé « *Le Grand Robert* » comme source de données pour le RPMH de dictionnaire. Nous remarquons, d'après le tableau 6.8, que plus le nombre de documents retrouvé par le système est important, plus la chance d'avoir des documents pertinents est important.

Nous remarquons aussi que l'écart entre le degré de pertinence possibiliste de la page la plus pertinente ($DPM(d_1)$) et celui de la page la moins pertinente ($DPM(d_N)$) de la collection de documents pertinents augmente lorsque le nombre de pages Web pertinentes augmente, ce qui prouve que le but premier qui nous a motivé dans l'usage des RPMH est bien vérifié ici : faire en sorte que les réponses renvoyées suite à une requête ne soient plus le vrac "à la Google",

mais quelque chose de structuré en RPMH de sorte que si une page parmi les réponses renvoyées semble pertinente alors toutes celles qui lui sont "proches" dans ce RPMH le seront aussi. Ainsi, nous augmentons très considérablement le nombre de pages pertinentes récupérées pour les mots-clés de départ et nous éliminons les pages non pertinentes (que Google livrait malgré tout). En fait, nous changeons le PageRank (au sens de Google) des pages résultats de la requête [Elayeb et al., 2007c].

A partir de la figure 6.21 et du tableau 6.12, nous remarquons bien que les pages Web collectées pour chaque requête sont des RPMH, ce que justifie nos choix de départ dans le modèle conceptuel proposé. En effet, à densité D égale, la moyenne des plus couts chemins entre les pages Web (L) est petit et le taux de clustering ou d'agrégation (C) est grand. En outre, les variations de L et C sont faibles en fonction des nombres des pages Web retrouvées. Ceci montre bien que le Web est un RPMH.

Expérience	**Nombre de documents à classer**	**La moyenne des plus courts chemins : L**	**Le taux de clustering ou d'agrégation : C**
Expérience 1: $RPMH_1$	24	1,0606	0,4510
Expérience 2: $RPMH_2$	172	1,0349	0,3500
Expérience 3: $RPMH_3$	349	1,0193	0,3155
Expérience 4: $RPMH_4$	297	1,0270	0,3520
Expérience 5: $RPMH_5$	207	1,0305	0,3389
	Moyenne	**1,03446**	**0,36148**

Tableau 6.12 : Les paramètres L et C des RPMH des documents

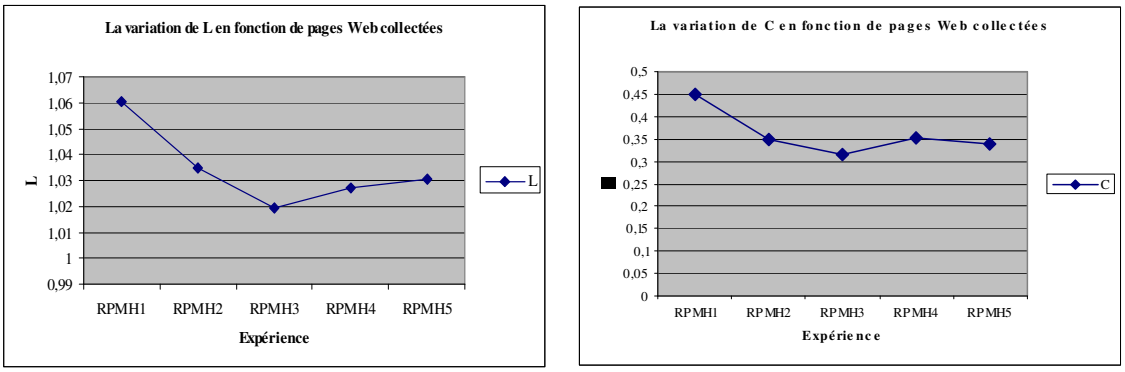


Figure 6.21 : Les variations de L et C en fonction du nombre de pages Web retrouvées

4.4 Comparaison avec le SRI SARCI

Le Système Agents pour la Recherche et la Classification d'Information (SARCI) est proposé par [Kammoun-Bouzaïene, 2006]. Ce système est à base d'un modèle pour une recherche d'information adaptative, évolutive et coopérative. En effet, SARCI met à la disposition de l'utilisateur plusieurs alternatives de recherche à travers deux principales phases : une analyse de surface qui constitue une étape préliminaire de recherche et une analyse en profondeur qui n'est activée que si la première ne satisfait pas l'utilisateur. L'analyse de surface permet de construire des connaissances liées aux requêtes antérieures, ce qui constitue un moyen pour résoudre la complexité d'exprimer une requête initiale, en assistant l'utilisateur à partir des expériences passées. L'analyse en profondeur permet de construire des connaissances liées aux utilisateurs et aux documents de la collection. Ces connaissances sont organisées par

point de vue exprimant différents points d'accès à une collection et permettant d'enrichir le niveau de recherche.

Les connaissances construites sont à caractère évolutif, l'auteur a introduit un apprentissage non supervisé à travers des classifieurs adaptatifs et incrémentaux. Pour l'analyse de surface, l'auteur a choisi une méthode symbolique (treillis de Galois) qui a l'avantage de s'adapter à la représentation de la requête et pour l'analyse en profondeur l'auteur a choisi une méthode numérique (cartes topographiques et auto-organisatrices de Kohonen) favorisant une représentation synthétique et thématique des connaissances, et constituant un support de navigation. D'autre part les résultats fournis par les classifieurs ont servi pour la reformulation de requêtes. L'auteur a introduit dans ce cadre en plus du *document feedback* le *query feedback.*

Un des apports de notre système SARIPOD par rapport au système SARCI consiste à modéliser d'une nouvelle manière la pertinence. En fait, nous avons défini la pertinence possible d'un document vis-à-vis d'une requête et sa pertinence nécessaire. La pertinence possible vise à éliminer les documents non pertinents, la pertinence nécessaire vise à renforcer la pertinence des documents non éliminés par la possibilité. Nous avons étendu cette définition d'un cadre quantitatif à un cadre qualitatif possibiliste. Cette double mesure de pertinence est censée aider le système dans sa décision concernant les documents à restituer ainsi que leur ordre de restitution. Pour ce faire nous comparons les performances de notre système à l'un de SRI multi-agent à savoir le système SARCI.

Une première constatation au vu des points de précision est que notre système obtient de meilleures performances. Nous présentons un comparatif de la précision moyenne obtenu suite aux expérientations. Nous remarquons que la précision varie entre 0,15 et 0,2 pour SARCI, alors que la moyenne des précisions de cinq expériences de SARIPOD est de l'ordre de 0,91 et la moyenne des rappels est de l'ordre de 0,695. Le système SARIPOD montre une amélioration dans la sélection des documents pertinents dans l'ensemble de document retrouvés par le système, ce qui prouve bien l'efficacité de l'extension proposée pour l'approche possibiliste quantitative. Ainsi, le système SARIPOD propose une nette amélioration des performances et sa courbe de Rappel-Précision est souvent au-dessus de celle de SARCI.

Quant à la composante classificatoire proposée par SARIPOD, elle est proposée d'une nouvelle manière à base d'une approche générique valable dans le cas des mots d'un dictionnaire que dans le cas des documents Web. En fait, les classes des thèmes des documents retrouvés sont conséquences des classes des sens des termes de la requête reformulée. Cette classification offre un cadre navigationnel pour l'utilisateur que se soit dans sa requête au cours de sa reformulation, soit dans les documents résultats de la recherche. Alors que pour le cas de SARCI, l'auteur a utilisé des méthodes existantes : les cartes de Kohonen pour la classification des documents et les Treillis de Galois pour la classification des requêtes.

5. Conclusion

Dans ce chapitre nous avons montré le caractère qualitatif possibiliste de notre SRI et précisément au cours des calculs des scores des pertinences possibilistes par l'agent mesure possibiliste. Ce dernier se base sur la structure logique du document, d'une part et les préférences proposées par l'utilisateur au système, d'autre part. Les expérimentations menées montrent que les résultats des sélections des documents pertinents parmi les documents retrouvés sont très encourageants et prouve l'apport de l'approche possibiliste proposée.

Nous avons pu démonter aussi que les résultats des classifications des documents pertinents, avec la méthode que nous avons proposée sont très proches de ceux proposés par l'expert, et ce, en se basant sur les deux mesures de Rappel et de Précision.

Par ailleurs, l'intérêt de faire combiner les deux RPMH via un Réseau Possibiliste (RP) dans un SRI permet d'enrechir le niveau d'exploration d'une collection. Ce dernier n'est pas limité aux documents mais l'étend en considérant les requêtes. En effet, la phase de reformulation sémantique de requête, assurée par l'agent lexicographique, permet à l'utilisateur de profiter des autres documents correspondants aux termes proches des termes de la requête initiale. Ces documents peuvent exister dans d'autres classes des thèmes. En conséquence, une reclassification proposée par le système s'avère pertinente afin d'adapter les résultats d'une requête aux nouveaux besoins des utilisateurs.

Conclusion générale et Perspectives

L'usage des réseaux probabilistes en RI est important grâce à leur capacité à représenter de manière naturelle les différents liens existants entre les objets manipulés en RI, à savoir les termes, les documents et la requête ainsi qu'à leur puissance pour inférer la pertinence des documents vis-à-vis d'une requête. Cependant, le cadre probabiliste dans lequel ces réseaux ont été définis traduit mal les deux notions de pertinence et de représentativité des termes dans les documents. En effet, cette théorie permet uniquement de mesurer la certitude d'un événement et de son contraire. Dans ces modèles la pertinence et la représentativité d'un terme dans un document sont des valeurs binaires. Un document donné est pertinent ou non vis-à-vis d'une requête à un certain degré. Un terme est représentatif d'un document ou non à un certain degré.

D'autre part, quel que soit le modèle de la RI, nous remarquons que la pertinence est vue comme un concept binaire. Cependant, certains travaux de la littérature ont montré que ce concept est graduel et dynamique [Rijsbergen, 1979] [Saracevic, 1996] [Kekäläinen et Järvelin, 2002] [Brini et Boughanem, 2003]. De plus, pour tous ces modèles, les termes de la requête absents des documents ne sont pas explicitement considérés dans le calcul des scores de pertinence.

Plusieurs travaux récents en Recherche d'Information traitent la problématique des documents semi-structurés. [Zayani, 2008] propose une contribution à la définition et à la mise en oeuvre de mécanismes d'adaptation de documents semi-structurés. [Ali Laouar, 2007] a proposé de sa part une contribution à l'interrogation flexible de données semi-structurées. [Sauvagna, 2005] a réalisé un modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés.

Le système SARIPOD proposé dans le cadre de cette thèse s'inscrit dans la problématique des systèmes multi-agents de Recherche d'Information sur Internet. Il est à base d'un modèle de RI permettant une nouvelle modélisation des deux notions de base en Recherche d'Information : la *pertinence* et le *profil*. Le modèle proposé par [Brini et al., 2005ab] [Brini et al., 2007] se base sur les réseaux possibilistes. Plus précisément, les nœuds de ce réseau représentent les documents, leurs termes d'indexation et la requête. La topologie du réseau permet de prendre en compte naturellement les relations de dépendance entre ces nœuds.

En fait, ce modèle ignore les dépendances entre les termes de la requête ainsi que les dépendances entre les documents de la collection. En outre, aucun processus de reformulation de la requête n'a été proposé. En conséquence, tous les termes de la même requête sont considérés de même poids ; il est supposé que l'utilisateur n'a pas de préférences entre les termes de sa requête. D'autre part, ce modèle est proposé uniquement dans un cadre quantitatif.

Nous avons proposé pour notre part une extension de ce modèle vers un cadre qualitatif possibiliste tout en tenant compte, non pas seulement de l'existence ou non du terme dans le document pour interpréter sa pertinence, mais aussi son poids dans ce document. En effet, le système SARIPOD répond aux limites du modèle possibiliste de [Brini et al., 2007] tout en

proposant une nouvelle modélisation faisant appel aux trois techniques: Réseaux Petits Mondes Hiérarchiques (RPMH), Réseaux Possibilistes (RP) et Systèmes Multi-Agents (SMA).

1. Choix principaux

Le choix de l'approche multi-agent est induit par les différentes caractéristiques de la problématique de la Recherche d'Information sur Internet : le parallélisme du traitement qui est souvent assuré dans un système multi-agent, la possibilité de mobilité de l'agent crawler (programmes distribués) qui peut toujours contribuer à la résolution des problèmes à distance, la sécurité de l'opération de recherche qui est souvent recommandée lorsque les agents sont censés se déplacer pour chercher l'information sur plusieurs sites distants. D'autres part, les sources d'information alimentent le réseau Internet par des milliers de sites chaque jour et qui nécessitent un traitement souvent rapide et distribué.

Quant au choix de conception adopté, le langage UML fournit un cadre convenable pour la modélisation de ce genre de système. L'organisation de l'application sous forme de couches d'agents semble être très naturelle pour ce genre de système. En effet, cette conception a fourni au système une grande généricité et indépendance de l'application.

L'utilisation de la plate-forme multi-agent, dans le développement du système SARIPOD, est imposée par la nécessité de la rapidité du prototypage. De plus, le choix de la plate-forme Jade est une bonne démonstration de l'applicabilité du concept de la réutilisation qui commence à convaincre les développeurs qui permet de fournir un gain considérable en temps de développement des logiciels.

2. Contribution principale

Nous avons proposé dans cette thèse un modèle possibiliste mixte (quantitatif et qualitatif) de RI. En fait, l'utilisation du cadre possibiliste permet d'éclaircir les définitions de la pertinence ainsi que la représentativité d'un terme dans un document. La notion de pertinence d'un document, étant donnée une requête, est modélisée par une double mesure. La pertinence possible permet de rejeter les documents non pertinents à une requête donnée. La pertinence nécessaire permet de se focaliser sur les documents à restituer ainsi que de renforcer la nécessité de faire figurer parmi les premiers de la liste des résultats en réponse à une requête. Les arcs reliant des paires de nœuds sont quantifiés par des degrés de possibilité et de nécessité. Ces degrés mesurent d'une manière générale le degré de possibilité et de nécessité de l'information véhiculée par les arcs du réseau possibiliste. Cette information concerne la représentativité d'un terme dans un document et permet de quantifier la pertinence d'un document étant donnée une requête.

Le système SARIPOD se déclenche par un processus itératif pour la reformulation de la requête. Pour ce faire, nous avons proposé une approche de recherche de composantes de sens dans un Réseau Petits Mondes Hiérarchiques (RPMH) de mots du dictionnaire Français « *Le grand Robert* ». Cette étape itérative nous facilite la définition des préférences entre les termes de la requête reformulée. En effet, l'évaluation de la pertinence d'un document vis-à-vis d'une requête est effectuée par un processus de propagation à travers les nœuds termes reliés à cette requête.

Les préférences entre les termes de la requête dans les représentations des documents sont naturellement et explicitement considérés dans le calcul des scores de pertinence contrairement aux systèmes actuels de RI. En effet, l'insertion des facteurs *Préférence*(t_i) dans

les calculs des possibilités et des nécessités, consiste à augmenter les scores de pertinence des documents contenant ces termes dans le but de pénaliser les scores de pertinence des documents ne les contenant pas. La pénalisation et l'augmentation des scores sont proportionnelles au pouvoir des termes à discriminer entre les documents de la collection. En outre, les facteurs de *Préférence* que nous avons proposé sont plus fins que le facteur *idf*, puisque la distribution des termes dans la collection de documents ne dépend pas seulement de la présence ou de l'absence des termes dans les documents de la collection (comme *idf*), mais de la distribution de leur densité dans les documents de la collection. Ainsi, ces mesures se sont avérées efficaces pour la discrimination négative, comparé notamment à *idf*. D'autre part, ces préférences permettent de restituer des documents classés par préférence de pertinence. Il est possible dans un tel cadre de mesurer à quel point un document d_1 est préféré au document d_2 ou de mesurer la préférence du document d_1 par rapport à un ensemble de documents $\{d_3, d_4\}$.

Nous avons aussi défini les relations de dépendance dans un cadre qualitatif. Les valeurs affectées à ces relations traduisent des ordres partiels de préférence. En fait, la théorie des possibilités offre deux cadres de travail : le cadre qualitatif ou ordinal et le cadre numérique. Rappelons que le modèle de [Brini et al., 2005ab] [Brini et al., 2007] s'inscrit dans le cadre quantitatif. Alors que notre approche traduit bien un cadre qualitatif possibiliste. En effet, nous avons bien introduit des relations de dépendance entre les termes de la requête via un Réseau Petits Mondes Hiérarchiques pour les mots de la langue Française. Ce premier RPMH permet de ne pas prendre les mots-clés de l'utilisateur tels quels mais de considérer une requête comme multiple en ce sens qu'on ne cherche pas seulement les mot-clés dans les pages Web mais aussi ses mots sémantiquement proches. Un deuxième RPMH permet d'intégrer des relations de dépendance entre les documents de la collection dont le rôle consiste à structurer les documents de la collection en zones denses de pages Web très fortement liées les unes aux autres et qui répondent toutes fortement à une requête.

Globalement, le développement du système SARIPOD engendre plusieurs contributions. D'une part, c'est un développement dédié au domaine de recherche d'information sur Internet où la maîtrise de la *pertinence* et du *profil* exige plus d'optimisation dans la Recherche d'Information. D'autre part, il s'agit d'une contribution dans la modélisation de l'information au niveau de la structuration des termes et des documents sous forme de deux RPMH définissant d'une part, les proximités sémantiques entre les nœuds termes et les proximités thématiques entre les nœuds documents. En effet, ces deux RPMH sont mixés via des réseaux possibilistes traduisant les pertinences existant entre les nœuds documents vis-à-vis d'une requête. De plus, c'est une contribution dans les Interfaces Homme-Machine (IHM) qui consiste à proposer une interface graphique du système SARIPOD permettant à l'utilisateur de présenter aisément sa requête ainsi que son profil, et qui reçoit, en contre partie, l'information optimale désirée. En effet, cette contribution permet de combler le fossé entre le système et son utilisateur. Quant à l'utilisation de l'approche multi-agent, la grande richesse de cette approche provient de la conformité au monde réel des applications de Recherche d'Information regroupant généralement tous les concepts de base déjà empruntés par cette approche, à savoir : la coopération, la compétition et l'autonomie.

Hormis tous ces avantages, SARIPOD offre plusieurs autres avantages. C'est ainsi qu'il est *portable* (puisque cette application est programmée en Java), *flexible* (facilement adaptable à d'autres applications) et *interopérable*.

D'autre part, les premiers essais effectués ont montré que SARIPOD est très efficace par rapport à une recherche Web classique, très facile d'utilisation et convient très bien aux applications faisant intervenir des bases de données héritées.

Bien que SARIPOD soit opérationnel, il y a toujours place pour son amélioration. Nous citons dans la suite quelques perspectives ainsi que de futurs développements.

3. Perspectives

Nos tests se sont limités aux occurrences verbales dans les définitions des verbes du RPMH de dictionnaire, mais nous envisageons d'étendre les tests à d'autres catégories grammaticales (nom, adjectif, adverbe, etc.), ainsi que d'affiner notre approche pour les substantifs en considérant par exemple également les occurrences verbales dans les définitions des noms, des adjectifs, des adverbes, etc.

Nous envisageons également de réaliser des mesures plus fines des performances du système SARIPOD en étendant les tests à d'autres formats de documents Web (Texte, Doc, PDF, etc.) ainsi qu'au dictionnaire (qui n'est pas un ensemble de pages Web mais un ensemble d'articles reliés les uns aux autres).

Nous proposons aussi d'étendre notre modèle pour représenter des documents particuliers à savoir les documents XML. En effet, la structure de document pourrait être traduite par la topologie du réseau, les nœuds intermédiaires correspondant aux balises du document et les nœuds feuilles aux termes des granules. D'autre part, l'architecture du réseau se prête naturellement à ce type de représentation. L'application de l'approche réalisée dans notre système permettrait de travailler à différents niveau de granularité (parties de documents, documents ou ensemble de documents) et d'obtenir des réponses nécessairement et possiblement pertinentes étant donnée une requête.

Dans l'intégration des relations de dépendance entre les termes ou les documents, les arcs sont mesurés par des valeurs numériques traduisant des quantités et non pas des ordres partiels. Afin de quantifier ces relations, nous pourrions nous baser sur la connaissance représentée dans une ontologie. En fait, une ontologie permet de formaliser des liens sémantiques entre des unités de sens [Ben Ahmed, 2007]. Définie dans un cadre possibiliste, elle pourrait ajouter de l'information pertinente à considérer lors du processus de propagation déclenchée par la requête. Le réseau serait composé d'un sous réseau de documents et d'un sous réseau de requête. Ces sous réseaux pourraient être reliés à travers une ontologie.

Nous envisageons aussi d'introduire dans SARIPOD la possibilité de navigation dans les documents restitués à travers une carte multi-dimentionnelle. Deplus, restructurer les pages Web pertinentes en RPMH propres aux besoins et aux préférences de l'utilisateur indépendamment des liens hypertextes existants entre ces pages.

Afin d'améliorer le processus de reformulation de requêtes dans SARIPOD, nous proposons une hybridation entre la reformulation sémantique proposée et une reformulation à base de technique de *Relevance Feedback* (*RF*).

Par ailleurs, nous proposons d'enrichir le nombre d'entités logiques exploitées par SARIPOD pour tenir compte d'autres structures textuelles (typologies des textes : narration, description, citations, …)

Une autre perspective est liée à l'architecture multi-agent du système SARIPOD consistant à inclure de nouveaux agents au système tel qu'un agent d'extraction des connaissances à partir des grandes bases de données. En effet, grâce à cet agent, le système pourra profiter des immenses informations stockées dans des bases de données géantes. Nous prévoyons aussi d'ajouter un autre agent simulateur mesurant le trafic dans la plate-forme pour éviter tout goulot d'étranglement en cas d'un nombre très élevé de pages Web collectées. D'autre part, nous envisageons améliorer la communication entre les différents agents de SARIPOD.

Notons enfin que pour une meilleure performance du système SARIPOD, ce dernier doit être installé sur un serveur assurant une accéleration dans les traitements et une puissance remarquable dans la manipulation des données géantes (dictionnaire de mots complet au format XML). Ce qui garantit une minimisation du temps de réponse à une requête utilisateur.

Bibliographie

[Abbadeni et al., 1998] Abbadeni N., Ziou D., et Wang S., "Recherche d'images basée sur leur contenu", Rapport de recherche, université de Sherbrooke, Canada, 1998.

[Abdallah et al., 2003] Abdallah H., Sleiman R., et Harati A., "Etude de la synonymie dans les dictionnaires et réalisation d'un outil de mesure de la proximité de sens", mémoire de fin d'études de maîtrise d'informatique, université libanaise, faculté des sciences I, 2003.

[Ali Laouar, 2007] Ali Laouar A., "Contribution à l'interrogation flexible de données semi-structurées", Thèse de Doctorat en informatique, Université Paul Sabatier, Toulouse (France), 2007.

[Aliane et al., 2004] Aliane H., Alimazighi Z., Boughacha R. O., et Djelliout T., "Un système de reformulation de requêtes pour la recherche d'information", Dans RIST Vol, 14 n°01, pp. 25-33, 2004.

[Antonella et al., 2003] Antonella D. S., Giuseppe P., Corrado S. et Emiliano T., "A Multi-Agent Reflective for Web Search Assistance", Université de Catania, Italie, 2003.

[Arfaoui, 2006] Arfaoui N., "Etude comparative entre les Réseaux Bayésiens et les Réseaux Possibilistes pour la détection des intrusions", Mémoire de Mastère en Informatique, Ecole Nationale des Sciences de l'Informatique, Tunisie 2006.

[Awada et Chebaro, 2004] Awada A., et Chebaro B., "Etude de la synonymie par l'extraction de composantes N-connexes dans les graphes de dictionnaires", Journées d'études linguistiques JEL2004, Nantes (France), 2004.

[Awada, 2005] Awada A., "Regroupement de synonymes en composantes de sens dans un dictionnaire", IC2005, France, 2005.

[Barabasi et al., 2000] Barabasi A. L., Albert R., et Jeong H., "Scale-free characteristics of random networks: The topology of the World Wide Web", Physica A281, pp. 69-77, 2000.

[Belew, 1989] Belew R. K., "Adaptative Information Retrieval", In Proceedings of the 12th Anuual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 11-20, 1989.

[Belkin et al., 1993] Belkin N. J., Cool C., Croft W. B., et Callan J. P., "The effect of multiple query representations on information retrieval performance", Dans : Proceedings of the 16th Annual ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 339-346, 1993.

[Bellifemine et al., 2003] Bellifemine F., Caire G., Trucco T., et al., "JADE, Programmer's Guide", Février 2003.

[Bellot, 2002] Bellot D., "Fusion des données avec des Réseaux Bayésiens pour la modélisation des systèmes dynamiques et son application en télémédecine", Thèse de Doctorat en Informatique, Université Henri Poincaré, Nancy I, Novembre 2002.

[Ben Ahmed, 2000] Ben Ahmed M., "Des sciences cognitives aux agents intelligents", Séminaire hebdomadaire du laboratoire RIADI-GDL, Ecole Nationale des Sciences de l'Informatique, Tunisie 2000.

[Ben Ahmed, 2007] Ben Ahmed M., "Cognition entre philosophie, science et technologie", Edition Centre de Publication Universitaire (CPU), Tunis, Tunisie 2007.

[Ben Amor et al., 2002] Ben Amor N., Benferhat S., Dubois D., Mellouli K., Prade H., "A theoretical framework for possibilistic independence in a weakly ordered setting", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10(2), p. 117-155,, 2002.

[Ben Amor et al., 2003] Ben Amor N., Benferhat S., Dubois D., Mellouli K., Prade H., "Anytime Propagation Algorithm for Min-based Possibilistic Graphs", Soft Computing, A fusion of foundations, methodologies and applications, vol. 8, p. 150-161, 2003.

[Ben Amor et al., 2006] Ben Amor N., Benferhat S., Smaoui S., "Inférence dans les réseaux possibilistes bases sur le conditionnement ordinal", dans la Revue d'Intelligence Artificielle (RIA), Numéro spécial sur les modèles graphiques dans les théories non probabilistes, Edition Hermès/Lavoisier, Octobre 2007.

[Benferhat et al., 1999] Benferhat S., Dubois D., Garcia L., and Prade H., "Possibilistic logic bases and possibilistic graphs", In Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 57–64, 1999.

[Ben Farhat et al., 2002] Ben Farhat S., Dubois D., Garcia L., Prade H., "On the transformation between possibilistic logic bases and possibilistic causal networks", Int. Journal of Approximate Reasoning, 29 (2) : 135-173, 2002.

[Ben Mena et al., 2005] Ben Mena T., Bellamine B. N., Ben Ahmed M., "Objective oriented approach evaluating multi-agents platforms", In SETIT'2005 Conference, Hammamet, Tunisia 2005.

[Berenji, 1988] Berenji H. R., "The Treatment of Uncertainty in Artificial Intelligence", NASA Armes Research Center/Heer Associates Inc (USA), 1988.

[Berry et Linof, 1997] Berry M. J. A., et Linof, G., "Data Mining : Techniques appliquées au marketing, à la vente, et aux services clients". Paris : InterEditions, 1997.

[Boissier et al., 1999] Boissier O., Guessoum Z., Occello M., "Plates-formes multi-agents", AFIA n°37, Octobre 1999.

[Boissier et al., 2002] Boissier O., Guessoum Z., Occello M., "Un essai de définition de critères pour l'étude comparative de plates-formes multi-agents", TSI, Numéro thématique : Environnement de développement de systèmes multi-agents, 21(4) : 549-553, avril 2002.

[Bookstein et Swanson, 1974] Bookstein A., et Swanson D., "Probabilistic models for automatic indexing", Journal of the American Society for Information Science (JASIS), 25, pp. 312 – 318, 1974.

[Borgelt et al., 1998] Borgelt C., Gebhardt J., Kruse R., "Possibilistic Graphical Models", Proceedings of International School for the Synthesis of Expert Knowledge (ISSEK'98), Udine (Italy), p. 51-68, 1998.

[Borgelt et al., 2000] Borgelt C., Gebhardt J., et Kruse R., "Possibilistic Graphical Models. Computational Intelligence in Data Mining", CISM Courses and Lectures 408, pp. 51-68, 2000.

[Boughanem, 2000] Boughanem M., "Formalisation et spécification des systèmes de recherche et de filtrage d'information", Mémoire d'habilitation (HDR), Université Paul Sabatier, Toulouse III, 2000.

[Bounhas, 2006] Bounhas I., "Un analyseur de contenu des documents scientifiques du Web", Mémoire de Mastère en Informatique, Ecole Nationale des Sciences de l'Informatique, Tunisie 2006.

[Brewington et al., 1999] Brewington B., Gray R., Moizumi K., Kotz D., Cybenko G., et Rus D., "Mobile agents indistributed information retrieval". In Intelligent Information Agents. Springer-Verlag, 1999.

[Brini et Boughanem, 2003] Brini A. H., et Boughanem M., "Relevance feedback : introduction of partial assessments for query expansion", In Proc. of the Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), Conf, Zittau, Allemagne, pp. 67-72, 2003.

[Brini et al., 2004a] Brini A. H., Boughanem M., et Dubois D., "Towards A possibilistic approach for information retrieval", In Proc. of the conference EUROFUSE, Data and Knowledge Engineering, pp. 92-102, 2004.

[Brini et al., 2004b] Brini A. H., Boughanem M., et Dubois D., "Une approche possibiliste pour la recherche d'information", In Logique Floue et ses Applications (LFA 2004), pp. 51-58, 2004.

[Brini et al., 2004c] Brini A. H., Boughanem M., et Dubois D., "Vers Une approche possibiliste pour la recherche d'information", In veille Stratégique Scientifique et Technologique (VSST 2004), pp. 55-65, 2004.

[Brini et al., 2005a] Brini A. H., Boughanem M., et Dubois D., "A model for information retrieval based on possibilistic networks", In Proc. of the symposium on String Processing and Information REtrieval (SPIRE 2005), Buenos Aires (Argentine), LNCS, Springer-Verlag, pp. 271-282, 2005.

[Brini et al., 2005b] Brini A. H., Campos, L., Dubois D., et Boughanem M., "Query propagation in possibilistic information retrieval networks", In Proc. of the Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2005), 2005.

[Brini, 2005] Brini A. H., "Un modèle de recherche d'information basé sur les réseaux possibilsites", Thèse de doctorat en informatique, Université de Toulouse III, Université Paul Sabatier (UPS), Toulouse (France), 2005.

[Brini et al., 2007] Brini A. H., Boughanem M., et Dubois D., "Un modèle de réseau possibiliste pour la recherche d'information", Dans : Information - Interaction - Intelligence, Cépaduès Editions, Vol. 7, N. 1, pp. 31-54, 2007.

[Buckley et Salton, 1995] Buckley C., et Salton G., "Optimization of relevance feedback weights", Dans : Proceedings of the Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 351-357, Seattle WA, 1995.

[Buntine, 1994] Buntine W., "Representing Learning with graphical Models". Technical Report, FIA 94-14, Artificial Intelligence Research Branch, NASA Armes Research Center, USA, 1994.

[Busetta et al., 1999] Busetta P., Ronnquist R., Hodgson A., et al., "Jack Intelligent Agents: Components for Intelligent Agents in Java ", AgentLink News Letter, Janvier 1999.

[Calado et al., 2003] Calado P., Cristo M., De Moura E., Ziviani N., Ribeiro-Neto B., et Gonçalves M. A., "Combining link-based and content-based methods for web document classification", In Proc. of ACM Conference on Information and Knowledge Management (CIKM), pp. 394 – 401, 2003.

[Cardon et al., 2001] Cardon A., Bertelle C., Olivier D., "Modélisation et implémentation des systèmes complexes – Mise en œuvre des systèmes multi-agents Exemples de pletes-formes", DEA Informatique Théorique et Applications, Le Havre (France), 2001.

[Chauhan, 1997] Chauhan D., "JAFMAS : A Java-Based Agent Framework for Multi-agent Systems Development and

Implementation", Master Thesis, ECECS Department, University of Cincinnati, Juillet 1997.

[Chavez et Cooper, 1990] Chavez R., Cooper G., "A Randomized Approximation Algorithm for Probabilistic Inference on Bayesian Belief Networks", Networks, vol. 20, p. 661-685, 1990.

[Chen, 1995] Chen H., "Machine learning for information retrieval: Neural Networks, Symbolic Learning and Genetic Algorithms", In Journal of the American Society for information Science, 46(3): 194-216, 1995.

[Chouaib, 2006] Chouaib H., "Reformulation de requêtes dans un modèle de réseau possibiliste pour la recherche d'information", Mémoire de DEA en Informatique, Faculté des Sciences, Université Libanaise, 2006.

[Cleverdon, 1960] Cleverdon C., "Cranfield research project: Report on the first stage of an investigation into the comparative efficiency of indexing systems", Cranfield: The College of Aeronautics, http://www.gnu.org, 1960.

[Cleverdon, 1970] Cleverdon C., "Progress in documentation, evaluation of information retrieval system", Journal of Documentation 26, pp. 55 – 67, 1970.

[Cleverdon, 1977] Cleverdon C., "Comparative evaluation of searching by controlled and natural language in a NASA database", European Space Agency Report 1/432, 1977.

[Cluzeau-Ciry, 1988] Cluzeau-Ciry M., "Typologie des utilisateurs et des utilisations d'une banque d'images", Le documentalistes, 25(3): 155-120, 1988.

[Coelho et al., 2002] Coelho H., Marietto M. B., David N., et al., "Requirements Analysis of Multi-agent Based Simulation Platforms: State of the Art and New Prospects", Proceedings of Multi-agent Based Simulation WorkShop, Bologna, Italy, July 2002.

[Cohen et Kjeldsen, 1987] Cohen P. R., et Kjeldsen R., Information Retrieval by constrained spreading activation in semantic networks, In Information Processing and Management, 23(4):255-268, 1987.

[Collis et Ndumu, 1999] Collis J., et Ndumu D., "The Zeus Agent Building ToolKit", Technical Manual, September 1999.

[Cooper, 1990] Cooper G., "The computational complexity of probabilistic inference using bayesian belief networks", In Artificial Intelligence, 42 (2-3): 393 – 405, 1990.

[Cornuéjols et Miclet, 2002] Cornuéjols A., Miclet L., "Apprentissage artificiel: Concepts et algorithmes", édition Eyrolles, page 364-365, 2002.

[Côté et al., 2002] Côté, M., Chaib-draa, B., et Troudi., N., "NetSA : une architecture multi-agent réutilisable pour les environnements riches en informations". Séries

Scientifiques. Centre Interuniversitaire de Recherche en ANalyse des Organisations (CIRANO), Canada, 2002.

[Crawford et al., 1991] Crawford S. L., Fung R., Appelbaum L. A., et Tong R. M., "Classification Trees for information retrieval", Dans : Proceedings of the 8th International Workshop on Machine Learning, CA : Morgan Kaufman, pp. 245-249, 1991.

[Crestani et al., 2003] Crestani F., De Campos L. M., Fernandez-Luna J. M., et Huete J. F., "A Multi-layered Bayesian Network Model for Structured Document Retrieval", ECSQARU 2003, LNAI 2711, Berlin Springer-Verlag, pp. 74-86, 2003.

[Croft, 1981] Croft W. B., "Document representation in probabilistic models of information retrieval", In Journal of the American Society for Information Science, pp. 451-457, Novembre 1981.

[Croft et Thompson, 1987] Croft W. B., et Thompson R. H., I3R : A new approach to the Design of Document Retrieval Systems, In The Journal of the Americain society for Information Science, 38: 389-404, 1987.

[Daniels, 1986] Daniels P. J., "Cognitive models in information retrieval – an evaluation review", in Journal of documentation, 42(4) : 272-304, Décembre 1986.

[De Campos et al., 1999a] De Campos L., Huete J. F., "Independence concepts in possibility theory: Part I", Fuzzy Sets and Systems, vol. 103, p. 127-152, 1999a.

[De Campos et al., 1999b] De Campos L., Huete J. F., "Independence concepts in possibility theory: Part II", Fuzzy Sets and Systems, vol. 103, p. 487-505, 1999b.

[De Campos et al., 2002] De Campos L., Fernandez-Luna J., et Huete J., "A layered bayesian network model for document retrieval", In Proc. of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval, pp. 169 – 182, 2002.

[De Campos et al., 2003] De Campos L. M., Fernandez-Luna J. M., et Huete J. F., "The BNR Model: foundations and performance of Bayesian Network-based retrieval model", JASIST, 54(4): 302-313, 2003.

[Dechter, 1996] Dechter R., "Bucket elimination: A unifying framework for probabilistic inference", In Proc. of the Twelth Conf. on Uncertainty in Artificial Intelligence, Horvits E., and Jensen F., (Eds), pp. 211-219, 1996.

[Delgado, 2000] Delgado M., "A Multiagent Architecture for Fuzzy Modeling", Département de Systèmes Informatique de l'Université de Murcia (Spain), 2000.

[Deloach et Wood, 2001] Deloach S. A., et Wood M., "Developing Multi-agent Systems with AgentTool", ATAL'2000, Berlin, 2001.

[Demazeau, 1995] Demazeau Y., "From interactions to collective behaviour in agent-based systems", In Proc. Of the first European Conf. on Cognitive Science, Saint Malo (France), pp. 117-132, 1995.

[Denoue et Vignollet, 2000] Denoue L., et Vignollet L., "An annotation tool for Web browsers and its application to information retrieval", Recherche d'informations Assistée par Ordinateur (RIAO2000), 12-14, Paris, avril 2000.

[Dubois et Prade, 1987] Dubois D. et Prade H., "Théorie des Possibilités : Application à la Représentation des Connaissances en Informatique", Paris : Edition MASSON, 1987.

[Dubois et Prade, 1988] Dubois D., and Prade H., "Possibility Theory", Plenum, New York (USA), 1988.

[Dubois et al., 1994] Dubois D., Lang J., Prade H., "Possibilistic logic ", Handbook on Logic in Artificial Intelligence and Logic Programming, vol. 3, Oxford University press, p. 439-513, 1994.

[Dubois et Prade, 1998] Dubois D., and Prade H., "Possibility theory : qualitative and quantitative aspects", Dans : Quantified Representation of Uncertainty and Imprecision. Dov M. Gabbay, Philippe Smets (Eds.), KLUWER ACADEMIC PUBLISHERS, The Netherlands, p. 169-226, Vol. 1, Handbook of Defeasible Reasoning and Uncertainty Management Systems, 1998.

[Duvignau et al., 2000] Duvignau K., Fabre C., Ferraty F., Gasquet O., Gaume B., Jouve B., Lang J., et Pery-Woodley M.P., (2000). "Les dictionnaires de langue : des graphes aux propriétés topologico-sémantiques ? ", Etats Généraux du Programme de REcherches en Sciences COgnitives de Toulouse (PRESCOT), Toulouse (France), 2000.

[Duvignau, 2002] Duvignau K., "La métaphore berceau et enfant de la langue", Thèse de doctorat, Université Toulouse - Le Mirail 2002.

[Duvignau, 2003] Duvignau K., "Métaphore verbale et approximation", Revue d'Intelligence Artificielle (RIA), 17(5/6) : 869–881, 2003.

[Dominich, 2001] Dominich S., "Mathematical Foundations of Information Retrieval", Kluwer Academic Publishers, Dordrecht, Boston, London 2001.

[Douglas et Houseman, 2002] Douglas W., et Houseman M., "The navigability of strong ties: small worlds, ties strength and network topology", eScholarship Repository, University of California 2002.

[Dunlop, 1997] Dunlop M. D., "The effect of accessing non matching documents on relevance feedback", In ACM Transactions on Information Systems, 15(2):137-153, 1997.

[Efthimiadis, 2000] Efthimiadis E., "Interactive query expansion : a user based evaluation in relevance feedback environment", Journal of

the American Society for Information Science, vol. 51, no 11, pp. 989-1003, 2000.

[Elayeb et al., 2006] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "SARIPOD : a system based on hierarchical small-worlds and possibilistic networks for Internet information retrieval", In IADIS International Conference WWW/Internet 2006, Katia Sycara, Fausto Giunchiglia (Eds.), IADIS Digital Library, (on line), pp. 227-232, Murcia (Spain), October 2006.

[Elayeb et al., 2007a] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "SARIPOD: A Possibilistic System For Web Information Retrieval", In The international conferences on Internet Technologies and Applications (ITA'07), Grout V., Oram D., et Picking R., (Eds), pp. 82-91, Wrexham Wales (UK), September 2007.

[Elayeb et al., 2007b] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "SARIPOD: A Multiagent Possibilistic System For Web Information Retrieval", In The 2007 International Conference on Information and Knowledge Engineering (IKE'07), Hamid R. Arabnia et Ray R. Hashemi (Eds), pp. 72-78, Las Vegas Nevada (USA), June 2007.

[Elayeb et al., 2007c] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "SARIPOD: Towards A Multiagent Possibilistic System For Web Information Retrieval", In The 2nd International Conference on Software and Data Technologies (ICSOFT'07), Barcelona (Spain), July 2007.

[Elayeb et al., 2007d] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "Vers une architecture à base des Réseaux Petits Mondes Hiérarchiques et des Réseaux Possibilistes pour les environnements riches en informations", Dans l'Atelier d'Intelligence Artificielle et Web Intelligence (IAWI 2007), Grenoble (France), juillet 2007. En ligne : http://www.emse.fr/~vercouter/iawi/10-elayeb.pdf

[Elayeb et al., 2008] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "A Qualitative Approach for an Intelligent Possibilistic Web Information Retrieval using Multiagent System", In The IEEE - 6th International Conference in Human System Learning (IEEE-ICHSL6), Saïd Tazi and Khaldoun Zreik (Eds), pp. 2-13, Toulouse (France), 14-16 May 2008.

[Elayeb et al., 2009] Elayeb B., Evrard F., Zaghdoud M., et Ben Ahmed M., "Towards An Intelligent Possibilistic Web Information Retrieval using Multiagent System", Dans : The International Journal of Interactive Technologie and Smart Education (ITSE), Special issue: New learning support systems, Emerald Group Publishing Limited, Volume 6, issue 1, pp. 40-59, 2009.

[Eichmann, 1996] Eichmann D., "Interaction Protocols for Software Agents on the World Wide Web", in WWW5 Workshop on Artificial Intelligence-based tools to help W3 users, Paris, 6 may 1996.

[Fabiani, 1996] Fabiani P., "Représentation Dynamique de l'Incertain et stratégie de Prise d'Information pour un Système Autonome en Environnement Evolutif", Thèse de Doctorat en Automatique et Informatique Industrielle, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1996.

[Fan et Gauch, 1997] Fan Y. et Gauch S., "An Adaptative Multi-Agent Architecture for the ProFusion Meta Search System", Proc. of WebNet'97-The Second World Conference of the Web Society, Toronto (ON), November 1997.

[Fan et Gauch, 1999] Fan Y., et Gauch S., "Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources", AAAI Symposium on Intelligent Agents in Cyberspace, Stanford University, March 1999.

[Ferber, 1995] Ferber J., "Les systèmes multi-agents, vers une intelligence collective", InterEditions, Paris, 1995.

[Ferber et al., 2000] Ferber J., Gutknecht O., et Michel F., "Madkit: une expérience d'architecture de plate-forme multi-agent générique", dans le 8ième Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'2000), La Réunion, Hermès, pp. 223-236, 2000.

[Fernandez et al., 2003] Fernandez-Luna J. M., De Campos L. M., et Huete J. F., "Two Term-Layers: an Alternative Topology for Representing Term Relationships in the Bayesian Network Retrieval Mode", Advances in Soft Computing-Engineering, Design and Manufacturing, pp. 213-224, 2003.

[Finetti, 1937] Finetti D. B., "La Prévision : Ses Lois logiques, Ses Sources Subjectives", Annales de l'Institut de Henri Poincaré, 7, 1937.

[Fonck, 1994] Fonck P., "Réseaux d'inférence pour le raisonnement possibiliste", PhD thesis, Université de Liège, Faculté des Sciences, 1994.

[Fox et Shaw, 1994] Fox E. A., et Shaw J. A., "Combination of multiple searches", Dans : Proceedings of the 2nd Text REtrieval Conference (TREC-2), pp. 243-252, 1994.

[François et al., 2003] François C., Hoffmann M., Lamirel J. C., et Polanco X., "Multi-Maps SOM Platform", Rapport technique, IST-1999-20350, France, 2003.

[François et Leray, 2003] François O., et Leray Ph., "Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux

Bayésiens", In F. D. de Saint-Cyr, Ed., RJCIA2003 – 6ième Rencontres Nationales des Jeunes chercheurs en Intelligence Artificielle, Presse Universitaire de Grenoble, pp. 167-180, 2003.

[François et Leray, 2004] François O., et Leray Ph., "Étude Comparative d'Algorithmes d'Apprentissage de Structure dans les Réseaux Bayésiens", Dans Journal Électronique d'Intelligence Artificielle (JEDAI), 2004.

[Frisse, 1988] Frisse M., "Searching for information in a hypertext medical dandbook", In Communication of the ACM (CACM), 31(7) : 880–886, 1988.

[Frisse et Cousins, 1989] Frisse M., et Cousins S., "Information retrieval from hypertext: Update on the dynamic medical dandbook", In Proc. of ACM Hypertext Conference, pp. 199 –211, 1989.

[Fuhr et Buckley, 1991] Fuhr N., Buckley C., "Probabilistic learning approach for document indexing", In ACM Transactions on Information Systems, 9(3): 223-248, 1991.

[Fuhr, 1992] Fuhr N., "Probabilistic models in information retrieval", In The Computer Journal, 35(3) : 243–255, 1992.

[Garneau et Delisle, 2002] Garneau T., et Delisle S., "Programmation orientée-agent : évaluation comparative d'outils et environnements", JFIADSMA, 2002.

[Gaume et al., 2002] Gaume B., Duvignau K., Gasquet O. et Gineste M.D., "Forms of meaning, meaning of forms", Journal of Experimental and Theoretical Artificial Intelligence, 14(1) : 61–74, 2002.

[Gaume, 2004] Gaume B., "Balades aléatoires dans les petits mondes lexicaux", dans I3 Information Interaction Intelligence, 2004.

[Gaume et al., 2004] Gaume B., Hathout N., et Muller P., "Désambiguïsation par proximité structurelle", TALN 2004, Fès (Marroc), pp. 19-21, avril 2004.

[Gaume et Ferré, 2004] Gaume B., et Ferré L., "Représentation de graphes par ACP granulaire", In Actes d'EGC 2004 : 4èmes journées d'Extraction et de Gestion des Connaissances, Clermont-Ferrand 2004.

[Gaume, 2006] Gaume B., "Cartographier la forme du sens dans les petits mondes Lexicaux", In: JADT 2006, pp. 541-465, Besançon (France), Avril 2006.

[Gaume et al., 2006] Gaume B., Duvignau K., Mas J. M., "Petits Mondes Hiérarchiques et dynamique d'acquisition pour l'enseignement du lexique", In: Technologies langagières et apprentissage des langues ACFAS, Montréal, 2006.

[Gaume et Mathieu, 2007] Gaume B., Mathieu F., "PageRank Induced Topology for Real-World Networks", In Complex Systems, volume (year) 1–1+; year Complex Systems Publications, 2007.

[Gaume et al., 2007] Gaume B., Duvignau K., Vanhove M., "Semantic associations and confluences in paradigmatic networks", In: M. Vanhove (éd.), Typologie des rapprochements sémantiques (Linguistic), 2007.

[Gacôgne, 1997] Gacôgne L., "Eléments de la Logique Floue", Edition Hermes, 1997.

[Giarratano et al., 1989] Giarratano et Riley, "Expert System : Principle and Programming", PWS-KENT Publishing Company, 1989.

[Graham et al., 2000] Graham J. R., Mchugh D., Mersic M., et al., "Tools for Developing and Monitoring Agents in Distributed Multi-agent Systems", Agents Workshop on Infrastructure for Multi-agent Systems, pp. 12-27, 2000.

[Haines et Croft, 1993] Haines D., et Croft W.B., "Relevance Feedback and Inference Networks", Conference on Research and Development in Information Retrieval (SIGIR), 1993.

[Hallouli, 2004] Hallouli K., "Reconnaissance de caractères par méthodes markoviennes et réseaux Bayésiens", Thèse de Doctorat spécialité Signal et Images, Ecole Nationale Supérieur des Télécommunications, Télécom Paris, Mai 2004.

[Hamers et al., 1989] Hamers L., Hemeryck Y., Herweyers G., Janssen M., Keters H., Rousseau R., et Vanhoutte A., "Similarity measures in scientometric research: the Jaccard index versus Salton's cosine formula", Information Proceeding and Management, 25(3): 315-318, 1989.

[Hammer et Fiedler, 2000] Hammer J., et Fiedler J., "Using mobile crawlers to search the Web eficiently", Int. Journal of Computer and Information Science, 1(1) : 36-58, 2000.

[Harman, 1992] Harman D., "Relevance feedback and other query modification techniques", In Information Retrieval: Data Structures an Algorithms, William B., Frakes and Ricardo Baeza-Yates, editors, Prentice Hall, Englewood, Cliffs, NJ, pp. 241–263, 1992.

[Harman et al., 1992] Harman D., Fox E., Baeza-Yates R., et Lee W., "Information Retrieval: data structures and algorithms", Chapitre 3: Inverted files, pp. 28-43, William B., Frakes and Ricardo Baeza-Yates, Prentice Hall edition, 1992.

[Harter, 1975] Harter S., "A probabilistic approach to automatic keyword indexing. Part ii. an algorithm for probabilistic indexing", In Journal of the American Society for Information Science (JASIS), 35(3) : 280–289, 1975.

[Hermans, 1997] Hermans, B., "Intelligent software agent on the internet: an inventory of currently offered functionality in the

information society and a prediction of (near-) future developments", Master's thesis, Tiburg University, Tiburg, The Netherlands, 1997.

[Howard et Matheson, 1981] Howard R. A., et Matheson, J. E., "Influence diagrams", In Howard, R. A., and Matheson, J. (Eds.), The Principles and Applications of Decision Analysis, pp. 720–762. Strategic Decisions Group, CA, 1981.

[Huhns et Singh, 1997] Huhns M. N., et Singh M. P., "Readings in AGENTS", Morgan Kaufmann Publishers, Inc, ISBN 1- 55860-495-2, 523 P, 1997.

[Ide, 1971] Ide E., "New experiments in relevance feedback", In G. Salton (Ed.) The Smart System-Experiments in Automatic Document Processing, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1971.

[Ide et Véronis, 1998] Ide N., et Véronis J., "Introduction to the special issue on word sense disambiguation: The state of the art", Computational Linguistics, 24(1), 1998.

[Ingwersen, 1992] Ingwersen P., "Information retrieval interaction", Taylor and Graham, London (UK), 1992.

[Jaakkola et Jordan, 1999] Jaakkola T. S., et Jordan M. I., "Variational probabilistic inference and the QMR-DT network", Journal of Artificial Intelligence Research, vol. 10, pp. 291-322, 1999.

[Jensen et al., 1990] Jensen F. V., Lauritzen S. L., et Olesen K. G., "Bayesian updating in recursive graphical models by local computations", Computational Statistical Quaterly, vol. 4, pp. 269-282, 1990.

[Jensen, 1996] Jensen F., "An introduction to Bayesian Networks", UCL Press, University college, London, 1996.

[Jensen, 2000] Jensen F. V., "Bayesian Networks and Decision Graphs", Wiley, 2000.

[Jones et al., 2000] Jones K. S., Walker S., and Robertson S., "A probabilistic model of information retrieval: development and comparative experiments, parts 1 & 2", In Information Processing and Management (IPM), 36(6) : 779–808, 809–840, 2000.

[Jordan et al., 1999] Jordan M. I., Ghahramani Z., Jaakkola T. S., et Saul L. K., "An introduction to variational methods for graphical models", In Machine Learning, vol. 37, pp. 183-233, 1999.

[Jordan et Weiss, 2001] Jordan M. I., et Weiss Y., "probabilistic inference", In Graphical models, MIT Press, Five Cambridge Center, MA 02142-1493 USA, 2001.

[Joshi et al., 1999] Joshi A., Singh M. P., et Ma M., "Special section on multiagent systems on the net and agents in e-commerce", Communications of the ACM, 42(3), 1999.

[Joubert et al., 1991] Joubert M., Fieschi M., Botti G., et Fieschi D., "Etude d'un modèle de représentation des concepts médicaux pour la recherche d'information", Dans Informatique et Santé (4) : 101-112, Paris France, 1991.

[Kammoun-Bouzaïene, 2006] Kammoun-Bouzaïene H., "Collaboration de modèles symbolique et numérique pour une recherche d'information adaptative, évolutive et coopérative", Thèse de Doctorat en Informatique, Ecole Nationale des Sciences de l'Informatique (ENSI), Tunisie, 2006.

[Kekäläinen et Järvelin, 2002] Kekäläinen J., et Järvelin K., "Evaluating information retrieval systems under the challenges of interaction and multidimentional dynamic relevance", In Bruce H., Fidel R., Ingwersen P., Vakkari P., (eds). Emerging Frameworks and Methodes, Seattle, Colerado: Libraries Unlimited, pp. 253–270, 2002.

[Kettani, 1998] Kettani N., "De Merise à UML", Paris : Edition Eyrolles, 1998.

[Kim et Pearl, 1983] Kim J. H., et Pearl J., "A computational model for combined causal and diagnostic reasoning in inference systems", In Proceedings of IJCAI-83, pp. 190-193, Karlsruhe, Germany, 1983.

[Kim et Pearl, 1987] Kim J. H., et Pearl J., "CONVINCE: A Conversational Inference Consolidation Engine", In IEEE Trans. on Systems, Man and Cybernetics, vol. 17, pp. 120-132, 1987.

[Kjaerulf, 1990] Kjaerulf U., "Triangulation of graphs algorithms giving small total state space", Dept. of Maths. And Comp. Sci. Technical Report r-90-09 edition, 1990.

[Kohonen et al., 1996] Kohonen T., Kaski S., Lagus K., et Honkela T., "Self organization maps", In Proceedings ICNN'97, International Conference on Knowledge Discovery and Data Mining, pp. 238-243, 1996.

[Kosch et al., 2001] Kosch H., Doller M. et Boszormenyi L., "Content-based indexing and retrieval supported by mobile agent technology", In The Second International Workshop on Multimedia Databases and Image Communication, pp. 152-166, 2001.

[Kraft et al., 1983] Kraft, D. H., et Buell, D. A., "Fuzzy sets and generalized Boolean retrieval systems", in International Journal on Man-Machine Studies, 19, 49–56, 1983.

[Lamirel, 1995] Lamirel J. C., "Application d'une approche symbolico-connexioniste pour la conception d'un système documentaire hautement interactif, le prototype NOMAD", Thèse de Doctorat en Informatique, Université Henri-Poincaré, Nancy I, Nancy (France), 1995.

[Lauritzen, 1988] Lauritzen S., "Local computation with probabilities on graphical structures and their application to expert systems", In Journal of the Royal Statistical Society, vol. 50, pp. 157, 1988.

[Lauritzen et al., 1988] Lauritzen S. L., Spiegelhalter D. J., "Local computations with probabilities on graphical structures and their application to expert systems", Journal of the Royal Statistical Society, vol. 50, p. 157-224, 1988.

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

[Lelu et François, 1992] Lelu A., François C., "Information retrieval based on a neural unsupervized of thematic fuzzy clusters", Les réseaux neuromimétiques et leurs applications (NeuroNîmes), pp. 93-104, 1992.

[Lesk, 1986] Lesk M., "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone", In Proceedings of the 5th annual international conference on Systems documentation, pp. 24–26, Toronto (Canada), 1986.

[Liebermann, 1995] Liebermann H., "Letizia, An Agent for Web Browsing", International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, August 1995.

[Lopez et al., 1998] Lopez N., Migueis J., et Pichon E., "Intégrer UML dans vos projets", Paris : Edition Eyrolles, 1998.

[Mackay, 1999] Mackay D., "An introduction to Monte Carlo Methods", In Learning in Graphical Models, MIT Press, pp. 175-204, 1999.

[Macquoy et al., 2002] Macquoy A., Michat S., Parisot V., Sevestre M., et Zimero N., "Documentation Projet Dilan", Rapport de stage, Institut de Recherche en Informatique de Toulouse (IRIT), 2002.

[Maes, 1994] Maes P., "Agents that Reduce Work and Information Overload", Communications of the ACM, 37(7), 1994.

[Mandala et al., 1972] Mandala R., Tokunaga T., et Tanaka H., "Combining multiple evidence from different types of thesaurus for query expansion", Proc. of the International ACM-SIGIR Conference, 1, pp. 191–197, 1972.

[Meganck, 2006] Meganck S., Leray Ph., Maes S., Manderick B., "Apprentissage des réseaux bayésiens causaux à partir de données d'observation et d'expérimentation Learning causal bayesian networks from data and manipulation", Dans 15e congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle, Tours (France), 25 au 27 janvier 2006.

[Miller et Bharat, 1998] Miller, R. C., et Bharat, K., "SPHINX: A framework for creating personal, site-specific web crawlers", In the 7th International World Wide Web Conference (WWW7). Printed in Computer Network and ISDN System Vol.30, pp. 119-130, Brisbane, Australia, 1998.

[Mizzaro, 1997] Mizzaro S., "Relevance : the hole history", In the Journal of the American Society for Information Science, 48(9): 810-832, 1997.

[Mothe et Dkaki, 1998] Mothe J., et Dkaki T., "Interactive multidimentional document visualisation", In Proceedings of the Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 363-364, Sydney (Australia), 1998.

[Ndumu et al., 1998] Ndumu D. T., Lee L. C., et Nwana H. S., "ZEUS : An Advanced ToolKit for Engineering Distributed Multi-Agent Systems", In Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Systems, pp. 377-392, Londres, 1998.

[Newman, 2003] Newman M. E. J., "The structure and function of complex networks", SIAM Review, Vol. 45, 167–256, 2003.

[Nguyen, 1979] Nguyen H. T., "Some Mathematical Tools for Linguistic Probabilities", in Fuzzy Sets and Systems, 2, 53-65, 1979.

[Nie, 1988] Nie J-Y., "An outline of general model for information retrieval systems", In Proceedings of the Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp.495-506, 1988.

[Pearl, 1982] Pearl J., "Reverand Bayes on inference engines: A distributed hierarchical approach", In Proceedings of the AAAI National Conference on AI, pp. 133-136, Pittsburgh, 1982.

[Pearl, 1988] Pearl J., "Probabilistic reasoning in intelligent systems : Networks of plausible Inference", Morgan Kaufman Publishers, Inc., San Mateo, CA, 2nd Edition, 1988.

[Ploux et Victorri, 1998] Ploux S., et Victorri B., "Construction d'espaces sémantiques à l'aide de dictionnaires de synonymes", TAL, 39(1) :161-182, 1998.

[Porter, 1980] Porter M. F., "An algorithm for suffix stripping", Program, 14(13): 130-137, 1980.

[Porter, 1982] Porter M. F., "Implementing a probabilistic information retrieval system", In Information Technology: Research and Development, pp. 131-156, 1982.

[Portrait, 2003] Portrait Y., "Modélisation de la structure du langage", Rapport de stage de fin d'études en informatique et mathématiques appliquées, IRIT, Toulouse (France), 2003.

[Prade et Testemale, 1987] Prade H., et Testemale C., "Application of possibility and necessity measures to documentary information retrieval", In Uncertainty in Knowledge-Based Systems (B. Bouchon, R. Yager, Eds.), LNCS n°286, Berlin Springer-Verlag, 265-274, 1987.

[Qiu et Frei, 1993] Qiu Y., et Frei H., "Concept based query expansion", In Proc. of the International ACM-SIGIR Conference, pp. 160–169, 1993.

[Radecki, 1979] Radecki, T., "Fuzzy set theoretical approach to document retrieval". Information Processing & Management, 15: 247-259.

[Ravasz et Barabàsi, 2003] Ravasz E., et Barabàsi A., L., "Hierarchical Organisation in Complex Networks", In Phys.Rev. E, 67:026112-026118, 2003.

[Ribeiro-Neto et al., 1996] Ribeiro-Neto B., Silva I., et Muntz R., "A Belief Network Model for IR", Proc. of the 19th ACM-SIGIR Conf. on Research and Development in Information Retrieval, 253-260, 1996.

[Ricordel et Demazeau, 2000] Ricordel P. M., et Demazeau Y. "From Analysis to Deployment : A Multi-agent Platform Survey", ESAW, p. 93-105, 2000.

[Rijsbergen, 1977] Rijsbergen C. V., "A theoretical basis for the use of co-occurrence data in information retrieval", In Journal of Documentation, 33, 106 – 119, 1977.

[Rijsbergen, 1979] Rijsbergen C. V., "Information Retrieval", Butterworth – Heinemann, Newton (MA), 1979.

[Resnik et Yarowsky, 2000] Resnik P., et Yarowsky D., "Distinguishing systems and distinguishing senses : New evaluation methods for word sense disambiguation", Natural Language Engineering, 5(2), 113–133, 2000.

[Rimassa et al., 1999] Rimassa G., Bellifemine F., et Poggi A., "JADE – A FIPA Compliant Agent framework", PMAA'99, pp. 97-108, Londres, Avril 1999.

[Roberston et Sparck-Jones, 76] Roberston S. E., et Sparck-Jones J., "Relevance weighting of search terms", Journal of the American Siciety for Information Science, vol. 27, no 3, p. 129, 146, 1976.

[Roberston, 1986] Roberston S. E., "On relevance weight estimation and query expansion", Journal of Documentation, 42:182-188, 1986.

[Rocchio, 1971] Rocchio J., "Relevance feedback in information retrieval", The SMART retrieval system-experiments in automatic document processing, Prentice Hall Inc, pp. 313-323, 1971.

[Roques, 2001] Roques P., "UML par la pratique", Paris : Edition Eyrolles, 2001.

[Salton, 1971] Salton G., "The SMART retrieval system", Prentice-Hall, Englewood Cliffs, N. J., 1971.

[Salton et al., 1983a] Salton G., Fox E. A., et Wu, H., "Extended Boolean information retrieval", Communications of the ACM, 26(12): 1022-1036, 1983.

[Salton et al., 1983b] Salton G., Fox E. A., Buckley C., et Voorhees E., "Boolean query formulation with relevance feedback", Rapport technique, Ithaca, NY Cornell University, Departement of computer science, 1983.

[Salton et McGill, 1983] Salton G., et McGill M. J., "Introduction to modern information retrieval", McGraw-Hill, New York, 1983.

[Salton et Buckley, 1988] Salton G., et Buckley C., "On the use of spreading activation methods in automatic information retrieval", communications of the ACM, pp. 147-160, 1988.

[Salton, 1989] Salton G., "Automatic text processing", Addisson-Wesley, Reading, MA, USA, 1989.

[Salton et Buckley, 1990] Salton G., et Buckley C., "Improving Retrieval Performance By Relevance Feedback", In Journal of The American Society for Information Science (JASIS), 41(4) : 288-297, 1990.

[Salton et al., 1994] Salton G., Allan J., Buckley C., et Singhal A., "Automatic Analysis, Theme Generation and Summarization of Machine Readable Texts", Science, 264(3): 1421-1426. 1994.

[Sandri, 1991] Sandri S., "La Combinaison de l'Information Incertaine et ses Aspects Algorithmiques", Thèse de Doctorat en Informatique. Université de Paul Sabatier de Toulouse, 1991.

[Saracevic, 1970] Saracevic T., "The concept of relevance in information science: a historical review", Saracevic (ed.), Introduction to information science, chap. 3 - The concept of relevance, pp. 111-151, New York R.R. Bowker company, 1970.

[Saracevic, 1996] Saracevic T., "Relevance reconsidered", In Information science: Integration in perspectives, Proc. of the Conference on Conceptions of Library and Information Science, pp. 201 – 218, 1996.

[Sauvagna, 2005] Sauvagna K., "Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structurés", Thèse de Doctorat en informatique, Université Paul Sabatier, Toulouse (France), 2005.

[Savage, 1972] Savage L. J., "The Foundations of Statistics", Dover Publications, New York (USA), 1972.

[Savoy et al., 1991] Savoy J., Dubois D., et al., "Information Retrieval in Hypertext Systems an Approach Using Bayesian Networks", Electronic Pub., 4(2) : 87-108, 1991.

[Savoy, 1993] Savoy J., "Stemming of French words based on grammatical categories", Journal of the Americal Society for Information Science, 44(1): 1-9, 1993.

[Sergi et Ricard, 2007] Sergi V., et Ricard V. S., "Hierarchical Small-Worlds in Software Architecture", Dynamics of Continuous Discrete and Impulsive Systems: Series B; Applications and Algorithms 14, pp. 1-11, 2007.

[Schamber et al., 1990] Schamber L., Eisenberg M., et Nilan S. M., "A re-examination of relevance toward a dynamic, situational definition", Information Processing and Management, 26(6): 755-776, 1990.

[Scharffe, 2004] Scharffe F., "Croisements sémantiques dans les graphes petits mondes", Mémoire de DEA Représentation de la connaissance et Formalisation du Raisonnement. Université Paul Sabatier Toulouse III, Toulouse (France), 2004.

[Shibly et al., 2004] Shibly T., Choumane A., Gharib A., "Recherche des composantes de sens dans un graphe de dictionnaire de verbes par une approche basée sur l'étude des circuits", Mémoire de fin d'études, Faculté des sciences, université libanaise, 2004.

[Silva et al., 2000] Silva I., Ribeiro-Neto B., Calado P., Moura E., et Ziviani N., "Link-Based and Content-Based Evidential Information in a Belief Network Model", ACM/SIGIR 23rd Int. Conference on Information Retrieval, pp. 96–103, 2000.

[Singhal et al., 1996a] Singhal A., Buckley C., et Mitra M., "Pivoted document length normalisation", Proc. of the International ACM-SIGIR Conference, 32(2) : 21–29, 1996.

[Singhal et al., 1996b] Singhal A., Salton G, Mitra M., et Buckley C., "Document Length Normalization", Information Processing and Management (IPM), 32(5) : 619–633, 1996.

[Smeaton, 1983] Smeaton A. F., "The retrieval effects of query expansion on a feedback document retrieval system", The computer Journal, 26:239-246, 1983.

[Snowdon et al., 1996] Snowdon D., Fahlen L., et Stenius M., "WWW3D : A 3D multi-user Web browser", In WebNet96 Proceedings Online, San Fransisco, California (USA), Octobre 1996.

[Sparck-Jones, 1988] Sparck-Jones K., "A Look Back and a Look Forward", In Proceedings of the 11th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 13-29, 1988.

[Sugeno, 1974] Sugeno M., "Theory of Fuzzy integral and its Applications", Ph.D. Thesis, Tokyo Inst of Technology, Japan, 1974.

[Tanimoto, 1958] Tanimoto T., "An elementary mathematical theory of classification and prediction", Rapport IBM, 1958.

[Thati et al., 2001] Thati P., Chang P.H., et Agha G., "Crawlets: Agents for high performance Web search engines", In Mobile Agents 2001, LNCS 2240, pp. 119-134, Springer-Verlag, 2001.

[Tombros et Sanderson, 1998] Tombros A., et Sanderson M., "Advantages of query biased summaries in information retrieval", In Proceedings of the Anuual International ACM/SIGIR Conference on Research and Development in Information Retrieval, pp. 2-10, Sydney (Australia), 1998.

[Tsai et Lee, 2003] Tsai T. et Lee W., "An interactive agent-based system for concept-based Web search", Department of Management Information Systems, National Pingtung University of Science and technology, Taiwan, 2003.

[Turtle et Croft, 1990] Turtle H. R., et Croft W. B., "Inference networks for document retrieval", In Proc. 13th International Conference on Research and Development in Information Retrieval, 1–24, 1990.

[Turtle, 1991] Turtle H. R., "Inference networks for document retrieval", Ph.D. thesis, University of Massachusetts, USA, 1991.

[Turtle et Croft, 1991] Turtle H. R., et Croft W. B., "Evaluation of an inference network-based retrieval model", In ACM Transaction on Information System, 9(3) : 187–222, 1991.

[Van Rijisbergen, 1986] Van Rijisbergen C. J., "A non-classical logic for information retrieval", In Computer Journal, 29(6) : 481–485, 1986.

[Venant, 2003] Venant F., "Géométriser le sens. Les Journées Graphes, Réseaux et Modélisation", ESPCI, Paris, 2003.

[Victorri et Fuchs, 1996] Victorri B., et Fuchs C., "La polysémie, construction dynamique du sens", Paris : Edition Hermès, 1996.

[Vise et Malseed, 2006] Vise D., et Malseed M., "Google story", Paris : Edition Dunod, 2006.

[Veronis et Ide, 1990] Veronis, J., et Ide N. M., "Word Sense Disambiguisation whith Very Large Neural Networks Extracted from Machine Readable Dictionaries", COLING'90, 1990.

[Watts et Strogatz, 1998] Watts D. et Strogatz S., "Collective dynamics of 'small-world' networks", Nature, 393, 440–442, 1998.

[Waller et Kraft, 1979] Waller, W. G., et Kraft D. H., "A mathematical model for a weighted Boolean retrieval system", Information Processing & Management, 15, pp. 235-245, 1979.

[Walker et al., 1997] Walker S., Roberston S. E., Boughanem M., Jones G. J. F., Sparck-Jones K., "OKAPI at TREC-6", Dans : Proceedings of the 6th Text REtrieval Conference (TREC-6), NIST Special Publication, 1997.

[Wong et Raghavan, 1984] Wong S. K. M., et Raghavan V. V., "Development in information retrieval", Chapitre: Vector Space Model of Information Retrieval: a re-evaluation, pp. 167-185, University of Cambridge, England, 1984.

[Wooldridge et Jennings, 1995] Wooldridge M., et Jennings N. R., "Intelligents Agents: Theories and practice", The Knowledge Engeneering Review, 1995.

[Wooldridge et al., 1996] Wooldridge M., et al., "Agent Theories, Architectures and Languages : a Survey", Intelligent Agents, Wooldridge and Jennings (Eds), Berlin Springer-Verlag, 1-22, 1996.

[Wooldridge et Jennings, 1998] Wooldridge M., et Jennings N. R., "Pitfalls of agent-oriented development", Departement of Electronic Engineering, Queen Mary & Westfield College, University of London, Research report, 1998.

[Yates et Neto, 1999] Baeza-Yates R., et Ribeiro-Neto B., "Modern Information Retrieval", Addison-Wesley, 1999.

[Yuwono et al., 1997] Budi Yuwono, Savio L.Y. Lam, Jerry H. Ying, Dik L. Lee, A World Wide Web Ressource Discovery S ystem, Sino Software Research Center (SSRC), 1997, En ligne: http://www.w3.org/Conferences/WWW4/Papers/66/

[Zadeh, 1965] Zadeh L. A., "Fuzzy Sets", Information and Control, 8, 338-353, 1965.

[Zadeh, 1978] Zadeh L. A., "Fuzzy Sets as a basis for a theory of Possibility", Fuzzy Sets and Systems, Vol. 1, pp. 3-28, 1978.

[Zaghdoud, 2003] Zaghdoud M., "SAFIINA : Système basé multi-Agent de Fusion d'Informations INcertAines", Thèse de Doctorat en Informatique, Ecole Nationale des Sciences de l'Informatique, Tunisie 2003.

[Zhang et Poole, 1994] Zhang N. L., et Poole D., "A simple approach to Bayesian Network computations", In Proc. of the 10th Canadian Conference on Artificial Intelligence, vol. 71, pp. 171-178, 1994.

[Zayani, 2008] Zayani C., "Contribution à la définition et à la mise en oeuvre de mécanismes d'adaptation de documents semi-structurés", Thèse de Doctorat en informatique, Université Paul Sabatier, Toulouse (France), 2008.

[Zipf, 1949] Zipf G. K., "Human behaviour and the principle of least effort", Cambridge, Mass, Addison-Wesley, 1949.

Annexes

Annexe 1 :

Format XML du dictionnaire français Le Grand Robert

Le travail réalisé dans cette annexe consiste à reécrire une application faite sur la base d'une étude du dictionnaire initial 'dico_ini.xml' réalisée par [Macquoy et al., 2002].

En effet, nous présentons dans cette annexe les étapes de formalisation de la source de données utilisée pour le RPMH de dictionnaire. Il s'agit du dictionnaire français « *Le Grand Robert* » sous format XML. Cette source de données n'étant pas utilisable dans son état initial, il nous faut la traiter en plusieurs étapes pour arriver enfin à un format de données exploitable.

Au terme de ce traitement, nous obtenons un dictionnaire complet au format XML, mais de taille très importante (45 Mo) inmanipulable par le système SARIPOD en raison de contraintes d'espace mémoire. C'est ainsi que nous avons été obligés de subdiviser ce dictionnaire complet en quatre fichiers XML : un dictionnaire pour les verbes, un pour les noms, un pour les adjectifs et un dernier pour les adverbes. Les traitements réalisés sur ces quatre fichiers sont les mêmes.

La présente annexe est organisée de la façon suivante :

Nous présentons dans une première section les traitements préliminaires de récupération de la structure du dictionnaire. La création du fichier XML fera l'objet d'une seconde section. Dans la troisième section, nous présentons brièvement le taggage/lemmatisation et l'extraction des mots. La dernière section s'intéresse à la création des liens entrants pour les verbes.

1. Traitements préliminaires du dictionnaire

Les différentes catégories de texte dans le fichier 'dico_ini.xml' sont séparées, plus ou moins correctement par des balises. La première étape consiste à identifier ces balises, à préciser leur rôle, notamment en construisant la matrice états-transitions, puis à transformer ce fichier en un fichier XML en remplaçant au moyen du script perl 'taggerXML.pl' les balises suivant le tableau A1.1 [Macquoy et al., 2002].

dico_ini.xml	Balises XML	Fonction
ÎABAISSER [abese] v. tr	<DEF>	Nouvelle entrée
Ï\015\	<NIVEAU type= '1'>	1er niveau
Ï\007\	<NIVEAU type= '2'>	2ème niveau
Ï\004\	<NIVEAU type= '3'>	3ème niveau
Ï[a.]	<NIVEAU type= '4'>	4ème niveau
Ï\016\	<AUTREFORME>	p.p.adj. ou v.pron.
Í\016\	<MOTSLIES>	mots liés
É	<RE-MOTSLIES>	reprise des mots liés
Î	<META>	méta
Ë	<ITALIQUE>	italique
È	<CITATION>	citations
\009\	<CONTRAIRE>	contraires, dérivatifs, comparatifs et homonymes
Ì	<RETOUR>	fin de méta ou de mots liés
Ò	</ITALIQUE>	fin d'italique

Tableau A1.1 : Récupération de la structure du dictionnaire Le Grand Robert

2. Génération d'un dictionnaire au format XML

La première étape consiste à nettoyer le fichier initial des balises inutiles. Ensuite, il s'agit de remplacer les caractères accentués (qui apparaissaient dans le dictionnaire avec un code du type ^Z), le fichier 'dico.xml' respecte normalement la DTD suivante [Macquoy et al., 2002]. (voir figure A1.1) :

```
    <!-- Premier NIVEAU de l'arbre -->
    <!ELEMENT DICO (DEF+)>
    <!-- Deuxième NIVEAU de l'arbre -->
    <!ELEMENT DEF
(ENTRANT?,(STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU|AUTREFORME|PRONOMINAL)*
,(CONTRAIRE|DERIVATIF|COMPARATIF|HOMONYME)*)>
    <!ATTLIST DEF
              mot        CDATA #REQUIRED
              phonetique CDATA #REQUIRED
              cat        CDATA #REQUIRED>
    <!ELEMENT CONTRAIRE     (#PCDATA)*>
    <!ELEMENT DERIVATIF     (#PCDATA)*>
    <!ELEMENT COMPARATIF    (#PCDATA)*>
    <!ELEMENT HOMONYME      (#PCDATA)*>
    <!ELEMENT    AUTREFORME    (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>

    <!ELEMENT PRONOMINAL (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
    <!ELEMENT NIVEAU (STANDARD|ITALIQUE|META|MOTSLIES|NIVEAU)*>
    <!ATTLIST NIVEAU type (1|2|3|4) #REQUIRED>
    <!-- Troisième NIVEAU -->
    <!ELEMENT META        (STANDARD|ITALIQUE)*>
    <!ELEMENT MOTSLIES    (STANDARD|ITALIQUE)*>
    <!ELEMENT STANDARD    (#PCDATA)*>
    <!ELEMENT ITALIQUE     (#PCDATA)*>
```

Figure A1.1 : La DTD du fichier 'dico.xml'

Dans la plupart des cas, la phonétique du mot existe entre crochets. Cette phonétique facilite la reconnaissance des définitions des mots dans le dictionnaire. Ensuite, pour les entrées comportant des « ou », des « et » ou des virgules, il a été nécessaire de recopier la définition pour chaque synonyme. En revanche les féminins et pluriels des mots ont été supprimés.

Il reste le problème des entrées non repérées qui introduisent des défauts de structure. Ces dernières causent plusieurs autres anomalies dans le fichier 'dico.xml'. Une comparaison des entrées déjà reconnues avec la liste des entrées du dictionnaire s'avère nécessaire pour contribuer à la résolution d'un tel problème. La dernière phase consiste à faire un nettoyage manuel pour s'assurer de l'exhaustivité des données afin d'obtenir le fichier 'dico_final.xml'. La figure A1.2 décrit brièvement les étapes de création d'un fichier XML à partir du dictionnaire.

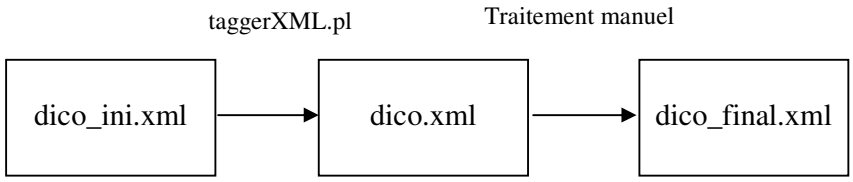


Figure A1.2 : Les étapes de création d'un fichier XML à partir du dictionnaire

3. Taggage/Lemmatisation et extraction des mots

La première phase consiste à appliquer les scripts 'TagScript1.pl', puis 'TagScript2.pl' afin de traiter le cas des verbes pronominaux.

Le script perl 'onlyVerbe.pl' permet d'engendrer un dictionnaire qui ne contient que les définitions des verbes intervenant dans toutes les définitions (dicoVerb.xml).

Le script perl 'TagScript1.pl' permet d'effectuer l'étape du balisage et de lemmatisation du dictionnaire XML-isé.

Le script perl 'TagScript2.pl' est chargé de traiter le cas des verbes pronominaux, une fois le dictionnaire XML-isé.

Ces mêmes traitements ont été faits pour obtenir les trois autres dictionnaires des noms, des adjectifs et des adverbes. En effet, les trois scripts perl 'onlyNom.pl', 'onlyAdjectif.pl' et 'onlyAdverb.pl' permettent d'engendrer respectivement les trois dictionnaires 'dicoNom.xml', 'dicoAdj.xml' et 'dicoAdv.xml'.

3.1 Taggage/Lemmatisation

Au début de cette étape, il faut que les balises du dictionnaire XML-isé ne contiennent pas d'attributs, sinon l'analyse avec le Tagger/Lemmatiseur est impossible. Or deux balises utilisées dans le dictionnaire XML-isé contiennent de tels attributs : la balise <DEF> et la balise <NIVEAU>. En effet, l'exécution du script 'TagScript1.pl' permet de remplacer les balises <DEF mot= …> et <NIVEAU type= …> par des simples balises <DEF> et <NIVEAU>, les balises remplacées étant stockées dans un fichier dédié. Ensuite, le fichier temporaire a été traité par le Tagger/Lemmatiseur qui fournit en sortie un fichier balisé et lemmatisé.

La réincorporation des balises <DEF mot=…> et <NIVEAU type=…> à leur place d'origine est fait par la dernière partie du script perl 'TagScript1.pl'. On obtient ainsi en sortie du script le fichier 'dico_Temp_Res_ini.xml' qui correspond au dictionnaire XML-isé. Ce fichier contient la catégorie grammaticale de chaque mot (verbe, adjectif, adverbe, nom, …), ainsi que la forme sous laquelle il apparaît comme entrée du dictionnaire.

3.2 Traitements particuliers : les verbes pronominaux et les autres formes

Le traitement des verbes pronominaux est un cas particulier dans l'analyse du dictionnaire XML obtenu. Cette étape est nécessaire pour obtenir le dictionnaire des verbes souhaité. En fait, ces verbes sont fréquents dans le dictionnaire et possèdent des sens différents de leurs verbes « racine ». A titre d'exemple, le verbe pronominal « se lever » a un sens qui n'est pas bien reflété par celui de sa racine « lever ».

D'autre part, il faut que le fichier XML référence ces verbes pronominaux comme ayant une définition à part entière dans le dictionnaire pour pouvoir définir un verbe pronominal comme une nouvelle entrée du dictionnaire. Cependant, un problème se pose dans la mesure où ces verbes ne sont pas recensés comme définition du dictionnaire, mais simplement incorporés dans la définition même de leur « racine ». En effet, ils interviennent comme « autre forme » du verbe, et toute « autre forme » du dictionnaire n'est pas forcément un verbe pronominal (par exemple, cela peut être un participe présent adjectif). L'ensemble des autres formes du verbe est contenu entre des balises <PRONOMINAL> et </PRONOMINAL>. Toutefois, le fichier XML 'dico_final.xml' permet de savoir s'il existe un verbe pronominal au sein des autres formes d'un verbe.

Une partie du script 'TagScript2.pl' permet de distinguer les verbes pronominaux des autres formes. Il s'agit en fait d'une étape assez délicate à cause du nombre de cas particuliers détectés au cours du codage (cas particuliers dus à la non standardisation dans le dictionnaire de la façon d'énumérer les verbes pronominaux).

En cas de modifications du script concernant cette partie, le fichier temporaire 'dico_Temp_Res_final.xml' s'avère très utile pour toute consultation ultérieure.

Un exemple de traitement est le suivant :

Fichier en entrée :

<DEF mot= « lever » phonetique=….>
…….
<PRONOMINAL>
<STANDARD> SE LEVER</STANDARD>
<META>
<STANDARD>
v. pron.
</STANDARD>
</META>
……..
</PRONOMINAL>
<CONTRAIRE>
…..
</DEF>

Fichier en sortie :

<DEF mot= « lever » phonetique=….>
…….
</DEF>
<DEF mot= « lever (se) » phonetique= « » cat= »v. pron. « >
…..
</DEF>

3.3 Insertion des verbes pronominaux et balisage dans les définitions

L'étape consiste ici à insérer les verbes pronominaux comme des entrées spécifiques du dictionnaire. En effet, il s'agit de traiter les verbes pronominaux dans les définitions comme des entités entières (autrement dit, il s'agit d'associer au pronom réflexif « se » ou « s' » le verbe correspondant). Par exemple, si le verbe « s'allumer » apparait dans la définition d'un mot, il faut que l'ensemble renvoie à une seule entité « allumer (s') », afin qu'on puisse retrouver l'entrée correspondante dans le dictionnaire.

Si le traitement s'applique au cœur d'une définition qui n'est pas entre les balises <MOTSLIES> et </MOTSLIES>, les verbes pronominaux se trouvent sous la structure suivante : « allumer (s') », c'est-à-dire que le pronom réflexif suit la forme verbale. Cela nécessite donc un traitement différent. C'est la deuxième partie du script 'TagScript2.pl' qui permet de traiter les verbes pronominaux de cette structure. De plus, cette partie de script permet de réaliser un balisage des mots et de leurs types.

Le fichier résultat en sortie du script 'TagScript2.pl' est le fichier 'dico_Result_final.xml' qui correspond au fichier XML final du dictionnaire.

Voici un exemple de ce que réalise ce balisage :

Fichier en entrée :

<DEF mot= « allumer » phonetique=….>
…….
<ITALIQUE>
avais
VER
avoir
</ITALIQUE>
<MOTSLIES>
<STANDARD>
lever
VER
lever
</STANDARD>
</MOTSLIES>
</DEF>

Fichier en sortie :

<DEF mot= « allumer » phonetique=….>
…….
<ITALIQUE>
<MOT>avoir</MOT><TAG>VER</TAG>
</ITALIQUE>
<MOTSLIES>
<STANDARD>
<MOT>lever</MOT><TAG>VER</TAG>
</STANDARD>
</MOTSLIES>
</DEF>

3.4 Conversion des accents

Les accents ont subi un certain traitement grâce à la procédure « Accent » qui s'applique sur un fichier XML afin de pouvoir valider la DTD. En effet, il faut convertir les accents pour que le fichier suive la norme XML.

Le script perl 'accent_dico.pl' est chargé de cette conversion pour fournir en sortie le fichier 'convert_dico.xml'.

3.5 Validation de la DTD et préparation au traitement des entrants

La validation de la DTD a été effectuée selon les trois étapes suivantes :

Etape 1 : Ouvrir sous *ULTRA-Edit* le fichier 'convert_dico.xml', y incorporer en début et en fin de fichier les lignes indiquées dans le fichier 'DTD_ini.xml'.

Etape 2 : Sans quitter *ULTRA-Edit*, taper en commandes DOS la commande suivante :
rxp –xsVc UTF-16 convert_dico.xml

Corriger *manuellement* (dans *ULTRA-Edit)* les quelques erreurs signalées (2 balises sur les verbes) en enlevant les balises qui posent problèmes.

La DTD est alors validée, et on peut quitter *ULTRA-Edit*, et repasser sous Linux.

Etape 3 : Le script perl 'balises_vides_ini.pl' permet d'enlever toutes les balises vides (par exemple de type <STANDARD></STANDARD> sans texte au milieu). On applique ce script sur le fichier 'convert_dico.xml' (qui est le fichier XML de référence pour les verbes) pour obtenir en sortie le fichier 'balises_vides_final.xml'.

4. Création des liens entrants : cas des verbes

Les "ENTRANTS" d'un mot sont tous les mots qui font référence dans leur propre définition au mot en question. En fait, on rajoute au début de chaque définition du fichier XML la liste des entrants grâce au script perl "*entree.pl*". Ce dernier permet de rajouter dans le fichier XML les balises <ENTRANT> et </ENTRANT> avec la liste de tous les entrants de ce mot répertoriés dans toutes les définitions du dictionnaire (avec suppression des doublons), exceptés ceux qui sont dans une balise <META> (que l'on ignore).

Ce script traite aussi le cas des homonymes. Pour chaque définition est rajoutée l'attribut "*ishomo*" qui peut prendre 3 valeurs :

- 0 : ce mot n'a pas d'homonyme ;
- 1 : ce mot a des homonymes (la phonétique et la catégorie sont vides ici) ;
- 2 : c'est un des homonymes du dernier mot dont la valeur de *ishomo* était 1.

- Dans le cas où il y a des homonymes, les entrants du mot sont placés seulement dans le mot générique (*ishomo* = 1).

- Si une forme pronominale est détectée pour un des homonymes elle est insérée après la définition de l'homonyme en question.

Exemple :

```
<DEF mot="adresser" phonetique="" cat="" ishomo="1">
<ENTRANT> admonester, adresser (s'), adjurer </ENTRANT>
</DEF>
<DEF mot="adresser_1" phonetique="[adrese]" cat="v. tr. " ishomo="2">
…
</DEF>
<DEF mot="adresser (s')" phonetique="[adrese]" cat="v. tr. " ishomo="0">
<ENTRANT> adresser_1 </ENTRANT>
…
</DEF>
<DEF mot="adresser_2" phonetique="[adrese]" cat="v. tr. " ishomo="2">
…
</DEF>
```

Après application de ce script on a donc le fichier XML final du dictionnaire, prêt à être traité par la procédure « Accent ».

Annexe 2 :
Les systèmes multi-agents et la Recherche d'Information

Le Web est difficile à appréhender de part sa forte évolutivité et l'absence de connaissances de synthèse sur la nature des informations accessibles : la recherche d'information s'opère dans un système ouvert, distribué et dynamique, où apparaissent et disparaissent des sites, où les contenus sont modifiés et où il devient impossible de maîtriser les contenus et l'organisation. Pour minimiser les fréquentes désorientations et surcharges cognitives des utilisateurs, des travaux visent une amélioration :

- Soit au niveau de la navigation : annotations et commentaires partagés [Denoue et Vignollet, 2000], carte graphique et représentation des hyperliens [Liebermann, 1995], etc.
- Soit de la recherche d'information proprement dite : sites portail, robots d'indexation, fouille de données, protocoles d'interaction [Eichmann, 1996], agents Internet [Maes, 1994] [Wooldridge et Jennings, 1995], etc.

Nous nous intéressons à cette seconde thématique. Certains outils mettent en œuvre les techniques de l'intelligence artificielle mais peu utilisent les techniques multi-agents. De plus, ils sont souvent prévus pour un domaine d'informations donné ou un groupe d'utilisateurs connu a priori.

1. Assistance de recherche Web basé multi-agent

L'architecture de l'assistance de recherche Web, proposée par Antonella et al. [Antonella et al., 2003], est présentée par la figure A2.1. Dans cette architecture le browser Web représente le bas-niveau du système tandis que les assistants représentent son méta-niveau. En fait, ces assistants sont reliés à l'application via des méta-objets appropriés. Les tâches accomplies par cette assistance sont les suivantes :

- *Observer l'activité de l'utilisateur* lors de sa navigation sur Internet avec un browser ;
- *Déterminer les préférences de l'utilisateur* en se basant sur l'analyse des pages visitées et sur d'autre paramètres, tels que la liste des mots-clés, les favoris[33], le temps passé dans la lecture des pages, etc.
- *Réorganiser la présentation* des liens retournés par les moteurs de recherche, selon les préférences détectées de l'utilisateur.

L'architecture proposée dans la figure A2.1 comporte trois agents : *le Coordinateur (COO)*, *l'Assistant de Profil Utilisateur (APU)* et *l'Assistant d'Organisation de Recherche (AOR)*. L'interaction entre ces agents est assurée par le langage ACL de communication entre agents.

[33] Adresse Internet dans un fichier comprenant la liste des sites préférés par un utilisateur du Web.

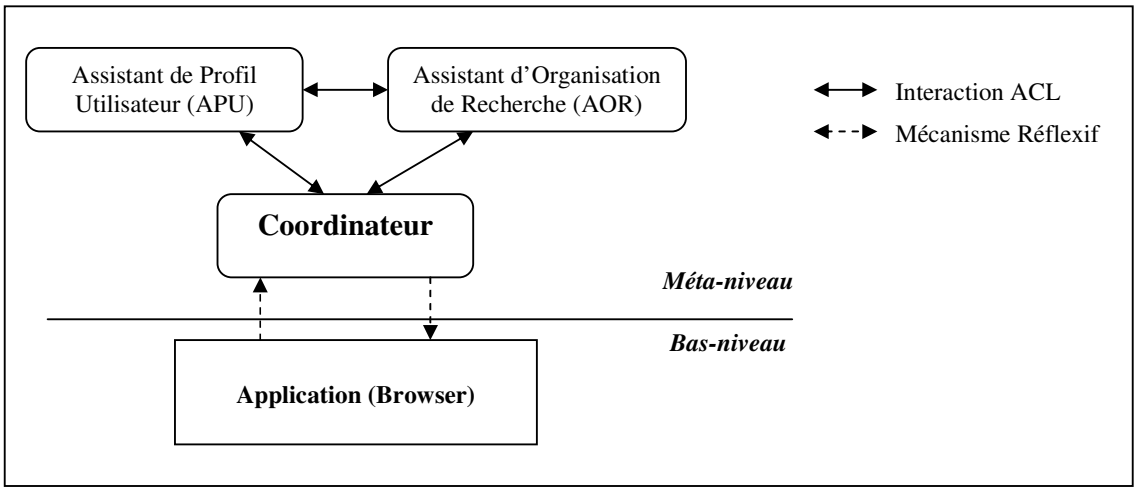


Figure A2.1 : Architecture réflexive de l'assistance de recherche Web

1. Le Coordinateur : Le COO assure la connexion entre l'application et les agents assistants. Il incorpore les méta-objets appropriés qui interceptent les actions de l'utilisateur auxquelles les assistants s'intéressent et qui autorisent les droits dans l'application selon les résultats des assistants. En outre, il permet une forme de coordination entre les agents assistants en agissant en tant que *base de connaissances commune* utilisée pour partager des informations utiles pour tous les assistants. Cette fonctionnalité est fournie par un composant du COO appelé tableau noir (*Blackboard*) ; utilisée dans cette application pour stocker le profil de l'utilisateur. Afin de régler ce genre d'interactions avec les agents assistants, un ensemble de protocoles de conversation, basés sur l'ACL, sont mis en application.

2. L'Assistant de Profil Utilisateur (APU) : C'est un service d'assistance approprié pour la recherche Web et il peut être fourni quand les préférences des utilisateurs sont connues. Pour accomplir ce but, l'APU est prévu pour le profil de l'utilisateur tout en observant ses activités lors de sa navigation sur Internet via un browser. En effet, l'APU détermine les préférences des utilisateurs en analysant, de façon autonome, les pages Web visitées par l'utilisateur lors de sa navigation sur Internet. Pour cela, l'APU rassemble des paramètres utiles des pages Web, tels que les ensembles de mots-clés pondérés pour chaque page Web, le temps passé dans la lecture de chaque page, la profondeur des pages visitées dans un domaine, etc.

L'APU est averti par le COO de la livraison de nouvelles pages Web et utilise un algorithme de classification pour caractériser la page courante. Cet algorithme adopte une liste de catégories classées par mots-clés. Ces listes sont prédéterminées dans la phase préliminaire de sorte qu'elles caractérisent les intérêts généraux des utilisateurs, toutefois elles seront modifiées par la suite pour refléter le comportement d'un utilisateur spécifique. Pour chaque page Web visitée une liste de mots-clés est extraite en calculant la fréquence (c'est-à-dire le poids) de chaque mot, puis en la normalisant avec la longueur de la page. Ensuite, à partir des mots-clés, on calcule le degré de "similarité" de chaque page Web avec la liste prédéterminée de catégories, par la formule suivante :

$$Sim(i) = \sum_{j=1}^{n} W(j) \frac{w(i,j)}{\sum_{k=1}^{m} w(k,j)}$$

Où : $Sim(i)$: la similarité avec la catégorie i,

$W(j)$: le poids du mot-clé *j* dans la page Web,

n : le nombre de mots-clés extraits de la page,

$w(i, j)$: le poids du mot-clé *j* dans la catégorie *i,*

m : le nombre de catégories.

En plus de l'algorithme cité ci-dessus, cet assistant change les poids des mots-clés de chaque catégorie selon les mots-clés les plus récurrents des pages visitées. Le résultat de la tâche de l'APU est une liste comprenant les mots-clés des catégories les plus importantes. Une telle liste est périodiquement communiquée au COO qui la stocke dans le tableau noir.

3. L'Assistant d'Organisation de Recherche (AOR) : Basé sur la connaissance accumulée par l'APU, l'assistant d'organisation de recherche (AOR) réorganise de nouveau les résultats des moteurs de recherche, tout en changeant leurs présentations selon le profil de l'utilisateur. En effet, l'AOR est averti quand la page Web téléchargée est retournée par un moteur de recherche. A la réception d'une nouvelle page, le COO vérifie si elle provient d'un moteur de recherche (en comparant l'adresse de la page aux adresses existantes dans une liste prédéterminée). En fait, l'AOR est le responsable de l'analyse des articles présentés comme résultats de recherche. Il extrait une courte description caractérisant chaque lien suggéré. Une telle description est utilisée pour ranger les articles selon le profil de l'utilisateur. Pour chaque article de la liste retournée de la recherche sur le Web, l'AOR calcule son score par la formule suivante :

$$Sc(i) = \sum_{k=1}^{n} W(k)$$

Où : $Sc(i)$: le score de l'article *i*,

$W(k)$: le poids du mot-clé *k* trouvé dans le tableau noir (fournit par l'APU),

n : le nombre de mots-clés de l'article *i.*

L'AOR range alors les articles selon les scores $Sc(i)$ donnés, et établit une liste des articles choisis en utilisant un score seuil (par exemple les articles qui ont des scores supérieurs à 5) ou choisis par nombre d'articles (par exemple les 10 premiers articles).

Le COO averti l'AOR de la nouvelle page téléchargée. Cet avertissement démarre simultanément l'activité de l'AOR et d'autres opérations exécutées par l'utilisateur ainsi que par l'application. Ainsi, l'AOR et l'application fonctionnent d'une manière asynchrone. Quand l'AOR est prêt à fournir des résultats, il compare l'adresse de la page actuelle à celle réorganisée (en agissant avec le COO), afin de déterminer si l'utilisateur consulte encore cette page résultat de recherche. S'il en est ainsi, l'AOR demande au COO d'intervenir sur cette même page pour changer l'ordre des articles. Autrement, si l'utilisateur consulte une autre page, la liste d'articles sera affichée dans une nouvelle fenêtre récapitulant les liens choisis.

2. Architecture multi-agent adaptatif du méta-moteur de recherche Web "ProFusion"

Le but de projet de Fan et Gauch [Fan et Gauch, 1997] à l'université d'Arkansas (USA) est de développer un outil intelligent et adaptatif de recherche Web. En effet, le travail est basé sur « ProFusion[34] », le méta-moteur de recherche Web développé à l'université du Kansas au début des années 1990. En fait, ProFusion analyse les requêtes entrantes, les classe par catégorie et sélectionne automatiquement les meilleurs moteurs de recherche pour la requête

[34] http://profusion.ittc.ukans.edu/

basée sur la connaissance a priori (facteurs de confiance) qui représente l'efficacité de chaque moteur de recherche pour chaque catégorie. Ces facteurs de confiance sont utilisés pour fusionner les résultats de la recherche dans une liste de documents retournés, ils enlèvent la redondance et présentent une liste finale rangée à l'utilisateur. Les buts principaux de ce type de recherche sont :

- Fournir au ProFusion une architecture multi-agent extensible, robuste et distribuée ;
- Inclure des algorithmes d'adaptation automatiques pour remplacer la connaissance a priori difficile à implémenter.

Le système multi-agent se compose de quatre types différents d'agents, à savoir, un agent d'expédition, un agent de recherche, un agent d'apprentissage, et un agent de surveillance. La figure A2.2 présente l'architecture multi-agent du système ProFusion [Fan et Gauch, 1999].

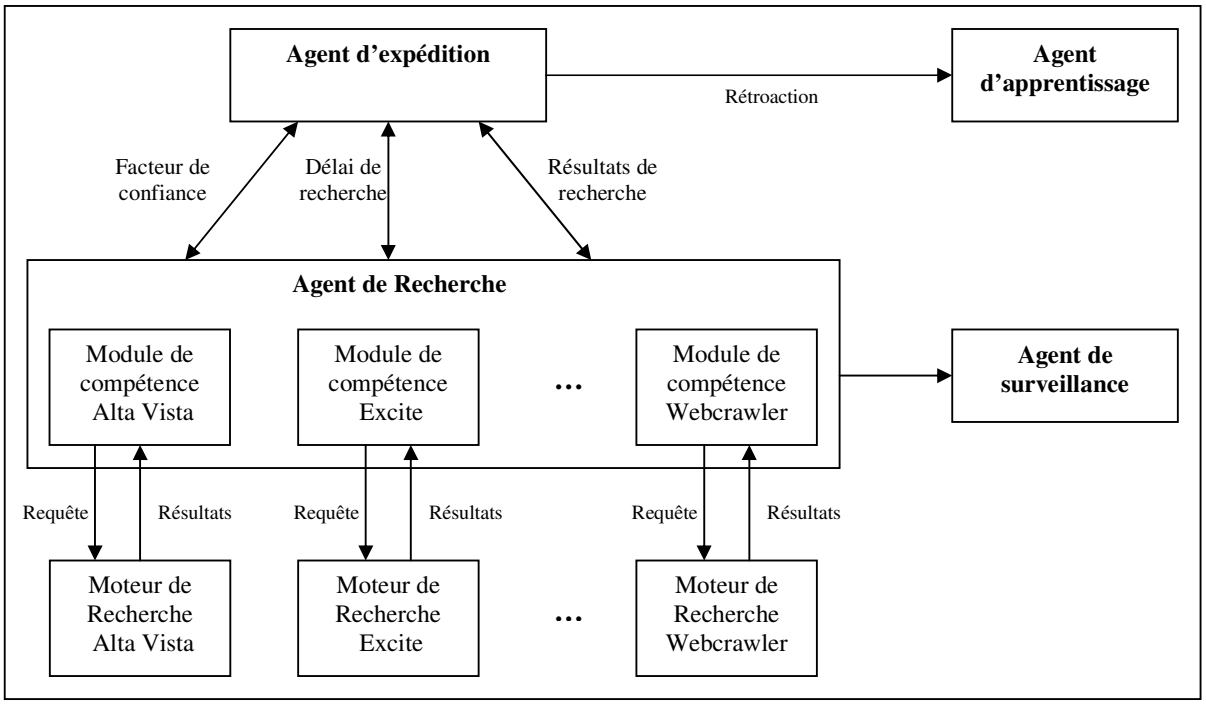


Figure A2.2 : L'architecture multi-agent du système ProFusion

1. Agent d'expédition : il communique avec l'utilisateur, puis expédie ses requêtes aux agents de recherche et d'apprentissage.

2. Agent de recherche : il agit sur les moteurs de recherche fondamentaux. En fait, il est le responsable des comptes rendus de résultats de recherche, des facteurs de confiance et des délais des recherches des moteurs de recherche pour les transmettre, par la suite, à l'agent d'expédition, aussi bien qu'à l'agent de surveillance en cas de besoin.

3. Agent d'apprentissage : il est chargé de l'étude et du développement des moteurs de recherche fondamentaux, en particulier l'ajustement des facteurs de confiance.

4. Agent de surveillance : il est appelé quand un moteur de recherche est inactif et il est le responsable de la protection de l'expédition des requêtes à un moteur de recherche non sensible, aussi bien que de la détection de la mauvaise performance de certains moteurs de recherche.

L'architecture multi-agent du système *ProFusion* est fortement distribuée et décentralisée. Chaque moteur de recherche maintient ses modules de compétence et ses représentations locales dans un annuaire séparé. Excepté l'agent d'expédition, tous les modules de compétence des agents de recherche, d'apprentissage et de surveillance fonctionnent en parallèle. Aucun des ces modules ne contrôle les autres. En raison de cette opération distribuée, le nouveau système peut réagir rapidement en cas de changement de l'environnement et propose les ajustements correspondants. En fait, ces ajustements sont faits par l'agent d'apprentissage qui utilise des algorithmes d'adaptation.

Le nouveau *ProFusion* s'adapte aux changements de performance du moteur de recherche, à son temps de réponse ainsi qu'aux formats des résultats obtenus. L'adaptation à l'exécution est réalisée en observant le comportement de l'utilisateur pour fournir la rétroaction qui change dynamiquement la performance de la base de connaissances. L'adaptation au temps de réponse est réalisée en utilisant des valeurs de temps de réponse modifiables dynamiquement. L'adaptation pour les formats des résultats de recherche est assurée en utilisant un modèle d'extraction dynamique (ou un analyseur).

Avec cette architecture multi-agent adaptative, le système ProFusion est devenu plus compétitif dans un environnement Web dynamique puisqu'il s'adapte automatiquement aux changements de son environnement. ProFusion est également extensible et facile à maintenir parce qu'il n'exige plus la connaissance a priori d'un facteur de confiance d'un nouveau moteur de recherche (ceci sera déterminé par l'agent d'apprentissage). En outre, l'incorporation d'un analyseur, par l'agent de recherche, exclu le besoin d'un code particulier pour extraire les résultats de recherche.

3. Système interactif basé multi-agent pour la recherche Web

Tsai et Lee [Tsai et Lee, 2003] proposent, à l'université des sciences et technologies de Taiwan, une méthodologie à base d'agents pour développer un système interactif basé multi-agent pour la recherche sur le Web. Chaque agent de ce système exécute, de façon autonome, une tâche spécifique et les différents agents fonctionnent simultanément pour accomplir la tâche globale. En effet, ce système comprend quatre agents :

- Un *agent d'interface* agissant avec l'utilisateur et collectant sa rétroaction ;
- Un *agent d'information* pour extraire les mots-clés des pages recueillies ;
- Un *agent de découverte* pour la formulation des requêtes ;
- Un *agent de filtrage* pour le classement et la recommandation des pages à l'utilisateur.

Les agents fonctionnent sans interruption jusqu'à ce que l'utilisateur soit satisfait des résultats de sa recherche. Pendant le processus de recherche, un profil est utilisé pour enregistrer les mots-clés critiques pour la génération automatique de requête. La figure A2.3 illustre l'architecture multi-agent de ce système.

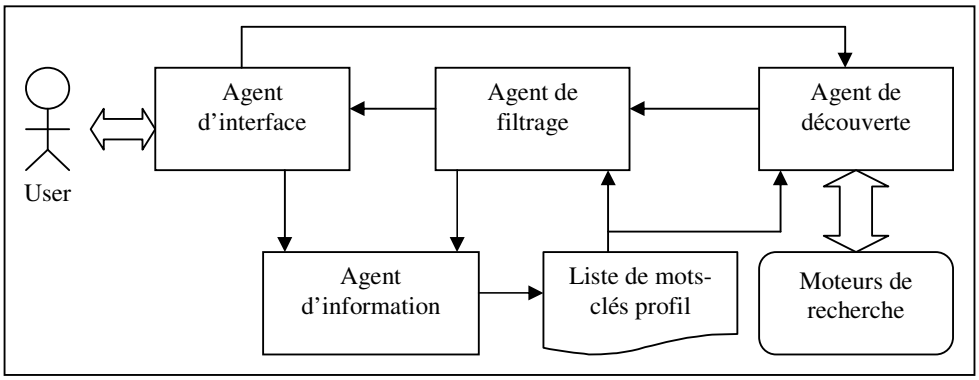


Figure A2.3 : Architecture du système interactif basé multi-agent pour la recherche Web

1. L'agent d'interface : il reçoit la requête initiale de l'utilisateur et l'envoie directement à l'agent de découverte qui est le responsable de l'interaction avec les moteurs de recherche ainsi que de l'organisation des résultats recherchés. En outre, l'agent d'interface reçoit les résultats de recherche de l'agent de filtrage pour les afficher à l'utilisateur, puis il enregistre sa rétroaction indiquant les pages dont il a besoin. Selon la rétroaction de l'utilisateur, le système peut ainsi connaître graduellement le sens de la requête demandée. En fait, l'agent d'interface affiche, dans une nouvelle fenêtre, les résultats typiques de recherche sous forme d'hyperliens pour évaluer les pages Web comme le font habituellement les moteurs de recherche traditionnels. Il marque alors les hyperliens pour l'identification. Après avoir collecté les hyperliens de la rétroaction positive, l'agent d'interface fournit les pages Web correspondantes à l'agent d'information pour les analyser davantage. Toutefois, le concept de la recherche d'information est une tâche subjective : la même requête proposée par différents utilisateurs pourra signifier des choses différentes pour eux, ou bien des utilisateurs différents peuvent utiliser leurs propres requêtes différemment pour chercher le même concept. Ça dépend complètement des opinions personnelles des utilisateurs. Par conséquent, l'interaction décrite, entre le système et l'utilisateur, est un dispositif important permettant au système de bien comprendre son utilisateur.

2. L'agent d'information : il se charge de l'analyse des pages Web sélectionnées, et maintien la liste des mots clés – profil – qui inclut l'information utile pour décrire le concept recherché par l'utilisateur. L'agent de découverte va utiliser ce profil pour la formulation de la requête. Pour les pages Web sélectionnées, l'agent d'information enlève d'abord les balises HTML et garde les termes importants tels que les pronoms et les prépositions, et exécute ensuite une procédure pour calculer les fréquences d'apparition de ces termes dans la page. En effet, les mots gardés seront employés pour représenter cette page, et leurs fréquences d'apparition dans la page Web indiquent leurs importances correspondantes. De cette façon, une page Web est représentée par un vecteur de mots. En conséquence la similitude entre deux pages peut être dérivée de la similarité entre ses vecteurs de mots. Alors les vecteurs des pages sélectionnées sont combinés dans un profil dans lequel les mots de différentes pages sont triés selon leurs fréquences accumulées.

Comme la montre la figure A2.3, le système fonctionne d'une manière itérative et l'utilisateur peut indiquer au système les pages désirées dans chaque itération. Par conséquent, le système peut mettre à jour le profil sans interruption pour continuer à prendre en considération les rétroactions de l'utilisateur. Les fréquences de mot sont accumulées d'une itération à l'autre. Du fait que certaines pages Web ne peuvent contenir qu'un nombre limité de mots (par exemple la page principale d'un site Web spécifique), ces mots ont ainsi des impacts

relativement faibles entre eux. Afin d'empêcher le biais causé par la longueur de la page (pages contenant un nombre de mots inférieur au seuil prédéfini), l'agent d'information explore les hyperliens dans ces pages pour prolonger leurs contenus. Le profil dérivé de cette procédure sera utilisé pour engendrer les nouvelles requêtes et réorganiser les pages recherchées plus tard.

3. L'agent de découverte : c'est le noyau du système ; il est chargé de produire les nouvelles requêtes pour améliorer le concept de capture dans l'esprit de l'utilisateur en se basant sur le rapport de rétroaction fourni par ce dernier. Le concept de recherche est plus abstrait et restrictif ; il est difficile de réaliser par une mesure directe de la similitude lexicologique. Une approche plus efficace est nécessaire pour extraire les ensembles de mots pour modéliser le comportement de l'utilisateur à partir des pages choisies. Les ensembles de mots produits peuvent alors former des requêtes afin d'explorer les différentes régions du Web pour plus précision dans les résultats. En effet, pour évaluer une certaine requête, l'agent de découverte envoie cette dernière à un mécanisme de recherche (par exemple, un moteur de recherche ordinaire ou un méta-moteur de recherche), il collecte les pages Web recherchées et les délivres à l'agent de filtrage pour la réorganisation. Après réception des ces pages Web, l'agent de filtrage active l'agent d'information pour les analyser, les transférer sous forme de vecteurs de mots et réorganiser ces pages selon leurs similarités avec la liste de mots accumulés dans le profil. En d'autres termes, les nouvelles pages Web recherchées seront rangées de la même manière que des pages Web similaires déjà utilisées par l'utilisateur.

4. L'agent de filtrage : pour préserver la durée de la transformation, l'agent de filtrage traite seulement les pages de rangs supérieurs (par exemple, les l premières pages données par le moteur de recherche) toute en calculant le produit de chacun des vecteurs des pages par le vecteur du profil. Selon les résultats de mesure, les pages les plus semblables au profil (par exemple, les k premières pages, $k \geq 1$) sont présentées à l'utilisateur par l'agent d'interface. L'utilisateur peut choisir alors les pages Web désirées. En fait, l'agent d'interface envoie les pages choisies à l'agent d'information qui analyse ces dernières et sélectionne, en conséquence, les mots nécessaires pour mettre à jour le profil.

Par la manière interactive et itérative décrite dans cette approche, le système de Tsai et Lee peut exploiter graduellement le profil de l'utilisateur et explorer de nouvelles régions pour rechercher ce dont il a besoin.

4. Agents mobiles de Recherche d'Information

Divers travaux s'appuient sur les concepts d'agent et de système multi-agent pour la programmation de l'Internet et le commerce électronique [Joshi et al., 1999], en particulier sur les mécanismes d'apprentissage [Fan et Gauch, 1999].

Brewington et al. [Brewington et al., 1999] présentent une application de recherche simple (sans expertise client) de documents textuels dans un réseau local (avec centralisation des informations sur les serveurs via un mécanisme de pages jaunes, sans contrainte de sécurité, ni découverte dynamique de nouveaux serveurs). Un agent mobile est créé pour chaque requête ; en fonction de l'état du réseau et de la complexité de la requête. Il peut choisir de se déplacer sur un site "proxy"[35] choisi dynamiquement. L'agent de recherche s'appuie sur des agents d'observation du réseau (network-sensing agents) et sur un agent stationnaire (par site) qui sert d'interface avec le serveur d'information local. En outre, les auteurs étudient les

[35] Un proxy peut avoir plusieurs utilisations tels que : le proxy peut vous protéger ; le proxy peut masquer les informations concernant votre ordinateur ; le proxy peut mémoriser les pages les plus demandées.

problèmes de planification du chemin et proposent d'utiliser des agents mobiles d'observation pour actualiser le contenu des pages jaunes.

Le système M3 (MultiMedia Database Mobile agents) de Kosch et al. [Kosch et al., 2001] est un système de recherche par le contenu de données multimédia (images, vidéo) qui repose sur Java et CORBA. Un agent mobile peut se déplacer de site en site et en extraire l'information au moyen d'un code spécifique. L'agent mobile peut mémoriser les informations recueillies sur un site, les utiliser sur les sites visités ensuite et les faire évoluer pendant le parcours. Il y a donc une véritable exploitation de la notion d'agent mobile (un programme explore depuis la machine serveur). Les problèmes de sécurité sont pris en compte grâce à des mécanismes de sessions indépendantes, aux mécanismes de sécurité de CORBA et à des restrictions des droits. L'évaluation des performances montre l'intérêt de la spécialisation pour retrouver l'information recherchée (et éviter l'intervention de l'utilisateur).

Enfin, plusieurs travaux exploitent les agents mobiles pour optimiser la phase de "crawlage" (parcours des liens hypertextes et indexation) des moteurs de recherche sur le Web (alternativement à la stratégie centralisée download first, process later) :

- Hammer et Fiedler [Hammer et Fiedler, 2000] suggèrent un agent mobile pour représenter le moteur de recherche sur le site serveur qui s'exécute localement et procède au "crawlage" du site. Il analyse les pages (à partir de mots-clés seulement) et mémorise un certain nombre d'informations (liens externes, méta-données, caractéristiques du serveur Web, etc.). Les liens externes ne sont pas explorés récursivement ; ainsi, l'exploration du Web est limitée et contrôlée de manière centralisée. Une fois le travail terminé, la seule information sélectionnée est déplacée sur le réseau.
- Thati et al. [Thati et al., 2001] proposent une implémentation qui ne demande pas de modification des serveurs Web (infaisable en pratique) mais seulement l'installation de pages actives (CGI, ASP) pour l'accueil des agents de "crawlage". En complément, ils proposent des solutions simples aux problèmes de sécurité.

5. Etude comparative des SMA de Recherche d'Information

La plupart des travaux étudiés et qui concernent l'implantation des systèmes multi-agents pour la recherche d'information sur Internet sont des systèmes à base de moteurs de recherche. En effet, ces systèmes organisent les documents recherchés grâce à ces moteurs, uniquement selon les fréquences des mots-clés existants dans ces documents, pour répondre aux profils des utilisateurs. D'ailleurs Fan et Gauch [Fan et Gauch, 1997] ont évoqué l'utilité d'un "*Web crawler*" dans le cadre d'un système multi-agent de recherche d'informations sur Internet. En outre, ces systèmes se limitent à la requête proposée par l'utilisateur et aucun processus de reformulation de la requête n'a été proposé. En fait, ces systèmes ne tiennent compte ni de la dépendance entre les mots-clés recherchés, ni de celle entre les pages Web résultats de la recherche. En outre, tous les termes de la requête utilisateur sont considérés de même poids et aucun modèle de préférences entre ces termes n'a été proposé dans ces systèmes. En conséquence de toutes ces limites, les deux notions de base dans un SRI, la *pertinence* et le *profil*, ne sont pas bien définies dans les systèmes proposés.

Le tableau A2.1 présente une étude comparative de ces SMA de Recherche d'Information au regard de plusieurs critères, à savoir :

- Les types d'agents existant dans la plate-forme ;
- Les techniques de recherche ;

- Reformulation de la requête utilisateur ;
- Modélisation de la requête utilisateur ;
- Préférences entre les termes de la requête ;
- Modélisation des documents recherchés.

SMA de Recherche d'Information	Types d'agents	Technique de Recherche	Reformulation de la requête utilisateur	Modélisation de la requête utilisateur	Préférences entre les termes de la requête	Modélisation des documents recherchés
Assistance de recherche Web. [Antonella et al., 2003]	- Coordinateur. - Assistant de profil utilisateur. - Assistant d'organisation de recherche.	Système à base d'un *web crawler.*	Non	Non	Non	Non
Méta-Moteur de recherche "ProFusion" [Fan et Gauch, 1997]	- Agent d'expédition - Agent de recherche - Agent d'apprentissage - Agent de surveillance	Système à base de moteurs de recherche.	Non	Non	Non	Non
Système Interactif basé multi-agent de recherche Web. [Tsai et Lee, 2003]	- Agent d'Interface - Agent d'Information - Agent de découverte - Agent de filtrage	Système à base de moteurs de recherche.	Non	Non	Non	Non
Agents mobiles de recherche d'information. [Brewington et al., 1999] [Kosch et al., 2001] [Hammer et Fiedler, 2000] [Thati et al., 2001]	- Agent mobile créé pour chaque requête utilisateur. - Agent mobile de mise à jour des sites Web.	Système à base de moteurs de recherche et de *web crawler.*	Non	Non	Non	Non

Tableau A2.1 : Comparaison des SMA de Recherche d'Information

En réalité, face à un problème donné de recherche d'information dans un environnement hétérogène comme l'Internet, un SRI associant les dépendances entre les mots-clés et les pages Web recherchées doit fournir une certaine souplesse et objectivité surtout dans la sélection des pages Web recherchées ainsi que dans leur organisation selon le profil de l'utilisateur de ce système. Les travaux que nous avons entrepris tentent de proposer un cadre conceptuel général de résolution de ce genre de problème, dans lequel il s'agit de fournir à l'utilisateur d'une part la possibilité de reformuler sa requête et d'autre part de choisir les paramètres appropriés pour définir son profil. En effet, le système multi-agent SARIPOD, développé dans le cadre de ce travail de recherche, propose un nouveau modèle de RI à base de Réseaux Petits Mondes Hiérarchiques (RPMH) et de Réseaux Possibilistes (RP). Le premier RPMH, pour les mots de la langue française, est utilisé pour tenir compte des dépendances entre ces mots. Le second RPMH est consacré aux pages Web recherchées et traduit de même les dépendances entre ces pages. Les Réseaux Possibilistes (RP) engendrent le mixage de ces deux RPMH afin d'organiser les documents recherchés selon le profil de l'utilisateur. De plus, SARIPOD est un système intelligent car il tient compte d'un profil dynamique de l'utilisateur formé d'une part des paramètres de pertinence possibiliste des documents et d'autre part des préférences entres les termes de la requête. Enfin, ce système contribue simultanément à l'enrichissement de l'approche du sujet par les RPMH et les RP et aussi à celui de la recherche d'information en tenant compte de l'hétérogénéité de cette information.

6. Conclusion

Nous avons présenté dans cette annexe un état de l'art de systèmes multi-agents de recherche d'information les plus connus dans la littérature. Nous avons commencé par détailler l'architecture multi-agent de chacun de ces systèmes afin de décrire brièvement la tâche de chaque agent ainsi que sa coopération avec le reste des agents de la plate-forme. Une étude comparative de ces SMA a été élaborée pour montrer leur insuffisance afin de combler les problèmes liés à la recherche d'information sur Internet. De notre côté, et pour résoudre certaines limites de ces SMA, nous avons proposé d'intégrer une nouvelle technologie dans notre SRI basé multi-agent faisant appel aux Réseaux Petits Mondes Hiérarchiques (RPMH) ainsi que leurs combinaisons via un Réseau Possibiliste (RP).

Annexe 3 : Données et résultats du RPMH de dictionnaire

Nous présentons dans cette annexe des expérimentations sur le choix de la longueur des circuits dans le RPMH de dictionnaire. En fait, nous prouvons que la longueur maximale prise en compte est de l'ordre de 4 arcs. Nous avons atteint ce chiffre après bien des tests sur la validité des résultats obtenus en fonction de la longueur des circuits étudiés.

Nos tests sont faits sur 5 verbes : vérifier, nettoyer, analyser, jouer et préserver. En effet, l'ordre de mots proches récupérés pour chacun de ces mots de test se stabilise à partir d'une longueur maximale de circuit égale à 4 (nous présentons ici uniquement les dix premiers mots proches).

Les tableaux A3.1, A3.2, A3.3, A3.4 et A3.5 sont constitués de 6 colonnes :

1. *Mot recherché* : c'est l'un de mots-clés de la requête utilisateur. À chaque itération, ce dernier choisi le nombre de termes proches désirés pour chacun de mot de la requête. Le système calcule la préférence de chaque terme par la formule suivante :

   Préférence (t_i) = *[Nbre termes proches choisis pour* t_i *dans Q'/ Nbre termes de Q]+ 1*

   Avec Q : la requête de départ et Q' : la requête reformulée

2. *Longueur de circuit* : c'est le nombre d'arêtes séparant les mots dans un circuit de mots en partant d'un mot de la requête reformulée et en y revenant à ce même mot de départ pour construire un cycle.

3. *Nombre de circuits pour chaque mot proche* : Nous présentons pour chaque mot de la requête reformulée les dix premiers mots proches ainsi que leurs nombres de circuits détectés.

4. *Proximité du mot proche par rapport au mot recherché* : c'est la proximité sémantique de chaque mot proche par rapport au mot-clé de la requête reformulée. Cette proximité est calculée par la formule suivante :

   Proximité_Dictionnaire (M_1, M_2) = *Nbre de circuits* (M_1, M_2)*/Nbre maximum de circuits détectés*

5. *Nombre de groupes* : C'est le nombre de petits mondes de mots sémantiquement proches. En effet, le groupement de ces mots est fait via l'algorithme de regroupement de mots par contrainte minimale, décrit dans le chapitre 3.

6. *Nombre de groupes fusionnés* : C'est le nombre de groupes de mots sémantiquement proches fusionnés. En effet, cette fusion est faite via l'algorithme de fusion des groupes potentiels en composantes de sens, décrit dans le chapitre 3.

Mot recherché	Longueur de circuit	Nombre de circuits pour chaque mot proche		Proximité du mot proche par rapport au mot recherché	Nombre de groupes	Nombre de groupes fusionnés
Vérifier	2	Justifier	1	1	0	0
		Confirmer	1	1		
		Expérimenter	1	1		
		Essayer	1	1		
		Eprouver	1	1		
		S'assurer	1	1		
		Voir	1	1		
		Reconnaître	1	1		
		Auditer	1	1		
		Contrôler	1	1		
	3	Examiner	12	1	22	4
		Eprouver	10	0,83		
		Essayer	8	0,66		
		Voir	8	0,66		
		Expérimenter	6	0,50		
		Contrôler	6	0,50		
		S'assurer	5	0,41		
		Reconnaître	5	0,41		
		Constater	3	0,25		
		Collationner	2	0,16		
	4	Examiner	87	1	115	14
		Voir	81	0,93		
		Eprouver	61	0,70		
		Reconnaître	55	0,63		
		Essayer	48	0,55		
		Contrôler	34	0,39		
		Expérimenter	33	0,37		
		Constater	31	0,35		
		S'assurer	18	0,20		
		Prouver	17	0,19		
	5	Voir	1021	1	1784	1
		Examiner	797	0,78		
		Reconnaître	739	0,72		
		Eprouver	533	0,52		
		Essayer	345	0,33		
		Constater	335	0,32		
		Expérimenter	226	0,221		
		Prouver	225	0,220		
		Contrôler	213	0,20		
		Rechercher	174	0,17		
	6	Voir	15150	1	27571	1
		Reconnaître	11736	0,77		
		Examiner	10370	0,68		
		Eprouver	7498	0,49		
		Essayer	4209	0,277		
		Constater	4182	0,276		
		Prouver	3944	0,26		
		Confirmer	3030	0,20		
		Connaître	2778	0,183		
		Rechercher	2737	0,18		
	7	nulle	0	0	0	0
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		

Tableau A3.1 : Résultats de la recherche de composantes de sens du verbe « vérifier »

Mot recherché	Longueur de circuit	Nombre de circuits pour chaque mot proche		Proximité du mot proche par rapport au mot recherché	Nombre de groupes	Nombre de groupes fusionnés
Nettoyer	2	Dépouiller	1	1	0	0
		Ruiner	1	1		
		Purifier	1	1		
		Purger	1	1		
		Déterger	1	1		
		Décrasser	1	1		
		Débarbouiller	1	1		
		Ecurer	1	1		
		Curer	1	1		
		Racler	1	1		
	3	Laver	19	1	49	14
		Purifier	15	0,78		
		Frotter	12	0,63		
		Balayer	9	0,47		
		Purger	7	0,36		
		Curer	7	0,36		
		Déterger	5	0,26		
		Epousseter	5	0,26		
		Ruiner	5	0,26		
		Décrasser	4	0,21		
	4	Laver	137	1	285	32
		Purifier	112	0,81		
		Frotter	83	0,60		
		Purger	66	0,48		
		Balayer	61	0,44		
		Dépouiller	45	0,32		
		Battre	39	0,28		
		Ruiner	36	0,26		
		Décrasser	30	0,21		
		Cribler	30	0,21		
	5	Dépouiller	815	1	3448	1
		Purifier	750	0,92		
		Laver	739	0,90		
		Balayer	676	0,82		
		Battre	656	0,80		
		Ruiner	586	0,71		
		Frotter	565	0,69		
		Purger	512	0,62		
		Arracher	512	0,62		
		Oter	474	0,58		
	6	Dépouiller	13747	1	59367	1
		Battre	13433	0,97		
		Ruiner	11864	0,86		
		Arracher	10308	0,74		
		Balayer	10169	0,73		
		Oter	9431	0,68		
		Enlever	9064	0,65		
		Tuer	7607	0,55		
		Détacher	7360	0,53		
		Vider	6543	0,47		
	7	nulle	0	0	0	0
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		

Tableau A3.2 : Résultats de la recherche de composantes de sens du verbe « Nettoyer »

Mot recherché	Longueur de circuit	Nombre de circuits pour chaque mot proche		Proximité du mot proche par rapport au mot recherché	Nombre de groupes	Nombre de groupes fusionnés
Analyser	2	Examiner	1	1	0	0
		Etudier	1	1		
		Résumer	1	1		
		Décomposer	1	1		
		Séparer	1	1		
		Rechercher	1	1		
		Diviser	1	1		
		Désosser	1	1		
		Réduire	1	1		
		Résoudre	1	1		
	3	Examiner	6	1	11	8
		Décomposer	6	1		
		Etudier	4	0,66		
		Réduire	2	0,33		
		Résumer	2	0,33		
		Rechercher	2	0,33		
		Diviser	1	0,16		
		Résoudre	1	0,16		
		Séparer	1	0,16		
		désosser	1	0,16		
	4	Examiner	38	1	43	13
		Etudier	30	0,78		
		Décomposer	16	0,42		
		Rechercher	10	0,26		
		Diviser	7	0,18		
		Séparer	7	0,18		
		Essayer	7	0,18		
		Réduire	6	0,15		
		Résumer	5	0,13		
		Comparer	4	0,10		
	5	Examiner	360	1	938	1
		Etudier	225	0,62		
		Décomposer	142	0,39		
		Séparer	105	0,29		
		Diviser	87	0,24		
		Rechercher	79	0,21		
		Réduire	75	0,20		
		Distinguer	60	0,16		
		Fouiller	60	0,166		
		Essayer	58	0,161		
	6	Examiner	5119	1	14347	1
		Etudier	2847	0,55		
		Décomposer	2190	0,42		
		Séparer	1736	0,33		
		Réduire	1611	0,31		
		Diviser	1139	0,22		
		Distinguer	1092	0,21		
		Rechercher	938	0,18		
		Résoudre	817	0,159		
		Résumer	800	0,156		
	7	Examiner	94701	1	248752	1
		Etudier	48691	0,51		
		Décomposer	42608	0,44		
		Réduire	34784	0,367		
		Séparer	34523	0,364		
		Diviser	19921	0,21		
		Distinguer	19574	0,20		
		Résoudre	16937	0,17		
		Diminuer	15300	0,16		
		Rechercher	14872	0,15		

Tableau A3.3 : Résultats de la recherche de composantes de sens du verbe « Analyser »

Mot recherché	Longueur de circuit	Nombre de circuits pour chaque mot proche		Proximité du mot proche par rapport au mot recherché	Nombre de groupes	Nombre de groupes fusionnés
Jouer	2	Imiter	1	1	0	0
		Simuler	1	1		
		Incarner	1	1		
		Entendre	1	1		
		Représenter	1	1		
		Donner	1	1		
		Se moquer	1	1		
		Echouer	1	1		
		Sonner	1	1		
		Exposer	1	1		
	3	Exposer	10	1	69	12
		Se moquer	10	1		
		Tromper	10	1		
		Folâtrer	10	1		
		Donner	9	0,9		
		Représenter	9	0,9		
		plaisanter	8	0,8		
		Berner	7	0,7		
		S'amuser	6	0,6		
		Risquer	6	0,6		
	4	Donner	130	1	504	19
		Tromper	111	0,85		
		Exposer	95	0,73		
		Se moquer	89	0,68		
		Représenter	78	0,60		
		Mettre	75	0,57		
		Plaisanter	61	0,469		
		Abuser	60	0,461		
		Entendre	58	0,44		
		Tourner	58	0,44		
	5	Donner	2722	1	11326	1
		Exposer	1486	0,545		
		Mettre	1477	0,542		
		Tromper	1338	0,49		
		Passer	1297	0,47		
		Entendre	1118	0,41		
		Prendre	1067	0,39		
		Représenter	1021	0,37		
		Tourner	794	0,29		
		Se moquer	710	0,26		
	6	Donner	57971	1	226341	1
		Mettre	30993	0,53		
		Passer	28402	0,489		
		Exposer	27973	0,482		
		Prendre	25270	0,43		
		Entendre	22583	0,38		
		Tromper	19371	0,33		
		Représenter	18052	0,31		
		Tourner	13791	0,23		
		Courir	12958	0,22		
	7	nulle	0	0	0	0
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		
		nulle	0	0		

Tableau A3.4 : Résultats de la recherche de composantes de sens du verbe « jouer »

Mot recherché	Longueur de circuit	Nombre de circuits pour chaque mot proche		Proximité du mot proche par rapport au mot recherché	Nombre de groupes	Nombre de groupes fusionnés
Préserver	2	Conserver	1	1	0	0
		Abriter	1	1		
		Exempter	1	1		
		Epargner	1	1		
		Soustraire	1	1		
		Sauver	1	1		
		Protéger	1	1		
		Prémunir	1	1		
		Garantir	1	1		
		Assurer	1	1		
	3	Protéger	14	1	33	1
		Garantir	14	1		
		Sauver	9	0,64		
		Epargner	8	0,57		
		Conserver	8	0,57		
		Assurer	5	0,35		
		Abriter	5	0,35		
		Sauvegarder	5	0,35		
		Soustraire	5	0,35		
		Garder	5	0,35		
	4	Garantir	98	1	133	4
		Protéger	97	0,98		
		Garder	60	0,61		
		Epargner	58	0,59		
		Conserver	56	0,57		
		Sauver	56	0,57		
		Assurer	48	0,48		
		Soustraire	48	0,48		
		Eviter	38	0,38		
		Abriter	35	0,35		
	5	Garantir	613	1	1260	1
		Protéger	606	0,98		
		Garder	585	0,95		
		Assurer	444	0,72		
		Epargner	396	0,646		
		Conserver	394	0,642		
		Eviter	392	0,63		
		Soustraire	386	0,62		
		Sauver	367	0,59		
		Arracher	268	0,43		
	6	Soustraire	5718	1	12683	1
		Assurer	5586	0,97		
		Protéger	4862	0,85		
		Garantir	4810	0,84		
		Eviter	4784	0,83		
		Arracher	4537	0,79		
		Garder	4515	0,78		
		Epargner	3686	0,64		
		Conserver	3561	0,62		
		Sauver	3290	0,57		
	7	Soustraire	100904	1	210634	1
		Assurer	96177	0,95		
		Arracher	85563	0,84		
		Eviter	75145	0,74		
		Protéger	60342	0,598		
		Garder	59720	0,591		
		Garantir	58511	0,57		
		Epargner	49288	0,48		
		Conserver	44976	0,44		
		Sauver	44266	0,43		

Tableau A3.5 : Résultats de la recherche de composantes de sens du verbe « Préserver »

Pour optimiser le choix du paramètre longueur de circuit, nous avons fait des expérimentations sur cinq verbes différents présentées dans les trois tableaux A3.1, A3.2, A3.3, A3.4 et A3.5. En effet, l'ordre de mots sémantiquement proches récupérés pour le mot de départ se stabilise à partir d'une longueur maximale de circuits égale à 4. A partir d'une longueur de circuits égale à 5, le nombre de circuits récupérés pour chaque mot proche devient très important (voir figure A3.1). En conséquence, nous avons plusieurs mots qui ne font pas partie de la composante de sens du mot de départ.

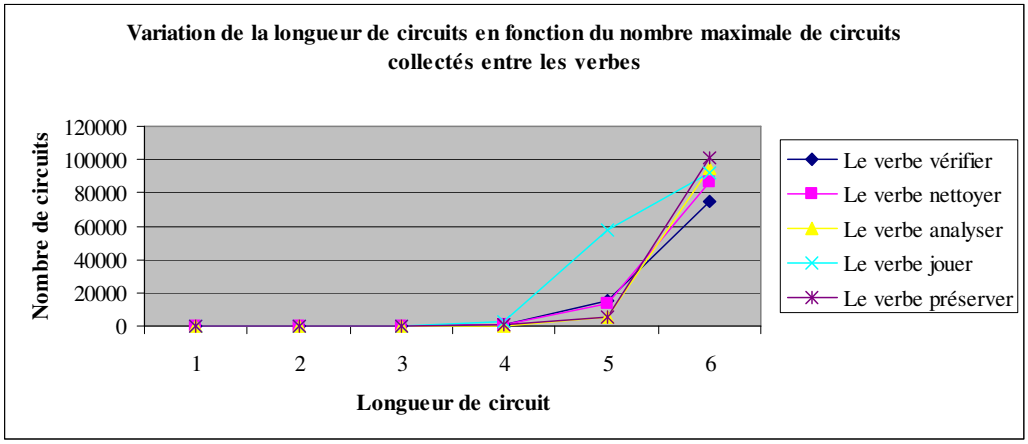


Figure A3.1 : Courbes de variation de la longeur de circuit en fonction du nombre maximale de circuits collectés entre les verbes

Nous remarquons aussi que pour les cinq verbes, toutes les composantes de sens seront fusionnées dans un seul groupe à partir d'une longueur de circuit égale à 5. Ceci montre bien la performance de l'approche à base de circuits dans la construction des composantes de sens dans le RPMH de dictionnaire.

Annexe 4 :
Données et résultats du RPMH de pages Web

Nous présentons dans cette annexe des expérimentations sur le RPMH de pages Web en utilisant notre approche à base de circuits présentée dans le chapitre 3. En fait, nous prouvons que les pages Web retournées suite à une requête donnée sont bien réparties sur plusieurs groupes thématiques. Ces groupes sont identifiés par un ensemble de mots-clés en commun entre les pages Web faisant partie du même thème.

De plus, nous montrons que la longueur maximale de circuits prise en compte est de l'ordre de 4 arcs. Nous avons atteint ce chiffre après bien des tests sur la validité des résultats obtenus en fonction de la longueur des circuits étudiés.

Nos tests sont faits sur un thème principale qui est l'informatique contenant 3 sous-thèmes différents : systèmes d'exploitation (Unix, Linux, Mac/MacOS, Windows 95/98/Me, Windows NT/2000/XP, MS-DOS, AS/400 – OS/400, …), Réseaux et protocoles (Protocoles, Transmission de données, équipements réseaux, Internet, Technologies, réseaux sans fil, WiFi (802.11), BleuTooth (802.15), Courants porteurs (CPL),…) et Bases de données (Conception MERISE, Modèle relationnel, Langage SQL, Langage PL/SQL, JDBC, ODBC, Annuaires LDAP, Active Directory, …).

Les tableaux A4.1, A4.2 et A4.3 sont constitués de 7 colonnes :

1. *Les URLs des pages Web sélectionnées* : C'est la liste de URLs affiché par le système comme résultat préliminaire de la recherche. Il s'agit en fait, de la totalité des pages Web avant ni groupement thématique, ni fusion de groupes.
2. *Longueur de circuit* : C'est le nombre d'arêtes séparant les pages Web dans un circuit de pages en partant d'une page Web racine et en y revenant à cette même page de départ pour construire un cycle.
3. *Nombre de circuits pour chaque page proche* : Nous présentons pour chaque page Web sélectionnée le nombre de circuits détectés. Nous nous sommes limités ici aux dix premières pages proches.
4. *Proximité d'une page Web proche par rapport à la page Web racine* : C'est la proximité hypertextuelle de chaque page Web proche par rapport à la page Web racine. Nous définissons cette proximité entre deux pages P_1 et P_2 en terme du nombre de circuits passant par P_1 et P_2 et revenant à P_1 de la manière suivante :

   $Proximité\ (P_1, P_2) = Nombre\ de\ circuits\ (P_1, P_2)/Nombre\ maximum\ de\ circuits\ détectés$
5. *Nombre de pages dans le groupe* : C'est l'ensemble de pages Web appartenant à un groupe thématique. En effet, ces composantes thématiques de pages Web seront fusionnées dans une deuxième étape pour former un petit monde de pages.
6. *Nombre de groupes* : C'est le nombre de petits mondes de pages Web thématiquement proches. En effet, le groupement de ces pages est fait via l'algorithme de construction des composantes thématiques des pages Web, décrit dans le chapitre 3.
7. *Nombre de groupes fusionnés* : C'est le nombre de groupes de pages Web thématiquement proches fusionnés. En effet, la fusion des groupes des pages Web est faite via l'algorithme de fusion des groupes potentiels en composantes thématiques, décrit dans le chapitre 3.

Les URLs des pages Web sélectionnées	Longueur de circuit	Nombre de circuits pour chaque page proche	Proximité de la page proche par rapport à la page racine	Nombre de pages dans le groupe	Nombre de groupes	Nombre de groupes fusionnés
file:///c:/RPMH/x.htm file:///c:/RPMH/a.htm file:///c:/RPMH/b.htm file:///c:/RPMH/c.htm file:///c:/RPMH/d.htm file:///c:/RPMH/f.htm file:///c:/RPMH/p.htm file:///c:/RPMH/q.htm file:///c:/RPMH/s.htm file:///c:/RPMH/t.htm file:///c:/RPMH/u.htm file:///c:/RPMH/v.htm file:///c:/RPMH/w.htm file:///c:/RPMH/z.htm	2	file:///c:/RPMH/w.htm 1 file:///c:/RPMH/v.htm 1 file:///c:/RPMH/u.htm 1 file:///c:/RPMH/t.htm 1 file:///c:/RPMH/q.htm 1 file:///c:/RPMH/p.htm 1 file:///c:/RPMH/d.htm 1 file:///c:/RPMH/c.htm 1 Nulle 0 Nulle 0	1 1 1 1 1 1 1 1 0 0	0	0	0
file:///c:/RPMH/P1.htm file:///c:/RPMH/P2.htm file:///c:/RPMH/P3.htm file:///c:/RPMH/P4.htm file:///c:/RPMH/P5.htm file:///c:/RPMH/P6.htm file:///c:/RPMH/P7.htm file:///c:/RPMH/P8.htm file:///c:/RPMH/P9.htm file:///c:/RPMH/P10.htm file:///c:/RPMH/P11.htm file:///c:/RPMH/P12.htm file:///c:/RPMH/P13.htm file:///c:/RPMH/P14.htm	3	file:///c:/RPMH/p.htm 17 file:///c:/RPMH/d.htm 16 file:///c:/RPMH/u.htm 16 file:///c:/RPMH/q.htm 15 file:///c:/RPMH/w.htm 15 file:///c:/RPMH/t.htm 15 file:///c:/RPMH/v.htm 15 file:///c:/RPMH/c.htm 12 file:///c:/RPMH/f.htm 07 file:///c:/RPMH/b.htm 04	1 0,94 0,94 0,88 0,88 0,88 0,88 0,70 0,41 0,23	2	43	1
file:///c:/RPMH/P15.htm file:///c:/RPMH/P16.htm file:///c:/RPMH/P17.htm file:///c:/RPMH/P18.htm file:///c:/RPMH/P19.htm file:///c:/RPMH/P20.htm file:///c:/RPMH/P21.htm file:///c:/RPMH/P22.htm file:///c:/RPMH/P23.htm file:///c:/RPMH/P24.htm file:///c:/RPMH/P25.htm file:///c:/RPMH/P26.htm file:///c:/RPMH/P27.htm file:///c:/RPMH/P28.htm	4	file:///c:/RPMH/p.htm 181 file:///c:/RPMH/u.htm 171 file:///c:/RPMH/d.htm 169 file:///c:/RPMH/t.htm 158 file:///c:/RPMH/w.htm 157 file:///c:/RPMH/v.htm 157 file:///c:/RPMH/q.htm 150 file:///c:/RPMH/c.htm 110 file:///c:/RPMH/f.htm 108 file:///c:/RPMH/s.htm 69	1 0,94 0,93 0,87 0,86 0,86 0,82 0,60 0,59 0,38	3	45	1
file:///c:/RPMH/P29.htm file:///c:/RPMH/P30.htm file:///c:/RPMH/P31.htm file:///c:/RPMH/P32.htm file:///c:/RPMH/P33.htm file:///c:/RPMH/P34.htm file:///c:/RPMH/P35.htm file:///c:/RPMH/P36.htm file:///c:/RPMH/P37.htm file:///c:/RPMH/P38.htm file:///c:/RPMH/P39.htm file:///c:/RPMH/P40.htm file:///c:/RPMH/P41.htm file:///c:/RPMH/P42.htm	5	file:///c:/RPMH/p.htm 1491 file:///c:/RPMH/u.htm 1423 file:///c:/RPMH/d.htm 1404 file:///c:/RPMH/t.htm 1316 file:///c:/RPMH/w.htm 1307 file:///c:/RPMH/v.htm 1301 file:///c:/RPMH/q.htm 1219 file:///c:/RPMH/f.htm 1046 file:///c:/RPMH/c.htm 874 file:///c:/RPMH/s.htm 716	1 0,95 0,94 0,88 0,876 0,872 0,81 0,70 0,85 0,48	4	36	1
file:///c:/RPMH/P43.htm file:///c:/RPMH/P44.htm file:///c:/RPMH/P45.htm file:///c:/RPMH/P46.htm file:///c:/RPMH/P47.htm file:///c:/RPMH/P48.htm file:///c:/RPMH/P49.htm file:///c:/RPMH/P50.htm file:///c:/RPMH/P51.htm file:///c:/RPMH/P52.htm file:///c:/RPMH/P53.htm file:///c:/RPMH/P54.htm file:///c:/RPMH/P55.htm file:///c:/RPMH/P56.htm	6	file:///c:/RPMH/p.htm 9994 file:///c:/RPMH/u.htm 9629 file:///c:/RPMH/d.htm 9542 file:///c:/RPMH/t.htm 8965 file:///c:/RPMH/w.htm 8941 file:///c:/RPMH/v.htm 8835 file:///c:/RPMH/q.htm 8277 file:///c:/RPMH/f.htm 7749 file:///c:/RPMH/c.htm 5983 file:///c:/RPMH/s.htm 5601	1 0,96 0,95 0,897 0,894 0,88 0,82 0,77 0,59 0,56	5	28	1
file:///c:/RPMH/P67.htm file:///c:/RPMH/P82.htm file:///c:/RPMH/P68.htm file:///c:/RPMH/P84.htm file:///c:/RPMH/P85.htm file:///c:/RPMH/P86.htm file:///c:/RPMH/P87.htm file:///c:/RPMH/P69.htm file:///c:/RPMH/P70.htm file:///c:/RPMH/P71.htm file:///c:/RPMH/P81.htm file:///c:/RPMH/P82.htm file:///c:/RPMH/P83.htm file:///c:/RPMH/P84.htm	7	file:///c:/RPMH/p.htm 55040 file:///c:/RPMH/u.htm 53474 file:///c:/RPMH/d.htm 53326 file:///c:/RPMH/w.htm 50395 file:///c:/RPMH/t.htm 50266 file:///c:/RPMH/v.htm 49454 file:///c:/RPMH/q.htm 46762 file:///c:/RPMH/f.htm 45828 file:///c:/RPMH/s.htm 34824 file:///c:/RPMH/c.htm 34718	1 0,97 0,96 0,915 0,913 0,89 0,84 0,83 0,632 0,630	6	21	1

Tableau A4.1 : Résultats de la recherche de composantes thématiques du thème « système d'exploitation »

Les URLs des pages Web sélectionnées	Longueur de circuit	Nombre de circuits pour chaque page proche	Proximité de la page proche par rapport à la page racine	Nombre de pages dans le groupe	Nombre de groupes	Nombre de groupes fusionnés
file:///c:/RPMH/p.htm file:///c:/RPMH/c.htm file:///c:/RPMH/a.htm file:///c:/RPMH/b.htm file:///c:/RPMH/d.htm file:///c:/RPMH/e.htm file:///c:/RPMH/f.htm file:///c:/RPMH/q.htm file:///c:/RPMH/s.htm file:///c:/RPMH/r.htm file:///c:/RPMH/t.htm file:///c:/RPMH/u.htm file:///c:/RPMH/v.htm file:///c:/RPMH/w.htm	2	file:///c:/RPMH/v.htm 1 file:///c:/RPMH/u.htm 1 file:///c:/RPMH/t.htm 1 file:///c:/RPMH/s.htm 1 file:///c:/RPMH/q.htm 1 file:///c:/RPMH/f.htm 1 file:///c:/RPMH/e.htm 1 file:///c:/RPMH/d.htm 1 Nulle 0 Nulle 0	1 1 1 1 1 1 1 1 0 0	0	0	0
file:///c:/RPMH/x.htm file:///c:/RPMH/y.htm file:///c:/RPMH/z.htm file:///c:/RPMH/P1.htm file:///c:/RPMH/P2.htm file:///c:/RPMH/P3.htm file:///c:/RPMH/P4.htm file:///c:/RPMH/P5.htm file:///c:/RPMH/P6.htm file:///c:/RPMH/P7.htm file:///c:/RPMH/P8.htm file:///c:/RPMH/P9.htm file:///c:/RPMH/P10.htm file:///c:/RPMH/P111.htm	3	file:///c:/RPMH/u.htm 16 file:///c:/RPMH/t.htm 15 file:///c:/RPMH/f.htm 15 file:///c:/RPMH/e.htm 15 file:///c:/RPMH/v.htm 15 file:///c:/RPMH/d.htm 14 file:///c:/RPMH/s.htm 13 file:///c:/RPMH/q.htm 12 file:///c:/RPMH/b.htm 06 file:///c:/RPMH/c.htm 03	1 0,93 0,93 0,93 0,93 0,87 0,81 0,75 0,37 0,18	3	36	1
file:///c:/RPMH/P112.htm file:///c:/RPMH/P113.htm file:///c:/RPMH/P114.htm file:///c:/RPMH/P115.htm file:///c:/RPMH/P116.htm file:///c:/RPMH/P117.htm file:///c:/RPMH/P118.htm file:///c:/RPMH/P119.htm file:///c:/RPMH/P110.htm file:///c:/RPMH/P120.htm file:///c:/RPMH/P121.htm file:///c:/RPMH/P122.htm file:///c:/RPMH/P123.htm file:///c:/RPMH/P124.htm	4	file:///c:/RPMH/u.htm 151 file:///c:/RPMH/f.htm 143 file:///c:/RPMH/e.htm 142 file:///c:/RPMH/t.htm 139 file:///c:/RPMH/v.htm 139 file:///c:/RPMH/d.htm 130 file:///c:/RPMH/s.htm 118 file:///c:/RPMH/q.htm 109 file:///c:/RPMH/b.htm 70 file:///c:/RPMH/c.htm 39	1 0,947 0,940 0,92 0,92 0,86 0,78 0,72 0,46 0,25	3	36	1
file:///c:/RPMH/P125.htm file:///c:/RPMH/P126.htm file:///c:/RPMH/P127.htm file:///c:/RPMH/P128.htm file:///c:/RPMH/P129.htm file:///c:/RPMH/P130.htm file:///c:/RPMH/P131.htm file:///c:/RPMH/P132.htm file:///c:/RPMH/P133.htm file:///c:/RPMH/P134.htm file:///c:/RPMH/P135.htm file:///c:/RPMH/P136.htm file:///c:/RPMH/P137.htm file:///c:/RPMH/P138.htm	5	file:///c:/RPMH/u.htm 1070 file:///c:/RPMH/f.htm 1024 file:///c:/RPMH/e.htm 1023 file:///c:/RPMH/v.htm 985 file:///c:/RPMH/t.htm 983 file:///c:/RPMH/d.htm 934 file:///c:/RPMH/s.htm 842 file:///c:/RPMH/q.htm 783 file:///c:/RPMH/b.htm 570 file:///c:/RPMH/c.htm 346	1 0,957 0,956 0,92 0,91 0,87 0,78 0,73 0,53 0,32	4	29	1
file:///c:/RPMH/P139.htm file:///c:/RPMH/P140.htm file:///c:/RPMH/P141.htm file:///c:/RPMH/P142.htm file:///c:/RPMH/P144.htm file:///c:/RPMH/P145.htm file:///c:/RPMH/P146.htm file:///c:/RPMH/P147.htm file:///c:/RPMH/P148.htm file:///c:/RPMH/P149.htm file:///c:/RPMH/P150.htm file:///c:/RPMH/P93.htm file:///c:/RPMH/P94.htm file:///c:/RPMH/P95.htm	6	file:///c:/RPMH/u.htm 6003 file:///c:/RPMH/e.htm 5834 file:///c:/RPMH/f.htm 5804 file:///c:/RPMH/v.htm 5570 file:///c:/RPMH/t.htm 5552 file:///c:/RPMH/d.htm 5400 file:///c:/RPMH/s.htm 4840 file:///c:/RPMH/q.htm 4571 file:///c:/RPMH/b.htm 3628 file:///c:/RPMH/c.htm 2368	1 0,97 0,96 0,927 0,924 0,89 0,80 0,76 0,60 0,39	5	21	1
file:///c:/RPMH/P96.htm file:///c:/RPMH/P97.htm file:///c:/RPMH/P98.htm file:///c:/RPMH/P99.htm file:///c:/RPMH/P100.htm file:///c:/RPMH/P101.htm file:///c:/RPMH/P102.htm file:///c:/RPMH/P103.htm file:///c:/RPMH/P104.htm file:///c:/RPMH/P105.htm file:///c:/RPMH/P106.htm file:///c:/RPMH/P107.htm file:///c:/RPMH/P108.htm file:///c:/RPMH/P109.htm	7	file:///c:/RPMH/u.htm 26900 file:///c:/RPMH/e.htm 26487 file:///c:/RPMH/f.htm 26254 file:///c:/RPMH/v.htm 25243 file:///c:/RPMH/t.htm 25159 file:///c:/RPMH/d.htm 25049 file:///c:/RPMH/s.htm 22409 file:///c:/RPMH/q.htm 21547 file:///c:/RPMH/b.htm 18301 file:///c:/RPMH/c.htm 12791	1 0,98 0,97 0,938 0,935 0,931 0,833 0,801 0,68 0,47	6	16	1

Tableau A4.2 : Résultats de la recherche de composantes thématiques du thème « Réseaux et protocoles »

Les URLs des pages Web sélectionnées	Longueur de circuit	Nombre de circuits pour chaque page proche	Proximité de la page proche par rapport à la page racine	Nombre de pages dans le groupe	Nombre de groupes	Nombre de groupes fusionnés
file:///c:/RPMH/P1.htm file:///c:/RPMH/P2.htm file:///c:/RPMH/P3.htm file:///c:/RPMH/P4.htm file:///c:/RPMH/P5.htm file:///c:/RPMH/P50.htm file:///c:/RPMH/P51.htm file:///c:/RPMH/P6.htm file:///c:/RPMH/P7.htm file:///c:/RPMH/P8.htm file:///c:/RPMH/P9.htm file:///c:/RPMH/P10.htm file:///c:/RPMH/P52.htm file:///c:/RPMH/P11.htm	2	file:///c:/RPMH/P94.htm 1 file:///c:/RPMH/P93.htm 1 file:///c:/RPMH/P92.htm 1 file:///c:/RPMH/P91.htm 1 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0	1 1 1 1 0 0 0 0 0 0	0	0	0
file:///c:/RPMH/b.htm file:///c:/RPMH/a.htm file:///c:/RPMH/c.htm file:///c:/RPMH/d.htm file:///c:/RPMH/e.htm file:///c:/RPMH/f.htm file:///c:/RPMH/p.htm file:///c:/RPMH/q.htm file:///c:/RPMH/s.htm file:///c:/RPMH/r.htm file:///c:/RPMH/t.htm file:///c:/RPMH/u.htm file:///c:/RPMH/v.htm file:///c:/RPMH/w.htm	3	file:///c:/RPMH/P38.htm 13 file:///c:/RPMH/P37.htm 13 file:///c:/RPMH/P25.htm 13 file:///c:/RPMH/P24.htm 13 file:///c:/RPMH/P22.htm 13 file:///c:/RPMH/P21.htm 13 file:///c:/RPMH/P75.htm 06 file:///c:/RPMH/P76.htm 06 file:///c:/RPMH/P78.htm 06 file:///c:/RPMH/P79.htm 06	1 1 1 1 1 1 0,46 0,46 0,46 0,46	2	78	1
file:///c:/RPMH/x.htm file:///c:/RPMH/y.htm file:///c:/RPMH/z.htm file:///c:/RPMH/P100.htm file:///c:/RPMH/P101.htm file:///c:/RPMH/P102.htm file:///c:/RPMH/P103.htm file:///c:/RPMH/P104.htm file:///c:/RPMH/P105.htm file:///c:/RPMH/P106.htm file:///c:/RPMH/P107.htm file:///c:/RPMH/P108.htm file:///c:/RPMH/P109.htm file:///c:/RPMH/P110.htm	4	file:///c:/RPMH/P94.htm 105 file:///c:/RPMH/P93.htm 105 file:///c:/RPMH/P92.htm 105 file:///c:/RPMH/P91.htm 105 file:///c:/RPMH/P21.htm 65 file:///c:/RPMH/P22.htm 65 file:///c:/RPMH/P24.htm 65 file:///c:/RPMH/P25.htm 65 file:///c:/RPMH/P37.htm 65 file:///c:/RPMH/P38.htm 65	1 1 1 1 0,61 0,61 0,61 0,61 0,61 0,61	3	164	1
file:///c:/RPMH/P111.htm file:///c:/RPMH/P112.htm file:///c:/RPMH/P113.htm file:///c:/RPMH/P114.htm file:///c:/RPMH/P115.htm file:///c:/RPMH/P116.htm file:///c:/RPMH/P117.htm file:///c:/RPMH/P118.htm file:///c:/RPMH/P119.htm file:///c:/RPMH/P120.htm file:///c:/RPMH/P121.htm file:///c:/RPMH/P122.htm file:///c:/RPMH/P123.htm file:///c:/RPMH/P124.htm file:///c:/RPMH/P125.htm	5	file:///c:/RPMH/P38.htm 1937 file:///c:/RPMH/P37.htm 1937 file:///c:/RPMH/P25.htm 1937 file:///c:/RPMH/P24.htm 1937 file:///c:/RPMH/P22.htm 1937 file:///c:/RPMH/P21.htm 1937 file:///c:/RPMH/P75.htm 1046 file:///c:/RPMH/P76.htm 1046 file:///c:/RPMH/P78.htm 1046 file:///c:/RPMH/P79.htm 1046	1 1 1 1 1 1 0,54 0,54 0,54 0,54	2 et 3	269	6
file:///c:/RPMH/P126.htm file:///c:/RPMH/P127.htm file:///c:/RPMH/P128.htm file:///c:/RPMH/P129.htm file:///c:/RPMH/P130.htm file:///c:/RPMH/P131.htm file:///c:/RPMH/P132.htm file:///c:/RPMH/P133.htm file:///c:/RPMH/P134.htm file:///c:/RPMH/P135.htm file:///c:/RPMH/P136.htm file:///c:/RPMH/P137.htm file:///c:/RPMH/P138.htm	6	file:///c:/RPMH/P38.htm 10673 file:///c:/RPMH/P37.htm 10673 file:///c:/RPMH/P25.htm 10673 file:///c:/RPMH/P24.htm 10673 file:///c:/RPMH/P22.htm 10673 file:///c:/RPMH/P21.htm 10673 file:///c:/RPMH/P42.htm 9724 file:///c:/RPMH/P43.htm 9724 file:///c:/RPMH/P91.htm 8841 file:///c:/RPMH/P92.htm 8841	1 1 1 1 1 1 0,91 0,91 0,82 0,82	2 et 3	270	8
file:///c:/RPMH/P139.htm file:///c:/RPMH/P140.htm file:///c:/RPMH/P141.htm file:///c:/RPMH/P142.htm file:///c:/RPMH/P143.htm file:///c:/RPMH/P144.htm file:///c:/RPMH/P145.htm file:///c:/RPMH/P146.htm file:///c:/RPMH/P147.htm file:///c:/RPMH/P148.htm file:///c:/RPMH/P149.htm file:///c:/RPMH/P150.htm file:///c:/RPMH/P21.htm file:///c:/RPMH/P53.htm file:///c:/RPMH/P12.htm file:///c:/RPMH/P13.htm file:///c:/RPMH/P14.htm file:///c:/RPMH/P15.htm file:///c:/RPMH/P16.htm file:///c:/RPMH/P17.htm file:///c:/RPMH/P18.htm file:///c:/RPMH/P19.htm file:///c:/RPMH/P20.htm	7	Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0 Nulle 0	0 0 0 0 0 0 0 0 0 0	0	0	0

file:///c:/RPMH/P22.htm file:///c:/RPMH/P54.htm file:///c:/RPMH/P23.htm file:///c:/RPMH/P55.htm file:///c:/RPMH/P24.htm file:///c:/RPMH/P56.htm file:///c:/RPMH/P25.htm file:///c:/RPMH/P57.htm file:///c:/RPMH/P26.htm file:///c:/RPMH/P58.htm file:///c:/RPMH/P27.htm file:///c:/RPMH/P59.htm file:///c:/RPMH/P28.htm file:///c:/RPMH/P60.htm file:///c:/RPMH/P29.htm file:///c:/RPMH/P61.htm file:///c:/RPMH/P30.htm file:///c:/RPMH/P62.htm file:///c:/RPMH/P31.htm file:///c:/RPMH/P63.htm file:///c:/RPMH/P32.htm file:///c:/RPMH/P64.htm file:///c:/RPMH/P33.htm file:///c:/RPMH/P65.htm file:///c:/RPMH/P34.htm file:///c:/RPMH/P66.htm file:///c:/RPMH/P35.htm file:///c:/RPMH/P67.htm file:///c:/RPMH/P36.htm file:///c:/RPMH/P68.htm file:///c:/RPMH/P37.htm file:///c:/RPMH/P69.htm file:///c:/RPMH/P38.htm file:///c:/RPMH/P70.htm file:///c:/RPMH/P39.htm file:///c:/RPMH/P71.htm file:///c:/RPMH/P40.htm file:///c:/RPMH/P72.htm file:///c:/RPMH/P41.htm file:///c:/RPMH/P73.htm file:///c:/RPMH/P42.htm file:///c:/RPMH/P74.htm file:///c:/RPMH/P43.htm file:///c:/RPMH/P75.htm file:///c:/RPMH/P44.htm file:///c:/RPMH/P76.htm file:///c:/RPMH/P45.htm file:///c:/RPMH/P77.htm file:///c:/RPMH/P46.htm file:///c:/RPMH/P78.htm file:///c:/RPMH/P47.htm file:///c:/RPMH/P79.htm file:///c:/RPMH/P48.htm file:///c:/RPMH/P80.htm file:///c:/RPMH/P49.htm file:///c:/RPMH/P81.htm file:///c:/RPMH/P95.htm file:///c:/RPMH/P96.htm file:///c:/RPMH/P97.htm file:///c:/RPMH/P82.htm file:///c:/RPMH/P83.htm file:///c:/RPMH/P84.htm file:///c:/RPMH/P85.htm file:///c:/RPMH/P86.htm file:///c:/RPMH/P87.htm file:///c:/RPMH/P88.htm file:///c:/RPMH/P89.htm file:///c:/RPMH/P90.htm file:///c:/RPMH/P91.htm file:///c:/RPMH/P92.htm file:///c:/RPMH/P93.htm file:///c:/RPMH/P94.htm						

Tableau A4.3 : Résultats de la recherche de composantes thématiques du thème « Base de Données »

Les expérimentations présentées dans les trois tableaux A4.1, A4.2 et A4.3 justifient bien nos choix pour une longueur de circuit égale à 4. En effet, l'ordre de pages Web thématiquement proches récupérées pour la page racine se stabilise à partir d'une longueur maximale de circuits égale à 4. A partir d'une longueur de circuits égale à 5, le nombre de circuits récupérés pour chaque page proche devient très important (voir figure A4.1). En conséquence, nous avons plusieurs pages qui ne font pas partie de la composante thématique de la page Web racine.

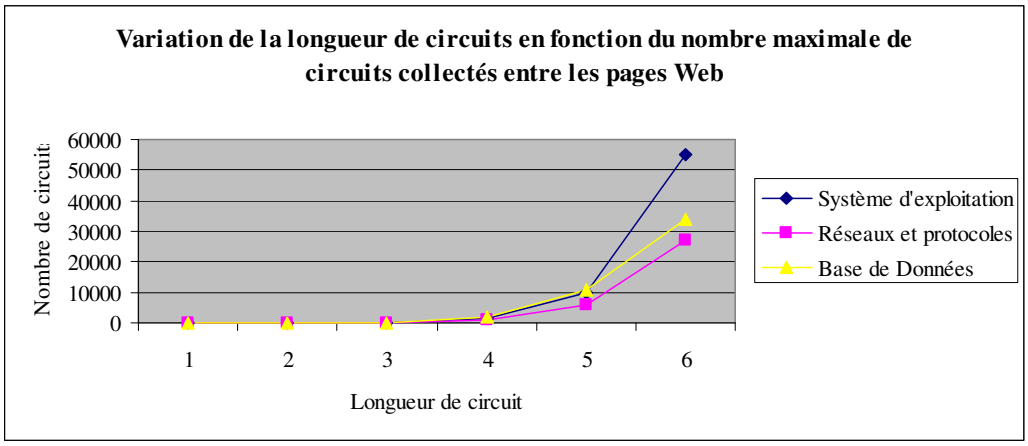


Figure A4.1 : Courbes de variation de la longeur de circuit en fonction du nombre maximale de circuits collectés entre les pages Web

Il s'est avéré que pour les trois thèmes différents (systèmes d'exploitation, Réseaux et protocoles, Bases de données), toutes les composantes thématiques seront fusionnées dans un seul groupe pour une longueur de circuit égale à 4. Ceci montre bien la performance de notre approche à base de circuits dans la construction des composantes thématiques dans le RPMH de pages Web.

Annexe 5 :

Résultats des expérimentations

Nous présentons dans cette annexe les résultats collectés suite aux cinq expériences de test de performance de notre SRI multi-agent SARIPOD.

Le tableau A5.1 récapitule ces résultats en terme de :

- Nombre de terme de la requête,
- Nombre de documents retrouvés par le système,
- Nombre de documents jugés pertinents par le système,
- Taux de documents pertinents par rapport aux documents retrouvés,
- Nombre de classes proposées par l'expert,
- Nombre de classes proposées par le système,
- Taux de classification du système par rapport à la classification de l'expert,

	Nombre de terme de la requête	**Nombre de documents retrouvés par le système**	**Nombre de documents jugés pertinents par le système**	**Taux de documents pertinents par rapport aux documents retrouvés**	**Nombre de classes proposées par l'expert**	**Nombre de classes proposées par le système**	**Taux de classification du système par rapport à la classification de l'expert**
Expérience 1	4	24	24	100%	1	1	100%
Expérience 2	7	172	146	84,88%	2	4	200%
Expérience 3	12	349	317	90,83%	4	5	125%
Expérience 4	16	297	262	88,21%	5	2	40%
Expérience 5	15	207	189	91,3%	4	3	75%

Tableau A5.1 : Récapitulations des résultats des cinq expériences de classification de documents

Dans le tableau A5.2, nous présentons le score de la pertinence possibiliste de chaque document retrouvé par le système. Les documents titulaires des scores des pertinences possibilistes négatifs ou nuls sont jugés non pertinents pour le système.

Expérience 1	Expérience 2	Expérience 3	Expérience 4	Expérience 5
20,22	53,65	58,817734	102,85487	66,36355
19,39	51,01	56,01483	67,07286	57,092854
17,55	50,51	52,71437	60,009636	56,514774
15,90	47,70	52,20844	55,949837	56,477985
13,45	47,26	51,522385	52,563118	55,42903
5,59	43,59	51,184376	51,73824	54,90814
5,33	42,14	50,259586	51,504044	54,33407
5,33	41,42	49,32683	51,00206	53,285316
5,33	40,54	47,89886	50,686104	52,289246
5,14	37,20	46,879864	49,06194	52,235878
5,14	32,43	46,858128	48,454086	49,040775
4,98	32,18	46,13898	47,1131	48,51172
4,98	31,29	45,707935	46,668518	48,1223
4,85	29,38	45,421528	46,468113	46,631073
4,73	28,96	45,250206	46,400677	46,43979
4,63	28,12	44,803947	43,862762	45,48456
4,54	27,85	44,704777	42,822388	44,763622
4,46	27,76	44,41924	39,795128	44,10233
4,38	27,50	44,239536	39,251328	42,48602
4,31	26,67	42,504345	38,224167	41,796158
4,25	25,82	40,876587	37,66309	40,36433
4,19	25,06	40,452824	33,84117	37,87319
4,13	24,99	40,19784	33,781223	37,186157
4,07	24,17	39,80896	33,252426	36,33839
	24,01	36,913837	32,893013	34,99326
	23,60	36,786964	32,78991	34,958103
	23,44	35,07938	32,628746	34,168713
	23,37	35,07408	31,143106	33,9636
	23,23	34,688137	30,743183	32,84442
	23,10	34,218185	30,426756	32,364227
	22,84	32,14723	30,307026	32,24236
	22,25	32,052048	29,919996	30,942766
	22,18	31,372522	29,349915	30,58426
	22,02	30,426914	28,930378	30,51868
	21,82	30,00851	28,878382	30,26866
	21,58	29,74598	28,700542	30,075129
	21,52	29,719501	28,614357	29,194088
	21,23	29,321217	28,53998	28,545162
	21,12	29,212599	28,442467	28,451214
	21,04	29,178679	27,586496	28,4023
	20,98	29,17271	27,286705	28,281687
	20,83	29,1354	26,505045	28,09709
	20,26	29,10649	25,937336	27,877016
	19,93	29,083166	25,700222	27,761475
	19,67	29,041706	25,5145	27,57396
	19,38	28,97194	25,424425	27,561884
	19,14	28,61219	25,052742	27,322163
	18,97	28,544659	25,03646	27,284626
	18,78	28,345373	24,67644	27,143082
	18,67	28,308882	24,647984	27,028929

	18,56	28,173225	24,56528	26,732925
	18,32	27,617582	24,289183	26,620544
	18,19	27,210487	24,123184	26,39511
	17,86	27,06193	23,596212	26,388937
	17,45	27,027508	23,458603	26,267977
	16,16	26,716825	23,169958	26,19879
	14,63	26,328983	23,142488	26,156282
	12,63	25,96592	22,708124	25,915047
	12,47	25,68135	22,618538	25,80925
	12,25	25,609777	22,45853	25,691277
	11,84	25,468338	22,367214	25,34619
	11,35	25,272038	22,283796	25,188223
	10,47	24,9538	22,277256	25,027967
	8,70	24,847345	22,059237	24,62368
	7,97	24,666597	22,026463	24,596767
	7,56	24,623234	21,81497	24,56366
	7,56	24,612713	21,378582	24,501038
	7,31	24,559395	21,328465	24,355154
	6,96	24,28071	21,135172	24,28278
	6,86	24,182405	21,07433	24,263863
	6,82	24,061277	20,992363	24,1804
	6,71	23,935366	20,966442	24,076426
	6,71	23,638987	20,875366	23,97662
	6,61	23,36378	20,84227	23,954126
	6,61	23,000593	20,630413	23,709024
	6,51	22,728685	20,397442	23,65819
	6,51	22,51168	20,379686	23,57935
	6,51	22,500042	20,317974	23,453272
	6,50	22,227163	19,838737	23,4529
	6,43	21,907192	19,79975	23,429764
	6,43	21,816511	19,668602	23,322662
	6,36	21,561031	19,629408	23,162416
	6,29	21,532568	19,55545	23,003902
	6,22	21,484833	19,386042	22,9091
	6,22	21,392485	19,170433	22,792103
	6,16	21,386574	19,146414	22,790518
	6,11	21,36895	18,629084	22,787537
	6,05	21,368074	18,472658	22,717783
	6,00	21,25997	18,25639	22,499296
	5,96	21,249783	18,255953	22,398369
	5,91	21,240591	18,254557	22,168003
	5,87	21,141457	18,213846	22,121878
	5,56	21,134016	18,201817	21,944426
	5,43	21,089989	18,172176	21,941801
	5,15	21,0721	17,564991	21,806942
	5,10	20,996286	17,521095	21,783184
	4,97	20,936962	17,360878	21,55336
	4,82	20,83631	17,352354	21,497417
	4,80	20,733881	17,309385	21,354908
	4,76	20,711805	17,250664	21,205982
	4,75	20,539225	16,982658	21,038036

	4,70	20,489592	16,608463	20,716362
	4,68	20,485954	16,562319	20,471416
	4,67	20,314577	16,192585	20,257689
	4,64	20,228205	16,02805	19,965096
	4,62	20,179502	16,006617	19,248837
	4,61	20,053694	15,791422	18,997986
	4,60	19,93748	15,705762	17,749416
	4,59	19,92392	15,601	17,425026
	4,58	19,879454	15,287069	17,353664
	4,58	19,874527	14,462973	15,801656
	4,57	19,789125	14,43328	14,38924
	4,56	19,695406	14,112987	14,377739
	4,55	19,572248	13,933811	13,585754
	4,54	19,56979	13,79191	13,469146
	4,53	19,444323	13,619904	13,336408
	4,52	19,441336	13,579743	12,439315
	4,51	19,42832	13,279229	9,008749
	4,50	19,35289	12,689574	8,484661
	4,50	19,290077	12,390535	7,5578933
	4,49	19,263647	12,322101	7,3079677
	4,48	19,099398	12,203095	7,3079677
	4,47	19,097609	11,95751	7,3079677
	4,46	18,837315	11,466781	7,2516828
	4,45	18,814867	11,385778	6,955689
	4,45	18,553457	11,160178	6,955689
	4,43	18,531258	10,909509	6,955689
	4,43	18,512629	10,831768	6,821892
	4,42	18,475822	10,808481	6,821892
	4,41	18,454208	10,80498	6,705908
	4,40	18,389202	10,4391575	6,705908
	3,74	18,154423	10,427081	6,705908
	3,72	17,91485	10,382826	6,705908
	3,59	17,86459	9,890645	6,603362
	3,51	17,781096	9,726243	6,603362
	3,24	17,665234	9,674974	6,53432
	2,53	17,65165	9,520321	6,5119915
	2,29	17,558712	9,458994	6,5119915
	2,15	17,487265	9,347502	6,5119915
	2,03	17,392496	9,148882	6,4291577
	2,03	17,299042	8,968245	6,4291577
	1,64	17,206709	8,912293	6,4291577
	1,33	17,169666	8,624437	6,4291577
	1,14	17,109879	8,558085	6,4291577
	0,30	16,99847	8,486207	6,3533397
	0,13	16,992899	8,40065	6,284153
	-0,87	16,95761	8,168892	6,219832
	-2,10	16,88633	8,114899	6,159809
	-2,77	16,657866	8,03533	6,103848
	-2,82	16,605307	7,8726654	6,103848
	-2,88	16,529015	7,7906747	6,103848
	-2,88	16,522335	7,3927794	6,103848

	-2,93	16,506842	7,1524525	6,0506115
	-2,99	16,324675	7,018405	6,0506115
	-3,17	16,215212	7,018405	6,0008683
	-3,24	16,169827	6,9704027	5,866975
	-3,37	16,145842	6,902344	5,866975
	-3,44	16,138056	6,708601	5,8265676
	-3,52	16,069258	6,5574365	5,8265676
	-3,55	15,97731	6,4899364	5,75128
	-3,58	15,883138	6,4610214	5,715871
	-3,59	15,854984	6,4362445	5,6814666
	-3,66	15,837788	6,3562264	5,6482515
	-3,67	15,745454	6,151094	5,6177716
	-3,75	15,425653	6,0636244	5,557749
	-3,78	15,384319	5,9890294	5,5283523
	-4,02	15,195276	5,948102	5,4485517
	-4,11	15,104489	5,948102	5,4236145
	-4,48	15,077839	5,912337	5,3979397
	-4,78	15,069502	5,912337	5,306425
	-6,27	14,971918	5,8780394	5,2849293
	-28,39	14,912968	5,8780394	5,2649145
		14,875858	5,845451	5,244428
		14,765024	5,8137474	5,223447
		14,652199	5,8137474	5,2041197
		14,489128	5,783075	5,186572
		14,106266	5,7535896	5,186572
		14,073114	5,7254553	5,175967
		13,780062	5,7254553	5,1316957
		13,7567835	5,6976094	5,112605
		13,741832	5,6976094	4,71587
		13,639359	5,645643	3,3690562
		12,715727	5,6204653	2,4227905
		12,407363	5,4462647	2,0026696
		12,389941	5,310277	1,9868947
		12,348577	5,2928076	1,9831243
		12,273359	5,2828193	1,9580564
		12,257402	5,2749796	1,7643362
		12,252163	5,1777573	0,6091099
		11,809918	5,0091558	**0**
		11,729418	4,970831	**-0,36443257**
		11,710194	4,921796	**-0,68725324**
		11,343309	4,8366027	**-3,5382848**
		11,085043	4,8096676	**-3,9474964**
		10,774433	4,788903	**-4,9880686**
		10,718988	4,7058825	**-5,3923907**
		10,41238	4,6729193	**-7,9135294**
		10,170755	4,6402926	**-7,997034**
		10,138035	4,5249023	**-8,746999**
		9,992364	4,5249023	**-9,132126**
		9,942102	4,505706	**-9,905433**
		9,290689	4,50084	**-10,292495**
		9,183305	4,440216	**-10,567971**

		9,028891	4,440216	**-10,797223**
		8,900472	4,434968	**-14,121737**
		8,798491	4,4292564	**-15,415162**
		8,139239	4,41903	**-29,262817**
		8,017206	4,2095313	
		7,8137093	3,824278	
		7,7384634	3,804915	
		7,3862224	3,6653848	
		7,3862224	3,62599	
		7,2522697	3,6136992	
		7,1047096	3,3988864	
		6,822481	3,1764593	
		6,445156	3,0818295	
		6,436856	2,9383495	
		5,5266323	2,867663	
		5,2888775	2,6928358	
		5,2506247	2,6389632	
		5,238187	2,6338053	
		5,2277346	2,5871358	
		5,2276835	2,4526253	
		5,1774383	2,4314833	
		5,148976	2,4306755	
		4,924796	2,245749	
		4,915764	2,239699	
		4,9066367	2,2374737	
		4,8974123	2,220019	
		4,8912086	2,1725776	
		4,881818	2,1654675	
		4,875501	2,1181493	
		4,8672385	2,1026342	
		4,8659387	2,0735219	
		4,8595047	2,0094852	
		4,849763	2,0041983	
		4,843208	2,0004437	
		4,833281	2	
		4,8265996	1,9988083	
		4,819866	1,9941947	
		4,8130803	1,9918282	
		4,806241	1,8892615	
		4,7958803	1,8354791	
		4,788903	1,8162884	
		4,78187	1,811647	
		4,7747793	1,696422	
		4,7604227	1,694921	
		4,753154	1,6789998	
		4,631941	1,6784356	
		4,437026	1,6275415	
		4,3908134	1,5585046	
		4,2892394	1,5216408	
		4,260668	1,2683628	
		4,2211795	1,1336935	

		4,2076073	1,1026486	
		4,1363716	1,022166	
		4,128916	0,99387324	
		4,1213956	0,7085576	
		4,117352	0,60000384	
		4,106669	0,47485054	
		4,106157	0,39252377	
		4,106157	0,07393527	
		4,090646	**-0,021297932**	
		4,066848	**-0,28438497**	
		4,066848	**-0,36338902**	
		4,0506115	**-0,95102096**	
		4,0506115	**-1,1720736**	
		4,0423784	**-1,4203012**	
		4,0256743	**-1,729958**	
		4	**-2,0460267**	
		3,5578308	**-2,0688987**	
		3,3775826	**-2,0987973**	
		3,2221465	**-2,23036**	
		3,1466963	**-2,559577**	
		3,1372252	**-2,8109088**	
		3,0287604	**-3,1001573**	
		2,7326393	**-3,6394954**	
		2,711725	**-3,709426**	
		2,5883906	**-3,776102**	
		2,5493746	**-4,537393**	
		2,441471	**-4,569243**	
		2,432969	**-4,894354**	
		2,3810232	**-5,5765266**	
		2,2083035	**-5,586808**	
		2,2059577	**-5,6946945**	
		2,192718	**-5,8215933**	
		2,1424398	**-6,3575964**	
		2,1164408	**-6,6993914**	
		2,0689666	**-7,5585575**	
		2,0661588	**-7,5669928**	
		2,0634491	**-7,952318**	
		2,051265	**-7,9952154**	
		2,0210712	**-11,863697**	
		2,0148616	**-15,781929**	
		2,0099323	**-26,021086**	
		2,0000322	**-57,39626**	
		2,000001	**-91,43486**	
		1,9826183		
		1,976609		
		1,9725448		
		1,9681702		
		1,9585527		
		1,954885		
		1,8932962		
		1,8516327		

		1,7779392		
		1,7495506		
		1,694269		
		1,6711214		
		1,6550298		
		1,5925949		
		1,4562621		
		1,4102452		
		1,2070794		
		1,0999503		
		0,92153263		
		0,07347238		
		-0,04367447		
		-0,36056328		
		-0,4416778		
		-0,6276047		
		-0,8392923		
		-0,92019606		
		-0,96275043		
		-1,0645797		
		-1,1928685		
		-1,3257718		
		-1,466121		
		-1,963647		
		-2,3476963		
		-2,6200395		
		-3,576488		
		-4,182135		
		-4,8814616		
		-5,354793		
		-5,511408		
		-7,5597396		
		-7,791536		
		-7,905372		
		-8,383428		
		-8,831229		
		-10,595719		
		-11,181103		
		-18,07492		
		-18,47083		
		-22,274036		
		-30,369854		
		-42,07043		
		-2388,9187		

Tableau A5.2 : Les scores des pertinences possibilistes des documents retrouvés