Ministère de l'enseignement supérieur et de la recherche scientifique
Université Mohamed Khider Biskra
Faculté des Sciences Exactes et Sciences de la nature et de la vie
Département d'informatique

N° d'ordre : .....
Série : .....

Mémoire
Présenté en vue de l'obtention du diplôme de Magistère en informatique

Option :
Intelligence Artificielle et système d'informations avancés

LA RECHERCHE D'INFORMATIONS SEMANTIQUE D'INFORMATIONS DANS LE CADRE DU WEB SEMANTIQUE

Présenté par :

Mr Ali SRITI

Soutenu le :25 Juin 2012

Devant le jury :

Mr Mahmoud BOUFAIDA	**Professeur Université Mentouri, Constantine**	**Président**
Mme Zizette BOUFAIDA	**Professeur Université Mentouri, Constantine**	**Rapporteur**
Mr Nacereddine ZAAROUR	**Professeur Université Mentouri, Constantine**	**Examinateur**
Mr Okba KAZAR	**Professeur Université Med Khider, Biskra**	**Examinateur**

Résumé

Le web sémantique assure une meilleure adéquation entre les besoins de l'utilisateur et les résultats renvoyés par les résultats renvoyés pas les moteurs de recherches. L'introduction d'un langage contrôlé, extrait des termes présents dans les ontologies de références améliore la qualité de l'index d'une part, mais optimise les fonctions de rappels d'autres part en limitant les résultats aux seules documents pertinents.

La collaboration entre des agents embarqués sur des fourmis permet de garantir la fraîcheur des ces dits index, en assurant une couverture quasi-permanente de petites partitions du web.

Dans notre travail, nous avons proposés un modèle distribué basé sur les algorithmes génétiques pour construire et exploiter un index sémantique du web.

Mots clés :

Recherche d'informations, Web sémantique, Algorithme génétiques, Agents fourmis.

الملخص

الويب الدلالي يوفر أفضل بين احتياجات المستخدمين والنتائج التي تم إرجاعها من قبل النتائج التي تم إرجاعها من قبل محركات البحث. إدخال لغة تسيطر عليها، ومقتطفات شروط موجودة في الأنطولوجيا مرجع يحسن نوعية المؤشر من جهة، ولكن يحسن النتائج من جهة أخرى عن طريق الحد من النتائج إلى الوثائق ذات الصلة فقط.

تعاون بين وكلاء المصممة على شكل مستوطنات النمل يضمن نضارة المؤشر تغطية دائمة تقريبا من أصغر أقسام من شبكة الإنترنت.

في عملنا، اقترحنا نموذج موزع على أساس الخوارزميات الجينية لبناء وتشغيل مؤشر دلالي على شبكة الإنترنت.

كلمات البحث:

استرجاع المعلومات، والويب الدلالي، الخوارزمية الجينية، وكلاء نملة.

REMERCIEMENTS

D'abords et avant toute chose, je remercie Dieu de m'avoir donné la force, la foi et la volonté de mener ce travail terme.

Je remercie mon encadreur, Professeur Zizette BOUFAIDA, de l'université Mentouri de Constantine d'avoir accepté d'encadrer ce travail, de m'avoir guidé et orienté d'un bout à l'autre d'un long et fastidieux chemin. Je la remercie tout spécialement pour sa patience, sa disponibilité et sa compréhension. Veuillez trouver ici l'expression de ma profonde gratitude.

Mes remerciements vont également aux membres du jury : Mrs Mahmoud BOUFAIDA, Nacereddine ZAAROUR et Okba KAZAR qui ont accepté d'examiner et juger ce travail.

Mes remerciements vont tout particulièrement au Dr Abderrezak DEBILOU, vice recteur de l'université de Biskra et au Pr Okba KAZAR d'avoir, sans compromis, cru en moi, de m'avoir soutenu, poussé au-delà de mes propres limites, qu'ils trouvent ici mon indéfectible gratitude et mon amitié.

Je remercie également Mr Djamel ASSASSI, conservateur de la bibliothèque centrale de l'université de Biskra pour son soutien et son amitié tout au long de ces années.

Je remercie le Pr Brahim MEZARDI et le Pr Abdelhamid GUETTALA vice-recteurs de l'université de Biskra pour leur disponibilité, leur gentillesse. Je leur suis redevable.

Mes remerciements vont à ma famille. A mes Parents d'abord qui, très tôt, m'ont fait comprendre qu'aucune voie ne mène seule à la connaissance. Que nul savoir n'est vain, qu'on ne peut prétendre évoluer, tout autant techniciens que nous espérons l'être, sans aimer les lettres et les arts.

Je remercie enfin mon épouse pour avoir été là au quotidien bien que ce ne fut point simple, d'avoir été disponible, patiente, aimante et réconfortante. Qu'elle sache que sans elle, ceci n'aurait jamais abouti.

Que ceux que j'ai oublié de citer sachent qu'ils trouveront en moi l'ami que j'ai toujours été.

A tous, Merci.

Ali SRITI

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES EQUATIONS

Introduction Générale

Ce n'est pas assez d'avoir l'esprit bon, mais le principal est de l'appliquer bien.
Discours de la méthode. René Descartes, 1637.

Internet est devenu le moyen de communication par excellence. La démocratisation de l'outil informatique a rendu aisées la création et la publication de pages web, personnelles ou professionnelles. La quantité d'informations, sous toutes les formes (pages, documents textuels, fichiers multimédias) est devenue très importante.

Comment trouver une information ? Comment ne retrouver que l'information recherchée ? Comment restreindre la recherche à un domaine ?

A toutes ces questions, les systèmes de recherches d'information, apportent des réponses plus ou moins satisfaisantes. Ils assistent l'utilisateur, pas forcément expert, à retrouver une information dans la masse colossale de données disponibles sur le web.

En fait, un système de recherche d'informations 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'informations désigne alors le processus qui permet, à partir d'une expression des besoins d'informations d'un utilisateur, de retrouver l'ensemble des documents contenant l'information recherchée, et ce par la mise en œuvre 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).

Ainsi, l'utilisateur doit formuler une requête, sensée exprimer ses besoins, en utilisant des

termes qu'il juge décrire au mieux son champ d'investigation. Le système tente d'identifier les documents et/ou enregistrements qui répondent à ce besoin et les renvoie sous forme de liste, traditionnellement triée selon la pertinence estimée de chaque document.

Cependant, nous savons que l'utilisateur connait ses besoins mais ne sait pas forcément les exprimer, du moins avec les termes exacts avec lesquels le système les a décrits. L'utilisateur est submergé par des milliers, voire des millions de réponses présentées « *en vrac* », dont une partie ne correspond pas à la requête, une autre correspond à des pages qui n'existent plus, sans parler de nombreux types de documents qui ne sont pas pris en compte dans la mémoire du moteur. Dans bien des cas, l'utilisateur passe un temps assez long à analyser les résultats du moteur, sans garantie de satisfaction.

Partant de ce constat, notre travail porte sur deux problèmes principaux : la pertinence des résultats par rapport à la requête ; la qualité de l'index, exhaustivité et fraicheur.

Problématique

L'avènement du web sémantique, défini par Berners-Lee (1999), et notamment l'utilisation des ontologies, a permis aux moteurs de recherches d'évoluer vers une meilleure organisation, indexation, du corpus, mais également vers un classement plus pertinent des résultats.

Les ontologies sont un moyen de fournir des représentations de connaissances qui correspondent à « *une spécification explicite et formelle d'une conceptualisation partagée* » et font l'objet de nombreux travaux de recherche. Ces travaux s'attachent, d'une part, à définir des méthodologies et des techniques permettant leur élaboration à partir de textes et, d'autre part, à leur utilisation dans les systèmes de recherche d'informations.

Une ontologie fournit une référence pour la communication entre les machines mais aussi entre humains et machines en définissant le sens des objets. Ceci est fait tout d'abord à travers les symboles qui les désignent et les caractérisent et ensuite à travers une représentation structurée ou formelle de leur rôle dans le domaine. L'utilisation d'ontologies dans un modèle de recherche d'informations a pour finalité de spécifier des connaissances qui seront interprétables, d'une part par l'utilisateur du système, et d'autre part par le système lui-même.

Les ontologies sont alors utilisées dans toutes les phases du processus de recherche d'informations. Elles permettent, d'une part, de réduire l'index du système aux seuls concepts qui définissent le domaine couvert par l'ontologie. Ainsi, seuls les documents pertinents par rapport au domaine sont pris en compte lors de la phase d'indexation : ceci réduit les bruits et améliore la *précision* : la proportion de documents pertinents parmi les documents renvoyés. D'autre part, en aidant l'utilisateur à formuler des requêtes plus précises, ou moins confuses, les ontologies permettent d'augmenter le taux de *rappel :* proportion de documents pertinents renvoyés par rapport aux documents pertinents dans le corpus, en reformulant la requête

utilisateur.

Cependant, même si l'utilisation des ontologies résout le problème de l'exhaustivité de l'index, sa fraicheur reste un problème récurrent. En effet, l'indexation du corpus étant un processus discret, le corpus peut évoluer entre deux cycles d'indexation. Il faut donc maintenir une surveillance quasi constante de l'ensemble du web.

La modélisation d'internet par des graphes orientés où les documents représentent les nœuds et les liens hypertextes les arcs reliant ces nœuds, nous permet d'appliquer les techniques d'optimisation issues des algorithmes génétiques. En effet, la taille du graphe du web, sa complexité, et sa perpétuelle mutation, nous amènent à considérer le problème d'exploration du web comme un problème NP-difficile, i.e. qui ne peut être résolu par les méthodes polynomiales dans des délais raisonnables.

Le partitionnement du graphe du web permet de distribuer le problème de parcours du web. En effet, une société d'agents peut parcourir en continu une partition du graphe du web. En s'inspirant des colonies de fourmis fourrageuses, la communauté scientifique a développé des algorithmes extrêmement simples mais qui, mis en collaboration, produisent des résultats extraordinaires. De plus, vu la taille réduite de la partition, un agent peut détecter les mises à jour dans le corpus et les intégrer à la représentation du corpus : l'index.

Organisation du mémoire

Ce mémoire tente de cerner l'ensemble des problèmes posés par les moteurs de recherches « *classiques* » et de voir comment l'introduction de la sémantique peut, dans une certaine mesure, améliorer la qualité de la recherche d'informations en prenant en compte les relations sémantiques entre ses entrées et en améliorant la requête utilisateur, en la réécrivant ou en l'enrichissant.

Le présent mémoire est organisé en quatre chapitres réunis en deux parties :

Partie 1 : Etat de l'art

Le Chapitre I : La recherche d'informations sur le Web est une revue, aussi complète que possible, des systèmes de recherche d'informations sur le web. Nous présenterons l'architecture générale, expliquerons le principe de fonctionnement de chacun de ses processus. Nous détaillerons tout particulièrement le processus d'indexation, cœur de tout système de recherche d'informations.

Le Chapitre II : La recherche sémantique d'informations traite de l'apport du web sémantique à la recherche d'informations. L'introduction des ontologies, comme langage de référence à l'indexation sera détaillée. Nous traiterons de l'indexation conceptuelle des documents et des mesures de similarités entre concepts de l'ontologie et de l'appariement

sémantique des documents par rapport aux requêtes de l'utilisateur.

Partie 2 : contribution

Dans le chapitre III : Métaheuristiques et recherche d'informations, nous proposons un modèle de recherche d'informations sémantique sur la base de métaheuristiques. Nous commencerons par définir les algorithmes génétiques. Une attention particulière est portée aux systèmes multi-agents à base de colonies de fourmis pour leur capacité à trouver le meilleur chemin, pas forcément le plus court, menant à la source de nourriture. Nous terminerons le chapitre en redéfinissant le problème de recherche d'informations pour un traitement métaheuristique.

Enfin, le Chapitre IV : Un modèle distribué pour la recherche sémantique d'information est une conceptualisation du modèle précédemment proposé. Nous traiterons de l'exploration du web par des colonies de fourmis, notamment les fonctions de dépôt et d'évaporation des phéromones, mais aussi des stratégies d'explorations retenues. Nous proposerons ensuite une architecture pour la pérennisation des données, ontologie et index, qui prend en compte les relations sémantiques entre les concepts eux-mêmes, et entre les concepts et les documents les décrivant. Nous terminerons par proposer une architecture globale du système.

Chapitre I :
La recherche d'informations sur le Web

Il y a des choses que l'intelligence seule est capable de chercher, mais que, par elle-même, elle ne trouvera jamais. Ces choses, l'instinct seul les trouverait; mais il ne les recherchera jamais.
L'Evolution créatrice. Henri Bergson, 1907.

Il existe actuellement près de 100 milliards de pages sur la toile. Comment trouver l'information qui répondrait exactement le mieux à nos besoins ? Comment formuler la requête qui exprimerait au mieux nos besoins ?

Trouver une aiguille dans une botte de foin ! Est-ce possible ? Est-ce raisonnablement faisable ? Les plus pessimistes diraient que c'est une vaine quête ! Les plus optimistes diraient que rien n'est plus facile.

Le paradoxe étant que les deux protagonistes ont à la fois tort et raison. Le problème se pose en ces termes : comment retrouver toutes les informations pertinentes et rien qu'elles ?

Quel que soit le moteur de recherche utilisé, le processus commencera toujours par la formulation d'une requête. Celle-ci devrait dans l'absolu exprimer exactement les attentes de l'utilisateur. Le SRI[1] tentera d'apparier les termes de la requête avec les documents du corpus indexés au préalable. Les résultats seront ensuite renvoyés à l'utilisateur qui décidera quels documents conviennent à ses besoins.

A chaque niveau, différentes techniques d'optimisation seront utilisées. Du choix de tel ou tel modèle de représentation, d'appariement et de restitution dépendra la qualité du système.

[1] SRI : Système de Recherche d'Informations

I.1 LES CONCEPTS DE BASE DE LA RECHERCHE D'INFORMATIONS

I.1.1 Principaux acteurs du processus

I.1.1.1. Le document

La notion de document est particulièrement complexe à définir ; dans son acception courante, l'une des définitions possibles de ce terme est de considérer un document comme le support physique d'une information. Dans le cas des données susceptibles d'être manipulées par un système de recherche d'informations, ce support physique, et plus particulièrement numérique, peut correspondre à un texte, une page Web, une image, une séquence vidéo ou sonore...

Un document est, dans ce cadre, défini comme toute unité susceptible de constituer une réponse à une requête d'un utilisateur. Un document-texte peut être représenté selon trois vues (SAUVAGNAT, 2005) : **la vue présentation** qui décrit la représentation sur un medium à deux dimensions (alignement de paragraphes, indentation, en-têtes et pieds de pages...), **la vue logique** qui contient des informations sur la structure et la partition d'un document (une structuration en chapitres, sections...) et **la vue contenu**, appelée également *vue sémantique*, qui se concentre sur le contenu textuel du document, l'information qui y est véhiculée.

L'unité documentaire est donc représentée par un support physique (le texte) associé à une information véhiculée par son contenu sémantique. L'ensemble des documents mis à disposition du SRI pour lui permettre de retrouver l'information recherchée par l'utilisateur constitue la base documentaire.

I.1.1.2. La requête

La requête se définit en recherche d'informations au sein d'un processus cognitif plus large représenté par le besoin d'informations d'un utilisateur. Ce besoin correspond à l'expression mentale de l'information qu'il recherche. Le passage d'un besoin d'informations à son expression en des termes compréhensibles par un système de recherche d'informations est une tâche difficile. Dans l'idéal, l'utilisateur devrait avoir des connaissances sur le système lui-même, sur la collection de documents disponibles, sur les thèmes associés à ces documents... Parallèlement, le système de recherche d'informations devrait pouvoir s'adapter au contexte précis du besoin de l'utilisateur, connaître son degré d'expertise dans le domaine de l'information recherchée, ses centres d'intérêts, ses préférences de recherches (type de documents, langue...).

Bien que les travaux en recherche d'informations portant sur une description plus précise

et plus pertinente du besoin d'informations soient de plus en plus nombreux[1], le processus utilisé en recherche d'informations traditionnelle pour passer de ce besoin à une forme directement exploitable par le système de recherche d'informations reste basique. Il s'appuie exclusivement sur l'utilisateur qui exprime son besoin d'informations en formulant une requête sous forme de mots-clés ou de phrases en langage naturel. La requête peut donc être considérée comme une description partielle du besoin d'informations (les mots-clés étant souvent imprécis et ambigus) à un instant donné. Le caractère évolutif de ce besoin[2] n'est par conséquent pas pris en compte. Les requêtes représentées par un ensemble de mots-clés sont généralement plutôt courtes. Selon le modèle de recherche d'informations utilisé pour la représentation du contenu des requêtes, ces mots-clés sont parfois reliés à l'aide d'opérateurs booléens (*et*, *ou*, *non*...).

Les requêtes formulées en langage naturel offrent quant à elles à l'utilisateur la possibilité d'exprimer plus librement ce qu'il recherche. Elles nécessitent généralement, pour pouvoir être représentées et interprétées par le système de recherche d'informations, de faire appel à des traitements linguistiques.

I.1.1.3. La pertinence

Le cœur du problème de la recherche d'informations réside dans la définition d'une fonction de correspondance entre la requête et l'ensemble des documents disponibles. La pertinence en recherche d'informations peut être vue sous différents angles :

- **du point de vue de l'utilisateur** ; elle représente alors la façon dont ce dernier évalue les documents retrouvés par le système de recherche d'informations en fonction de son besoin d'informations (on parle de ses jugements de pertinence). Il s'agit de la « *pertinence utilisateur* » ;
- **du point de vue du système**, ou la pertinence que le système a lui-même calculé à partir des méthodes utilisées pour comparer les documents et la requête. C'est la « *pertinence système* ».

Rendre compte de manière automatique de la pertinence « *utilisateur* » semble difficile, compte tenu notamment de son caractère fortement subjectif[3], de son aspect nuancé[4], et évolutif[5]. Cette difficulté est renforcée par le fait que les utilisateurs ont souvent également du mal à définir et à exprimer leur besoin d'informations.

[1] Voir notamment les diverses études liées à la personnalisation de l'information ou à la recherche d'informations basée sur le profil de l'utilisateur
[2] Le besoin de l'utilisateur peut évoluer au fur et à mesure que l'utilisateur acquiert des éléments d'information supplémentaires
[3] Un même document peut être jugé pertinent ou non selon les utilisateurs.
[4] Deux documents pertinents n'ont pas nécessairement la même importance, et les raisons de cette importance peuvent varier.
[5] Un document non pertinent à un moment donné peut le devenir au fur et à mesure que l'utilisateur progresse dans sa connaissance du sujet

En pratique, la notion de pertinence est représentée essentiellement au travers de la pertinence « *système* ». Cette dernière s'exprime sous la forme d'un score obtenu automatiquement par les systèmes de recherche d'informations, en comparant les représentations des documents et celles des requêtes suivant les méthodes définies par le modèle de recherche d'informations utilisé. Bien que ce score ne soit qu'une représentation imprécise de la pertinence « *utilisateur* »[1], il présente l'avantage de rendre « *mesurable* » la notion de pertinence et de permettre par conséquent l'évaluation des performances des systèmes de recherche d'informations.

Tout l'enjeu de la recherche d'informations réside dans la mise en œuvre de mécanismes visant à rapprocher la pertinence système de la pertinence utilisateur.

I.2 LES PROCESSUS GENERAUX DE LA RECHERCHE D'INFORMATIONS

Le but d'un système de recherche d'informations est de présenter à l'utilisateur des documents répondant à ses besoins. D'une manière générale, un système de recherche d'informations se modélise par le quadruplet :

SRI=<document, requete, modele, fonction_de_pertinence>

Equation 1 : Modélisation du processus de recherche d'information

La figure suivante illustre le processus de recherche d'informations :

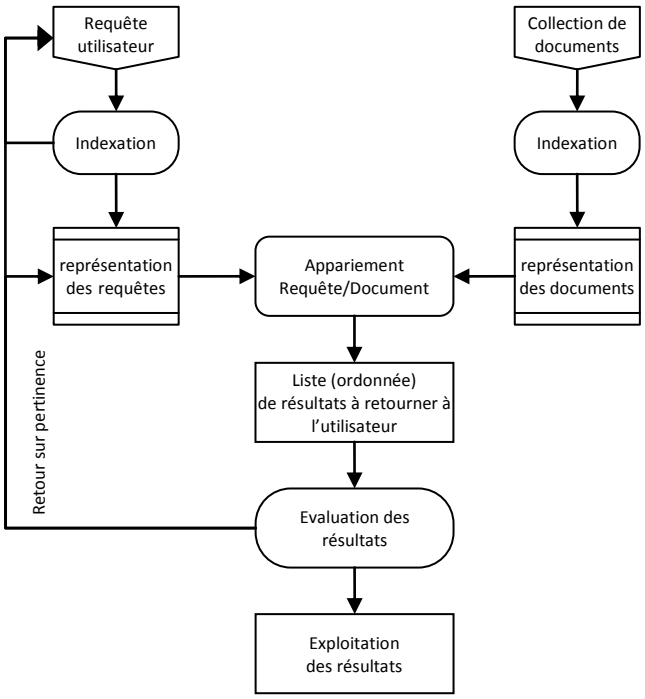


Figure 1 : Architecture générale d'un système de recherche d'informations.

[1] Un document considéré comme pertinent par le système ne l'est pas nécessairement par l'utilisateur

I.2.1 Indexation des documents et des requêtes

L'objectif d'un système de recherche d'informations est de retrouver et restituer une liste de documents pertinents par rapport à la requête utilisateur. Il est nécessaire alors de pouvoir effectuer une recherche sur l'ensemble de la collection de documents.

L'étape d'indexation consiste à analyser les documents et les requêtes pour produire une représentation de leur contenu textuel qui soit exploitable par le système de recherche d'informations. Chaque document (et requête) est alors associé à un descripteur représenté par l'ensemble des termes d'indexation extraits. La section I.2.1.3 détaille le processus d'indexation.

Le terme indexation désigne l'opération par laquelle le système crée une représentation structurée des documents du corpus. Ainsi, seule cette représentation est comparée aux termes de la requête. Cette représentation est pratiquement le seul moyen de retrouver de façon non ambigüe des documents ou un document ou une partie de document où le terme apparaît (CHAMPLAUX, 2009).

Indexer un document, c'est élire ses termes représentatifs afin de générer la liste des termes d'indexation et les ajouter à l'index de la collection avec la liste des références de chaque document le contenant.

I.2.1.1. Modes d'indexation

On identifie deux principaux modes d'indexation :

L'indexation manuelle

Elle repose sur l'intervention d'un documentaliste qui analyse le document pour en identifier le contenu et en construire une représentation. Les entrées d'index sont reliées manuellement aux documents les contenant. La principale critique faite à ce mode d'indexation est son coût. De plus, le documentaliste doit posséder les compétences, du moins minimales, à la compréhension des centres d'intérêts du document. Un autre aspect à considérer est celui de la variabilité de l'index d'un documentaliste à un autre causant ainsi des incohérences dans la base d'index[1].

L'indexation automatique

Elle est basée sur un système capable de générer, sans intervention humaine, les index des documents. Elle présente l'avantage d'un processus régulier et déterministe. En effet le même index est toujours renvoyé pour un même document. Cependant, elle est incapable d'interpréter un document et ne peut s'adapter, automatiquement, aux évolutions du

[1] Cette variation est vraie aussi bien dans le cas ou plusieurs personnes indexent un même document ou dans le cas où une même personne indexe un même document à des moments différents.

vocabulaire. De plus, si le système ne possède aucun moyen de lever les ambiguïtés des termes, les erreurs d'interprétation seront fréquentes et les incohérences de la base de données lourdes.

Cependant, et en pratique, nous avons le plus souvent recours à une combinaison des deux modes pour créer un index : l'indexation semi-automatique. Une première version de l'index est créée automatiquement puis elle est affinée en faisant appel aux experts.

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les fichiers inverses « *inverted files* », les tableaux de suffixes « *suffix arrays* » et les fichiers de signatures « *signature files* ». Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications.

I.2.1.2. Descripteur et langages d'indexation

Les descripteurs

Le terme descripteur désigne l'information atomique d'un index. Un des enjeux de l'indexation est de choisir parmi l'ensemble des mots d'un document, ceux qui sont sensés représenter au mieux l'information contenue dans un document.

On identifie globalement cinq types de descripteurs :

- Les **mots simples** non vides et les **lemmes** ou **racines** qui permettent de dépasser les variations mots dans un texte (formes, conjugaisons, participes, etc.)
- Les **n-grammes** qui sont une représentation originale d'un texte en séquence de N caractères consécutifs.
- Les **groupes de mots** qui sont souvent plus riches sémantiquement que les mots qui les composent pris isolément. Certains considèrent les groupes de mots comme unités de base dans le langage d'indexation.
- Les **contextes** ou termes n'apparaissant pas explicitement dans le texte du document mais ayant un lien sémantique et/ou de cooccurrence avec les mots du document.
- Les **concepts** qui sont des expressions contenant un ou plusieurs mots. Ils peuvent être écrits de manière libre par l'utilisateur ou, ce qui est souvent le cas, choisis dans une liste de concepts (vocabulaire contrôlé : un thésaurus, une ontologie, une hiérarchie de concepts, etc.).

L'ensemble des descripteurs constitue le langage d'indexation. Un langage est habituellement constitué de vocabulaire et de syntaxe. La syntaxe définit les règles permettant la construction, à partir d'association d'éléments du vocabulaire, d'expressions valides dans un langage donné. Aussi, la richesse du vocabulaire et/ou la qualité de la syntaxe permettent

d'optimiser le langage d'indexation.

Nous présentons ici deux types de langage :

Le langage libre

C'est un langage proche du langage naturel. Il n'est fait aucune hypothèse sur le nombre ni le sens des termes du vocabulaire où tous les descripteurs peuvent être choisis pour indexer un document. Il n'est régi par aucune syntaxe définie à priori. Par conséquent, le vocabulaire évolue rapidement sans contrôle ; il peut contenir des termes synonymes, polysémique etc. ce qui induit des incohérences et réduit significativement les performances des moteurs de recherche. De plus, l'indexation des requêtes en est considérablement affectée. Ainsi, un document sera retrouvé pour une requête, du fait que son index contient le descripteur de la requête, alors qu'il ne traite pas du sujet de la requête et inversement, des documents pertinents seront ignorés car la requête ne contenait pas les descripteurs qui y sont associés.

Le langage contrôlé

L'objectif d'un tel langage est d'éviter les polysémies et les synonymies du langage libre. Une liste des termes d'indexation, appelée liste d'autorité, est prédéfinie et ne contient pas de termes polysémiques ni synonymiques. Ainsi, un terme désigne un et un seul sens et inversement, un sens n'est associé qu'à un terme unique. Par conséquent, l'utilisation d'un langage contrôlé devrait permettre de limiter le nombre de représentations possibles du contenu du document, si l'on ne tient pas compte de la subjectivité de l'interprétation des documents et des termes, car un sujet donné ne peut être décrit que par un seul ensemble de termes d'indexation.

Le principal avantage de l'utilisation d'un langage contrôlé associé à un thésaurus ou à une ontologie est la possibilité de composer, automatiquement, des expressions évoluées de recherche associant des descripteurs.

I.2.1.3. Processus d'indexation

Le processus d'indexation a pour rôle de construire un index, et éventuellement un index inversé, reliant les documents trouvés et les termes d'indexation selon le schéma suivant :

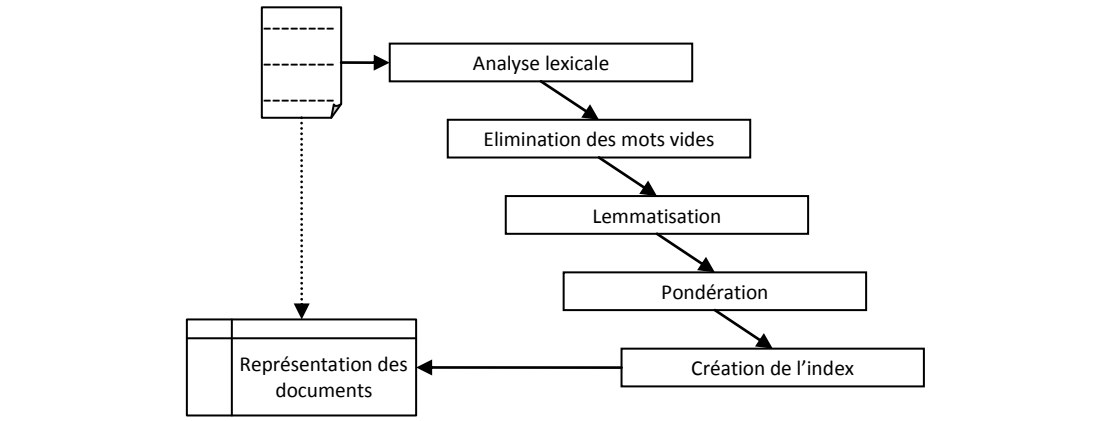


Figure 2 : Etapes du processus d'indexation

L'élimination des mots vides

Un des problèmes majeurs de l'indexation consiste à extraire les termes significatifs et à éviter les mots vides (pronoms personnels, prépositions,...).

Les mots vides peuvent aussi être des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc.). On distingue deux techniques pour éliminer les mots vides :

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

Même si l'élimination des mots vides a l'avantage évident de réduire le nombre de termes d'indexation, elle peut cependant réduire le taux de rappel, c'est à dire la proportion de documents pertinents renvoyés par le système par rapport à l'ensemble des documents pertinents.

La lemmatisation

Un mot donné peut avoir différentes formes dans un texte, mais le sens reste le même ou très similaire. On peut, par exemple citer économie, économiquement, économétrie, économétrique, etc. Il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept véhiculé. Pour résoudre le problème, une substitution des termes par leur racine, ou lemme, est utilisée.

Cette phase de passage à la forme canonique n'est pas obligatoire. Elle présente le principal avantage d'indexer par exemple le mot « *camions* » et le mot « *camion* » de la même façon (« *camion* »), ce qui évite à l'utilisateur de devoir entrer les formes de pluriel des noms ou les formes conjuguées des verbes lors de sa recherche. Cependant, dans certains cas, le passage à la forme canonique supprime la sémantique originale du mot. Par exemple, la forme

conjuguée « *portera* » du verbe « *porter* » sera indexée sous « *porte* », de la même façon que le nom « portes ». Ainsi, lorsque l'utilisateur formulera une requête avec le verbe « *porter* », il aura très certainement, parmi la liste des documents résultats, des documents non pertinents relatifs au nom « *porte* ». Si la lemmatisation a pour but d'augmenter le rappel, la précision en fait souvent les frais.

La pondération

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

Selon Luhn (1958) et Salton (1987) le facteur de signification d'un document est basé sur deux mesures :

- La fréquence d'apparition d'un mot dans un document établit une mesure efficace de la signification des documents.
- La position relative des mots apportant le plus d'informations dans une phrase fournit une bonne mesure de signification.

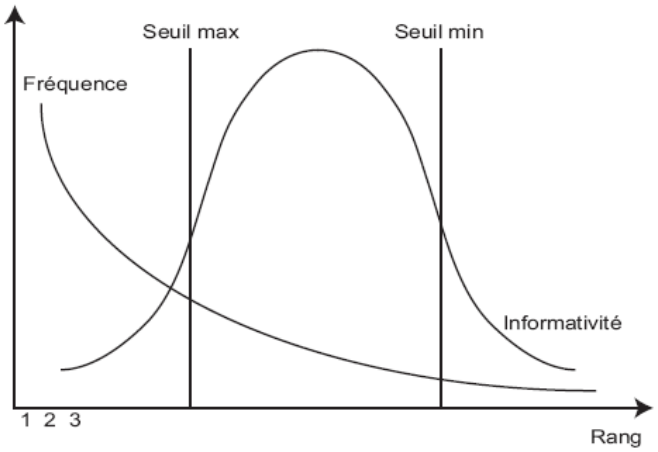


Figure 3 : Rang et fréquence d'apparition d'un mot dans un document.

Salton et Buckley (1988) cités dans (KARBASI, 2007) ont introduit une mesure du profil lexical d'un terme *ti* : *tf * idf*

- **tf** (term frequency): indique l'importance d'un terme dans un document
- **idf** (inverted document frequency) : indique le pouvoir de discrimination de ce terme dans un document du corpus.

$tf = log(f(ti, d)+1)$ f(ti,d) : fréquence d'occurrence du terme ti dans d

$idf = log(1/n)$ n : nombre de documents dans lesquels figure le terme

Equation 2 : pondération des termes : calcul de tf/idf

Illustration des étapes d'indexation

Pour mettre en évidence le processus d'indexation, nous allons construire un index à partir du texte suivant :

Texte original :

« Un SRI a pour fonction de permettre à l'utilisateur d'accéder à des documents qui contribuent à combler son besoin d'information, exprimé sous forme de requête, qui motive sa recherche. Ainsi le système peut être vu par l'utilisateur comme un instrument de prédiction de la pertinence des documents d'un corpus par rapport à sa requête. ».

Après la suppression des mots vides

« sri fonction permettre utilisateur accéder documents contribuent combler besoin information exprimé forme requête motive recherche système utilisateur instrument prédiction pertinence documents corpus rapport requête».

Après la lemmatisation

« sri fonction permet utilisat accé documen contrib combl besoin inform exprim form requête motiv cherche systèm utilisat instrum prédi pertinen document corpus rapport requête ».

Résultat de l'indexation

Le résultat de l'indexation donne un ensemble de termes et leur pondération en application de la règle tf*idf comme suit[1] :

Terme	Occurrence dans d	TF	occurrence dans le corpus	idf	tf*idf
sri	1	0,3010	24	0,6198	0,1866
Accé	1	0,3010	63	0,2007	0,0604
besoin	1	0,3010	85	0,0706	0,0212
cherche	1	0,3010	100	0,0000	0,0000
combl	1	0,3010	12	0,9208	0,2772
contrib	1	0,3010	53	0,2757	0,0830
corpus	1	0,3010	20	0,6990	0,2104
documen	2	0,4771	38	0,4202	0,2005
exprim	1	0,3010	85	0,0706	0,0212
fonction	1	0,3010	48	0,3188	0,0960
form	2	0,4771	72	0,1427	0,0681
instrum	1	0,3010	10	1,0000	0,3010
motiv	1	0,3010	17	0,7696	0,2317
permet	1	0,3010	95	0,0223	0,0067
pertinen	1	0,3010	18	0,7447	0,2242
prédi	1	0,3010	5	1,3010	0,3916
rapport	1	0,3010	43	0,3665	0,1103
requête	2	0,4771	19	0,7212	0,3441
systèm	1	0,3010	28	0,5528	0,1664
utilisat	2	0,4771	32	0,4949	0,2361

Figure 4 : Résultat du processus d'indexation

[1] Les fréquence dans le corpus de 100 documents sont supposées et proposées à titre d'exemple.

I.2.2 La restitution des résultats

Généralement et par abus de langage appelé processus de recherche. Ce processus vise à apparier les documents et la requête de l'utilisateur en comparant leurs descripteurs respectifs. Pour cela, il s'appuie sur un formalisme précis défini par un modèle de recherche d'informations (voir Annexe I :). Les documents présentés en résultat à l'utilisateur, et considérés comme les plus pertinents, sont ceux dont les termes d'indexation sont les plus proches de ceux de la requête.

I.2.2.1. L'appariement document / requête

La comparaison entre le document et la requête revient à calculer un score, supposé représenter la pertinence du document vis-à-vis de la requête. Cette valeur est calculée à partir d'une fonction ou d'une probabilité de similarité notée *rsv(Q, d)* (*Retrieval Status Value*), où *Q* est une requête, et d'un document.

Cette mesure tient compte du poids des termes dans les documents, déterminé en fonction d'analyses statistiques et probabilistes. La fonction d'appariement est très étroitement liée aux opérations d'indexation et de pondération des termes de la requête et des documents du corpus. D'une façon générale, l'appariement document-requête et le modèle d'indexation permettent de caractériser et d'identifier un modèle de recherche d'informations.

La fonction de similarité permet ensuite d'ordonner les documents renvoyés à l'utilisateur. La qualité de cet ordonnancement est primordiale. En effet, l'utilisateur se contente généralement d'examiner les premiers documents renvoyés (les 10 ou 20 premiers). Si les documents recherchés ne sont pas présents dans cette tranche, l'utilisateur considérera le système de recherche d'informations comme mauvais vis-à-vis de sa requête.

I.2.2.2. Le classement en recherche d'informations

Le classement des résultats est un des aspects les plus importants dans les approches pour la recherche d'informations. Il doit permettre à l'utilisateur de retrouver l'information pertinente en priorité.

La pertinence des résultats d'une recherche s'observe lors du classement des pages retournées par les moteurs de recherche. Les techniques de classement ont été largement étudiées. Les approches existantes se répartissent en deux grandes catégories : celle focalisant sur la structure du Web, en particulier les hyperliens reliant les pages entre elles, et celle utilisant le contenu des pages jugées pertinentes.

Dans beaucoup d'approches, le Web est vu comme un graphe dont les sommets sont les pages Web et les arêtes les hyperliens pointant d'une page à l'autre. L'exploitation de cette

structure a donné lieu à de nombreuses métriques pour le classement des résultats d'une recherche parmi lesquelles le *pageRank* et toute une variété d'approches s'inspirant de cet algorithme.

L'algorithme du *pageRank* proposé par (PAGE & BRIN, 1998) mis en œuvre dans *Google* est incontestablement le plus connu. Cet algorithme fonctionne sur le principe de popularité d'un document ; en conséquence plus une page Web est pointée par d'autres pages, plus son score est élevé et la page apparaîtra parmi les plus pertinentes. Cependant, le *PageRank* ne tient pas compte de la sémantique du contenu des pages Web. L'étude menée par Dhyani et al. (2002) et citée dans (PRUSKI, 2009) distingue deux grandes catégories d'approche de classement des résultats pour définir la pertinence et la qualité des pages, à savoir celle exploitant les relations (sémantiques ou physiques) existantes entre les pages et celle considérant les pages indépendamment les unes des autres.

D'autres approches utilisent le modèle booléen pour déterminer l'ensemble des pages pertinentes. Elles exploitent la structure de graphe du Web pour classer les résultats en fonction de la longueur des chemins (nombre minimal d'hyperliens reliant les pages) entre les pages représentant les sommets du graphe.

I.2.2.3. La visualisation des résultats

L'affichage des résultats d'une requête est l'un des problèmes majeurs pour un moteur de recherche. En effet, que les résultats soient pertinents ou non, si l'utilisateur ne comprend pas le plus rapidement possible la manière dont s'organisent les éléments de réponses, ces derniers ne lui seront d'aucune utilité.

Mann (1999) cité dans (PICAROUGNE, 2004) avance que l'apport de la visualisation pour l'utilisateur d'un système d'informations est séparé en trois niveaux distincts. Pour chaque niveau, le but recherché est différent et il faut donc adapter les techniques utilisées pour chacun d'eux :

- Représentation de l'ensemble des résultats,
- Visualisation de la structure d'un site Web, d'une partie d'Internet ou du chemin suivi par l'utilisateur afin de trouver ses résultats,
- Visualisation du document de manière plus compréhensible pour l'utilisateur.

Par ailleurs, toujours selon Mann (1999), il existe principalement quatre facteurs influençant l'utilité relative d'une visualisation donnée. Ces facteurs sont regroupés sous le terme de l'environnement des *4T* :

- **Target user group** : Adaptation au public visé,

- **Type and number of data** : Adaptation au type de données,
- **Task to be done** : Adaptation à la tâche à accomplir,
- **Technical possibilities** : Adaptation aux possibilités techniques de présentation des résultats.

On distingue globalement deux classes de visualisation :

Visualisation à base de texte

Ce mode de visualisation est le plus couramment utilisé ; il présente les résultats sous forme d'une liste de liens accompagnés de quelques lignes de résumé. La liste est triée selon un score de pertinence système.

Une première technique d'amélioration de la visualisation consiste à rajouter pour chaque document retourné une barre de visualisation caractérisant les critères qui ont permis de sélectionner cette page comme étant pertinente à la requête.

L'étude du comportement des utilisateurs a conduit à regrouper les résultats en catégories. La principale problématique de cette approche est de trouver la classification adéquate. Certaines techniques utilisent les similarités sémantiques ou conceptuelles entre les documents ; d'autres ont recours à une hiérarchisation des résultats selon une taxonomie.

Les approches textuelles des résultats, même si elles sont intuitives et faciles à mettre en œuvre, n'offrent pas de grandes opportunités à la visualisation de grandes quantités de résultats. Les études statistiques effectuées par (PRUSKI, 2009) montrent que les utilisateurs ne consultent pas plus des 50 premiers résultats.

Visualisation graphique

La visualisation des résultats sous forme graphique présente l'avantage de faire passer beaucoup d'informations très rapidement à l'utilisateur. Ces représentations peuvent intervenir en deux, trois et même quatre dimensions en prenant en considération la notion de temps. Les travaux dans ce domaine sont nombreux et variés et semblent avoir un avenir prometteur dans le domaine de la recherche d'informations.

I.2.3 La reformulation de requête

Quel que soit le système de recherche utilisé, le résultat d'une recherche n'est efficient que s'il décrit 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'informations. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin (ELAYEB, 2009). 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 nouveaux documents qui vont apparaître comme résultat de la nouvelle recherche. Ces techniques (interactives) peuvent être assistées par l'utilisateur, comme elles peuvent être menées d'une manière automatique.

Il existe essentiellement deux techniques de reformulation des requêtes : la première technique est la plus répandue en recherche d'informations. Il s'agit de la reformulation par réinjection (rétroaction ou propagation) 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. Dans la deuxième technique, il s'agit de l'expansion de requête soit à partir du corpus, soit sur la base d'une ressource terminologique.

I.2.3.1. La propagation de pertinence

La propagation de pertinence (*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 général 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.

I.2.3.2. L'expansion des requêtes sur la base d'un corpus

Une grande famille d'approche pour l'expansion de requêtes s'appuie sur l'étude d'un corpus de documents pour le choix de mots clés à rajouter à la requête initiale. L'analyse du

corpus est faite en utilisant des outils statistiques comme la fréquence de termes, la cooccurrence ou la distribution des termes dans le corpus.

Une des premières approches fonctionne sur l'hypothèse que deux termes apparaissant fréquemment sur les mêmes documents ont un lien sémantique. Cette approche exploite cette observation pour le choix des mots clés. Ces derniers sont sélectionnés suivant leur cooccurrence dans le corpus de documents.

Une autre approche utilise des modèles de Markov sur plusieurs sources de documents afin d'en extraire les mots clés les plus pertinents à rajouter à la requête.

La popularité des réseaux sociaux a inspiré Bender et al. (2008) pour la proposition d'une méthode pour l'expansion de requête où le choix des mots clés est déterminé suivant l'étude des tags et des signets définis par l'utilisateur ayant soumis une requête initiale. Un graphe est construit à partir de ces informations et un algorithme en déduit les mots clés pertinents.

Même si ces approches ont montré des résultats intéressants en termes de précision et de rappel des résultats lors de l'interprétation de la requête étendue, leur efficacité est dépendante de la qualité du corpus de documents à analyser. Un des problèmes majeurs de ce type d'approche basée sur l'exploitation d'un corpus de documents réside, par ailleurs, dans le fait que la sémantique des termes de la requête n'est pas prise en compte.

I.2.3.3. L'utilisation de ressources terminologiques

Une façon de corriger le problème de la sémantique des relations entre les mots clés posé par les approches présentées précédemment est d'utiliser des ressources terminologiques comme les ontologies ou les thésaurus.

L'utilisation du thésaurus *WordNet* pour l'expansion de requête a montré que les termes sélectionnés pour l'expansion donnent de meilleurs résultats lorsque ceux-ci sont liés par une relation lexicale à ceux de la requête initiale. De plus, l'expansion de requête améliore peu les résultats lorsque la requête est longue car celle-ci contient suffisamment d'informations pour être discriminante.

Une façon différente d'utiliser les ontologies dans le processus d'enrichissement des requêtes a été proposée par Stojanovic et al. (2004). L'approche proposée se sert des ontologies pour mesurer une distance sémantique entre la requête posée et les besoins en informations de l'utilisateur déterminés de manière automatique par le système à travers l'analyse du comportement de celui-ci. Un ensemble de requêtes étendues par des termes de l'ontologie sélectionnés en fonction des informations collectées sur l'utilisateur lui est proposé.

Chirita et al. (2007) combinent l'utilisation d'ontologies pour désambiguïser les mots de la requête et les informations sur l'utilisateur pour la sélection des mots clés adéquats.

Les approches à base de ressources terminologiques n'utilisent pas, toutes, la palette des relations ontologiques offerte par ces ressources, seules les relations linguistiques que sont la synonymie et l'hypéronymie sont exploitées. Par ailleurs, la pertinence des résultats retournés suivant une requête étendue par le vocabulaire d'une ontologie dépend de plusieurs facteurs. Le premier d'entre eux est la qualité de l'ontologie. La représentation du domaine par l'ontologie doit être cohérente, à jour, précise et non-ambiguë. Dans le cas contraire, les mots clés extraits et rajoutés à la requête ne donneront pas les résultats escomptés. Ensuite, l'utilisateur doit être familier avec l'ontologie utilisée. Enfin, la taille de l'ontologie peut jouer un rôle important dans l'efficacité et la précision des algorithmes d'extraction car les termes de la requête doivent être mis en relation avec les concepts de l'ontologie.

De manière générale, l'utilisation de techniques d'expansion de requête est destinée à augmenter la précision des résultats retournés au détriment du rappel. En favorisant la précision, des résultats pertinents qui répondent aux besoins des utilisateurs ne lui sont pas retournés. Il est nécessaire pour les applications de trouver un compromis entre précision et rappel pour optimiser la qualité des résultats.

I.2.3.4. 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 le 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 éventuelle 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* ».

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

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

La plupart des approches d'expansion considèrent qu'il vaut mieux choisir des mots qui sont reliés à la requête qu'aux mots individuels de la requête. 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 représentations du

même terme.

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 requête 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.

Une requête reformulée va contenir plus de termes reliés pour repérer de nouveaux documents. Ainsi, ce traitement est souvent vu comme un moyen d'augmenter le taux de rappel. Cependant, nous savons qu'il n'y 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.

I.2.4 Evaluation des Systèmes de Recherche d'Informations

L'évaluation des systèmes de recherches d'informations a toujours constitué un centre d'intérêt important. (CHAMPLAUX, 2009) dresse un bilan complet des méthodes et des outils développés à cet effet. Globalement, la qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses que l'utilisateur espère. Plus les réponses du système correspondent à celles que l'utilisateur espère, meilleur est le système.

La figure suivante illustre la position des différentes partitions de documents les unes par rapport aux autres :

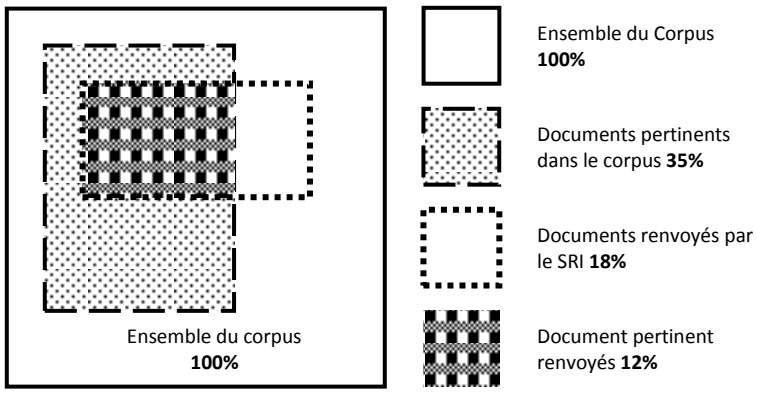


Figure 5 : Représentation des partitions de la collection lors d'une interrogation.

Ainsi, les documents de la collection forment deux partitions selon deux caractéristiques :

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

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

I.2.4.1. Rappel et précision

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

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

Pour reprendre l'exemple de la Figure 5, nous obtiendrons

> Rappel = 12/35 = 0.3542
> Silence = (30-12)/30 = 0.6571
> Précision = 12/18 = 0.6666
> Bruit = (18-12)/18 = 0.3333

La valeur de ces taux est influencée par le processus d'indexation. En effet, plus l'indexation est exhaustive, plus le taux de rappel est potentiellement important : toutes les informations susceptibles d'être pertinentes peuvent être restituées, mais certaines ne seront pas pertinentes pour l'utilisateur. Symétriquement, plus l'indexation est spécialisée, plus le taux de précision est élevé, mais cela induit le risque d'avoir un taux de rappel faible : les informations restituées seront pertinentes, mais d'autres informations pertinentes ne seront pas restituées.

Un système qui aurait *100 %* à la fois pour la précision et pour le rappel signifie qu'il a trouvé tous les documents pertinents et rien que les documents pertinents. En pratique, cette situation n'arrive pas. Selon Jian-Yun Nie (2004), il est possible, le plus souvent, d'obtenir un taux de précision et de rappel aux alentours de 30%, ce qui implique que le problème auquel répond la recherche d'informations est non trivial.

Les deux métriques ne sont pas indépendantes: quand l'une augmente, l'autre diminue. Il est facile d'avoir *100%* de rappel: il suffit de donner tous les documents disponibles comme réponse à chaque requête. Cependant, la précision dans ce cas serait très basse. De même, il est

possible d'augmenter la précision en donnant très peu de documents en réponse, mais le rappel en souffrira.

Il faut donc utiliser les deux métriques conjointement en améliorant la précision sans trop sacrifier le rappel et vice-versa.

I.2.4.2. Mesures alternatives

D'autres mesures permettent l'évaluation des systèmes de recherche d'informations. Une mesure basée sur la notion de *rappel/précision* nommée la *R-précision* est la précision à *n* quand *n* est égal au nombre total de documents pertinents. Cette mesure est plus réaliste pour l'étude de l'ordonnancement en tête de liste, mais pour l'obtenir, il est nécessaire de connaître au préalable le nombre de documents pertinents disponibles dans le corpus pour une requête donnée. Une *R-précision* de *1.0* signifie une précision et un rappel optimaux.

Cependant, d'autres critères peuvent être pris en compte lors de l'évaluation des systèmes de recherche d'informations, par exemple le temps de réponse, l'effort fourni par l'utilisateur, la présentation du résultat, éventuellement la taille de l'index.

D'autres travaux portent sur la notion de l'utilité par rapport à l'utilisateur et se basent sur d'autres critères d'évaluation comme le ratio de couverture qui fait référence aux documents restitués connus et reconnus pertinents par l'utilisateur, le ratio de nouveauté qui fait référence aux documents restitués reconnus pertinents par l'utilisateur et qui lui étaient inconnus jusque là, l'effort de rappel qui correspond au rapport entre le nombre de documents que l'utilisateur souhaitait retrouver et le nombre de documents lus pour les trouver.

I.3 VERS LA RECHERCHE SEMANTIQUE D'INFORMATIONS

Dans ce chapitre, nous avons essayé de passer en revue les nombreuses techniques mises à contribution pour retrouver une information. Après avoir défini les principaux acteurs de la recherche d'informations, nous avons décrit en détail le processus typique de recherche d'informations.

Nous avons décrit les principales techniques d'indexation de documents et de requêtes ; ainsi que les techniques et algorithmes utilisés par les grands moteurs de recherche pour classer et restituer les résultats à l'utilisateur pour qu'il puisse en faire une interprétation efficace.

Une attention toute particulière a été donnée à la reformulation des requêtes. Il a été démontré que, selon le formalisme adopté, des requêtes correctement réécrites ou enrichies peuvent améliorer significativement la qualité d'une recherche.

La prise en compte de la sémantique ne peut plus être ignorée dans aucune phase du processus. Ainsi, le recours à une indexation en langage libre a montré ses limites, le recours à

un vocabulaire est nécessaire. L'usage des ontologies, et à moindre mesure les thésaurus, permet une reformulation convenable des besoins de l'utilisateur, menant inéluctablement à de meilleurs résultats.

Dans le chapitre suivant, nous essayerons d'introduire les concepts du web sémantique (voir Annexe II :) et comment intégrer les ontologies dans le processus de recherche d'informations. Nous nous attarderons particulièrement sur les mesures de similarité conceptuelles.

Chapitre II :
La recherche sémantique d'informations

Définir, c'est entourer d'un mur de mots un terrain vague d'idées
Samuel Butler.

Les *ontologies* en sciences de l'information sont issues de la nécessité de manipulation sémantique de l'information. Il s'agit de mettre en place des représentations permettant la compréhension par la machine du sens de ce qu'elle manipule. Cela permet de connaître les liens qui unissent des concepts. Le but étant bien évidemment que la machine comprenne le sens de l'information qu'elle véhicule.

Bien évidemment, les ontologies développées pour le Web doivent utiliser le même formalisme, ou au moins être formellement compatibles. De plus, les sémantiques qu'elles expriment doivent aussi être compatibles. Si la compatibilité des formalismes peut être assurée par la définition d'un standard de représentation, celle des sémantiques peut difficilement être garantie a priori. Poser des règles sur la construction des ontologies et imposer des méta-ontologies standards pourraient constituer un moyen de limiter les incohérences.

II.1 RECHERCHE D'INFORMATIONS GUIDEE PAR LES ONTOLOGIES

De par sa définition, une ontologie est une conceptualisation d'un domaine d'application ; cette notion permet d'accroître considérablement la qualité des moteurs de recherche d'informations en :

- **réduisant les silences** : dans SRI, les silences sont tous les documents pertinents par rapport à une requête mais qui ne figurent pas dans le jeu des résultats renvoyés. À cet effet, les relations et les axiomes d'une ontologie permettent de rechercher des concepts qui ne sont pas explicitement fournis dans la requête initiale.
- **réduisant les bruits** : les bruits sont tous les documents renvoyés par le moteur de recherche mais qui ne correspondent pas aux attentes de la requêtes exprimée. Le cas le plus flagrant est celui des documents contenant les termes de la requête mais avec un sens différent ou une interprétation hors du domaine d'application. La restriction du champ de recherche aux documents relatifs à un domaine et annotés en tant que tels en utilisant une ontologie, permet de réduire les bruits. Parce qu'elle contient des définitions non ambiguës, une ontologie devrait être assez précise pour fournir une définition unique à des termes, ou pour traiter la synonymie et l'ambiguïté d'une manière satisfaisante, que ce soit dans la représentation d'ontologie ou dans l'annotation de documents (BAZIZ, BOUGHANEM et AUSSENAC-GILLES, 2005), (BAZIZ, 2005).
- **enrichissant les requêtes** : afin de guider l'utilisateur, des étapes peuvent lui être suggérées pour préparer sa requête à une nouvelle formulation avec des termes plus appropriés. L'ontologie permet d'établir une interface qui le guidera. Le parcours de l'ontologie mène à choisir des concepts et à définir une requête composée de concepts choisis et de leur description. Le fait d'exprimer une requête revient alors à une instanciation des concepts de l'ontologie.

II.1.1 Architecture d'un système de recherche guidée par les ontologies

Un Système de recherche d'informations guidée par les ontologies se caractérise par la notion d'espace conceptuel dans lequel les documents et les requêtes sont représentés par opposition à l'espace mots simples qu'on trouve dans les modèles classiques (BAZIZ, 2005).

Un système de recherche d'informations guidée par les ontologies est construit en trois couches. La **couche application** considérée comme l'interface du système permet à l'utilisateur de formuler des requêtes, de sélectionner les documents et de les annoter. La **couche logicielle** fournit les implémentations adoptées pour la recherche sémantique d'informations. A ce niveau, les requêtes sont traitées, l'appariement entre concepts est effectué ainsi que l'indexation proprement dite des documents par rapport aux ontologies. La plus basse est la **couche pérennisation** des données. On y retrouve deux bases de données : celle des ontologies et bases de connaissances et celle des documents annotés. Elle constitue le fond « *documentaire* » du moteur de recherche.

Globalement, le processus d'une recherche sémantique peut être illustré comme suit :

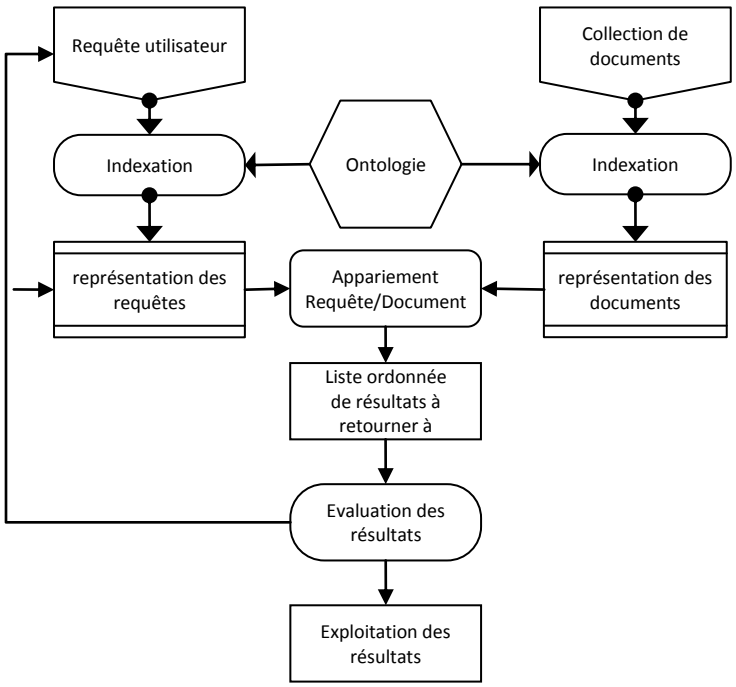


Figure 6 : Schéma synoptique d'une recherché sémantique.

Une fois les concepts extraits, la phase d'appariement se fait en comparant les représentants des concepts extraits des documents avec ceux de la requête. L'avantage qu'offre cette représentation, est que le processus d'appariement peut exploiter les relations entre concepts présents dans l'ontologie pour réaliser un appariement plus élaboré que la simple comparaison de chaînes de caractères. En effet, le rapprochement entre le document et la requête se fait via des concepts similaires mais pas nécessairement « *identiques* ». Ces systèmes gagnent donc en souplesse lors de l'association d'un document à une requête, et ce, à deux niveaux : (i) au moment de la représentation du document et de la requête, (ii) au moment d'apparier les concepts de la requête avec ceux du domaine.

II.1.2 Contraintes liées à la recherche sémantique

Effectuer une recherche sémantique d'informations implique le respect de certaines contraintes principalement liées au degré d'opérationnalisation des ontologies existantes et utilisées pour supporter le processus recherche. On distingue habituellement trois types de contraintes :

II.1.2.1. Présélection des ontologies

La première contrainte consiste à choisir la ou les ontologies du domaine. Cette phase, dite présélection des ontologies, lorsque elle est réalisée avec succès, réduit considérablement le bruit : les informations non directement sollicitées par la recherche ou ayant une faible pertinence. Il existe globalement deux méthodes : la recherche par navigation et la recherche par

exploration.

II.1.2.2. Intégration des ontologies

La deuxième contrainte peut être posée de cette façon : comment travailler avec les ontologies sélectionnées ? Doit-on les fusionner ou les laisser séparées ? Il s'agit de l'épineux problème de l'intégration. Dès lors que plusieurs ontologies peuvent concerner un même domaine, il existe donc une possibilité de différence entre les termes utilisés dans deux ontologies distinctes.

Plusieurs techniques de fusion d'ontologies sont disponibles et utilisées à cet effet, mais elles nécessitent toutes une intervention humaine. Lortal, (2002) et Koli, (2005) dressent un bilan complet de l'intégration des ontologies et distinguent entre l'intégration par assemblage et l'intégration par fusion : *merging/matching*.

Cependant, quel que soit le mode d'intégration choisi, le facteur humain reste significatif bien que le *merging/matching* apparaît comme plus évident. Néanmoins, le problème de classification des concepts reste entièrement posé. L'intervention humaine est jusqu'alors nécessaire.

II.1.2.3. Annotation du corpus

La dernière contrainte est celle de la construction de l'index sémantique qui consiste à décrire au mieux le contenu des documents et seconde la phase de recherche proprement dite qui doit restituer à l'utilisateur les documents les plus pertinents par rapport à sa requête. Ce processus repose sur l'annotation.

Selon (DESMONTILS, JACQUIN, & MORIN, 2002) et (ABROUK, 2006), une annotation est une information graphique ou textuelle attachée à un document et le plus souvent placée dans ce document. Cette place est donnée par un ancrage. Les annotations font référence à des entités diverses : un ensemble de documents, un passage, une phrase, un terme, un mot, une image … Les annotations peuvent pendre plusieurs formes comme des icônes, des symboles de liens, des notes textuelles, des mises en formes, des redécoupages de texte, des images, des sons etc. Une revue complète de l'annotation des documents guidée par les ontologies est présentée dans (ABROUK, 2006).

II.2 INDEXATION CONCEPTUELLE

L'indexation conceptuelle peut être vue comme un cas particulier du processus d'indexation en recherche d'informations (*RI*). L'ingénierie documentaire en donne la définition suivante : «*L'indexation est un processus destiné à représenter par les éléments d'un langage*

documentaire ou naturel des données résultant de l'analyse du contenu d'un document ou d'une question.» Dans son acception informatique, elle consiste à parcourir un corpus, repérer toutes les informations associées et stocker le résultat afin de pouvoir retrouver celles-ci facilement.

Aujourd'hui encore, la plupart des moteurs classiques de recherche d'informations met en œuvre des techniques statistiques appliquées au contenu lexical des textes, auxquelles viennent s'ajouter des critères basés sur le nombre et la popularité des références. Cette solution a l'avantage d'être facilement automatisable, toutefois elle ne permet pas de représenter les informations à un niveau sémantique. Dans le cas d'une ressource aussi ouverte que l'Internet, les moteurs de recherche pêchent surtout par leur manque de précision, directement lié à leur incapacité à lever les ambiguïtés sémantiques des requêtes (Berners-Lee, 1999).

Pour pallier à ces problèmes, le principe de l'indexation sémantique consiste à se placer au niveau conceptuel du document. Les objets manipulés ne se réduisent plus à des chaînes de caractères pondérées mais un lien est fait entre celles-ci et les notions qu'elles désignent. L'index sert alors à stocker la présence de ces notions dans chacun des textes de la base de recherche. Ce passage à un niveau d'abstraction supérieur s'accompagne de certaines contraintes : disposer d'un modèle des connaissances présentes dans l'ensemble des documents et connaître les expressions linguistiques qui leur sont associées. Il est donc nécessaire, pour une indexation sémantique, de manipuler les notions d'ontologie et de terminologie.

II.2.1 L'indexation sémantique ou terminologique

L'indexation sémantique est une approche d'indexation basée sur le sens des mots (BAZIZ, 2005). Elle s'appuie sur des algorithmes de désambiguïsation de mots (*wsd*) pour indexer les documents et les requêtes avec le sens des mots (*mots-sens*) plutôt qu'avec des mots simples. Une manière d'indexer serait, par exemple, d'associer aux mots extraits, des mots du contexte qui aident à déterminer leur sens. D'autres approches de désambiguïsation plus élaborées, utilisent des représentations hiérarchiques pour calculer la distance sémantique ou similarité sémantique entre les mots à comparer.

L'indexation sémantique se base essentiellement sur le processus de « *désambigüisation* » qui permet, dans l'idéal, de ramener tous les mots à leur sens originel. Deux scénarios antagonistes existent (BAZIZ, 2005), (DESMONTILS, JACQUIN, & MORIN, 2002) :

Il faudrait que l'outil de désambiguïsation soit capable de désambiguïser de façon exacte une collection de documents et qu'un *SRI* soit capable de traiter une telle collection, il est facile de croire que l'efficacité d'un tel *SRI* serait améliorée grâce à cette représentation plus sophistiquée.

La désambiguïsation est parfois peu susceptible d'être utile. Un cas usuel est celui où

l'utilisateur utilise dans sa requête plusieurs termes sémantiquement proches. Par conséquent, la désambiguïsation de ce document n'est pas nécessaire étant donné la nature « *auto-désambiguïsante* » de la requête.

II.2.1.1. La méthode de Voorhees

Voorhees (1993) a construit un outil de désambiguïsation basé sur *WordNet*. *WordNet* est un réseau sémantique organisé autour de la notion de *synset*[1]. Un *synset* regroupe des termes (simples ou composés) ayant un même sens dans un contexte donné. Les *synsets* sont liés par différentes relations telles que l'Hyperonymie (*is-a*) et son inverse, l'Hyponymie (*instance-de*).

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

Voorhees a expérimenté cette approche sur une collection de test désambiguïsée (les requêtes de la collection de test sont aussi désambiguïsées manuellement) par rapport aux performances du même processus sur la même collection dans son état d'origine (ambigu).

Les résultats de ces expérimentations ont montré que pour chacune de ces collections, les performances du système de recherche d'informations diminuent sensiblement dans le cas de l'utilisation des collections désambiguïsées. Voorhees n'a pas mesuré la précision de son outil de désambiguïsation, elle a néanmoins conduit ce qu'elle a appelé une « *évaluation subjective* » de la désambiguïsation. Sa conclusion sur cette évaluation est que l'étiquetage avec les sens tel qu'il est réalisé n'est pas exact, ce qui est la cause la plus probable, selon elle, de la dégradation des performances. De plus, pour un certain nombre de requêtes courtes, elle a trouvé que l'outil de désambiguïsation était incapable de déterminer de façon exacte le sens attendu des mots dans les requêtes.

II.2.1.2. La méthode de Krovetz & Croft

Krovetz et Croft (1992) ont conduit une vaste étude sur certaines hypothèses ayant trait à la pertinence de la relation de correspondance du sens des mots dans la requête et les documents. En utilisant les collections de test *CACM* et *Time*, ils ont examiné les dix premiers documents restitués pour chaque requête. Leur but était de trouver l'existence d'une relation entre, d'une part, la **correspondance/non-correspondance** des sens et la **pertinence/non-pertinence** des documents restitués, d'autre part. Leurs résultats ont montré que la relation de non-correspondance de sens a tendance à apparaître de façon significative dans les documents non pertinents. De plus, ils ont remarqué que la non-correspondance de sens est plus rare dans

[1] Synset : Synonyme set (ensemble de synonymes)

les documents classés en premier. Krovetz et Croft expliquent ceci par deux raisons : **L'effet de la collocation** des mots des requêtes qui fait qu'ils contribuent ensemble à exprimer un même sens, même si ils sont isolément ambigus. Et **la distribution non uniforme** du sens des mots dans les documents. Krovetz et Croft ont montré que les différents sens d'un mot n'occurrent pas dans une même proportion et que souvent un sens d'un mot (le sens dominant) occurre plus fréquemment que les autres sens.

Globalement, les travaux réalisés dans le cadre de l'utilisation du sens en recherche d'informations ont rapporté des résultats mitigés. Mihalcea et Moldovan (2000), ont observé une amélioration de 16% dans le rappel et de 4% dans la précision quand ils ont utilisé une combinaison de l'indexation basée sur les mots clés et de l'indexation basée sur les *synsets* de *WordNet*. Par contre, pour Voorhees (1993) et Katz et al. (1998), l'utilisation du sens des mots avec un désambiguïseur automatique (imparfait) n'améliore pas sensiblement les résultats de la recherche.

II.2.2 L'indexation conceptuelle

II.2.2.1. Le modèle OntoSeek

Le système *OntoSeek* de Guarino et al (1999) est l'une des premières tentatives expérimentales sur l'application des ontologies à la recherche d'informations ; il est considéré comme une excellente référence dans ce domaine. Il est conçu pour traiter les pages jaunes et les catalogues de produits (en ligne). Le contenu des pages ainsi que les requêtes sont modélisés par un formalisme basique de graphes conceptuels. *OntoSeek* combine un appariement basé sur le contenu et guidé par ontologie (*ontology-driven content-matching*).

L'idée principale derrière ce projet est que contrairement aux autres systèmes, l'utilisateur n'est pas censé être familier avec le vocabulaire utilisé pour le codage des composantes des documents à chercher. Le système se base sur une large ontologie linguistique, *Sensus*, contenant 50000 nœuds issus de la fusion de l'ontologie linguistique *WordNet* et de l'ontologie *Penman*.

Les objectifs assignés au système *OntoSeek*, sont (BAZIZ , 2005) :

- Une utilisation de termes « *arbitraires* » du langage naturel pour décrire le contenu des documents,
- Une flexibilité terminologique pour formuler les requêtes,
- Une assistance à la formulation, la généralisation ou la spécialisation des requêtes,
- Une efficacité raisonnable avec des volumes de données importants,
- Une grande portabilité et extensibilité.

Le fonctionnement *d'OntoSeek* peut être schématisé, comme dans la figure suivante :

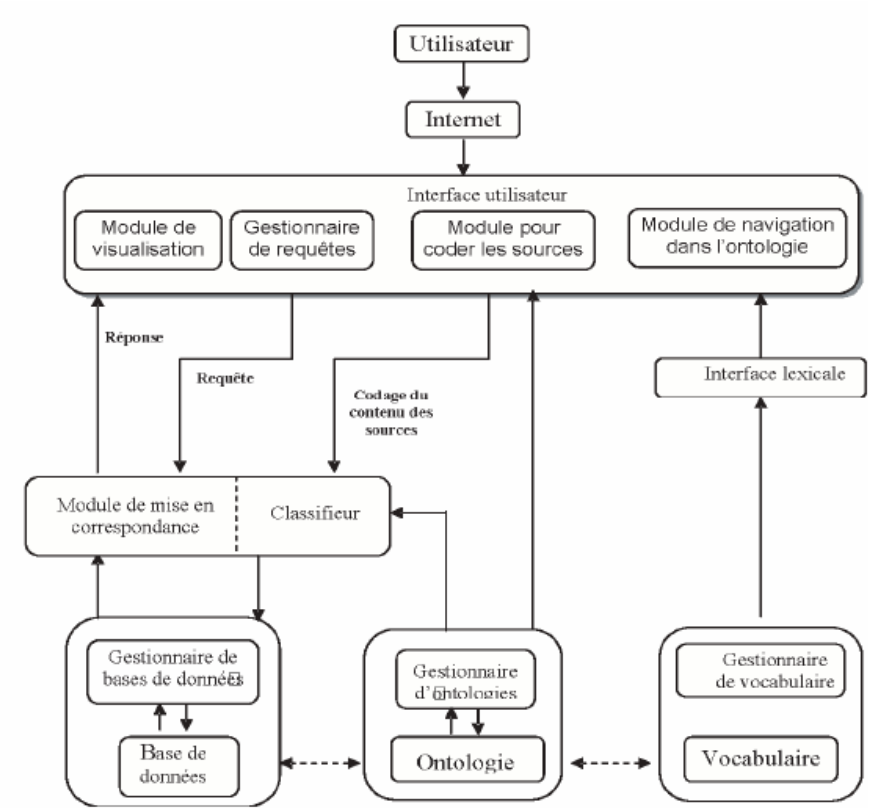


Figure 7 : Architecture fonctionnelle du système OntoSeek (BAZIZ, 2005)

De manière générale, dans le système *OntoSeek*, des gestionnaires d'ontologies et des moteurs de recherche coopèrent dans le processus de recherche d'informations constituant plus globalement des fournisseurs d'informations appelés «*Brokers*». L'objectif de ces Brokers est d'améliorer le degré d'interactivité du système avec l'utilisateur.

Notons que dans ce système, le processus de désambiguïsation se fait de manière interactive.

II.2.2.2. Le modèle DocCore

Dans ce modèle présenté dans (BAZIZ, 2005), la représentation du document, appelée noyau sémantique, est une représentation analogue à celle du réseau sémantique à la différence que les arcs ne sont pas étiquetés, mais quantifiés par une valeur réelle dénotant la proximité sémantique entre les deux concepts. Cette valeur de proximité sémantique est calculée en utilisant des mesures de similarité sémantique[1].

Le choix des valeurs au détriment des étiquettes permet la désambiguïsation des termes du document, ainsi que l'assignation d'un « *poids sémantique* » aux différents nœuds du réseau.

[1] Le sens de l'arc est bidirectionnel puisque la relation de proximité sémantique entre les deux valeurs est symétrique. Si un concept C1 est proche sémantiquement d'un concept C2, alors l'inverse est valable aussi

L'objectif du modèle est de représenter le contenu sémantique de documents. L'approche consiste à projeter les documents sur une ontologie linguistique générale, telle que *WordNet*. Il s'agit d'identifier pour chaque document les représentants de concepts de l'ontologie. Ces derniers peuvent être des mots simples ou des groupes de mots. Un critère de cooccurrence (*cf.idf*) est utilisé pour extraire les concepts importants. Un deuxième critère qui est la similarité sémantique entre concepts, permet de les désambiguïser via le réseau sémantique de l'ontologie. Le résultat de cet appariement entre le document et l'ontologie est un ensemble de concepts (appelés aussi nœuds) avec des liens pondérés entre eux, formant le noyau sémantique de document qui représente au mieux le contenu sémantique du document (BAZIZ, 2005).

L'approche proposée comprend deux étapes principales : la première sert à extraire des documents les concepts les plus fréquents qui correspondent à des entrées dans l'ontologie. Dans la seconde, l'ontologie est utilisée pour récupérer les différents sens possibles pour les concepts retenus dans un premier temps, puis pour calculer les similarités entre les différents sens de ces concepts en utilisant des mesures de proximité sémantique connues dans la littérature.

Un réseau sémantique est alors construit en calculant un score pour chaque terme du document. Le score d'un concept candidat est obtenu en faisant la somme des valeurs de similarité qu'il a avec les autres concepts candidats. Ceci permet de désambiguïser les termes compte tenu du contexte du document. Les concepts candidats ayant les plus grands scores sont alors sélectionnés pour représenter les nœuds du noyau sémantique. Les liens (*arcs*) entre ces différents nœuds sont matérialisés alors par les valeurs de similarité sémantique préalablement calculées.

L'évaluation du modèle *DocCore* montre que l'indexation conceptuelle utilisée seule ne permet pas d'augmenter les performances du système, mais une fois combinée aux mots clés issus de l'indexation classique, elle améliore les résultats sur tous les points de précision.

Cependant, les tests montrent qu'avec un taux de couverture des concepts encore plus important, on n'aurait pas besoin de combiner l'indexation classique à la conceptuelle.

II.3 SIMILARITES ENTRE CONCEPTS DANS UNE ONTOLOGIE

L'évaluation du lien sémantique entre deux concepts dans une ontologie est un problème de longue date dans le domaine de l'intelligence artificielle et de la psychologie. Selon Resnik (1999), la similarité sémantique est une évaluation du lien sémantique entre deux concepts dont le but est d'estimer le degré par lequel les concepts sont proches dans leur sens. La similarité entre deux concepts est liée aux caractéristiques qu'ils ont en commun et à leurs différences. La similarité maximale est obtenue lorsque deux concepts sont identiques.

Deux grandes familles de mesure de similarité dans une ontologie existent : la première considère que la similarité peut être évaluée uniquement à partir des liens taxonomiques (ou lien « *est un* »), la seconde, au contraire, estiment que ce calcul doit intégrer les autres types de liens.

Dans cette section, nous développerons ces deux familles de calcul de similarité ainsi qu'une évaluation des différentes approches.

L'exemple de hiérarchie de concepts présenté dans la Figure 8 est utilisé pour illustrer les différentes mesures. Les concepts sont représentés par des rectangles et les flèches symbolisent la relation « *est un* ».

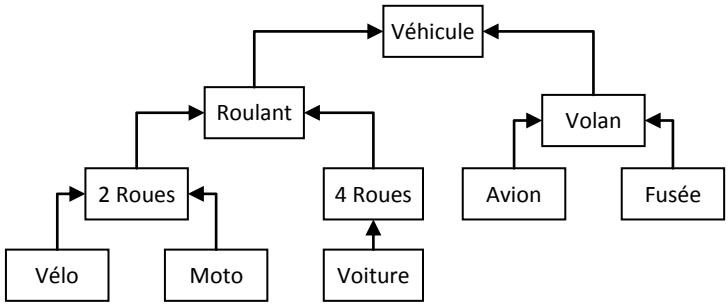


Figure 8 : Exemple d'une taxonomie de véhicules

II.3.1 Extension des requêtes via une ontologie

Les ontologies peuvent être impliquées pour la modification de requêtes dans un système de recherche d'informations. Dans ce cas, leur utilisation peut se situer à l'extérieur (en amont) du système de recherche d'informations. L'objectif est d'améliorer le rendement des requêtes en élargissant leur champ de recherche. La démarche est d'abord de détecter les termes de la requête qui renvoient à des concepts de l'ontologie, puis de les étendre par des termes représentant d'autres concepts proches de ceux de la requête. Ces termes sont identifiés grâce aux liens sémantiques entre concepts qu'offre l'ontologie. Un premier problème à résoudre est la désambiguïsation des termes de la requête. Il s'agit d'arriver à assigner les termes de la requête de manière univoque aux concepts de l'ontologie.

Les ontologies peuvent intervenir dans le processus de modification ou de reformulation de requêtes en recherche d'informations. L'idée, comme représentée dans la Figure 9, est de faire passer la requête utilisateur par le réseau conceptuel d'une ontologie pour l'enrichir avec de nouveaux termes issus du vocabulaire de cette ontologie.

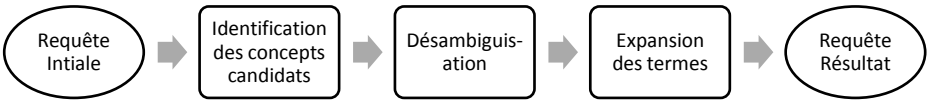


Figure 9 : Cheminement d'une requête via l'ontologie.

L'intérêt est double :

- permettre à l'utilisateur de faire usage d'une terminologie autre que celle présente dans les documents, donc de formuler ses requêtes dans son propre langage. C'est l'ontologie qui assure la correspondance entre les termes de l'utilisateur et ceux des documents.
- permettre au système de recherche d'informations d'aider l'utilisateur à reformuler ses requêtes sur la base d'une proximité sémantique (processus automatique).

Globalement, l'approche de modification de requête propose, étant donné une requête initiale, de la modifier en ajoutant des termes extraits de l'ontologie. Les termes ainsi rajoutés représentent des concepts liés à ceux de la requête initiale via différentes relations sémantiques.

II.3.1.1. Identification des concepts candidats

Pour réaliser ces modifications à la requête, il faut d'abord identifier les termes de la requête susceptibles de représenter des concepts dans l'ontologie. Pour ce faire, avant même d'élaguer les mots vides du document, il faut détecter les termes formés par des mots uniques ou des groupes de mots. Ces termes peuvent renvoyer à différents concepts et donc correspondre à différentes entrées dans l'ontologie. Les termes ainsi identifiés peuvent être des groupes nominaux ou des entités.

La principale méthode utilisée consiste à concaténer des mots adjacents dans le texte et voir s'ils correspondent à une (des) entrée(s) dans l'ontologie. Deux cas peuvent alors être distingués :

- **Projeter l'ontologie sur le document** : Il s'agit dans ce cas, de mettre l'ontologie à plat puis identifier les termes qui occurrent dans le document. Cette méthode a l'avantage d'être rapide et permet la construction d'une ressource réutilisable même si le corpus change. Son inconvénient est le risque d'omettre des concepts qui apparaissent dans le texte du document et dans l'ontologie sous des formes différentes.
- **Projeter le document sur l'ontologie** : Pour chaque concept candidat formé en combinant les mots adjacents dans le texte, on interroge d'abord l'ontologie en utilisant les mots tels qu'ils se présentent dans le texte ; s'ils ne correspondent pas à des entrées dans l'ontologie, on utilise leurs formes de base. Ceci permet de résoudre partiellement les problèmes liés aux variations de la morphologie des mots.

Concernant la combinaison des mots, le terme le plus long qui correspond à un concept est retenu. On peut noter ici l'intérêt d'utiliser une ontologie dont l'espace conceptuel possède une terminologie suffisamment vaste pour couvrir celle des documents de la collection.

De l'intérêt des termes composés

On peut s'interroger de l'intérêt d'extraire les termes composés ou multi mots dans le texte des documents, du moment qu'à l'issue d'une indexation classique par mots simple, on retrouve tous les mots y compris ceux formant des termes composés. En fait, cette démarche est importante dans la mesure où elle permet de réduire l'ambiguïté lors de l'affectation d'un terme à un concept de l'ontologie. En effet, les termes composés sont en général monosémiques (ils n'ont qu'un seul sens) même si les mots qui les composent peuvent être ambigus.

Pondération des termes

Une fois les termes extraits du document, il s'agit de leur affecter un poids qui détermine leur importance dans le document. Dans les systèmes classiques, plusieurs méthodes de pondération qui sont en général des variantes de *tf.idf*, sont utilisées.

II.3.1.2. Désambiguïsation

Pour cela, nous partons de l'idée que les mots-clés qui occurrent ensemble dans un même contexte (la requête), déterminent des concepts appropriés pour désigner ensemble un autre concept. Autrement dit, même si chaque mot-clé dans une requête est individuellement plusieurs fois ambigu, ces mots pris ensemble, contribuent à exprimer une même idée (sens). Dans notre cas, la désambiguïsation est contextuelle, c'est-à-dire, pour affecter un nœud à un terme de la requête, on prend en compte tous les autres termes de la requête. Une fois le nœud approprié sélectionné, on peut rajouter à la requête les autres termes (synonymes) de ce nœud. Cette opération est également effectuée quand on veut sélectionner le meilleur nœud retourné par une relation sémantique donnée, pour le même terme. Ainsi, les processus de désambiguïsation et d'expansion sont indissociables dans ce cas.

Pour réaliser une modification optimale à la requête, des questions doivent être élucidées : quel est l'apport de chaque relation sémantique prise séparément dans l'amélioration des performances de la recherche? Comment pondérer les termes rajoutés? Jusqu'à quel point, quel niveau de la hiérarchie, peut-on étendre?

Les formalismes de désambiguïsation-expansion

Soit une relation R_i appartenant à l'ensemble des relations $R = \{R_1, R_2, .., R_n\}$ de l'ontologie et un terme T_k de la requête Q_c.

Premièrement, chaque terme de la requête est étendu en utilisant les différentes relations sémantiques. Le résultat de cette expansion est un ensemble de concepts candidats C_k reliés au terme : T_k.

$$R_i(T_k) = \{C_k^1, C_k^2, .., C_k^t\}$$

Equation 3: concepts candidats à l'expansion

Une première expansion, que l'on qualifie d'« aveugle » consisterait alors à ajouter à la requête initiale tous les concepts possibles pour toutes les relations :

$$R^n(Q) = \{R_i(T_k)\}_{i,j}$$

Equation 4 : expansion « aveugle » de la requête.

Une autre manière d'étendre serait de n'utiliser que les « *meilleurs concepts* » de R^n pour étendre la requête. C'est une expansion « *modérée* » où les meilleurs concepts seraient ceux ayant un taux de recouvrement assez important avec la requête initiale. Cette désambiguïsation contextuelle permet de sélectionner les concepts en prenant en compte tous les termes de la requête. La sélection dans ce cas se fait comme suit :

$$Best\big(R^n(Q)\big) = ArgMax_k^i \|\{R_i(T_k) \simeq\} \cap Q\|$$

Equation 5 : expansion « *modérée* » de la requête.

Cette façon d'étendre traduit l'hypothèse qu'une requête, même composée de plusieurs termes, exprime un seul concept.

La troisième possibilité d'expansion, qualifiée de « *prudente* » est de sélectionner pour chaque relation, donc pour chaque, $R_i(T_k)$ le meilleur concept :

$$Best\big(R_i(C_k)\big) = ArgMax_k^i \|\{C_k^1, C_k^2, .., C_k^t\} \cap Q\|$$

Equation 6 : expansion « *prudente* » de la requête.

Dans ce cas, étant donné une relation R_i, le nombre de concepts à rajouter est égal au nombre de termes dans la requête. Cette façon d'étendre traduit l'hypothèse qu'une même requête peut faire allusion à plusieurs concepts différents.

II.3.2 Mesures reposant sur la distance

Les mesures reposant sur la distance considèrent que la similarité entre deux concepts peut être calculée à partir du nombre de liens qui séparent les deux concepts. Plusieurs variantes existent en fonction du chemin pris en compte pour calculer la distance entre les concepts.

II.3.2.1. Mesure de RADA

La mesure du *Edge Counting* introduite par Rada (1989) (HERNANDEZ, 2006) évalue la distance sémantique à partir du nombre de branches séparant les concepts par le plus court chemin dans la hiérarchie.

A partir de l'exemple de la Figure 8, la distance se calcule de la façon suivante :

$Dist_{edge}$(Moto, Vélo)=2
$Dist_{edge}$ (Volant, Avion)=1
$Dist_{edge}$ (2 Roues, Roulant)=1
$Dist_{edge}$ (Vélo, Volant)=4
$Dist_{edge}$ (Vélo, Voiture)=4

La similarité sémantique entre deux concepts correspond à l'inverse de la distance entre deux concepts. Plus deux concepts sont distants, moins ils sont similaires. A partir de la mesure de distance précédente, Leacock (1998) a proposé la formule suivante pour calculer la similarité. Elle est issue de la proposition de Resnik (1998) et assure la normalisation de cette dernière.

$$Sim_{edge}(c_1, c_2) = -log \frac{Dist_{edge}(c_1,c_2)}{2*Max}$$

Equation 7 : mesure « *Edge* » de similarité entre concepts.

où max est la profondeur maximale de la taxonomie.

L'utilisation de la fonction *–log* permet de normaliser la similarité entre 0,1) (1 signifiant que les concepts sont totalement similaires).

$Dist_{edge}$(Moto, Vélo)=0.6
$Dist_{edge}$ (Volant, Avion)=0.9
$Dist_{edge}$ (2 Roues, Roulant)=0.9
$Dist_{edge}$ (Vélo, Volant)=0.3
$Dist_{edge}$ (Vélo, Voiture)=0.3

II.3.2.2. Mesure de Wu-Palmer

Wu et Palmer (1994) ont proposé une autre mesure de similarité prenant en compte à la fois la profondeur des concepts dans la hiérarchie de concepts et la structure de la hiérarchie de concepts.

Pour calculer la similarité entre deux concepts c_1 et c_2, la formule suivante est utilisée :

$$Sim_{WP}(c_1, c_2) = \frac{2 * depth(c)}{depth(c_1) + depth(c_2)}$$

Equation 8 : Mesure de similarité de « Wu-Palmer »

où depth(ci) correspond au niveau de profondeur du concept c_i et c représente le concept le plus spécifique qui généralise c_1 et c_2.

La valeur de la similarité est comprise entre 0 et 1 (1 signifiant que les concepts sont totalement similaires).

$Dist_{WP}$ *(Moto, Vélo)=2*3/(4+4) = 0.75*
$Dist_{WP}$ *(Volant, Avion)=0.8*
$Dist_{WP}$ *(2 Roues, Roulant)=0.8*
$Dist_{WP}$ *(Vélo, Volant)=0.33*
$Dist_{WP}$ *(Vélo, Voiture)=0.50*

Cette mesure est plus pertinente que les mesures précédentes reposant uniquement sur le chemin le plus court entre les deux concepts, car elle prend en compte l'organisation hiérarchique des concepts, c'est-à-dire le concept généralisant les deux concepts considérés. Cependant, cette mesure admet que la distance sémantique entre deux concepts reliés par la relation « *est un* » est égale, alors que ce n'est pas forcément le cas. Les choix arbitraires pris lors de la construction de la hiérarchie de concepts influencent grandement la valeur de la similarité.

Afin de signifier que les liens dans une ontologie ne représentent pas la même distance sémantique, plusieurs solutions ont été proposées. L'une d'elles consiste à prendre en compte la densité locale de l'ontologie au niveau des concepts considérés pour contrecarrer la subjectivité dans le choix des relations. L'utilisation de la densité repose sur l'observation suivant laquelle les concepts appartenant à une partie dense en concepts de l'ontologie sont sémantiquement plus proches que ceux appartenant à une partie éparse de l'ontologie McHale (1998) (HERNANDEZ, 2006)

Une autre solution consiste à prendre en compte le contenu en informations des concepts.

II.3.3 Mesures reposant sur le contenu en informations des concepts

Les approches reposant sur le contenu en informations supposent que l'information détenue par les concepts puisse être quantifiée. La similarité est alors calculée à partir de l'information partagée par les concepts. Les différentes mesures proposées pour évaluer la similarité entre concepts reposent sur ce calcul.

II.3.3.1. Calcul du contenu en informations (CI) d'un concept

Afin de calculer l'information contenue par les concepts, un corpus de référence est choisi. Les concepts sont alors pondérés par une fonction correspondant à l'information portée par un concept dans le corpus. Le contenu en informations du concept est défini par ses occurrences dans le corpus ainsi que celles des concepts qu'il subsume. Il a pour objectif d'utiliser la probabilité d'obtenir un concept dans un corpus documentaire afin de contrecarrer la subjectivité dans le choix des relations « *est un* » de l'ontologie.

Le contenu en informations d'un concept c se calcule de la façon suivante Resnik (1995) :

$$\begin{aligned} &CI = -log\big(p(c)\big), \\ &Freq(c) = \sum_{n \in word\ (c)} count(c) \\ &p(c) = \frac{freq(c)}{N} \end{aligned}$$

Equation 9 : calcul du contenu informationnel (*CI*) d'un concept.

où *word(c)* est l'ensemble des termes représentant le concept *c* et ses subsumés, *count(n)* le nombre d'occurrences du terme *n* dans le corpus et *N* le nombre total d'occurrences des labels de concepts retrouvés dans le corpus

L'utilisation de la fonction *–log* permet de réduire le contenu en informations d'un concept ayant une forte probabilité d'apparition dans le corpus.

Un inconvénient du calcul du contenu en informations d'un concept proposé par Resnik est que la probabilité d'obtenir un concept est calculée à partir des labels retrouvés dans le corpus sans vérifier que chacune des occurrences se rapporte effectivement au concept. Ceci peut poser problème dans le cas où le label est ambigu et désigne différents concepts. Le contenu en informations d'un concept peut ainsi être amplifié par la forte occurrence d'un label désignant un autre concept dans le corpus.

Une solution plus adaptée serait de désambigüiser le label dans les documents et de comptabiliser uniquement les occurrences qui se rapportent au concept en question. Cette solution ne peut pas être mise en place dans la mesure où Resnik (1999) utilise ces mesures justement pour désambiguïser les termes. Afin de prendre en compte les labels ambigus, une autre formule a été proposée par Anderson (1995). Dans cette formule (*freq2*), la fréquence d'apparition d'un label est normalisée par rapport au nombre de concepts auxquels il se rapporte.

$$Freq(c) = \sum_{n \in word\ (c)} \frac{count(n)}{nbclass(n)}$$

Equation 10 : fréquence d'apparition d'un terme.

où *nbclasse(n)* correspond au nombre de concepts dont le terme *n* est label

II.3.3.2. Interprétation du contenu en informations

Plusieurs mesures de similarité ont été définies à partir du contenu en informations des concepts Resnik (1995), Lin (1998) ou Jiang (1997). Elles reposent sur le principe que l'information commune aux deux concepts est capturée dans le contenu en informations du concept le plus spécifique qui subsume les deux concepts.

Afin d'illustrer l'intérêt de la prise en compte du contenu en informations des concepts, la Figure 10 reprend la hiérarchie de concepts présentée dans la Figure 8 pour laquelle le contenu en informations de chaque concept est précisé.

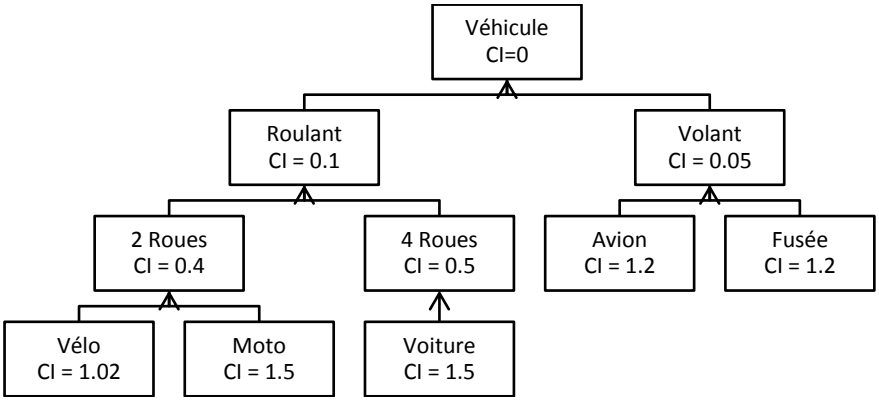


Figure 10 : Hiérarchie de concepts augmentée par le contenu en information (CI).

En prenant par exemple les concepts « *Roulant* » et « *2 Roues* », l'information partagée par ces deux concepts est le contenu en informations du concept *Roulant*, c'est-à-dire *0,1*. L'information commune aux concepts *Volant* et *Avion* est *0,05*. Ainsi l'information partagée par deux concepts permet de capturer le poids sémantique du lien « *est un* ».

Afin de pouvoir calculer la similarité entre deux concepts, la probabilité d'obtenir dans le corpus le concept le plus spécifique subsumant c_1 et c_2 est définie de la façon suivante :

$$Pms(c_1,c_2) = min_{c \in S(c_1,c_2)}\{p(c)\}$$

Equation 11 : probabilité d'obtention du plus spécifique subsumant.

où $S(c_1,c_2)$ est l'ensemble de concepts qui subsument à la fois c_1 et c_2.

La probabilité d'obtenir un concept prend en considération à la fois la probabilité d'obtenir un concept ainsi que tous les concepts qu'il subsume. *pms* (ou *min{p(c)}*) revient donc à choisir le concept subsumant c_1 et c_2 ayant la plus faible probabilité, c'est-à-dire le concept le plus spécifique de c_1 et c_2 dans la hiérarchie au sens où son contenu en informations est le plus faible. Quand l'ontologie permet l'héritage multiple et que plusieurs concepts subsument les deux concepts considérés, cette formule permet de choisir dans l'ensemble des concepts candidats le concept le plus spécifique au sens où c'est celui qui a la probabilité la plus faible.

II.3.3.3. Mesure de Resnik

La première mesure de similarité calculée à partir du contenu en informations des concepts a été proposée dans Resnik (1995). Elle prend des valeurs entre *0* et *1*.

$$Sim_{Resnik}(c_1,c_2) = -log(Pms(c_1,c_2))$$

Equation 12 : mesure de « *Resnik* »

Pour reprendre l'exemple précédent :

SimResnik(Moto, Vélo)=CI(2 Roues) = 0.4
SimResnik (Volant, Avion)=0.05
SimResnik (2 Roues, Roulant)=0.1
SimResnik (Vélo, Volant)=0
SimResnik (Vélo, Voiture)=0.1

L'inconvénient de cette mesure est que deux couples de concepts qui ont le même subsumeur le plus spécifique ont la même similarité. Ceci est par exemple le cas entre (*Vélo* et *voiture)* et (« *2 Roues* » et *Roulant*).

II.3.3.4. Mesure de Lin

La mesure proposée par Lin vise à contrecarrer cet inconvénient Lin (1998). Dans le cas de l'évaluation de la similarité entre deux concepts dans une taxonomie, Lin étend ces mesures en considérant que la description de chacun de ces concepts est le contenu en informations des concepts et que les caractéristiques communes aux deux concepts sont quantifiées par le contenu en informations du concept le plus spécifique généralisant les deux concepts.

$$Sim_{Lin}(c_1, c_2) = \frac{2 * log(Pms(c_1, c_2))}{log(P(c_1) + log(P(c_2)}$$

Equation 13 : Mesure de « *Lin* »

Cette mesure a l'avantage de prendre en compte le concept le plus spécifique subsumant c_1 et c_2 ainsi que le contenu en informations des concepts comparés. Contrairement à la mesure précédente proposée par Resnik, elle permet de différencier la similarité entre plusieurs couples de concepts ayant le même *subsumeur* le plus spécifique. Elle prend des valeurs entre *0* et *1*.

Pour l'exemple précédent, on obtient :

SimLin (Volant, Avion)=0. 8
Sim_{Lin} (2 Roues, Roulant)=0.4
Sim_{Lin} (Vélo, Volant)=0
Sim_{Lin} (Vélo, Voiture)=0.09

II.3.3.5. Mesure de Jiang

Selon JIANG et al (1997) les liens les plus courts entre deux concepts sont pondérés à partir de quatre facteurs :

Le contenu en informations du lien

Calculé à partir de la différence du contenu en information du concept fils c_{fils} et celui du père c_{pere} :

$$CI(c_{fils}, c_{pere}) = CI(c_{fils}) - CI(c_{pere})$$

Equation 14 : calcul du CI d'un concept fils par rapport au père

La profondeur du père

Qui est relative à la profondeur maximale et est évalué comme suit :

$$Prof(c_{pere}) = \frac{profondeur(c_{pere}) + 1}{profondeur\_max}$$

Equation 15 : profondeur relative du concept parent.

La densité locale du concept père

Qui est calculée à partir du nombre de nœuds fils du concept $E(c_{pere})$ et le nombre moyen $\bar{E}$ du nœud fils pour un concept :

$$Densite(c) = \frac{\bar{E}}{E_c}$$

Equation 16 : densité locale d'un concept père.

Le poids du lien reliant les concepts père et fils

$$Poids(c_{fils}, c_{pere}) = \left(\beta + (1-\beta) \times Densite(c_{pere})\right) \times \left(Prof(c_{pere})\right)^{\alpha} \times CI(c_{fils}, c_{pere})$$

Equation 17 : poids du lien reliant les concepts père et fils.

Donc, la distance entre deux concepts c_1 et c_2 est alors calculée par :

$$Dist_{Jiang}(c_1, c_2) = \sum_{c \in pcc(c_1,c_2)} Poids(c, pere(c))$$

Equation 18 : distance de « Jiang ».

où pcc(c1,c2) représente l'ensemble des concepts appartenant au plus court chemin entre c1 et c2.

Ainsi, Jiang propose de convertir la distance en similarité en adoptant la formule de Resnik :

$$Sim_{Jiang}(c_1, c_2) = (2 \times max) - Dist_{Jiang}(c_1, c_2)$$

Equation 19 : Mesure de « Jiang ».

où max est la distance maximale entre deux concepts obtenue par $Dist_{Jiang}$.

II.3.4 Evaluation des mesures de similarité

Commençons par récapituler les résultats des différentes mesures précédentes :

	Méthodes basées sur la distance			Mesures basées sur le contenu informationnel		
	Rada	Leacock	Wu-Palmer	Resnik	Lin	Jiang
Sim (Moto, Vélo)	2.00	0.60	0.75	0.40	0.15	0.20
Sim (Volant, Avion)	1.00	0.90	0.80	0.05	0.08	0.12
Sim (2 Roues, Roulant)	1.00	0.90	0.80	0.10	0.40	0.35
Sim (Vélo, Volant)	4.00	0.30	0.33	0.00	0.00	0.00
Sim (Vélo, Voiture)	4.00	0.30	0.50	0.10	0.09	0.10

Figure 11 : récapitulatif des différentes mesures de similarité

Une comparaison des différentes mesures sur l'ensemble des paires a été menée dans différents travaux Lin (1998), Jiang (1997) et McHale (1998). En considérant la hiérarchie de concepts issue de *WordNet*, la mesure qui permet d'obtenir les résultats les plus proches des jugements humains est la méthode proposée par Lin, suivie de très près par la mesure de Jiang. La pertinence de l'utilisation du contenu en informations des concepts est alors vérifiée.

Cette conclusion n'est pas vraie pour les évaluations utilisant le thésaurus *Roget*. Pour ce thésaurus, les meilleurs résultats sont obtenus par la mesure reposant sur le « *Edge Couting* ». Cette différence s'explique par le fait que le corpus considéré pour le calcul du contenu en informations des concepts est particulièrement adapté à l'utilisation de *WordNet* car le corpus est désambiguïsé à partir de ses *synsets* (HERNANDEZ, 2006).

II.4 CONCLUSION

Dans ce chapitre, nous nous sommes intéressés à l'apport du web sémantique à la recherche d'informations, en intégrant les ontologies comme composante dans le processus de recherche.

Nous avons ensuite soulevé quelques problèmes inhérents à cette intégration, à savoir la présélection des ontologies ainsi que l'annotation des documents.

Le processus d'indexation a été revisité. Quelques méthodes existantes ont été présentées. Nous avons également fait la différence entre l'indexation sémantique, basée sur les relations sémantiques entre les termes, et l'indexation conceptuelle qui, au sens large, se base sur les relations conceptuelles entre les termes à l'intérieur d'un corpus ou d'un document.

Le dernier point abordé, et non le moindre, fut celui de l'extension des requêtes. L'identification des concepts, leur désambiguïsation et l'expansion des requêtes ont été détaillées.

Chapitre III : Métaheuristiques et recherche d'informations

La fourmi est un animal intelligent collectivement et stupide individuellement, l'homme c'est l'inverse !
Karl Von Frish

Le Web est un environnement complexe, défini parfois comme cas extrême de distribution de documents connectés par les hyperliens. Face à cette taille imposante et sans cesse grandissante, il semble pertinent de chercher à restreindre, sans limiter, selon les attentes de l'utilisateur, les champs d'exploration pour constituer de mini-Web thématiques personnalisés.

Dans ce chapitre, nous nous intéresserons à la problématique de l'exploration thématique d'Internet en adoptant un point de vue orienté vers les techniques de vie artificielle.

Le terme « *vie artificielle* » est très générique, il est utilisé pour regrouper toutes les techniques informatiques fondées sur l'observation du comportement complexe des espèces vivantes : l'apprentissage, l'auto-organisation, l'émergence de comportements complexes à partir de règles simples, la coopération entre agents simples etc.

III.1 LES ALGORITHMES GENETIQUES

Les algorithmes génétiques ont été introduits par Holland (1975). Ils sont principalement inspirés de la *théorie de l'évolution* de Charles Darwin (1859) selon laquelle les espèces évoluent de génération en génération dans le sens d'une meilleure adaptation des individus à

l'environnement.

La nature assure, de cette manière, la pérennité des meilleures caractéristiques de chaque individu. La recombinaison de ces caractéristiques entre deux individus permet de former au fil des générations de nouveaux individus pouvant être mieux adaptés à leur environnement ; chaque enfant reçoit en effet des caractéristiques à la fois de son père et de sa mère.

La mutation de gènes introduit des erreurs dans ce processus de copie. Elle peut contribuer à améliorer les individus et notamment de les adapter à un changement d'environnement.

Les algorithmes génétiques (*AG*) sont des algorithmes d'optimisation s'appuyant sur les principes d'évolution naturelle et de sélection des espèces comprenant des croisements entre individus, des mutations apparaissant aléatoirement au sein d'une population et une sélection des individus les mieux adaptés à l'environnement Holland (1975) (PICAROUGNE, 2004). Face à un problème pour lequel il existe un grand nombre de solutions, l'AG va explorer l'espace des solutions en se laissant guider par les principes décrits précédemment.

Les algorithmes génétiques sont appliqués dans divers domaines pour résoudre des problèmes d'optimisation ou de recherche. Par exemple : la conception topologique de réseaux téléinformatiques, à la commutation et routage de d'information, la recherche d'informations etc.

Nous présenterons ici les principes et les fonctionnalités de base d'un algorithme génétique ainsi que ses caractéristiques. Nous décrirons les différentes opérations génétiques telles que la mutation et le croisement.

III.1.1 Principes et fonctionnalités

III.1.1.1. Principe de base et mode de fonctionnement

Les algorithmes génétiques sont des approches d'optimisation qui utilisent des techniques dérivées de la science génétique et de l'évolution naturelle : la sélection, la mutation et le croisement. La modélisation d'un problème pour une résolution avec des algorithmes génétiques doit fournir :

- **Le codage** d'un élément de population: une fonction qui permet de modéliser les données du problème réel dans des données utilisables par l'algorithme génétique.
- Une fonction pour générer **la population initiale** : la génération de la population initiale est importante puisque cette génération représente le point de départ de l'algorithme et son choix influe sur la rapidité et l'optimalité de la solution finale.
- **La fonction objectif** (une fonction à optimiser) : est une fonction qui retourne une valeur

d'adaptation pour chaque individu. Cette valeur permet de déterminer la solution pertinente puisque le problème se restreint à chercher le groupe d'individus qui ont les valeurs optimums.

- **Des opérateurs** qui permettent d'évoluer d'une population à une autre tout en améliorant la fonction « *objectif* ». L'opérateur de croisement recompose les gènes d'individus existant dans la population, alors que l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.
- **Des paramètres** de dimensionnement : taille de la population, nombre total de générations (critère d'arrêt), probabilités d'application des opérateurs de croisement et de mutation, etc.

La figure suivante illustre le cycle de vie d'un algorithme génétique :

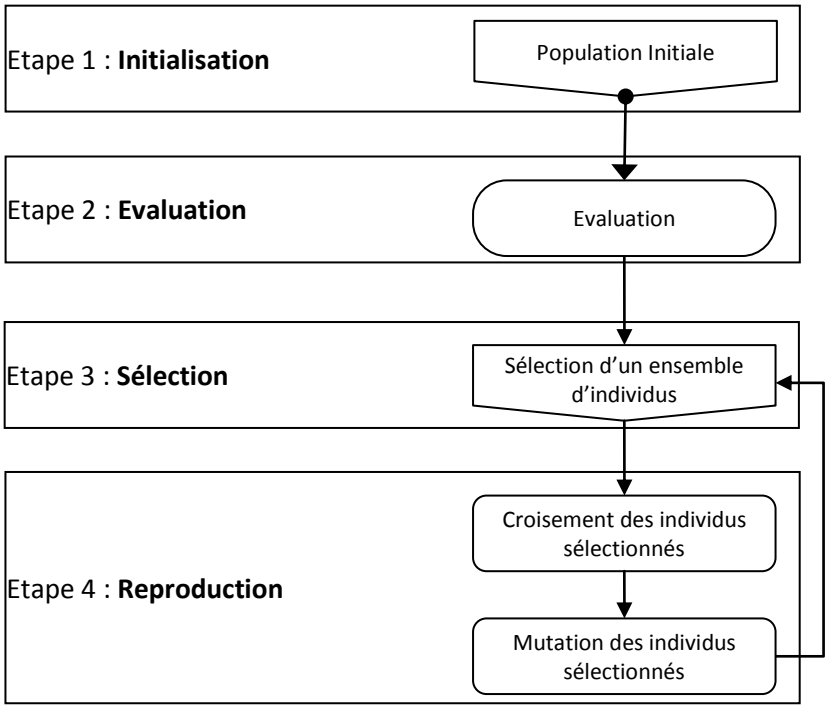


Figure 12 : Cycle de vie d'un algorithme génétique

- **Initialisation** : on choisit μ individus qui représentent la population initiale;
- **Évaluation** : on évalue chaque individu par la fonction objectif;
- **Sélection** : on définit les individus de la génération P qui vont être dupliqués dans la nouvelle population. A chaque génération, il y a deux opérateurs de sélection : la sélection de reproduction, ou plus simplement sélection, qui détermine les individus qui vont se reproduire durant une génération et la sélection pour le remplacement, ou plus simplement le remplacement, qui détermine quels individus devront disparaître de la population.

- **Reproduction** : on utilise des opérateurs génétiques (croisement et mutation) pour produire la nouvelle génération. Les opérateurs de mutation modifient un individu pour en former un autre tandis que les opérateurs de croisement engendrent un ou plusieurs enfants à partir de combinaisons de deux parents.

Plus formellement, un algorithme génétique peut être écrit comme suit :

```
variables
    P : Population initiale
    P' : population après sélection et croisement
    P'' : population après mutation
Algorithme Genetique
    Initialiser la population initiale P
    Evaluer(P)
    tantque (non Convergence) faire
        P'   := Sélection(P)
        P'   := Croisement(P')
        P''  := Mutation(P')
        P    := P''
        Évaluer(P)
    finTantque
Fin
```

Algorithme 1 : Déroulement d'un algorithme génétique

La notion de convergence peut prendre différentes définitions et varie d'un champ d'application à un autre. Elle peut être par exemple un temps de calcul maximum, un seuil minimal que l'on désire atteindre ou un nombre maximum de solutions etc.

III.1.2 Caractéristiques des algorithmes génétiques

Les algorithmes génétiques, en tant qu'approche de résolution de problèmes, se caractérisent par certains aspects particuliers: le codage des paramètres du problème à traiter, l'espace de recherche et la fonction d'évaluation qui permet de déterminer la pertinence d'une solution trouvée et l'évolution d'une génération à une autre par la sélection des chromosomes qui participent à la reproduction et les chromosomes à disparaître.

III.1.2.1. Le codage

Le codage est une fonction qui permet de passer de la donnée réelle du problème traité à la donnée utilisée par l'algorithme génétique (Figure 13). Le choix du codage est l'élément le plus important dans la conception de l'algorithme puisqu'il permet d'une part de représenter les données, les paramètres et les solutions et d'autre part il influe sur la mise en œuvre des opérations génétique telles que le croisement et la mutation qui influent directement sur le bon déroulement de l'algorithme génétique et de leur convergence vers la bonne solution.

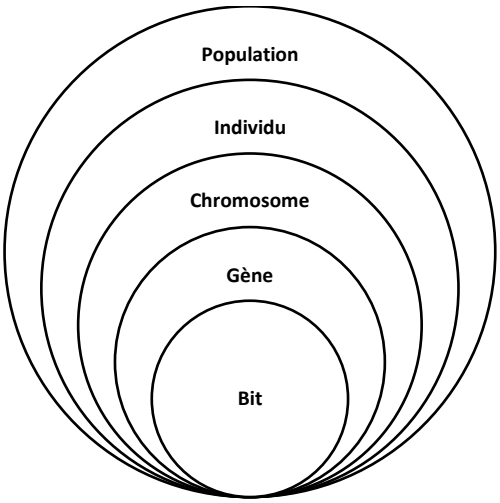


Figure 13 : niveaux d'inclusion des populations d'un algorithme génétique

Représentation binaire

Chaque gène dispose du même alphabet binaire *0*, *1*. Un gène est alors représenté par un entier, les chromosomes, qui sont des suites de gènes. Ils sont représentés par des tableaux de gènes et les individus de l'espace de recherche sont représentés par des tableaux de chromosomes.

Représentation avec réels

Contrairement au codage binaire, un gène est représenté par une suite de bits (un bits dans le code binaire) qui est associé à un réel. Ce type de codage peut être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle.

Représentation à l'aide d'arbres syntaxiques

Ce type de codage utilise une structure arborescente (une racine de laquelle peuvent être issus un ou plusieurs fils, eux mêmes des arbres). Un arbre syntaxique est un arbre contenant deux types de nœuds :

- les nœuds internes ou « *symboles non terminaux* »
- les feuilles ou «*symboles terminaux* »

Ce type de codage peut être utilisé lorsque la taille du problème ou de la solution n'est pas finie. Son inconvénient est qu'on peut trouver des arbres de solutions de taille importante difficile à analyser.

III.1.2.2. La fonction d'évaluation

La fonction d'évaluation, qu'on appelle aussi fonction d'adaptation, fonction objectif,

fonction de performance ou fonction fitness, associe une valeur de performance à chaque individu ce qui offre la possibilité de le comparer à d'autres individus et permet à l'algorithme génétique de déterminer quel individu sera sélectionné pour la reproduction ou pour un remplacement.

III.1.3 Les opérateurs génétiques

La reproduction est le processus qui permet de construire une population P' à partir d'une population P. Ce processus est constitué par l'utilisation de l'opération de sélection, de l'opération de croisement ou/et de l'opération de mutation (Figure 14).

III.1.3.1. La sélection

L'opération de sélection permet de déterminer quels individus sont plus enclins à obtenir les meilleurs résultats. On trouve deux types de sélection:

La sélection pour la reproduction:

Appelée simplement « *opération de sélection* », elle permet de choisir les individus qui participent à une reproduction (croisement ou mutation). Cette opération sélectionne, généralement, les individus les plus forts (meilleurs scores d'adaptation) pour produire les enfants les plus performants.

La sélection pour le remplacement:

On l'appelle tout simplement « *opération de remplacement* », et elle choisit les individus les plus faibles pour être remplacés par les nouveaux.

On trouve plusieurs techniques de sélection:

- **Sélection par rang** : Choisir toujours les individus possédant les meilleurs scores.
- **Sélection par tournoi** : Utiliser la probabilité de sélection proportionnelle à l'adaptation sur des paires d'individus, puis choisir parmi ces paires celles qui a le meilleur score d'adaptation.
- **Sélection uniforme** : Choisir aléatoirement sans faire intervenir la valeur d'adaptation.

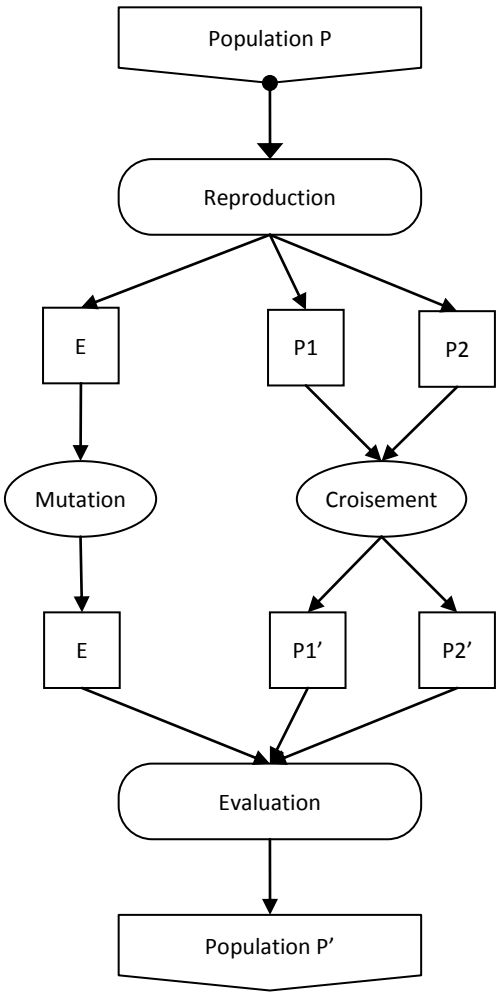


Figure 14 : Evolution d'une population dans un algorithme génétique

III.1.3.2. Le croisement

Le phénomène de croisement est une propriété naturelle de l'ADN. C'est par analogie qu'ont été conçus les opérateurs de croisement dans les AG. Cette famille d'opérateurs a pour but de répartir aléatoirement les individus sélectionnés en couples afin d'engendrer une nouvelle génération. Cette génération est obtenue en copiant et recombinant les deux chromosomes parents de manière à générer deux chromosomes enfants possédant des caractéristiques issues des deux parents.

L'opérateur de croisement favorise l'exploitation de l'espace de recherche en examinant de nouvelles solutions à partir de deux solutions actuelles, en assurant le brassage du matériel génétique de la population et l'accumulation des gènes favorables en les multipliant.

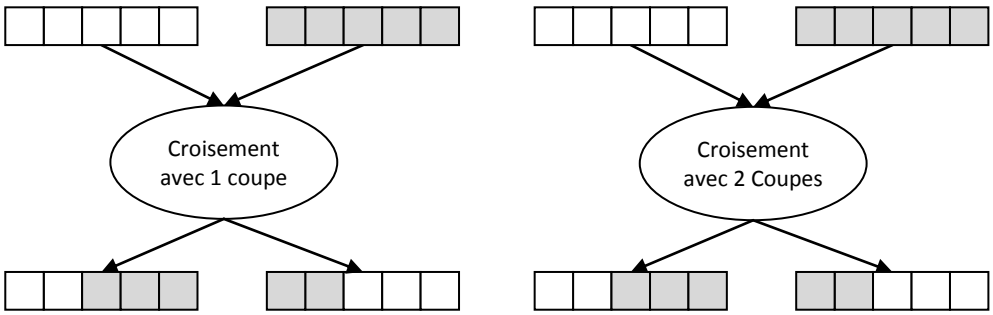


Figure 15 : les croisements dans un algorithme génétique

Le taux de croisement détermine la proportion des individus qui sont croisés parmi ceux qui remplaceront l'ancienne génération.

III.1.3.3. La mutation

L'opérateur de mutation consiste à modifier un gène sélectionné dans un chromosome et à le remplacer par une valeur aléatoire. La probabilité de mutation d'un gène est généralement très faible dans un environnement stable afin de favoriser la capitalisation des gènes favorables tout en permettant d'élargir l'espace des solutions évaluées.

Cependant, dans un environnement dynamique subissant une évolution rapide, il peut être judicieux d'employer un taux de mutation relativement élevé permettant une adaptation rapide de la population. Le taux de mutation est ainsi à adapter en fonction de l'application et du type de mutation employé.

L'opérateur de mutation apporte aux AG la propriété d'ergodicité de parcours de l'espace. Cette propriété indique que l'AG est capable d'atteindre tous les points de l'espace des solutions, sans pour autant avoir la nécessité d'énumérer l'ensemble de points de l'espace. D'un point de vue théorique, les propriétés de convergence des AG sont fortement dépendantes de cet opérateur. Il garantit ainsi que l'optimum global peut être atteint (PICAROUGNE, 2004).

La Figure 16 présente un exemple de mutation de chromosome, tel que le gène g est retiré aléatoirement et est remplacé par le gène g'.

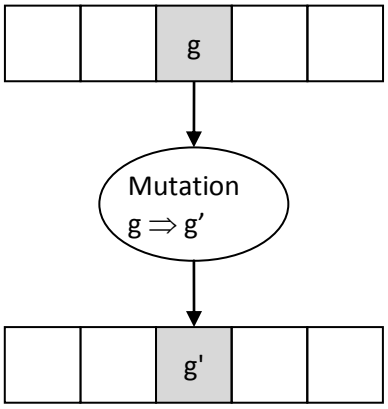


Figure 16 : la mutation du gène « g » en « g' » dans un algorithme génétique

La mutation est un phénomène rare mais qui permet d'explorer de nouvelles zones dans l'espace de recherche et aide l'algorithme génétique à possiblement aller vers une solution optimale globale, sans rester bloqué dans une solution optimale locale.

III.1.4 Parallélisation des algorithmes génétiques

Plusieurs auteurs ont exploré les mécanismes de parallélisation des algorithmes génétiques. L'idée motrice est évidemment le gain de temps de calcul, mais pas seulement. La diversité des environnements peut, sous certaines conditions, faire évoluer les individus en exploitant des ressources nouvelles pour produire des solutions originales.

On recense principalement deux grandes méthodes de parallélisation :

III.1.4.1. Parallélisation de l'évolution des individus

L'idée principale de cette méthode est de conserver la totalité de la population générée par les algorithmes génétiques sur une seule machine appelée « *maître* » qui est chargée d'effectuer toutes les opérations génétiques sur la population : *sélection, croisement* et *mutation*.

Les opérations de calcul, qui sont potentiellement plus lourdes, sont effectuées par des machines ou des processeurs dits « *esclaves* ».

La machine « *maitre* » attend que toutes les machines « *esclaves* » aient terminé l'évaluation de toute la population pour construire la nouvelle génération.

La première implémentation de cette méthode a été effectuée par Forgatti et Huang (1991) et (PICAROUGNE, 2004). Ils ont noté que le trop grand nombre de communications entre la machine « *maître* » et les machines « *esclaves* » influe sur la qualité du système.

Anbramson & Abela (1991) et (PICAROUGNE, 2004) ont implémenté le même algorithme sur une machine à mémoire partagée. Ils ont observé une progression linéaire des performances, mais une augmentation importante des ressources mises en œuvre.

Cette stratégie de parallélisation n'est intéressante que si le temps nécessaire à l'évaluation des individus est important, d'où l'intérêt de considérer plusieurs populations d'individus gérées d'une manière plus ou moins asynchrone.

III.1.4.2. Parallélisation par sous-populations d'individus.

Contrairement à l'approche précédente, celle-ci fait fonctionner plusieurs algorithmes génétiques en parallèle sur différentes populations d'individus. Chaque machine exécutant un algorithme génétique fait évoluer sa population indépendamment des autres.

Suivant certains critères, une machine peut décider de partager les gènes de ses meilleurs individus avec une ou plusieurs machines. La machine réceptrice décide d'intégrer ou non les

nouveaux individus dans sa population. De cette manière, chaque algorithme a la possibilité de converger vers des optima différents de ceux calculés sur d'autres machines. Ainsi, il devient possible de traiter de grandes populations et restituer des résultats dans un temps raisonnable.

Ce procédé de parallélisation a été très largement traité. Nous pouvons néanmoins regrouper tous les travaux sous deux grandes familles :

Parallélisme à gros grain (coarse grains)

L'idée initialement développée était d'exécuter le même algorithme génétique pour toutes les sous-populations d'individus.

Dans un parallélisme à gros grain, des opérateurs et des règles sont ajoutés pour gérer la communication entre les différentes sous populations. Selon la politique d'émigration adoptée (meilleure, aléatoire, etc.), un opérateur de migration est ajouté afin de permettre l'échange d'informations entre les sous-populations.

Des stratégies et des politiques sont généralement développées pour gérer le remplacement dans la population d'accueil à l'arrivée d'un nouvel individu.

Notons que le modèle « *island* » développé par Cohoon et al. (1987) est très intéressant dans l'exploration du web. Dans ce modèle, les individus peuvent migrer dans n'importe quelle autre population. Il a été observé que les meilleures solutions apparaissent majoritairement rapidement après l'introduction d'un nouvel individu.

Mühlenbein (1991) a introduit le modèle « *stepping-stone* » dans lequel les échanges sont limités aux populations voisines dans l'espace de recherche. L'originalité du modèle vient de l'utilisation d'une heuristique de type *hill-climbing* qui autorise, lorsque la qualité d'une population ne s'améliore pas au bout d'un certain nombre d'itérations, l'ouverture des frontières.

Parallélisme à grains fins (fine grain)

Ce sont des méthodes asynchrones qui divisent la population en un nombre conséquent de sous-éléments. Ces sous-éléments peuvent aller du cas limite (PICAROUGNE, 2004) pour lequel un seul individu est représenté par un processeur. Chaque entité est connectée à plusieurs autres entités dans son voisinage formant un groupe et les opérateurs génétiques sont appliqués à ce groupe grâce à des échanges entre les individus le composant.

Ces techniques ont été implémentées notamment par Spiessens et Manderick (1991) où chaque individu est mis à jour sur une architecture massivement parallèle.

Whitley (1993) qualifie même ce type d'algorithme de modèle cellulaire car on retrouve un parallèle avec les automates cellulaires notamment avec des règles probabilistes de réécriture et un alphabet composé des éléments de l'espace de recherche.

Peu d'études ont comparé les performances respectives d'une parallélisation à grains fins et à gros grains et les résultats sont souvent contradictoires comme le souligne Cantú-Paz (1999) et Cantú-Paz (2000) (PICAROUGNE, 2004).

Quelle que soit la méthode de parallélisation adoptée, il convient de respecter certaines règles : ne pas subdiviser la population en de trop petites sous-populations et sélectionner la méthode de parallélisation adaptée au problème.

III.2 SYSTEMES MULTI-AGENTS ET COLONIES DE FOURMIS

La notion de multi-agents est apparue dès lors que l'intelligence artificielle a montré les limites de la capacité d'un super-individu à résoudre des problèmes complexes dans un temps raisonnable. L'idée était de trouver une nouvelle approche pour résoudre des théorèmes en considérant plusieurs entités actives appelées « *acteurs* ». Le plus souvent, on a recours à la décomposition des problèmes en petites tâches plus simples à résoudre. Ces tâches peuvent ainsi être traitées par des systèmes moins lourds et plus faciles à maîtriser.

Cependant, la gestion de plusieurs entités coopérant à la résolution d'un problème commun génère des nouvelles problématiques, notamment la mise en œuvre de la coopération, les communications et la synchronisation. La gestion de toutes ces interactions a conduit à la notion de systèmes multi-agents.

III.2.1 Agent et environnement

Les systèmes multi-agents sont un paradigme de modélisation de systèmes complexes et de résolution de problèmes construit autour de la notion d' « *agent* ».

III.2.1.1. L'agent

Un agent peut être décrit comme une entité autonome capable de réaliser des actions. Si cette définition reste assez floue, c'est qu'il n'existe pas encore de consensus autour d'un formalisme de conception ou d'implémentation, mais plutôt un ensemble de règles, de guides décrivant les grandes propriétés qu'un agent doit respecter.

Un agent est donc une entité :

- **Autonome** : l'agent doit être capable d'agir de sa propre volonté, i.e. il ne doit être soumis à aucune autorité extérieure mais prend ses décisions en fonction de son état interne et de son environnement. Son état est la résultante de l'ensemble de ses perceptions, de ses connaissances et de ses croyances.
- **située** : un agent est nécessairement immergé dans un environnement sur lequel il va

agir. Il doit également être capable de percevoir cet environnement pour pouvoir réagir aux changements de celui-ci.

- **sociale** : l'agent doit être capable de communiquer et d'interagir avec les autre agents de son environnement. Ces interactions, directes ou indirectes, sont toujours effectuées à travers l'environnement.
- **Pro-active** : l'agent prend des décisions, basées sur son état ou sur son environnement, motivées par la poursuite d'un but.

III.2.1.2. L'environnement

L'environnement est un espace dans lequel évoluent les agents. Il peut être vu comme physique et disposer d'une métrique permettant de définir une mesure de proximité et fournir un support d'interaction entre agents. A l'inverse, il peut également être considéré comme étant sans dimensions, auquel cas, il sera limité à la transmission des communications entre agents. Le concept d'agent est un non-sens en dehors d'un environnement.

Figure 17 : boucle perception/action entre un agent et son environnement

L'environnement possède lui aussi plusieurs propriétés définissant et conditionnant directement le comportement des agents. Ainsi, un environnement peut être :

accessible/inaccessible

Cette propriété décrit la possibilité pour un agent de connaître/percevoir l'intégralité de son environnement. Mais généralement, puisque l'agent n'est qu'une entité immergée, il ne peut pas connaître à un instant donné l'état de l'environnement dans son ensemble. Il ne possède qu'une vue locale de l'état du système.

déterministe/non déterministe

Dans un système déterministe, chaque action effectuée par un agent fait évoluer l'environnement vers un état unique. A l'inverse, dans un système non déterministe, chaque action est régie par une probabilité de réalisation. Cette notion est très intéressante pour modéliser l'incertitude.

statique/dynamique

Cette propriété indique si un environnement peut être modifié autrement que par l'action des agents. Un environnement doté d'une force de pesanteur, un environnement capable de faire évaporer une phéromone ou un environnement capable de modifier sa topologie sont autant d'exemples d'environnements dynamiques.

continu/discret

Un environnement est dit continu lorsqu'il est possible d'associer à chacun de ses points une position réelle. Un environnement est discret, lorsqu'il est fini, et n'admet qu'un nombre fini dénombrable de positions accessibles aux agents. Concrètement, un environnement discret est modélisé par un graphe.

III.2.1.3. L'architecture des agents

Les agents possèdent un état interne qui est la dynamique de leur comportement. Un agent est souvent considéré comme un processus cyclique à trois étapes : *perception - décision - action* agissant sur la base de l'état sur l'état lui-même.

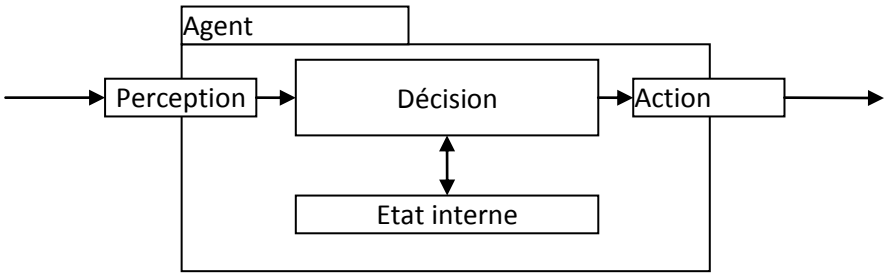


Figure 18 : représentation de l'architecture d'un agent

Les phases de perception et d'action sont globalement indépendantes de l'architecture physique des agents. La perception peut être ramenée à un opérateur transformant les données perçues en informations exploitables. L'action transforme les décisions en actions sur l'état via ses effecteurs.

La prise de décision est sûrement la phase la plus critique de l'architecture d'un agent. Elle détermine le comportement de l'agent. On distingue deux familles d'architectures basées sur les perceptions et la rationalité du raisonnement effectué par l'agent.

L'architecture cognitive

L'architecture cognitive est fondée sur une approche symbolique qui permet un raisonnement logique. Elle repose sur le traitement des informations symboliques et suppose l'existence d'un langage sur lequel les agents peuvent raisonner. On fait souvent référence,

lorsqu'on évoque cette architecture, à des agents *rationnels* ou *délibératifs*. Le modèle le plus connu est celui basé sur « *croyances-désirs-intentions* » qui permet aux agents de raisonner sur une base de *croyances* pour répondre à des *désirs.* Ils produisent ensuite des *intentions* d'actions qu'ils pourront ou non exécuter.

Cette architecture est particulièrement intéressante quand il s'agit d'effectuer des traitements complexes dans un environnement distribué.

Dans notre cas, cette architecture sera utilisée pour évaluer les documents d'un corpus sur la base d'une ontologie dans le but de produire un index sémantique d'une partition ou de tout le web.

L'architecture réactive

Minsky présente dans « *the society of mind* » en 1986, un modèle de l'intelligence humaine bâti à partir des interactions entre entités simples, incapables de produire un raisonnement élaboré, qu'il appelle agents. Ce concept d'agents a été repris et développé, sous l'impulsion de travaux en biologie et en informatique, pour aboutir à la modélisation des sociétés d'insectes. Même si ces agents réactifs sont individuellement extrêmement limités, leur interaction permet de développer un comportement *intelligent* pouvant résoudre des problèmes complexes.

Intuitivement, cette architecture est adaptée à l'exploration des graphes. Nous verrons comment, en utilisant des algorithmes extrêmement simples, on arrive à optimiser le parcours du web.

III.2.2 Les systèmes Multi-agents

L'intérêt du domaine des systèmes multi-agents repose sur l'utilisation simultanée d'une population d'agents pour modéliser des systèmes ou résoudre des problèmes. L'ajout d'agents permet au système de développer de nouvelles propriétés ou d'améliorer celles existantes.

Cependant, l'étude des systèmes multi-agents ne se limite pas à l'observation des agents isolés, mais s'étend à la gestion et l'harmonisation de l'ensemble.

III.2.2.1. Complexité des systèmes multi-agents

Les systèmes multi-agents sont des systèmes complexes où les composantes (agents) sont en perpétuelle interaction. Il en résulte la création de boucles de rétroaction complexes, de nouvelles propriétés impossibles à déduire depuis les spécifications du comportement des agents, etc.

Ainsi, comme tout système complexe, les systèmes multi-agents sont caractérisés par des comportements non linéaires qui mettent en échec les approches analytiques classiques. Ce type de systèmes se caractérise par les éléments suivants :

Comportement global imprévisible

Les boucles de rétroaction[1] et les interactions entre les nombreuses entités composant les systèmes multi-agents sont responsables de sa non-linéarité. Il devient alors impossible d'obtenir une modélisation analytique satisfaisante du système[2].

Emergence de structure et/ou de propriété

Les interactions entre les agents eux-mêmes et entre les agents et leur environnement peuvent produire des propriétés, des structures cohérentes ou des motifs sans qu'ils soient explicitement décrits dans les spécifications initiales des composantes du système. On parle alors d' « *émergence* » ou d'« *auto-organisation* »

Sensibilité aux conditions initiales

L'évolution d'un système peut être profondément modifiée par les variations des conditions initiales, connues aussi sous le nom d'« *effet papillon*[3] » qui fait référence aux phénomènes d'amplification observables dans les boucles de rétroaction. En fait, une infime variation d'un paramètre peut faire basculer le système vers un attracteur ou vers un autre ou le faire entrer dans un état « *chaotique* ».

Robustesse

Cette propriété exprime la capacité du système à conserver son organisation quand il est perturbé. La redondance, nombre d'agents dans un systèmes multi-agents, lui permet de résister à pratiquement toutes les fluctuations non prévues dans l'environnement ou à des comportements illicites.

III.2.2.2. Convergence des systèmes multi-agents

Les systèmes multi-agents, comme tout système complexe, sont amenés à converger vers des optima (attracteurs dans la littérature spécifique aux systèmes complexe). Même si ces optima sont préalablement établis et définis dans les spécifications du système, celui-ci peut s'en écarter. On dénombre quatre (04) types d'attracteurs :

- **Les points fixes** qui correspondent à un état unique vers lequel le système doit converger
- **Les attracteurs simples** qui correspondent à un ensemble fini d'états autour desquels le

[1] Feedback en anglais
[2] Il est éventuellement possible de modéliser le système dans un espace de fonctionnement. Mais se modèle perdra son pouvoir explicatif et prédictif dès que l'on sort de cet espace.
[3] D'origine météorologique, la métaphore dit que le battement d'aile d'un papillon peut déclencher une tempête à l'autre bout du monde : « *does the flap of a butterfly's wings in Brazil set off a tornado in Texas* ».

système va osciller ou évoluer de manière cyclique.

- **Les attracteurs complexes** qui correspondent à un ensemble fini d'états que le système va visiter de façon stochastique.
- **Le chaos** qui correspond au cas, possible, mais non souhaité, où le système, par des mécanismes d'amplification et récurrence, ne converge jamais vers un régime stable. On dit alors qu'il est « *chaotique* ».

Plusieurs familles d'algorithmes multi-agents existent. Elles s'inspirent toutes de l'observation de la nature. Chaque méthode essaye de tirer meilleur parti d'un comportement observé dans les groupes sociaux. On parle alors d'«*intelligence en essaim* ».

Dans ce qui suit, nous traiterons d'une méthode particulière inspirée de l'observation des fourmis fourrageuses. Nous verrons comment de telles colonies peuvent être efficaces pour la recherche d'informations sur le web.

III.2.2.3. Algorithmes multi-agents à base de fourmis.

En observant les insectes sociaux, on s'aperçoit qu'une communauté d'individus simples peut développer au niveau collectif des capacités supérieures à la somme des capacités individuelles de chacun de ses membres. Les colonies de fourmis en sont une parfaite illustrations. Selon Höllderboler et Wilson (1990), une fourmi n'est pas considérée comme une créature « *très* » intelligente, mais prise dans sa globalité, une colonie de fourmis est capable d'accomplir des tâches complexes et faire preuve d'une intelligence étonnante.

Les fourmis sont naturellement dotées de mécanismes leurs permettant d'identifier rapidement le chemin le plus court menant à la nourriture. Dans une fourmilière, chaque fourmi est spécialisée dans une tâche particulière. Les individus sont regroupés en des familles *spécialisées* mais non figées. Si, par exemple, le nombre d'individus nécessaire à l'accomplissement d'une tâche est insuffisant, des individus d'autres familles peuvent venir en renfort.

Les interactions entre les fourmis sont réglées généralement par des substances chimiques : les phéromones. Grâce à elles, les fourmis peuvent retrouver leur chemin ou encore émettre une alarme. Le seuil de phéromone définit l'appartenance d'un individu à telle ou telle famille.

Les algorithmes à base de fourmis artificielles s'inspirent du comportement collectif des fourmis fourrageuses pour résoudre les problèmes d'optimisation combinatoire ou de classification. Ainsi, les fourmis sont considérées comme des agents qui agissent dans un environnement donné. Dans ce qui suit, nous examinerons les algorithmes inspirés des colonies de fourmis et nous verrons comment ils peuvent servir à la construction d'un index

sémantique du Web.

ACO : l'algorithme fondateur

La première démonstration de la capacité et de la performance des algorithmes génétiques à base de fourmis artificielles a été réalisée par Dorigo (1992) avec l'heuristique *ACO* : *Ant Colony Optimisation* pour résoudre le Problème du Voyageur de Commerce.

L'algorithme se base sur des agents, organisés en colonies de fourmis, qui, partant chacun de son coté, vont essayer d'optimiser le chemin hamiltonien du graphe orienté *G(A,V)* représentant l'ensemble des villes considérées dans le problème (les sommets V) et des chemins les reliant (les arcs A).

Les fourmis sont disposées sur les sommets du graphe et peuvent voyager de sommet en sommet en suivant les arcs. À chaque voyage d'une ville i à une ville j, elles déposent une trace de phéromone sur l'arête (i, j). Le choix d'une nouvelle ville à parcourir se fait alors en fonction des villes précédemment visitées et selon une probabilité dépendant de la quantité de phéromone présente sur l'arête ainsi que de la distance entre cette ville et la position courante de la fourmi.

Une fois que toutes les fourmis ont effectué un parcours complet, le taux de phéromone présent sur chaque arête est mis à jour.

Cet algorithme permet aux fourmis artificielles de sauvegarder quelques données en mémoire et de percevoir une partie de leur environnement.

Ces facultés permettent d'adapter le comportement des fourmis artificielles au problème : chaque fourmi doit se souvenir des villes dans lesquelles elle est déjà passée afin de ne pas y retourner. Dans cette optique, une liste des villes à ne pas visiter est maintenue au sein de la mémoire de la fourmi et réinitialisée une fois qu'un parcours complet a été effectué (i.e. toutes les villes ont été visitées).

Pour cela, le taux présent à l'itération précédente est diminué proportionnellement à une constante d'évaporation définie à l'origine puis renforcée en fonction du parcours des fourmis sur l'arête prise en compte.

L'algorithme se termine au bout d'un temps défini et la solution retenue correspond au chemin de moindre coût, le plus court, parcouru par les fourmis au cours de l'exécution de l'algorithme.

L'algorithme API

L'algorithme API est un algorithme d'optimisation qui s'inspire de la stratégie de fourragement observé chez les fourmis de la famille Pachycondyla Apicalis (BENNOUAS, 2005).

Ces fourmis peuvent être considérées comme assez primitives, elles n'utilisent pas de communication poussée à base de phéromone lors de la recherche de nourriture et leur comportement de chasse reste relativement simple. En effet, les fourmis chassent individuellement en tentant de couvrir un espace particulier autour du nid de la colonie.

Pour cela, elles sélectionnent plusieurs sites de chasse et effectuent des explorations locales autour de ces sites. Il s'agit à la fois d'une stratégie globale et locale qui s'est révélée assez efficace pour maintenir jusqu'à nos jours l'existence de ces fourmis.

Chaque fourmi s'attribue des sites particuliers qu'elle va explorer aléatoirement en mémorisant toutefois préférentiellement ceux sur lesquels elle a pu rencontrer un certain succès. Au fur et à mesure de leur sortie, les ouvrières chargées de récolter de la nourriture s'éloignent de plus en plus du nid couvrant petit à petit une grande partie de leur espace de recherche.

Trois règles décrivent le comportement de ces fourmis (PICAROUGNE, 2004) :

- La découverte d'une proie entraîne toujours le retour sur le même site de chasse lors de la sortie suivante,
- La découverte d'une proie pèse sur la décision de sortie des fourrageuses en réduisant l'intervalle de temps passé au nid,
- Les fourrageuses semblent apprendre progressivement une association entre une direction de recherche opposée au nid et l'augmentation de la probabilité de succès.

Les algorithmes inspirés par les colonies de fourmis sont des algorithmes à base d'agents particuliers. Dans la section suivante, nous allons détailler comment tous ces mécanismes sont mis à profit lors de la recherche d'informations.

Nous nous inspirerons très fortement du comportement des fourmis pour concevoir un modèle algorithmique relativement simple de recherche d'informations sur Internet par algorithme de fourmis.

III.3 RI ET METAHEURISTIQUES : UN PAYSAGE DIFFERENT A CHAQUE RECHERCHE.

En effet, au vu du temps supplémentaire alloué à la recherche, nous pouvons effectuer des traitements supplémentaires sur l'ensemble « *résultats* » d'un moteur de recherche classique pour en améliorer la pertinence. Ces opérations sont principalement :

- **Reformuler** la requête utilisateur,
- **Télécharger** des pages actualisées pour une meilleure analyse,
- **Apparier** un document à une requête en fonction d'une similarité sémantique,
- **Classer l**es résultats en fonction de leur pertinence sémantique par rapport à la requête.

L'espace de recherche utilisé par les moteurs de recherche est Internet dans sa globalité. Nous choisissons de modéliser le Web en un graphe dont les nœuds sont les pages, et les arcs, les liens hypertextes qui existent entre les pages. Cette modélisation, bien que simple, permet d'optimiser l'espace de recherche en reproduisant fidèlement le comportement de l'utilisateur.

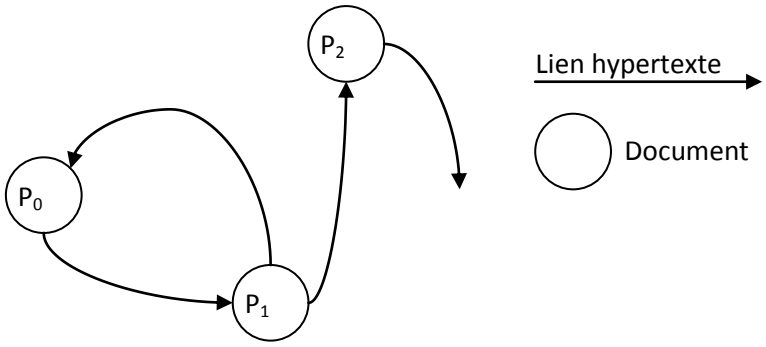


Figure 19 : Modélisation d'internet sous forme de graphe.

Cependant, l'avantage majeur d'une telle représentation est l'utilisation d'algorithmes à base d'agents parcourant le web en suivant les arcs pour aller d'un nœud à un autre.

Le graphe ainsi obtenu, noté S, nous permet de considérer chaque nœud du graphe correspondant à un point de S. La notion d'arc établit une relation de voisinage pour chaque point de cet espace. Il est alors possible de valuer les nœuds du graphe en définissant une fonction similarité sémantique dont les paramètres sont les termes de la requête utilisateur.

L'espace original, tout internet à la base, est alors transformé en un espace de recherche sémantique et le problème de recherche d'informations consiste alors à trouver les optima de cet espace de recherche.

Ainsi, chaque requête utilisateur va présenter une fonction d'évaluation différente et un paysage de qualité différent (PICAROUGNE, 2004). Il devient donc possible d'appliquer les algorithmes d'optimisation pour obtenir des solutions personnalisées aux problèmes (requêtes utilisateur).

La table suivante illustre l'analogie entre la modélisation du problème de recherche d'informations sur le Web et le problème d'optimisation d'une fonction par méta-heuristique :

Internet	⇔	Optimisation
Ensemble de pages Web	⇔	Espace de recherche
Adéquation de la page à la requête	⇔	Fonction d'évaluation
Ensemble de documents pertinents	⇔	Solution optimale
Liens sortants d'une page	⇔	relation de voisinage

Figure 20 : modélisation du problème de la RI en un problème d'optimisation

Les sections suivantes décrivent les opérations particulières permettant l'exploration et plus généralement l'espace de recherche ainsi adapté au problème d'optimisation.

III.3.1 Les opérateurs de recherche

Les opérateurs de recherche servent à parcourir l'espace de recherche *S*. Nous pouvons distinguer différents types d'opérateurs permettant d'explorer de manière relativement distincte cet espace *S*.

Le premier opérateur, noté O_{entree} est celui de génération d'un espace indépendamment de tout point. Il est nécessaire à la création d'un point d'entrée dans l'espace. Le second, noté $O_{explore}$ permet d'examiner le voisinage local d'un point particulier de *S*.

III.3.1.1. La création heuristique ou l'espace d'entrée

Cet opérateur est utilisé afin de créer des points de *S* sans tenir compte d'aucune information issue d'un autre point.

La première approche consiste à élire aléatoirement un ensemble de pages du web en se basant sur leur adresse IP et un chemin d'accès. Evidemment, cette méthode comporte de nombreux inconvénients. Lawrence et Giles (1999), Kishi et al. (2000) ont montré que la probabilité d'avoir une adresse valide dans le cas d'une génération aléatoire d'adresse IP est très faible. En effet, leurs expérimentations ont montré qu'une (01) adresse sur 269 était valide et que seulement un (01) serveur sur 6 contenait des pages accessibles directement.

Pour notre modèle, nous choisissons d'utiliser une heuristique qui va construire une solution initiale sur la base des résultats « *R* » retournés par un moteur de recherche classique en fonction de la requête « *q* », ainsi : $S = O_{entree}(R_q)$

Notons que la majorité des moteurs de recherche renvoient uniquement les 1000 premiers résultats, nous associerons un agent à chaque entrée de l'espace initial.

III.3.1.2. L'exploration locale

L'opérateur d'exploration locale $O_{explore}$ va se servir des points existants pour en générer de nouveaux. Ainsi, à partir d'une page Web donnée, il est possible d'obtenir l'ensemble de ses voisins, de les explorer un à un et d'en choisir le meilleur, en fonction de sa pertinence, comme nouveau point de départ.

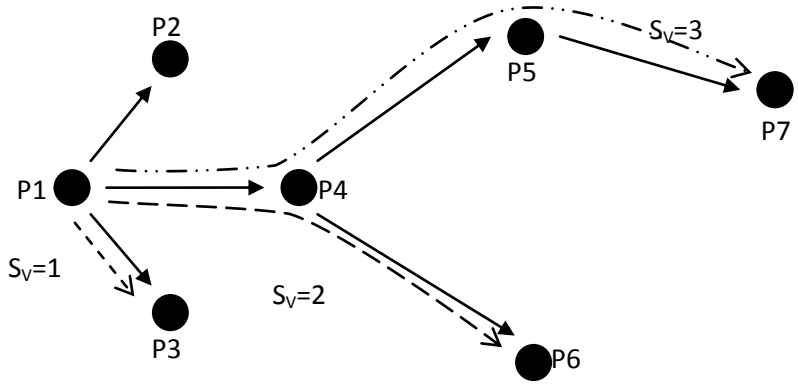


Figure 21 : taille du voisinage S_V utilisé par $O_{explore}$.

A partir des pages web composant le réseau, on effectue la transformation sous forme de graphe. La page *P3* correspond à la cible des liens hypertextes de la page *P1* dans un voisinage S_V=1. Les pages *P6* et *P7* correspondent aux cibles des liens de la page *P1* dans les voisinages respectifs S_V=2 et S_V=3.

La première étape est donc celle de l'extraction de la liste des liens contenus dans une page. L'opérateur d'exploration va donner en sortie un nouveau point qui peut être dans le voisinage immédiat de la page (S_V=1), i.e. une page pointée directement par un lien hypertexte, ou bien plus loin dans le graphe des relations de voisinage.

Nous discuterons plus loin de la stratégie de sélection des points à explorer. Mais en général, sachant que le diamètre d'Internet est d'une moyenne de 21 liens, ce type d'opérateur peut couvrir une grande zone de l'espace de recherche avec une fonction de voisinage relativement faible. Ceci rejoint les constatations de Albert et al. (1999) qui concluent qu'un agent intelligent suivant uniquement les liens hypertextes sur Internet peut retrouver relativement rapidement n'importe quelle information (CHAMPLAUX, 2009).

Un paramètre important à définir est la fonction d'ordonnancement des liens par l'opérateur de recherche : la sémantique des liens d'ancrage, le voisinage du lien, le poids des termes, etc. sont autant de paramètres qu'il convient de considérer. Pour cela, une fonction d'évaluation est construite et associée à chaque requête utilisateur.

III.3.2 La fonction d'évaluation

Quelle que soit la fonction d'évaluation retenue, les trois premières étapes de l'indexation que nous qualifierons d'évaluation statistique (I.2.1.3, page 11) seront exécutées. Nous nous intéresserons ensuite à l'évaluation sémantique, i.e. à la pondération et la mesure de similarité entre les termes de la requête et le document.

III.3.2.1. L'évaluation des termes

Nous supposons dans ce qui suit que les requêtes introduites par l'utilisateur sont enrichies avec un vocabulaire, une ontologie dans notre cas. Ainsi, si $t_1,..t_m$ sont les termes de la requête initiale, une requête enrichie comprendra les termes $k_1..k_n$ augmentés de la similarité du terme générique k_i , notée sim_i, avec le terme original t obtenu en sélectionnant les termes K d'une ontologie similaire au terme T de la requête avec la mesure de Jiang (voir section II.3.3.5 page 42).

Différents critères peuvent être ajoutés à une requête, à savoir :

- La présence obligatoire du terme i dans le document O_i;
- L'absence obligatoire du terme i dans les documents nO_i,
- La présence souhaitée du terme i dans les documents S_i;
- La présence non souhaitée du terme i dans les documents nS_i;

Ainsi, le poids d'un terme i, noté w_i dans un document est calculé comme suit :

$$w_i = occur(k_i) \times sim_i$$

Il est cependant nécessaire de s'assurer qu'aucun terme ne domine les autres. Une fonction $uniform_i$ favorise une représentation équitable des mots-clés et prend une valeur maximale de 1 quand les proportions d'apparition des termes k dans le document D sont égales. Elle sert principalement à retarder les pages où un terme domine tous les autres et irait à l'encontre du principe de diversification des espaces de recherche.

$$uniform_i = \frac{\sum_1^n w_i}{occur(K)}$$

Equation 20 : représentation équitable des termes d'un document.

III.3.2.2. L'évaluation des liens

Les liens présents dans une page n'ont certainement pas la même valeur informationnelle. Il est important pour la fonction d'évaluation d'établir un ordre entre les liens. Cet ordre est défini par la valeur d'une fonction notée $evalLink_i$ qui évalue le $i^{ème}$ lien de la page D.

La valeur de $evalLink_i$ est d'autant plus élevée que L_i est fréquent dans D et que la distance entre un terme k et le lien $L_{i,j}$ notée $dist(L_{i,j} | k)$.

Ainsi, pour chaque occurrence j du lien L_i, notée $L_{i,j}$ on calcule le poids du lien en fonction des termes dans son entourage avec la fonction $wL_{i,j}$ comme suit

$$wL_{i,j} = \sum_{n}^{k=1} \frac{w_k}{dist(L_{i,j} | k)}$$

$$evalLink_i = \frac{\text{occur}(L_i)}{count(L)} \times \sum_{n}^{j=1} wL_{i,j}$$

Equation 21 : Evaluation d'un lien entre deux noeuds.

Ainsi la valeur de retour de *evalLink* est comprise dans l'intervalle [0..1].

III.3.2.3. L'évaluation sémantique

On entend par évaluation sémantique la fonction d'appariement sémantique des documents aux termes de la requête. Cette évaluation se base sur l'importance relative du document par rapport à chaque terme de la requête. Pour ceci, une mesure de similarité est effectuée entre chaque terme de la requête et le document.

Deux contextes sont alors définis :

Evaluation locale

Elle représente la pertinence du document par rapport à la requête actuelle et désignée par le terme *contexte*.

Trois paramètres sont pris en considération :

- **Fréquence du terme** dans le document : elle permet de favoriser les termes les plus fréquents et associés à un concept de l'ontologie de référence.
- **La similarité sémantique** entre les termes du document et chaque terme du contexte.
- **L'importance du lien** entre les concepts, ce qui permet de favoriser les concepts les plus reliés.

Ainsi, la pertinence d'un document par rapport à la requête, notée *rsv*(*d*, *Q*), devient :

$$rsv(d,q) = \sum_{\text{i}} \frac{\sum_{\text{j}} \max\left(\text{sim}\left(\text{t}_\text{j}^\text{d}, \text{q}_\text{i}\right) \times \text{w}\left(\text{t}_\text{j}^\text{d}\right)\right)}{\sum_{\text{j}} \text{w}\left(\text{t}_\text{j}^\text{d}\right)}$$

Equation 22 : évaluation locale de la pertinence d'un document par à la requête.

Où :

- $\text{w}\left(\text{t}_\text{j}^\text{d}\right)$ désigne le poids du terme *j* dans le document *d*
- $sim\left(t_j^d, q_i\right)$ désigne la similarité entre un terme *j* du document *d* et un terme *i* de la requête *q* selon la mesure de Jiang.
- L'opérateur max permet de favoriser la sémantique du document par rapport à l'analyse syntaxique qui, elle, ne favorise que les occurrences.

III.3.3 Quelques approches génétiques pour la RI

Il existe plusieurs types de systèmes de recherche d'informations sur Internet. Mais généralement, seuls les méta-moteurs de recherche utilisent des algorithmes évolutionnaires afin d'améliorer les résultats produits par une méthode plus classique à base d'index inversé. Les méta-moteurs ont en effet de plus grandes ressources matérielles à disposition pour effectuer une recherche parce qu'ils se présentent généralement comme une application mono-utilisateur et ne doivent par conséquent pas répondre à des milliers d'utilisateurs en parallèle. Sur ce principe, des agents intelligents ont vu le jour, agissant comme des assistants personnels de recherche capables de s'adapter à l'utilisateur les manipulant.

III.3.3.1. Webnaut

Webnaut est un système décrit par Nick et Themis (2001) qui combine un méta-moteur interrogeant les moteurs de recherche classiques (*Google*, etc.) avec un algorithme génétique permettant de générer de nouvelles requêtes afin de trouver de meilleurs résultats, le tout étant contrôlé par l'utilisateur au travers d'un système de *relevance feedback*. Cinq étapes se succèdent ainsi afin d'obtenir les meilleurs résultats possibles :

- Collecter des données sur les préférences de l'utilisateur (pages de résultats visitées du méta-moteur),
- Extraire des informations de ces données pour créer un profil de recherche (mots les plus représentatifs des documents),
- Découvrir de nouveaux documents en accord avec le profil créé précédemment,
- Évaluer et ne garder que les meilleurs documents découverts,
- Demander à l'utilisateur d'indiquer les meilleurs résultats retournés.

La première étape consiste donc classiquement à demander à l'utilisateur de poser une question au système de recherche à l'aide de mots-clés. Le méta-moteur interroge plusieurs moteurs de recherche et en extrait les résultats, élimine les doublons, et les présente à l'utilisateur sous forme de résumés. Ce dernier peut alors visiter les pages lui paraissant convenir le mieux à sa question. Pendant ce temps, le système enregistre les pages visitées et les propose en entrée du système génétique afin de créer de nouvelles requêtes. Le système se compose de deux populations d'individus distincts : une composée de termes et une composée d'éléments de la logique booléenne (*et*, *ou*, . . .).

Dans *Webnaut*, les documents pris en compte sont représentés sous forme vectorielle dans laquelle les poids des mots sont calculés à partir de leur fréquence d'apparition et de la fréquence du mot le plus représenté dans le document.

Deux opérateurs génétiques sont utilisés dans chaque *AG* : un opérateur de croisement à un point de coupure agissant sur deux individus, et un opérateur de mutation qui consiste à choisir aléatoirement un élément d'un vecteur représentant un individu et à le remplacer par un autre, tiré au hasard dans le cas de la population de vecteurs de booléens, et issu des documents initiaux dans l'autre population.

Les résultats produits par la meilleure requête sont alors présentés à l'utilisateur afin que celui-ci sélectionne les meilleures pages. Le processus peut alors recommencer dans le but d'améliorer encore les résultats obtenus.

III.3.3.2. Une approche multi-agents : *InfoSpiders*

Les moteurs classiques de recherche d'informations utilisent des *crawlers* afin de parcourir Internet dans le but d'indexer un maximum de pages. Ces robots agissent comme des agents relativement peu évolués. En effet, ils cherchent uniquement à parcourir l'ensemble d'Internet, sans stratégie particulière pour trouver une page intéressante pour une requête donnée. L'effort d'indexation est rentabilisé dans les moteurs de recherche par la très grande réutilisation de cet index afin de répondre aux requêtes des utilisateurs.

En partant de l'hypothèse que des agents intelligents, qui arriveraient à déterminer efficacement les liens hypertextes utiles et intéressants d'une page, peuvent trouver relativement rapidement des documents pertinents à une requête donnée, Menczer et Monge (1999) puis Menczer (2003) ont conçu le modèle *InfoSpiders*.

Ce système est composé de plusieurs agents chargés de retrouver en permanence les documents correspondants à une requête donnée. Un tel système, au contraire d'une approche classique à base d'index, a l'avantage de prendre en compte naturellement le caractère de plus en plus dynamique des documents du réseau évoluant de jour en jour.

L'utilisation d'agents complètement autonomes décompose le problème général en plusieurs sous-problèmes que chaque entité va se charger de résoudre. Cette stratégie est inspirée de l'adage «*diviser pour régner*». Chaque individu va ainsi naviguer de documents en documents en prenant lui-même ses décisions sur les liens à suivre pour trouver des pages.

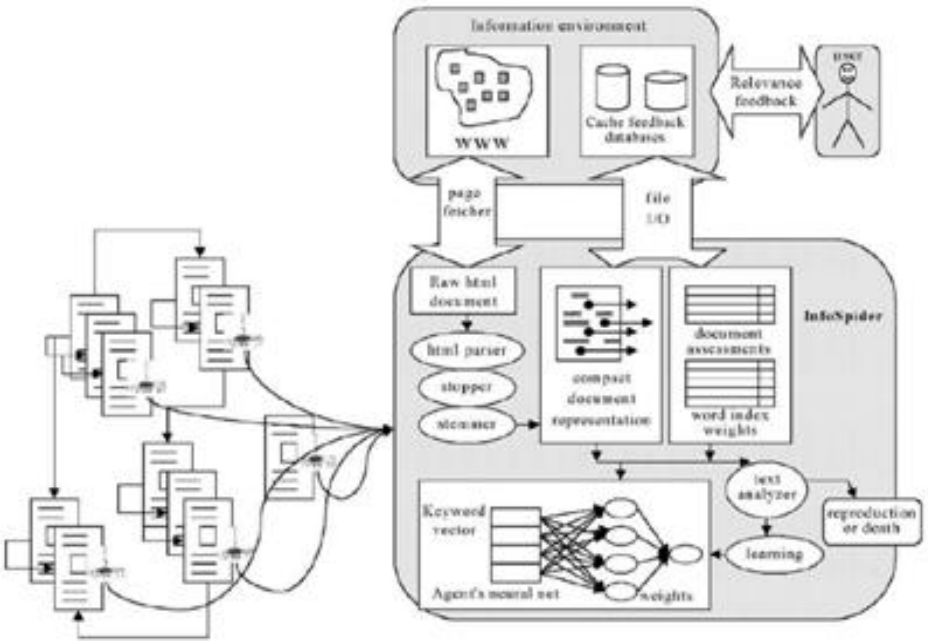


Figure 22 : Architecture de *Infospiders*

A des fins d'évaluation, tous les individus possèdent une certaine énergie fixée uniformément à l'initialisation. Afin de garantir de bons points de départ, les moteurs classiques de recherche sont interrogés à l'aide de la requête de l'utilisateur, ou de signets définis par l'utilisateur lui-même.

III.4 CONCLUSION

Dans ce chapitre, nous nous sommes intéressés aux algorithmes génétiques comme alternative métaheuristiques pour la résolution des problèmes complexes.

Après avoir expliqué le principe de fonctionnement des ces algorithmes, nous avons décrit les opérateurs génétiques qui permettent à l'algorithme de converger vers un optimum. La sélection, le croisement et les mutations permettent à chaque génération de préserver et de partager les meilleures qualités de ses individus.

Ainsi, la redéfinition du problème, ou son codage pour les *AG*, la collaboration entre les individus, la communication et le respect de règles simples permettent d'atteindre le but suprême : *optimiser*.

Les algorithmes devenant plus fins, les ressources matérielles quasi-intarissables, la parallélisation des traitements et la distribution des tâches sont devenus des évidences. L'apport va de soi : moins complexe, moins long. Que demander de plus ?!

En effet, une modélisation en agents autonomes organisés dans une société, permet de mettre en œuvre très rapidement des architectures parallèles pour résoudre des problèmes complexes.

Enfin, la première inspiration n'étant autre que la nature, en imitant les fourmis fourrageuses, dans le processus de recherche de sources d'alimentation, nous avons développé un modèle représentant le web par un graphe orienté dont les nœuds sont les documents et les arcs sont les liens hypertextes.

Ce modèle nous permet d'appréhender très simplement le problème de parcours du web et de l'indexation de son contenu. Chaque fourmi de la colonie n'a conscience que de l'endroit où elle se trouve (le document courant) et des endroits où elle peut aller (les documents pointés par des liens hypertextes). Nous avons alors redéfini les opérateurs d'exploration ainsi que la fonction *fitness* pour permettre une évaluation sémantique du document en cours et d'optimiser la fonction d'exploration en explorant en premier les documents jugés les plus pertinents.

Dans le chapitre suivant, nous allons proposer une palette d'algorithmes permettant l'implémentation d'une distribution de recherche sémantique d'informations guidée par une ontologie.

Chapitre IV :
Un modèle distribué pour la recherche sémantique d'informations

Le contrat social actuel est implicitement fondé sur une logique de non-coopération. Les coopérations existent bien sûr, mais elles ne sont pas exigées ni attendues par le corps social. Elles sont informelles. Il est certain que l'on n'avancera pas vers une Intelligence Collective sociétale sans un nouveau contrat social que l'on pourrait appeler contrat collaboratif
Management de l'intelligence collective. Oliver ZARA.

Dans ce chapitre, nous allons décrire un modèle distribué pour la recherche sémantique d'informations sur internet. Nous essayerons d'aborder les moteurs de recherche sous l'angle des algorithmes génétiques distribués ou parallèles.

Deux hypothèses fondatrices sont posées :

- L'utilisateur dispose d'une définition du domaine de la recherche (ontologie) qu'il maitrise.
- L'utilisateur est disposé à attendre les résultats plus longtemps au profit d'un meilleur résultat de la recherche.

IV.1 ARCHITECTURE DU SYSTEME

Le système est organisé en quatre couches distinctes comme illustré dans la Figure 23.

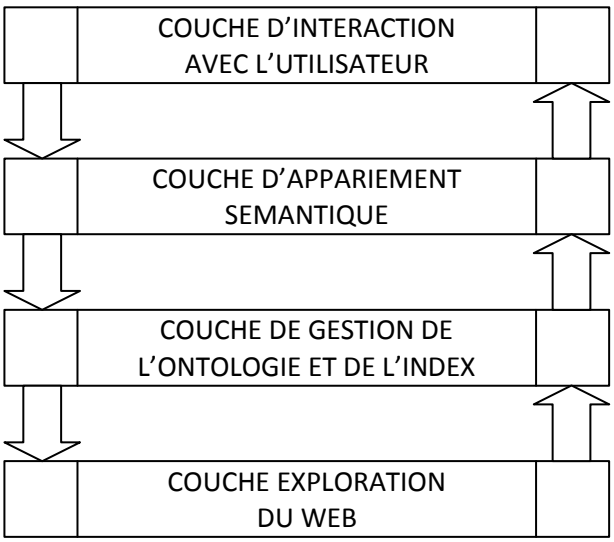


Figure 23 : Les différentes couches du système.

La définition de ces couches est motivée par la projection des différences conceptuelles entre les agents dans l'architecture du système. En effet, nous avons identifié quatre classes d'agents :

- **les agents interface** : chargés de l'interaction avec l'utilisateur. Ils assurent la transmission de la requête et l'affichage des résultats de la recherche ;
- **les agents sémantiques** : chargés de l'appariement, et éventuellement du classement, des documents répondant à la requête de l'utilisateur ;
- **les agents d'index** : chargés de la gestion de l'index, du maintien de sa cohérence et de son évolution ;
- **les agents crawlers ou fourmis** : chargés du parcours du web.

IV.1.1 Description des agents du système

La figure suivante illustre la hiérarchie des classes des agents du système :

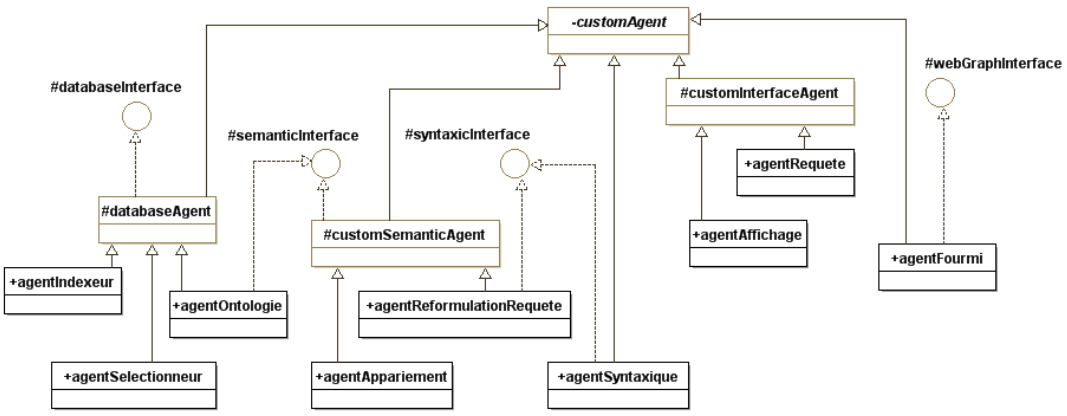


Figure 24 : Hiérarchie des agents du système

IV.1.1.1. Les agents d'interface

Les agents d'interface sont chargés de l'interaction du système avec l'utilisateur. Ce sont des agents réactifs qui assurent la transmission des informations dans les deux sens : utilisateur/système et système/utilisateur.

Les agents de requête

Ces agents, très simples par ailleurs, récupèrent la requête exprimant les besoins de l'utilisateur et la transmettent à la couche sémantique pour y être reformulée.

Les agents d'affichage

Ces agents sont chargés de l'affichage des résultats de la recherche. En effet, ils sont capables d'interagir avec l'utilisateur pour adapter la présentation des résultats à ses besoins. Une telle application de ses agents est très largement détaillée dans (PICAROUGNE, 2004).

IV.1.1.2. Les agents de la couche d'appariement sémantique

La mission première de la couche d'appariement sémantique est de calculer la pertinence des documents renvoyés par la couche inférieure par rapport à la requête de l'utilisateur. Cependant, nous avons établi que la reformulation des requêtes et la mesure de similarité sémantique entre concepts sont des processus nécessaires. De ce fait, nous identifions les agents suivants :

Les agents de reformulation des requêtes

Ces agents sont chargés, à partir d'une requête, de proposer une nouvelle formulation capable de mieux répondre aux besoins de l'utilisateur. Ils procèdent soit à la réécriture de la requête si celle-ci est jugée ambigüe ou imprécise, soit à son enrichissement si elle jugée incomplète.

Les agents d'appariement

Ces agents sont chargés d'apparier les documents renvoyés par la couche inférieure avec les termes de la requête, reformulée ou non, de l'utilisateur. En effet, la couche index calcule la pertinence de chaque document par rapport à un terme. Ainsi, un même document, jugé pertinent par rapport à plusieurs termes sera renvoyé avec une valeur de pertinence pour chaque terme. Ces agents calculent alors la pertinence du document par rapport à l'ensemble des termes.

IV.1.1.3. Les agents d'index et d'ontologie

Ces agents gèrent principalement la persistance des données dans le système : l'index du corpus et les ontologies de référence.

Les agents superviseurs

Ces agents sont le cœur du système. Ils n'ont pas de tâches, stricto sensu, dans le processus de recherche d'informations. Leur rôle se limite, si l'on puisse dire, à l'ordonnancement des tâches des différents agents, à la planification des processus en cas de conflits ou d'erreurs et l'allocation des ressources critiques.

Les agents sélectionneurs

Ces agents sont chargés de sélectionner dans la base d'index les documents pertinents par rapport à un terme de la requête. Prenant un à un les termes de la requête, ils extraient les documents pertinents et leur associent la valeur de pertinence relative à chacun des termes pour lesquels ils sont jugés importants.

Les agents décideurs

Ces agents décident, en recevant une notification d'insertion, de modification ou de suppression d'un document de l'action à accomplir. L'action décidée ainsi que la notification, reçue, sont envoyées aux agents indexeurs.

Les agents indexeurs

Ces agents mettent à jour la base de l'index pour refléter les modifications dans le corpus. Ils transmettent le corps des documents notifiés aux agents syntaxiques et décident de la mise à jour ou non du rang des documents dans le corpus par rapport à l'ontologie de référence.

Les agents syntaxique

Ces agents sont chargés de l'analyse syntaxique des documents du corpus.

Les agents ontologiques

Ces agents sont chargés de la gestion de l'ontologie de référence. Dans notre système, ils sont chargés de la mesure de similarités entre les concepts.

IV.1.2 Les agents crawlers

Les agents crawlers ou fourmis sont chargés d'explorer le graphe du web pour la détection de nouveaux documents ou de nouveaux liens. De leur qualité dépend l'efficience de

l'index.

IV.1.2.1. La classe *fourmi*

Chaque fourmi exploratrice contient en mémoire trois listes :

- **Liste des arcs suivis,** notée *arcTabou* : L'intérêt de cette liste est d'empêcher une fourmi de repasser par un chemin qu'elle a déjà emprunté. Elle est initialisée à ϕ (ensemble vide) à la création de la fourmi.
- **Liste des nœuds visités,** notée *noeudTabou* : à chaque fois qu'une fourmi transite d'un nœud vers un autre, le nouveau nœud est ajouté à cette liste.
- **Liste des arcs à suivre,** notée *instanceArcs* : elle contient tous les arcs que la fourmi doit suivre. elle est mise à jour à chaque nouvel arc découvert.

De plus, chaque fourmi implémente les opérations suivantes

- **suivreLien(***arc***)** : assure la transition entre deux nœud en suivant le lien arc.
- **deposerPheromone(***arc***,** *qPh***)** : dépose la quantité *qPh* de phéromone sur *arc*.
- **pheromone(***arc***)** : lit la quantité de phéromone déposée sur *arc*.
- **distance(***voisin***)** : retourne la distance entre *nœudCourant* et *voisin*.

Les algorithmes suivants implémentent les méthodes suscitées :

suivreLien (*arc*)

```
empiler noeudCourant dans listeNœuds

deposerPheromone(arc, qPh)
noeudCourant := cible(arc)

purger(instanceArcs)

Ajouter arc à arcTabou
Ajouter nœudCourant à noeudTabou
```

fin

Algorithme 2 : déplacement de la fourmi vers un nouveau nœud.

La fonction suivante filtre la liste des liens sortants du nœud courant. Les arcs déjà suivis ainsi que les nœuds déjà explorés sont systématiquement ignorés. Si un nouveau nœud ou un nouvel arc sont découverts, ils ne seront pas explorés mais sont notifiés à l'agent décideur.

filtrerListeArcs()

```
listeArcs = arcSortants(nœudCourant) - arcTabou
listeNoeuds  = voisigane(noeudCourant) - nœudTabou

pourChaque arc dans listeArcs  faire
    si arc ∉liste des arcs de la partition alors
        message :: nouveauArc(partition, nœudCourant, arc)
    sinon Ajouter arc à listeArc
```

```
finPour

pourChaque nœud de cibles(listeArcs) faire
    si nœud ∉ liste des nœuds de la partition alors
        message :: nouveauNoeud(cible(nœud))
    sinon ajouter nœud à listeNœuds
finPour

instancesArcs = listeArcs
instancesNoeuds = instancesNoeuds ∪ listeNœuds
reOrdonner(instanceArcs)

selectionner arc dans instanceArcs pour la transition
suivreLien(arc)
fin
```

Algorithme 3 : exploration du voisinage

IV.1.2.2. La visibilité du voisinage

Dans cette section, nous proposons un modèle de dépôt d'information dynamique pour le marquage des arcs visités par les fourmis exploratrices.

Dans une colonie de fourmis, le choix d'un arc dépend de deux facteurs :

- La **visibilité** qui désigne la longueur de l'arc. Elle mesure l'importance de l'arc dans l'ensemble du système calculé sur la base des explorations précédentes.
- La **phéromone** déposée sur l'arc. Elle mesure l'importance de l'arc dans le contexte précis du processus d'exploration courant.

La visibilité globale

Notée **rang**(*nœud*), elle désigne l'importance du document dans le corpus par rapport à l'ontologie de référence. Lors de l'ajout d'un document à la partition, cette valeur est initialisée au rang du document attribué par un moteur de recherche classique.

La visibilité locale

Notée **poids**(*nœud*, *contexte*), elle désigne l'importance relative du document par rapport à la requête : la pertinence locale. Elle est calculée sur la base de pertinence du document par rapport à chaque terme de la requête comme décrite dans la section III.3.2.3 page 67.

Ainsi, la fonction **visibilite***(noeud, contexte)* décrit l'importance du *nœud* dans un *contexte.* En effet, nous savons qu'un document n'a pas la même importance pour tous les documents qui le pointent avec un lien. Cette fonction s'écrit alors :

$$\boldsymbol{visibilite}(noeud, contexte) = \boldsymbol{evalLink}_{arc} \times \frac{\boldsymbol{poids}(noeud, contexte)}{\boldsymbol{rang}(noeud)}$$

Equation 23 : visibilité d'un nœud dans un contexte de recherche.

La visibilité prend ses valeurs dans l'intervalle [0..1].

IV.2 LES PRINCIPAUX PROCESSUS DU SYSTEME

IV.2.1 Parcours du corpus

Le problème du parcours du web se pose en deux termes : une couverture complète et exhaustive de tous les nœuds du graphe modélisant le web et une détection dans les plus brefs délais des mises à jour des nœuds ou des arcs du graphe modélisant le web.

IV.2.1.1. Exploration du voisinage

Une fourmi se déplace du nœud où elle se trouve, noté *nœudCourant*, vers un nœud du voisinage, noté *voisin,* en suivant un lien, noté **arc***(nœudCourant, voisin)*, elle dépose une quantité de phéromone sur l'arc pour marquer son passage. Les informations *arc* et *voisin* sont ajoutées à la mémoire locale de la fourmi et à la mémoire globale de la partition.

La fourmi compare la liste des arcs sortants du nœud au moment *t* avec sa liste d'arcs suivis et avec celle enregistré dans la mémoire de la partition.

Plusieurs cas de figures peuvent apparaître :

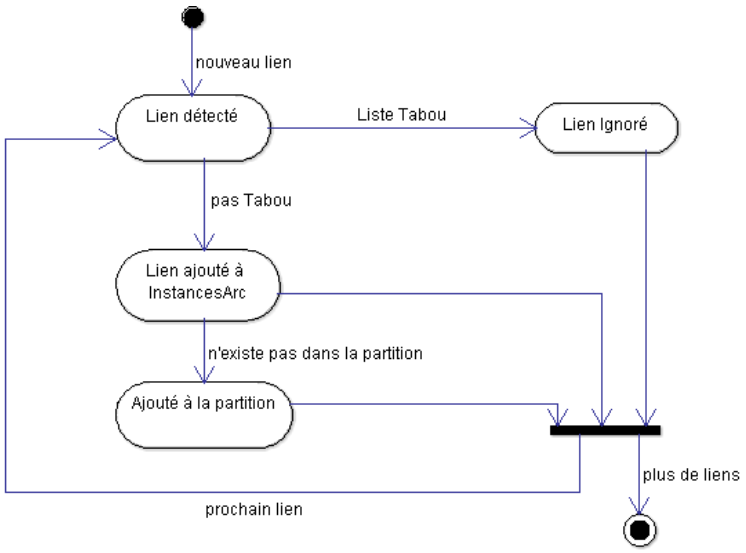


Figure 25 : exploration du voisinage d'un nœud du graphe par l'agent *fourmi*.

- le lien est ignoré s'il appartient à sa liste des liens suivis
- si le lien n'appartient pas mais appartient à la liste des arcs de la partition, alors le nœud destination de l'arc est ajouté à la liste des liens à visiter
- si l'arc est nouvellement introduit dans le graphe, alors il est ajouté à liste des arcs de la partition.
- Si le nœud destination de l'arc est différent de celui existant au niveau de la mémoire de la partition ou qu'il n'existe pas, alors ajouter le nœud à la liste des nœuds à indexer.

Après avoir analysé tous les liens, la liste des documents à visiter est réordonnée selon la fonction fitness, la fourmi suit alors le meilleur arc dans sa liste d'arcs à visiter. La fourmi s'arrête quand sa liste d'arcs à suivre est vide.

IV.2.2 Dépôt et évaporation de la phéromone

IV.2.2.1. Dépôt de phéromone

La quantité *qPh* de phéromone déposée par la fourmi sur un arc lorsqu'elle se déplace d'un nœud vers un autre s'écrit :

$$\boldsymbol{qPh} = \boldsymbol{maxPh} \times \boldsymbol{visibilite}(noeud, contexte)$$

Equation 24 : quantité de phéromone à déposer sur un arc.

Dans le cas, utopique mais absolu, où un document encapsule à lui seul toute l'information pertinente par rapport à une requête, c'est-à-dire $evalLink_{arc}=1$ et *poids=rang*, une fourmi, déposera la quantité *maxPh* sur l'arc la menant à ce document.

La quantité de phéromone d'un arc est augmentée de *qPh* à l'instant où une fourmi traverse l'arc pour aller vers un nouveau nœud.

IV.2.2.2. Fonction d'évaporation

On définit la fonction d'évaporation de la phéromone sur les arcs comme étant le taux *Δph* de phéromone qui s'évapore à chaque itération du système.

De la définition des paramètres *Δph* et du cycle d'itération dépend grandement la qualité du parcours. Si *Δph* est trop élevé, les phéromones s'évaporent trop rapidement, le rapport poids/rang tendrait trop rapidement vers 1. On se retrouvera dans le cas d'un parcours type *pageRank* sans aucune prise en compte de la sémantique. Par contre, si *Δph* est trop faible, le système sera bloqué dans une solution locale, où, seul un nombre limité de nœuds seront explorés.

$$qPh_{arc}^{t+1} = (1 - \Delta Ph) \times qPh_{arc}^{t}$$

Equation 25 : fonction d'évaporation des phéromones.

L'algorithme s'écrit :

evaporation

pourchaque *arc* dans *InstanceArcs* ∪ *arcTabou* **faire**
pheromone(*arc*) = (1-ΔPh) × pheromone(*arc*)
finPour

fin

Algorithme 4 : application de la fonction d'évaporation des phéromones

IV.2.3 Mise à jour de l'index

IV.2.3.1. Rang d'un document par rapport à l'ontologie de référence

Le calcul du rang d'un document *D* par rapport à l'ontologie *O* revient à calculer la fonction *rsv(D, C)*, où *C* est l'ensemble des concepts de *O* comme décrit dans la section III.3.2.3.

A la réception du message **nouveauNoeud**(*noeud*) (Algorithme 3 page 77) le document *nœud* est ajouté à la liste des documents en attente de traitement.

Pour chaque nouveau document, le processus d'indexation syntaxique (décrit dans la Figure 2) est exécuté. Il permet de mesurer le poids syntaxique de chacun de ses termes.

Une première liste d'index est alors constituée :

$$index_{intial} = \{\langle t_i, d, rsv\rangle \ / \ \forall t_i \in d\}$$

Equation 26 : Modélisation de l'index initial.

Cette liste est alors comparée à la liste des concepts de l'ontologie. Le but est évidemment de n'indexer le document que par rapport aux concepts de l'ontologie de référence :

$$index_{sem} = \{\langle c_j, d, rsv\rangle / \forall t_i \in d \wedge \forall c_j \in C, sim(c_i, t_i) \geq minsim\}$$

Equation 27 : modélisation d'un index sémantique.

Ceci assure une évaluation sémantique du document par rapport à l'ontologie de référence. Cette évaluation sera optimisée au fur et à mesure des requêtes des utilisateurs. Enfin, l'index sémantique est inséré dans l'index global de la partition.

La figure suivante décrit ce processus

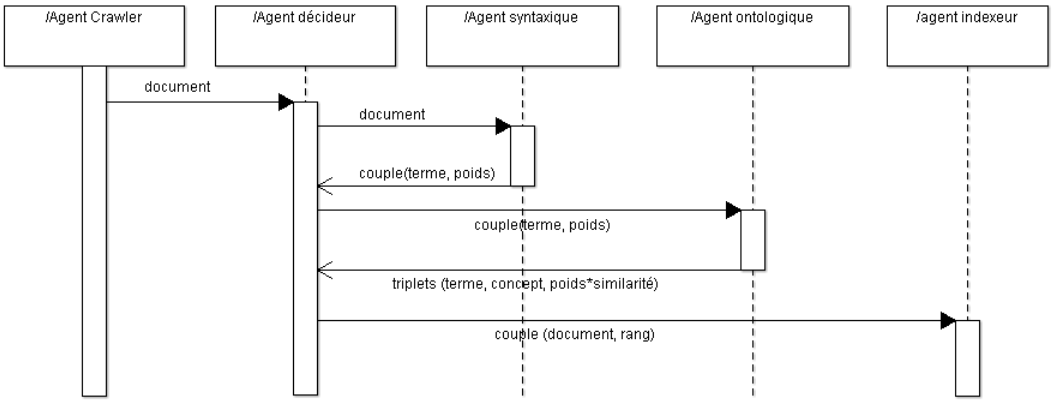


Figure 26 : mise à jour du rang d'un document en fonction de l'ontologie de référence.

L'algorithme suivant insère un document, nouvellement introduit ou mis à jour dans la base d'index du système :

insererDocument(*document*)

intialiser $index_{initial}$
pourChaque *terme* du *document* **faire**
    inserer <terme, document, rsv> dans $index_{initial}$

```
    finPour

    intialiser index_sem
    pourChaque concept de l'ontologie faire
        si sim(concept, terme)≥minSim alors
            rang(document) = rsv× sim(concept, terme)
            inserer <concept, document, rang> dans index_sem
        finSi
    finPour

    pourChaque élément de index_sem faire
        inserer <concept, document, rang> dans l'index de la partition
        inserer <partition, concept, document, rang> dans l'index du système
    finPour

    vider index_sem
    vider index_sem
fin
```

Algorithme 5 : insertion d'un nouveau document dans l'index

IV.2.3.2. Mise à jour de la pertinence

A chaque fois qu'un document est sélectionné par une fourmi dans le contexte d'une requête utilisateur, le poids du document est mis à jour en fonction de son rang et de son poids dans le contexte.

```
Actualiser(document)
    pourChaque terme de la requête faire
        poids(document, terme) =  poids(document, terme) × qPh/maxPh
        rang(document) = rang(document) × poids(document, terme)/rsv(document, terme)
    finPour
fin
```

Algorithme 6 : Actualisation du poids et du rang d'un document sur la base de sa pertinence

Ainsi, à la première itération après son insertion, le rang du document est égal à son poids pour l'unique requête dans laquelle il a figuré.

De cette manière, le poids et le rang sont mis à jour pour refléter les pertinences absolue et relative d'un document dans et en dehors d'un contexte. Il est alors possible d'utiliser l'index existant ou choisir d'effectuer une recherche, et donc une ré-indexation de la partition, pour prendre en charge de nouveaux concepts ou pour rafraîchir l'index jugé obsolète.

La fraction ($qPh/maxPh \geq 0$) permet de juger de l'importance d'un document en mesurant la quantité de phéromone déposée avec celle qui a été retrouvée quand la fourmi rebrousse chemin. S'il tend vers 0, cela signifie que le lien menant vers ce document n'a pas été utilisé fréquemment. Alors, comme pour le problème du voyageur de commerce, il sera abandonné au fur et à mesure par les autres fourmis dans ce contexte. Plus il est grand, plus en suppose que le document cible de l'arc est jugé important, car visité par d'autres fourmis.

La mise à jour du rang du document est alors fonction d'une constante, son *rsv*, et de son importance relative à chaque contexte : si un contexte augmente le poids d'un document, son

rang en sera systématiquement affecté.

IV.2.4 Enrichissement de la requête

L'enrichissement d'une requête consiste en l'ajout de nouveaux termes à la requête sur la base de leur similarité sémantique avec les termes initiaux.

Un paramètre important à considérer est la distance minimale entre les termes. Cette distance permet d'ignorer tous les termes dont la similarité sémantique est inférieure au seuil minimal. Deux approches existent :

- Le seuil est fixé par le système, il s'applique à toutes les requêtes.
- Le seuil est défini indépendamment pour chaque requête.

Un seuil dynamique de similarité a l'avantage certain de personnaliser la requête enrichie en fonction de sa sémantique propre ; mais, selon qu'il est trop élevé ou trop faible, les silences ou les bruits dans le jeu de résultats seront importants.

Les travaux menés dans le cadre de la recherche flexible d'informations (FANKAM, 2009), (HIGNETTE, 2007) et (LOISEAU, 2004) montrent la difficulté de mise en œuvre de cette distance. Pour notre part, notons *minSim* le seuil minimum de similarité autorisé.

Le processus est décrit par la figure suivante :

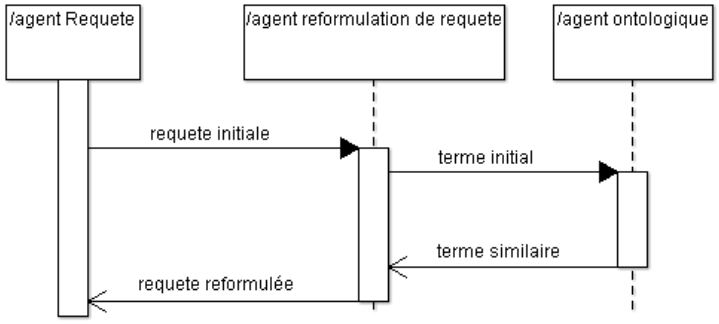


Figure 27 : processus de reformulation de requête.

Pour chaque terme, noté q_i, de la requête initiale, on sélectionne l'ensemble des concepts C de l'ontologie O, tel que : $sim(c, qi) >= minSim$. La nouvelle requête Q_{finale} s'écrit alors :

$$Q_{finale} = \{c \in C / \forall q_i \in Q, sim(c, q_i) \geq nimSim\}$$

Equation 28 : modélisation d'une requête après reformulation.

L'algorithme s'écrit alors :

enrichirRequete(Q)

```
Q_finale = Q
pourChaque concept ∈ C faire
    pourChaque terme ∈ Q faire
        si sim(concept, terme) >= minSim alors
            ajouter le couple <concept, sim> à Q_finale
```

Fin

Algorithme 7 : enrichissement de la requête avec les concepts similaires dans l'ontologie.

IV.3 LA PERENNISATION DES ONTOLOGIES

Dans cette section, nous décrivons un modèle basé sur les travaux de (FANKAM, 2009) relatifs aux bases de données d'ontologies, appelées communément « *bases ontologiques* ».

IV.3.1 Le modèle OntoDB2

En réponse aux problèmes de stockage et d'accessibilité aux données des ontologies, l'architecture OntoDB2 a été proposée par (FANKAM, 2009).

OntoDB2 supporte des constructions canoniques communes aux formalismes d'ontologie. Elle est fondée sur une architecture de type 3 qui lui confère la possibilité de faire évoluer le formalisme d'ontologie supporté.

Les motivations principales d'OntoDB2 étaient de fournir des moyens simples pour pouvoir :

- modifier aisément le formalisme d'ontologie pour pouvoir intégrer des ontologies de différents formalismes ;
- étendre le système de types de données afin de représenter de nouvelles données.

IV.3.1.1. Flexibilité du formalisme d'ontologie

OntoDB2 permet la modélisation d'un domaine donné, on définit des classes, des propriétés, des types de données, et la construction des ontologies pouvant se décomposer suivant un modèle en oignon contenant :

- Une **couche canonique** qui contient les primitives de base permettant de décrire de façon unique les informations à modéliser.
- Une **couche non canonique** constituée de constructeurs d'équivalences conceptuels qui permettent de définir différentes vues sur les classes de couche supérieure, la couche canonique.
- Une **couche linguistique** qui permet à travers les différents descripteurs textuels de décrire un document.

IV.3.1.2. Architecture d'OntoDB2

L'architecture de type 3 offre la possibilité de représenter au sein de la base de donnée le méta-modèle du formalisme d'ontologie et le formalisme d'ontologie en tant qu'instance de ce

méta-modèle. Dans cette optique, l'ontologie n'est pas figée, la disponibilité du méta-modèle et la représentation du formalisme d'ontologie permettent d'assurer l'évolution de la partie ontologie.

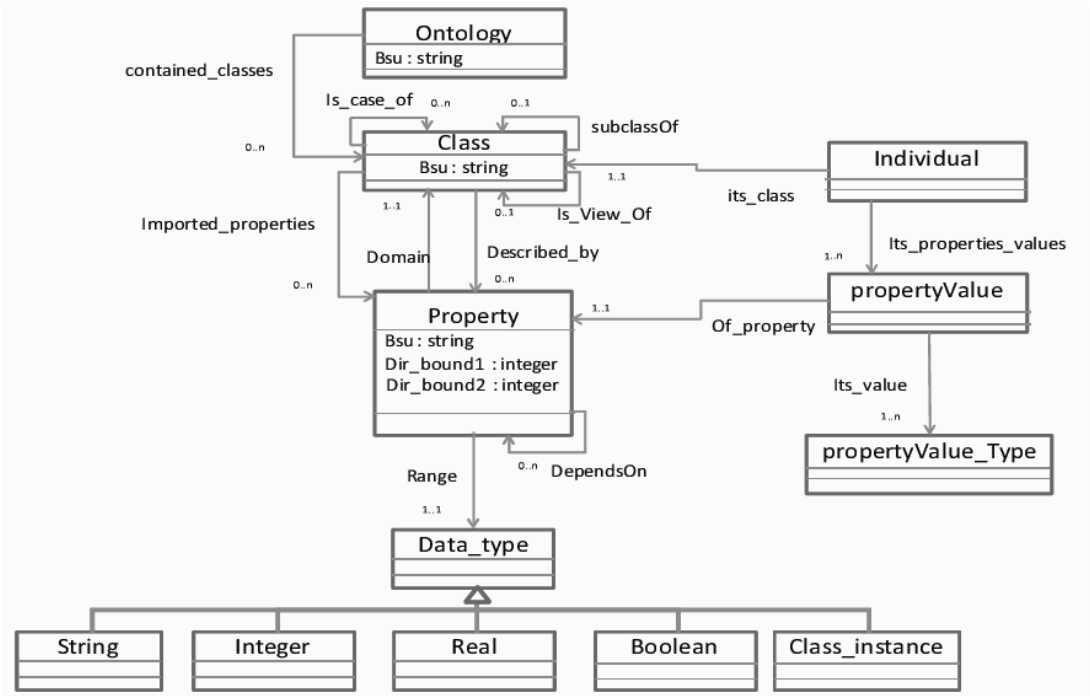


Figure 28 : schéma conceptuel des données dans OntoDB2

IV.3.2 Schéma de la base ontologique selon OntoDB2

Dans cette section, nous implémentons un schéma de la base de données ontologique. Nous étendons le modèle OntoDB2 pour prendre en charge la similarité sémantique entre les concepts d'une même ontologie.

Le diagramme de base de données est[1] :

[1] Le schéma de la base ontologique ainsi que les requêtes proposées sont écrits avec ms SQL Server 2008 R2.

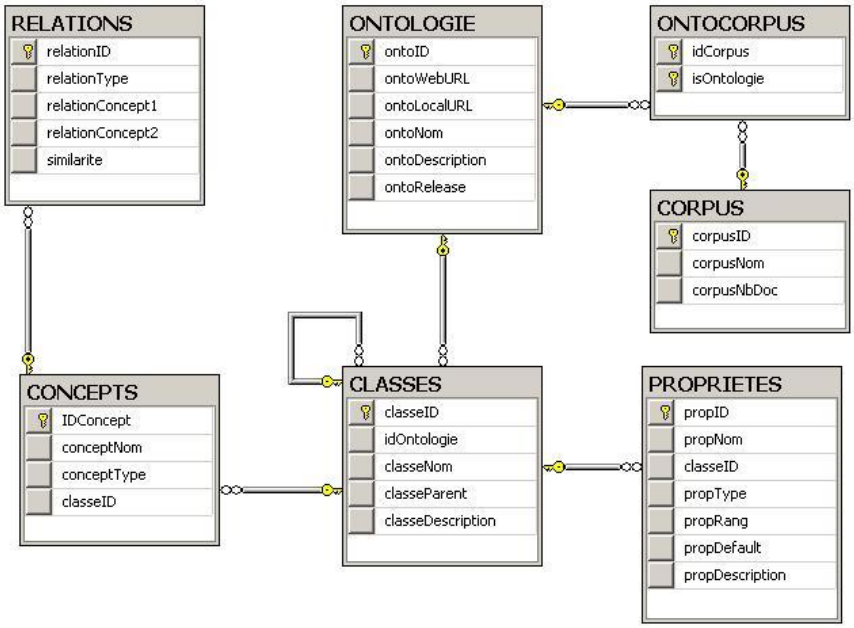


Figure 29 : Diagramme de la pérennisation des données des ontologies de références

Ainsi, à titre d'exemple, la similarité entre deux concepts c_1 et c_2 sera obtenue par la fonction SQL suivante :

```
Create fonction similarite (C1, C2)  AS
(
    Declare @result

    Select @result = similarite
    from RELATIONS
    where relationConcept1 = c1
        and relation Concept2 = c2
    return @result
)
```

Algorithme 8 : récupération de la similarité entre deux concepts depuis la base ontologique.

IV.3.3 Pérennisation de corpus

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

- Un **dictionnaire** : Le fichier dictionnaire contient tous les mots ou groupes nominaux spécifiques pouvant servir de mots-clés pour l'indexation et la recherche dans l'ensemble des fichiers à traiter. A chaque entrée du dictionnaire est associé le nombre de fois où l'entrée apparaît dans l'ensemble documentaire.
- Un **fichier de hachage** : Ce fichier contient pour chaque entrée du dictionnaire une liste décrivant dans quel fichier apparaît cette entrée. Cette méthode permet de restreindre l'étude sur les fichiers qui nous intéressent et pas les autres. A signaler que dans certains cas, la position dans le fichier est aussi stockée.
- Les **fichiers de données** : Qui représentent les documents du corpus à indexer.

Nous proposons, pour notre modèle, le schéma de base de données suivant : *

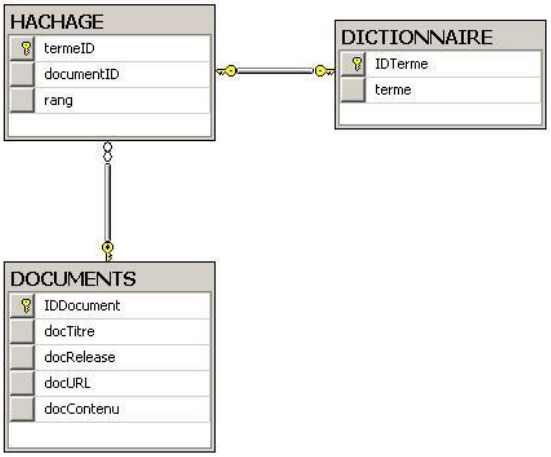


Figure 30 : schéma de la base de données de l'index

Ainsi, les documents pertinents par rapport à un terme C seraient renvoyés par la fonction SQL suivante :

```
create function pertinentDocs(terme)
as
(
    return
        select documentID , rang
        from HACHAGE
        where termeID = terme
)
```

Algorithme 9 : sélection des documents pertinents par rapport à un terme de la requête

Beaucoup de travaux traitent de la création, de la gestion et de l'optimisation des index de corpus. D'autres techniques d'indexation du corpus existent. Elles cherchent toutes à optimiser la table de hachage pour accélérer l'accès aux documents notamment quand la taille du corpus est très importante et que le dictionnaire est ouvert. (DAHAK, 2006) dresse un bilan complet de ces techniques.

IV.4 CONCLUSION

Dans ce chapitre, nous avons décrit et modélisé les principaux processus d'un système distribué de recherche sémantique d'information. Bien qu'il soit impossible d'être exhaustif, nous avons tenté de nous focaliser sur les principales fonctions du système. Une attention particulière a été portée à la définition des agents du système. Après avoir proposé une architecture multicouches du système, nous avons identifié les agents de chaque couche, leur rôle et leurs limites. Nous avons également, pour quelques processus importants, montré comment ces agents communiquent et collaborent.

Nous avons également proposé des algorithmes pour les principales fonctions du

système, notamment les fonctions d'exploration et de mise à jour du corpus ainsi que les fonctions de dépôt et d'évaporation des phéromones.

Enfin, nous avons proposé un modèle simple, mais qui mérite d'être détaillé, pour le stockage des données. Ce modèle de pérennisation des données est basé sur les bases de données relationnelles mais peut faire l'objet d'une modélisation objet (SGBDRO) ou de bases de données XML, puisque c'est la tendance actuellement.

Conclusion générale

La connaissance et le bois de construction ne devraient pas être beaucoup employés jusqu'à ce qu'ils soient chevronnés.
Oliver Wendell Holmes

Personne ne se doute, en entrant quelques termes dans un champ de saisie, de la complexité du processus qui retournera les résultats de la recherche d'informations.

Dans ce mémoire, nous avons vu à quel point la recherche d'informations sur le web est un processus compliqué et difficile à mettre en œuvre. Du choix de telle ou telle stratégie d'indexation, d'exploration ou d'appariement dépend la qualité des résultats.

Dans le premier chapitre, nous avons établi un état de l'art décrivant les composantes essentielles d'un moteur de recherche : les méthodes de formulation d'une requête, les algorithmes de détection de pertinence et les principes de visualisation utilisés pour mettre en valeur les résultats. Ces éléments nous permettent de fonder une nouvelle stratégie de recherche et d'identifier les principes fondateurs de notre outil de recherche d'informations.

Dans le chapitre II, nous avons présenté le processus de recherche d'informations du point de vue de la sémantique. L'extraction des termes et leur pondération, la similarité entre les concepts d'une même ontologie et l'appariement entre les termes de la requête utilisateur et les documents. Nous avons montré que ces techniques peuvent accroitre, en simultané, les taux de *rappel* et de *précision*, optimisant, un peu plus, les résultats des moteurs de recherche. Cela nous a permis, en particulier, d'identifier les points sur lesquels se concentre ce travail : la représentation de l'information, la reformulation de la requête et la stratégie de recherche.

Nous avons ensuite, dans le chapitre III, proposé un modèle de recherche sémantique d'informations basé sur les méthodes heuristiques de recherche locale, inspiré du

comportement des fourmis fourrageuses. Ce modèle nous permet d'appréhender la complexité inhérente à la recherche classique d'informations sous un angle complètement différent.

Les agents, organisés en sociétés et dotés de comportements simples, sont capables de résoudre pratiquement l'ensemble des problèmes que pose un système de recherche d'informations. Nous avons décrit leur rôle, quand ils sont modélisés comme des fourmis fourrageuses, dans l'exploration du graphe du web. En effet, l'information déposée sur les arcs, la phéromone, nous permet de juger de l'importance d'un document dans le graphe. La fonction d'évaporation quant à elle, permet la prise en compte de l'importance relative du document dans un contexte de recherche. La combinaison de ces deux évaluations retourne la pertinence réelle d'un document par rapport à une ontologie de référence.

Enfin, dans le chapitre IV, nous avons essayé de définir un cadre général à l'implémentation d'une telle approche. Nous avons décrit comment le graphe du web est exploré, comment l'information est générée et partagée par l'ensemble de la communauté d'agents. Une mise en œuvre du codage du problème de recherche d'informations est alors proposée. Les fonctions de dépôt, d'évaporation de la phéromone sont formalisées ainsi que la fonction fitness, permettant de calculer la pertinence des documents. Nous avons ensuite proposé une modélisation du stockage des données ontologiques, à l'aide du modèle OntoDB2 proposé par (FANKAM, 2009), et de l'index avec le modèle relationnel des bases de données.

Perspectives

Il y a toutes sortes de gris. Il y a le gris plein de rose qui est le reflet des deux Trianons. Il y a le gris bleu qui est un regret du ciel. Le gris beige couleur de la terre après la herse. Le gris du noir au blanc dont se patinent les marbres. Mais il y a un gris sale, un gris terrible, un gris jaune tirant sur le vert, un gris pareil à la poix, un enduit sans transparence, étouffant, même s'il est clair, un gris destin, un gris sans pardon, …
Aurélien. Aragon, 1944.

On ne pouvait se douter, avant de lire Aragon, qu'il y avait autant de gris. Le concept « *gris* » est devenu une information incertaine, imprécise et floue.

Là est tout l'enjeu des moteurs de recherche. Prendre en compte toutes les nuances qu'un concept peut avoir dans les différents documents. En effet, lorsque les sources d'information se multiplient, il n'est pas rare que le vocabulaire utilisé pour ces termes soit hétérogène, et souvent incompatible ou contradictoire entre les différentes sources. Ce problème d'hétérogénéité se pose aussi lors de la formulation des requêtes, puisque l'utilisateur doit connaître a priori le vocabulaire utilisé dans la représentation de l'information.

Un outil issu de la théorie des possibilités, appelé « *filtrage flou* » se prête particulièrement bien à cette application, que ce soit dans le domaine des bases de données ou de la recherche documentaire. Il permet en effet de représenter dans quelle mesure il est possible et certain qu'une information satisfasse un besoin. Ces besoins peuvent de plus être exprimés de manière flexible, c'est à dire en tenant compte des préférences de l'utilisateur.

Nous proposons, dans ce cadre, d'utiliser et d'étendre le filtrage flou pour définir une technique de représentation, de recherche, de classification des documents du web qui prendrait en compte le caractère flou, incertain et imprécis des termes sensés décrire les concepts et des documents, support de l'information. Ainsi, la correspondance entre les termes

de la requête et les données ne nécessitera plus une identité parfaite, mais découlera d'un processus d'appariement graduel et qualitatif.

Au niveau ontologique, la similarité entre les concepts sera alors redéfinie pour représenter au mieux les relations sémantiques ambigües propres au langage naturel, seul utilisé par l'utilisateur, définissant ainsi des ontologies possibilistes de concepts.

De plus, ce modèle permettra l'élaboration de requêtes complexes pouvant représenter les préférences de l'utilisateur en prenant en compte la priorité entre les différents paramètres entrant dans l'expression de ses besoins dans le but de fournir des résultats répondant pleinement à ses attentes.

Bibliographie

ABROUK, L. (2006). *Annotation de documents par le contexte de citation basée sur une ontologie.* Thèse de doctorat, Université Montpellier II.

AUSSENAC-GILLES, N., HERNANDEZ, N., & BAZIZ, M. (2006). *Ontologies pour la recherche d'information, importance de la dimension terminologique.*

BAZIZ, M. (2005). *Indexation conceptuelle guidée par les ontologies pour la recherche d'information.* Thèse de doctorat, Université Paul Sabatier, Toulouse.

BAZIZ, M., BOUGHANEM, M., & AUSSENAC-GILLES, N. (2005). Conceptual Indexing Approach for the TREC Robust Task. *TREC 2005 Robust Task.*

BAZIZ, M., BOUGHANEM, M., & AUSSENAC-GILLES, N. (2005). Conceptual Indexing Approach for the TREC Robust Task. *TREC 2005 Robust Task.*

BENNOUAS, T. (2005). *Modélisation du parcours du Web et calcul des communautés par émergence.* Thèse de doctorat, MODÉLISATION DE PARCOURS DU WEB.

BENOIT, A. (2006). *Internet et le Web (Note de cours).* ENS Lyon.

BOUGHANEM, M., LOISEAU, Y., & PRADE, H. (2005). Improving Document Ranking in Information Retrieval Using Ordered Weighted Aggregation and Leximin Refinement. *EUSFLAT - LFA.*

CAO, T. D. (2006). *Exploitation du Web Sémantique pour la veille technologique.* Thèse de doctorat, Université de Nice .

CHAMPLAUX, Y. (2009). *Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information.* Thèse de Doctorat, Université de Toulouse.

DAHAK, F. (2006). *Indexation des documents Semi-Structurés :Proposition d'une approche basée sur le fichier inversé et le "Trie".* INI Oued Smar.

DESMONTILS, C., JACQUIN, C., & MORIN, E. (2002). Indexation sémantique de documents sur le Web:application aux ressources humaines.

ELAYEB, B. (2009). *SARIPOD: Système multi-Agent de Recherche Intelligente POssibiliste de Documents Web.* Thèse de doctorat, Université de tooulouse.

FAESSEL, N. (2008). Indexation de blocs extraits de pages Web en utilisant le rendu visuel. *CORIA - Conférence en Recherche d'Information et Applications.*

FANKAM, C. (2009). *OntoDB2 : un système flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données Base Ontologique pour le Web sémantique et les données.* Ecole nationale supérieure de mécanique et d'aéronautique, Lyon.

GERY, M. (2002). *Indexation et interrogationde chemins de lecture en contexte pour la Recherche d'Information Structurée sur le Web.* Thèse de doctorat, Université Joseph FOURIER, Grenoble.

GOMEZ DA SILVA, A. (2009). *Analyse des données évolutives : application aux données d'usage du Web.* Thèse de doctorat, Université Paris IX Dauphine, PAris.

HARRATHI, F. (2009). *Extraction de concepts et de relations entre concepts à partir des documents multilingues : Approche statistique et ontologique.* Thèse de doctorat, Institut National des Sciences Appliquées, Lyon.

HERNANDEZ, N. (2006). *Ontologies de domaine pour la modélisation du contexte en recherche d'informations.* Thèse de Doctorat, Université Paul Sabatier, Toulouse.

HIGNETTE, G. (2007). *ANNOTATION SEMANTIQUE FLOUE DE TABLEAUX GUIDEE PAR UNE ONTOLOGIE.* Thèse de doctorat, Institut des Sciences et Industries du Vivant et de l'Environnement Agro Paris Tech.

HOLI, M., & HYVÖNEN, E. (2005). *Modeling Degrees of Conceptual Overlap in Semantic Web Ontologies.* Helsinki Institute for Information Technology, Helsinki.

HUYNH-KIM-BANG, B. (2009). *Indexation de documents péfagogiques : Fusion des approches du Web sémantique et du Web Participatif.* Doctorat de l'université Henri Poincaré, Nancy.

KARBASI, S. (2007). *Pondération des termes en recherche d'informations : Modèle de pondération basé sur le rang des termes dans les documents.* thèse de doctorat, Université Paul Sabatier - Toulouse III.

KLEINBERG, J. (1998). Authoritative sources in a hyperlinked environment. *Journal of the ACM* (pp. 604-632). ACM Press.

KLEINBERG, J. (2000). The small-world phenomenon: an algorithm perspective. *Proceedings of the thirty-second annual ACM symposium on Theory of computing* (pp. 163-167). ACM Press.

LAURENT, A. (2009). *Fouille de données complexes et logique floue : Extraction de motifs à partir de bases de données multidimensionnelles.* Habilitation à diriger les recherches, Université de Montpellier 2.

LOISEAU, Y. (2004). *Recherche flexible d'informations par filtrage flou qualitattif.* Thèse de doctorat, Université Paul SABATIER, Toulouse.

MALIK, M. M. (2004). Le rôle de la logique floue dans le web sémantique.

MERCIER, S. (2012). *Contrôle du partage de l'autorité dans un système d'agents hétérogènes.* Thèse de

doctorat, Université de Toulouse.

PAGE, L., & BRIN, S. (1998). The anatomy of a large-scale hypertextual web search. *Proceedings of the seventh international conference on World Wide* , pp. 107-117.

PICAROUGNE, F. (2004). *Recherche d'information sur Internet par algorithmes évolutionaires.* Thèse de doctorat, Université François Rabelais, Tours.

PRADE, H., BAZIZ, M., BOUGHANEM, M., & PASI, G. (2005). A fuzzy set approach to concept-based information retrieval. *EUSFLAT - LFA.*

PRUSKI, C. (2009). *Une approche adaptative pour la recherche d'information sur le Web.* Thèse de doctorat, Université du Luxembourg et de l'université Paris-Sud.

SABOU, M., LOPEZ, V., & MOTTA, E. (2006). *Ontology Selection for the Real Semantic Web:.* The Open University, Milton Keynes, Knowledge Media Institute (KMi) & Centre for Research in Computing.

SAUVAGNAT, K. (2005). *Modèle flexible pour la Recherche d'Information dans des corpus de documents semi-structuées.* Thèse de doctorat, Université Paul Sabatier, Toulouse.

SIMONIAN, S., & ENEAU, J. (2011). L'usage de l'hypertexte : une dynamique favorisant les processus métacognitifs. *Environnements Informatiques pour l'Apprentissage Humain* .

SUR, F. (1998). *PRésentation de la Logique Floue.* Mémoire de DEA Mathématique, Ecole Normale Supérieure , Cachan.

TAMBELLINI, C. (2007). *Un système de recherche d'information adapté aux données incertaines : adaptation du modèle de langue.* DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER, Grenoble.

VERDIN, D. (2006). *Intégration et recherche de données environnementales dans un système d'information.* Thèse de doctorat Mention Mathèmatique-Physique-Informatique, Ecole nationale supérieure agronomique de Rennes.

VIVIAN, R., & DINET, J. (1998). RCI WEB : un système collaboratif de recherche d'information centré utilisateur. *Revue des Interactions Humaines Médiatisées , 9* (2).

YEN, J., & WIDYANTORO, D. H. (2001). *A Fuzzy Ontology-based Abstract Search Engine and Its User Studies.*

ZARGAYOUNA, H. (2005). *Indexation sémantique de documents XML.* Thèse de doctorat, Uuniversité Paris XI Orsay.

Annexe I : Les modèles-piliers de la Recherche d'Informations

L'efficience d'un système de recherche d'informations réside dans sa capacité à établir la pertinence d'un document vis-à-vis d'une requête. Pour parvenir à cet objectif, il s'appuie sur un modèle de recherche d'informations, dont le principal rôle est de proposer un formalisme mathématique pour représenter cette notion de pertinence. D'une manière formelle, un modèle de recherche d'informations peut être défini par le quadruplet suivant :

> {d, q, F, RSV(d,q)} (DESMONTILS, JACQUIN, & MORIN, 2002) où :
> d et q correspondent respectivement à la représentation d'un document et d'une requête de la collection (obtenue à la suite de l'indexation),
> F est le formalisme d'expression interne des représentations de d et q et leurs relations,
> RSV (d; q) est la fonction de correspondance.

De nombreux modèles ont été proposés en recherche d'informations[1]. En fonction du cadre théorique sur lequel ils s'appuient, ils sont généralement regroupés autour des trois familles suivantes : **les modèles ensemblistes** considèrent le processus de recherche comme une succession d'opérations à effectuer sur des ensembles de mots contenus dans les documents ; **les modèles algébriques** au sein desquels la pertinence d'un document par rapport à une requête est envisagée à partir de mesures de distance dans un espace vectoriel, ; **les modèles probabilistes** qui représentent la recherche d'informations comme un processus incertain et imprécis où la notion de pertinence peut être vue comme une probabilité de pertinence.

La Figure 31 présente une taxonomie, non exhaustive, des différents modèles de recherche d'informations.

[1] (ELAYEB, 2009) et (SAUVAGNAT, 2005) font un état détaillé des principaux modèles de recherche d'informations

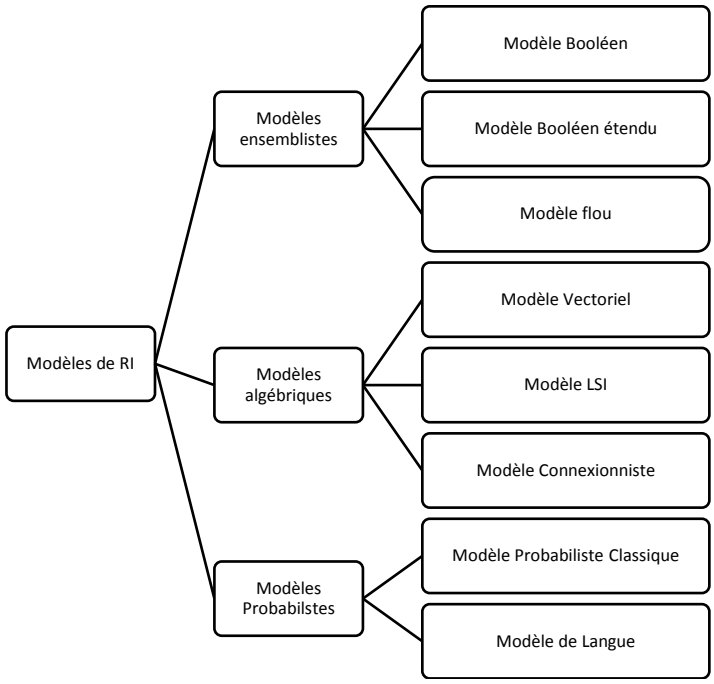


Figure 31 : Taxonomie des modèles de représentation des RIs.

I.1. LES MODELES ENSEMBLISTES

I.1.1. Le modèle booléen

Le modèle booléen a été introduit par Salton et McGill, 1983) ; il se caractérise par la simplicité et la rapidité de mise en œuvre en utilisant une liste de mots clés combinés à des opérateurs logiques

Un document <d> est représenté par son ensemble de termes t_i et une requête $q = f(t_i)$, où f est une expression logique des termes t_i. Le document <d> ne correspond à une requête que si d => q est vraie. Cette correspondance, notée RSV(d,q) est déterminée comme suit :

$$\begin{aligned}
&RSV(d, ti) = ai \\
&RSV(d, q1 \wedge q2) = 1 \text{ ssi } RSV(d, q1) = 1 \text{ et } RSV(d, q2) = 1 \\
&RSV(d, q1 \vee q2) = 1 \text{ ssi } RSV(d, q1) = 1 \text{ ou } RSV(d, q2) = 1 \\
&RSV(d, \neg q1) = 1 \text{ ssi } RSV(d, q1) = 0
\end{aligned}$$

La principale faiblesse de ce modèle est de ne pas établir de relation d'ordre de pertinence entre les différents documents. Il est de ce fait généralement utilisé pour la présélection de documents.

I.1.2. Le modèle booléen étendu

Introduit par Salton ,1983), ce modèle est une extension du modèle booléen classique en ceci qu'il tient compte de la pondération des termes dans le corpus. Il permet donc un ordonnancement des documents retrouvés par les systèmes de recherche d'informations.

Un document <d> est représenté par le couple (t_i, w_i) où :

t_i : le ieme terme du document

w_i : poids du terme défini par son occurrence dans le document ou par certains caractères typographiques.

$RSV(d, t_i) = a_i$

$RSV(d, q_1 \wedge q_2) = RSV(d, q_1) \times RSV(d, q_2)$

$RSV(d, q_1 \vee q_2) = RSV(d, q_1) + RSV(d, q_2) - RSV(d, q_1) \times RSV(d, q_2)$

$RSV(d, \neg q_1) = 1 - RSV(d, q_1)$

Cependant, $RSV(d, q \wedge \neg q) \equiv 0$ et $RSV(d, q \vee \neg q) \equiv 1$ ne sont toujours pas vérifiées mais satisfaisantes. Une autre formulation est introduite par Robertson, 1984) :

$RSV(d, t_i) = a_i$

$$RSV(d, q_1 \wedge q_2) = 1 - \sqrt{\frac{\left(1 - RSV(d, q_1)\right)^2 + \left(1 - RSV(d, q_2)\right)^2}{2}}$$

$$RSV(d, q_1 \vee q_2) = 1 - \sqrt{\frac{\left(RSV(d, q_1)\right)^2 + \left(RSV(d, q_2)\right)^2}{2}}$$

$RSV(d, \neg q_1) = 1 - RSV(d, q_1)$

I.1.3. Le modèle flou

Ce modèle est construit sur l'hypothèse que la correspondance d'un document avec les termes d'une requête est approximative. Ceci peut être modélisé en considérant que chaque terme de la requête définit un ensemble flou et que chaque document possède un degré d'appartenance à cet ensemble. Le degré d'appartenance (RSV) est utilisé pour représenter l'incertitude ou l'ambiguïté. Les trois opérations les plus couramment effectuées sur des ensembles flous (le complément, l'union et l'intersection) sont ainsi définies :

Dj= {…, (ti,ai), …} où : ai est le degré d'appartenance du terme ti au document Dj.

$RSV(D_j, q_1 \wedge q_2) = \min(RSV(D_j, q_1), RSV(D_j, q_2))$,

$RSV(D_j, q_1 \vee q_2) = \max(RSV(D_j, q_1), RSV(D_j, q_2))$,

$RSV(D_j, \neg q_i) = 1-(RSV(D_j, q_i)$

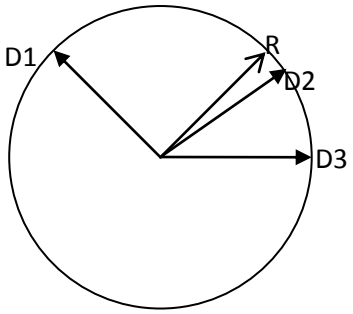
Une autre application du modèle flou est la construction et l'utilisation d'un thesaurus ou d'une ontologie pour étendre la requête. Le modèle peut enfin être utilisé pour former une nouvelle requête dans un cycle de reformulation de la requête.

I.2. LES MODELES ALGEBRIQUES

I.2.1. Le modèle vectoriel

Dans ce modèle, les requêtes et les documents sont représentés dans l'espace vectoriel engendré par les termes d'indexation Salton, 1983) dans le cadre du projet SMART. L'espace est

de dimension N (N étant le nombre de termes d'indexation de la collection de documents).



Trois documents D_i et une requête R dans le modèle d'espace vectoriel

D2 est le document le plus pertinent par rapport à la requête R

Figure 32 : Représentation des documents et des requêtes dans le modèle vectoriel

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

$$La\ mesure\ cosine\ :\ RSV(Q,D_j) = \frac{\sum_{i=1}^{N} q_i d_{ij}}{\left[\sum q_i^2\right]^{1/2} \times \left[\sum d_{ij}^2\right]^{1/2}}$$

$$Le\ produit\ scalaire\ :\ RSV(Q,D_j) = \sum_{i=1}^{N} q_i d_{ij}$$

I.2.2. Latent Semantic Indexing Model (LSI)

L'idée principale du modèle LSI (Latent Semantic Model) est que les idées dans un texte sont plus reliées aux concepts décrits par elles que les termes de l'index utilisés pour leur description. Ainsi, la correspondance entre un document et une requête donnée devrait être basée sur la correspondance des concepts plutôt que sur la correspondance des termes de l'index. L'objectif fondamental est d'aboutir à une représentation conceptuelle des documents. Ainsi, les documents qui partagent des termes co-occurrents ont des représentations proches, ce qui permet de sélectionner un document même s'il ne contient aucun mot de la requête. Pour ce faire, on se place dans un espace de moindre dimension associé aux concepts. Les vecteurs des termes de l'index sont convertis dans cet espace, et le modèle affirme que la recherche dans l'espace réduit donne de meilleurs résultats que la recherche dans l'espace des termes de l'index.

La similarité entre le document et la requête est calculée de la même façon. Le principal inconvénient de cette méthode est qu'elle n'est pas souple pour certains types d'application dont le filtrage. En effet, la performance et la stabilité du système dépendent largement de la quantité et de la qualité des données traitées. Si le nombre de documents est faible, le processus devient erroné.

I.2.3. Le modèle connexionniste ou neuronal

Sous le terme réseaux de neurones, on regroupe un certain nombre de modèles dont

l'objectif est d'imiter quelques fonctions du cerveau humain en reproduisant certaines de ses structures de base. Le fonctionnement du réseau se fait par propagation de signaux de la couche d'entrée vers la couche de sortie. Chaque neurone de la couche d'entrée reçoit une valeur d'activation, calcule une valeur de sortie et la transmet vers les neurones qui lui sont reliés dans la couche suivante. Ce processus se reproduit jusqu'à arriver à la couche de sortie, les valeurs de sorties dans la couche de sortie servant de critère de décision.

La notion de réseau en général est très intéressante pour représenter les différentes relations et associations qui existent entre les termes et les documents. Ceci est d'autant plus vrai quand ces relations sont valuées. Différentes relations peuvent exister entre les termes et les documents :

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

Une représentation sous forme de réseau permet de mettre en évidence l'importance des relations et des interactions qui peuvent exister entre les différents éléments d'un système documentaire. Il n'existe pas de représentation unique d'un réseau de neurones pour la recherche d'informations, c'est au constructeur du système de la définir (nombre de couches, nombre de neurones par couche, fonction de sortie de chaque neurone, liens entre les neurones et poids des neurones, couche d'entrée et couche de sortie).

I.3. LES MODELES PROBABILISTES

I.3.1. Le modèle probabiliste

Le modèle probabiliste aborde le problème de la recherche d'informations dans un cadre probabiliste. Le premier modèle probabiliste a été proposé par Maron et Kuhns au début des années 1960 (SAUVAGNAT, 2005). Le principe de base, résumé par « *principe de classement probabiliste* », consiste à présenter les résultats de recherche d'un système de recherche d'informations dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête.

Pour chaque requête formulée, il existe un ensemble de documents contenant exactement le résultat recherché : « *la réponse idéale* ». Cependant, les propriétés de cet ensemble ne sont pas connues au moment de la requête, il faut d'abord deviner ce qu'il pourrait être. Cette première tentative permet de générer une première description probabiliste de l'ensemble, qui est ensuite utilisée pour retrouver un premier ensemble de documents. Il faut ensuite une interaction avec

l'utilisateur pour améliorer la description probabiliste de l'échantillon représentant cet ensemble idéal.

Le processus de recherche se traduit par calcul, de proche en proche, du degré ou probabilité de pertinence d'un document relativement à une requête. Pour ce faire, le processus de décision complète le procédé d'indexation probabiliste en utilisant deux probabilités conditionnelles :

- $P(w_{ij}/Pert)$: probabilité que le terme t_i occurre dans le document D_j sachant que ce dernier est pertinent pour la requête.
- $P(w_{ij}/NonPert)$: que le terme t_i occurre dans le document D_j sachant que ce dernier n'est pas pertinent pour la requête.

Le calcul d'occurrences des termes d'indexation dans les documents est basé sur l'application d'une loi de distribution sur un échantillon représentatif de documents d'apprentissage. En posant les hypothèses suivantes :

- la distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents,
- les variables « *document pertinent* » et « *document non pertinent* » sont indépendantes.

La fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document D, notée P(Pert/D) :

$$RSV\left(\frac{Pert}{D}\right) = \sum_{i=1}^{j} \frac{P(w_{ij}/Pert)}{P(w_{ij}/NonPert)}$$

Parmi les applications du modèle probabiliste, citons le modèle *2-Poisson* développé par Robertson et Walker ou bien encore le moteur de recherche *Okapi*

I.3.2. le modèle de langue

Les modèles de langue, qui calculent sur des corpus d'apprentissage des probabilités de succession de mots, sont habituellement utilisés en reconnaissance de la parole et en traduction. En recherche d'informations, leur principe est le suivant :

- un document est considéré comme un échantillon d'un langage particulier. Un modèle de langue (noté Md) est donc entraîné pour chaque document de la collection ;
- la requête (notée q) est vue comme un processus de génération d'une phrase dans les différents langages des documents (i.e. leur modèle de langue) ;
- la fonction de correspondance entre une requête et un document est calculée en estimant

la probabilité que la requête q puisse être générée par le modèle de langue d'un document M_d ;

- les documents retournés à l'utilisateur sont alors classés dans l'ordre décroissant de la probabilité $P(q_j M_d)$.

La particularité de ces modèles, par rapport aux approches probabilistes classiques, est qu'ils ne cherchent pas à modéliser la notion de pertinence. Dans les modèles de langue, la pertinence d'un document vis-à-vis d'une requête est considérée uniquement comme la probabilité que cette dernière puisse être générée par le modèle de langue d'un document.

La probabilité qu'un mot q d'une requête soit généré par le modèle de langue d'un document d est estimée par comptage, et revient généralement (dans le cas d'un modèle unigramme) à calculer la fréquence des termes de q dans d.

Plusieurs raisons conduisent à penser que ces modèles, basés sur un cadre mathématique simple, sont très prometteurs. Ils présentent notamment l'avantage, contrairement aux modèles algébriques, d'intégrer au sein d'un seul modèle la phase d'indexation et de recherche, réduisant ainsi les calculs. Ils introduisent également une nouvelle fonction de classement basée sur la génération de la requête. De plus, ils ne nécessitent, à la différence des modèles probabilistes, aucun jugement de pertinence pour fonctionner. La difficulté d'incorporer les stratégies de relevance feedback et les préférences de l'utilisateur constitue toutefois l'un des points faibles de ces modèles (SAUVAGNAT, 2005).

Annexe II : Web sémantique et ontologies

Le Web actuel est un web tourné vers l'utilisateur : les contenus, les interfaces et les outils disponibles ont été conçus pour être lisibles et utilisables par les hommes. Cette approche, bien qu'ayant été utile et efficace un temps, ne peut plus répondre aux nouveaux besoins d'échange, de partage, de réutilisation de l'information devenus nécessaires. Cet état de faits est dû non seulement au manque de normes et sémantiques formelles des informations, mais aussi au fait que l'information contenue actuellement sur le web n'est généralement pas interprétable et exploitable « *intelligemment* » par des machines. Il s'agit donc de transformer le web actuel, centré représentation, vers un web futur où les « *connaissances, et non plus les informations, sont reliées pour permettre aux machines d'accéder aux sources d'informations et aux services Web : Le web sémantique* ».

L'expression de Web Sémantique, énoncée par Berners-Lee et al., 2001) fait d'abord référence à la vision du Web du futur comme un vaste espace d'échange de ressources supportant la collaboration entre humains et machines en vue d'exploiter plus efficacement de grands volumes d'informations et de services variés disponibles sur le Web (ZARGAYOUNA, 2005).

L'objectif du Web sémantique est de rendre explicite le contenu sémantique des ressources dans le Web (documents, pages web, services, etc.). Les machines et les agents logiciels pourraient « *comprendre* » les contenus décrits dans les ressources et faciliter les tâches de traitement des informations de façon plus automatique et plus efficace.

II.1. LE WEB SEMANTIQUE

Le web sémantique définit des ensembles de relations entre caractères ou groupes de caractères et leur signification indépendamment de la façon dont ils sont utilisés ou interprétés. Si en linguistique, la sémantique porte sur l'étude des sens à partir d'une combinaison de mots,

en IA, elle porte sur la capacité d'un réseau de termes à représenter de la manière la plus humaine possible des relations entre objets, idées ou situations.

II.1.1. Architecture du Web sémantique

La vision courante du Web sémantique proposée par Berners-Lee et al., 2001) peut être représentée dans une architecture en plusieurs couches différentes(CAO, 2006) :

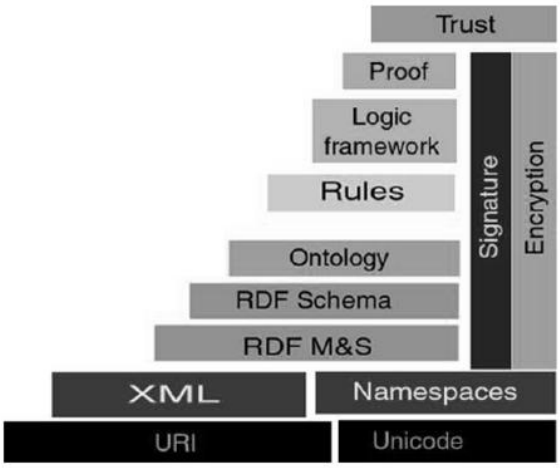


Figure 33 : Architecture du Web sémantique

La couche URI et Unicode

Elles assurent l'interopérabilité syntaxique. La notion d'**URI** fournit un adressage standard universel permettant d'identifier les ressources tandis que **Unicode** est un encodage textuel universel pour échanger des symboles.

La couche XML et namespace

XML fournit une syntaxe pour décrire la structure du document, créer et manipuler des instances des documents. Il utilise l'espace de nommage (**namespace**) afin d'identifier les noms des balises (tags) utilisées dans les documents *XML*. Le schéma *XML* permet de définir les vocabulaires pour des documents *XML* valides. Cependant, *XML* n'impose aucune contrainte sémantique à la signification de ces documents, l'interopérabilité syntaxique n'est pas suffisante pour qu'un logiciel puisse « *comprendre* » le contenu des données et les manipuler d'une manière significative.

Les couches RDF M&S et RDF Schéma

Elles sont considérées comme les premières fondations de l'interopérabilité sémantique. Elles permettent de décrire les taxonomies des concepts et des propriétés (avec leurs signatures). *RDF* fournit un moyen d'insérer de la sémantique dans un document, l'information

est conservée principalement sous forme de déclarations *RDF*. Le schéma *RDFS* décrit les hiérarchies des concepts et des relations entre les concepts, les propriétés et les restrictions domaine/co-domaine pour les propriétés. Les trois langages *XML*, *RDF* et *RDFS* seront détaillés dans la section réservée aux langages du web sémantique.

La couche Ontologie

Elle décrit des sources d'information hétérogènes, distribuées et semi-structurées en définissant le consensus du domaine commun et partagé par plusieurs personnes et communautés. Les ontologies aident la machine et l'humain à communiquer avec concision en utilisant l'échange de sémantique plutôt que de syntaxe seulement. La deuxième section de ce chapitre sera entièrement dédiée aux ontologies.

La couche règles

Elle est aussi un élément clé de la vision du Web sémantique, la couche Règles offre la possibilité et les moyens de l'intégration, de la dérivation, et de la transformation de données provenant de sources multiples.

La couche Logique

Elle se trouve au-dessus de la couche Ontologie. Certains considèrent ces deux couches comme étant au même niveau, comme des ontologies basées sur la logique et permettant des axiomes logiques. En appliquant la déduction logique, on peut inférer de nouvelles connaissances à partir d'une information explicitement représentée.

Les couches Preuve (Proof) et Confiance (Trust)

Ce sont les couches restantes qui fournissent la capacité de vérification des déclarations effectuées dans le Web Sémantique. On s'oriente vers un environnement du Web sémantique fiable et sécurisé dans lequel nous pouvons effectuer des tâches complexes en sûreté.

D'autre part, la provenance des connaissances, des données, des ontologies ou des déductions est authentifiée et assurée par des signatures numériques, dans le cas où la sécurité est importante ou le secret nécessaire ; le chiffrement est utilisé.

II.1.2. Les langages du Web sémantique

XML

XML est un langage conçu à partir des travaux du W3C en 1998. Il a été créé pour combler certaines lacunes du langage HTML. Son principal atout est qu'il hérite de la notion de sémantique du langage SGML « *Standard Generalized Markup Language* », et la facilité de la mise

en œuvre du langage HTML en dissociant le contenu de la forme (présentation). C'est un langage générique qui permet de structurer le contenu d'un document à l'aide de balises, ces dernières caractérisent la sémantique présente dans le document. Il facilite la gestion, l'exploitation et la diffusion de l'information en étant indépendant des plateformes.

XML permet de définir un vocabulaire (sémantique) personnel en créant ses propres balises. Ces balises étant extensibles, il permet aux utilisateurs d'ajouter à leurs documents des structures arbitraires (MALIK, 2004).

La structure de XML est arborescente et très similaire au principe de l'orienté objet, chaque type de documents est représenté par plusieurs éléments. Chaque élément est lui-même décrit par d'autres propriétés et ainsi de suite, le tout est bien sûr traduit en langage de balisage.

XML est un langage extrêmement important pour structurer le contenu de Web car il est capable d'apporter de la sémantique aux ressources.

XML se caractérise en particulier par :

- Son indépendance par rapport aux plateformes.
- Sa souplesse et puissance.
- La gestion de plusieurs formats de données.
- Son pouvoir de normalisation des données.

On pourrait se demander si cela est suffisant pour pouvoir créer un Web Sémantique, la réponse est non. Car si XML est un élément essentiel pour structurer et standardiser les données, il se trouve que la plupart des ressources ne sont pas au format XML.

RDF et RDFs

Le langage *RDF* (Ressource Description Framework) a été adopté par *W3C* comme un des formalismes standards de représentation de connaissances sur le Web. Utilisant la syntaxe *XML* qui constitue déjà un standard, le *RDF* permet de décrire des ressources Web en termes de ressources, propriétés et valeurs. Une ressource peut être une page Web (identifiée par son *URI*, United Ressource Identifier) ou une partie de page (identifiée par une balise). Les propriétés couvrent les notions d'attributs, relations ou aspects et servent à décrire une caractéristique d'une ressource en précisant sa valeur. Les valeurs peuvent être des ressources ou des littéraux. *RDF* dispose d'une sémantique formelle analogue à celle des graphes conceptuels, c'est-à-dire identique à celle d'un fragment de la logique du premier ordre.

Pour décrire n'importe quel type de connaissances à l'aide de ce formalisme, on doit d'abord décrire en *RDF* le modèle sémantique à utiliser. Par exemple, pour décrire des connaissances en terme de concepts et de relations hiérarchisés, l'introduction des types

« *concepts* » et « *relations* » et des propriétés de subsomption et d'instanciation est nécessaire. Un schéma de base incluant les primitives sémantiques généralement utilisées a ainsi été ajouté au *RDF* et constitue ce qu'on appelle le *RDF SCHEMA (RDF(S))*.

Le *RDF(S)* n'est cependant pas un langage opérationnel de représentation, au sens où il ne permet pas la représentation des axiomes et leur utilisation pour raisonner. Des extensions ont été proposées afin de pallier cette lacune. Plusieurs propositions ont été faites tendant à représenter les axiomes (règles) et définitions de classes et de propriétés en introduisant d'autres classes et/ou propriétés.

OIL

Dans l'optique d'une utilisation d'ontologies sur le Web, le langage *RDF(S)* a été enrichi par l'apport du langage *OIL* (Ontology Interchange Language) qui permet d'exprimer une sémantique à travers le modèle des frames tout en utilisant la syntaxe de *RDF(S*). OIL offre de nouvelles primitives permettant de définir des classes à l'aide de mécanismes ensemblistes issus des logiques de description (intersection de classes, union de classes, complémentaire d'une classe). Il permet également d'affiner les propriétés de *RDF(S)* en en contraignant la cardinalité ou en en restreignant la portée.

Le langage OIL a été fusionné avec le langage *DAML* pour former le *DAML+OIL*. *DAML* (Darpa Agent Markup Language) est conçu pour permettre l'expression d'ontologies dans une extension du langage *RDF*. Il offre les primitives usuelles d'une représentation à base de frames et utilise la syntaxe *RDF*. L'intégration de *OIL* rend possible les inférences compatibles avec les logiques de description, essentiellement les calculs des liens de subsomption.

OWL

La combinaison de *RDF/RDF(S)* et de *DAML+OIL* a permis l'émergence d'un langage standard de représentation de connaissances pour le Web, *OWL*, retenu par le *W3C*. Cependant, si *OWL* offre la possibilité d'exprimer différentes sortes de propriétés conceptuelles (restrictions de propriétés, axiomes de classes, cardinalités, etc.), il ne permet pas encore de représenter tout type d'axiome dans une ontologie.

D'autre part, *OWL* n'est pas un langage opérationnel en soi car il n'offre pas de mécanisme permettant le raisonnement, même s'il peut servir de format d'échange de représentations conceptuelles de connaissances entre des systèmes qui, eux, permettent le raisonnement (en particulier des raisonneurs basés sur les logiques de description). *OWL* offre trois sous langages d'expression croissante : *OWL Lite*, *OWL DL* et *OWL Full*.

Le Langage *OWL Full* est celui qui offre le plus de liberté au développement de systèmes à base d'ontologie, mais un système de raisonnement ne pourrait probablement pas gérer un

raisonnement complet de toutes ses caractéristiques de *OWL Full* et il n'existe actuellement aucune mise en œuvre *OWL Full* complète.

A terme, *OWL* est amené à voir son expressivité augmenter, et devra devenir un véritable langage opérationnel pour permettre l'émergence d'un Web sémantique conforme aux ambitions proposées par T. BERNERS-LEE.

II.2. LES ONTOLOGIES

Introduit en Intelligence Artificielle (IA) vers la fin des années 60, le terme d'ontologie est cependant utilisé en philosophie depuis le XIXème siècle où il désigne l'étude de l'«*être* », ou plus généralement de ce qui existe. L'IA, ayant de moindres ambitions que la philosophie, ne considère les ontologies que relativement aux différents domaines de connaissances. L'émergence de l'Ingénierie des Connaissances a propulsé les ontologies comme réponses aux problématiques de représentation et de manipulation des connaissances au sein des systèmes informatiques.

L'Ingénierie des Connaissances, branche de l'Intelligence Artificielle issue de l'étude des Systèmes Experts, est censée permettre le stockage et la consultation de connaissances, le raisonnement automatique sur les connaissances stockées, la modification des connaissances stockées, et, avec le développement des réseaux, le partage de connaissances entre systèmes informatiques.

D'une façon plus générale, il ne s'agit plus de manipulation stricte des connaissances par des systèmes « *aveugles* » mais de coopération entre système informatique et utilisateur, et éventuellement expert, pour produire des résultats plus proches de la nature « *humaine* » des problèmes posés que de la rigidité « *logique* » à laquelle sont soumis les systèmes experts. Un système à base de connaissances doit donc avoir accès non seulement aux termes utilisés par l'être humain, mais également à la sémantique que ce dernier associe aux différents termes, faute de quoi aucune communication efficace n'est possible. Plus précisément, les représentations symboliques utilisées dans les machines doivent avoir du sens aussi bien pour la machine que pour les utilisateurs.

La représentation des connaissances sous forme de règles logiques, utilisée dans les systèmes experts, ne suffit donc plus. Modéliser la richesse sémantique des connaissances, nécessite l'introduction de nouveaux formalismes[1] capables de représenter les connaissances au niveau conceptuel en y incluant la « *structure cognitive* » d'un domaine.

Les ontologies permettent l'intégration de la sémantique associée à un concept, en

[1] Les langages à bases de frames, les logiques descriptives et les graphes conceptuels sont des exemples de tels formalismes. Ils permettent de représenter les concepts sous jacents à un domaine de connaissance, les relations qui les lient et la sémantique de ces relations.

restreignant les interprétations possibles dans un domaine sans restreindre la sémantique opérationnelle. Ainsi, la désambigüisation d'une conceptualisation n'est possible que dans un et un seul domaine d'application, restriction nécessaire mais non suffisante garantissant l'unicité du sens associé à une connaissance désignée par des termes du domaine. Les ontologies, par leur essence propre, étant destinées à être réutilisées, la sémantique qu'elles intègrent ne doit pas dépendre d'un objectif opérationnel.

Plus précisément, la sémantique des connaissances à représenter peut en partie reposer sur le choix d'un contexte applicatif, mais la représentation formelle des connaissances exprimant cette connaissance ne doit pas dépendre du contexte.

Les ontologies sont donc des représentations de connaissances, contenant des termes et des énoncés qui spécifient la sémantique d'un domaine de connaissance donné, dans un cadre applicatif donné. Dans cette optique, le terme « *ingénierie ontologique* » a été proposé par Mizoguchi, 1997) pour désigner un nouveau champ de recherches ayant pour but la construction de systèmes informatiques tournés vers le contenu, et non plus vers les mécanismes de manipulation de l'information.

Aucune définition précise n'a encore été donnée aux ontologies, bien qu'un consensus soit né autour de la formulation de Gruber, 1993) : «*an ontology is a formal, explicit, specification of a shared conceptualization*»[1]. La construction d'une ontologie intervient donc après un travail de conceptualisation qui consiste à identifier, au sein d'un corpus, les connaissances spécifiques au domaine de connaissance à représenter, et reconnues comme relevant du domaine. «*A conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose* »[2].

Guarino, 1995) affine cette définition en considérant les ontologies comme des spécifications partielles et formelles d'une conceptualisation. Les ontologies sont formelles car exprimées sous un formalisme doté d'une sémantique formelle, et partielle car une conceptualisation ne peut pas toujours être entièrement formalisée dans un tel cadre du fait d'ambigüités ou du fait qu'aucune représentation de leur sémantique n'existe dans le langage de représentation choisi.

II.2.1. Les concepts, les relations et leurs propriétés

Les ontologies fournissent le vocabulaire commun d'un domaine et définissent, de façon plus ou moins formelle, le sens des termes et les relations entre ces derniers. Le propos général d'une ontologie est de catégoriser un même monde. Pourtant les ontologies peuvent être très

1 Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée.

2 Une conceptualisation est une abstraction, une vue simplifiée du monde que l'on désire représenter pour une certaine utilisation.

différentes aussi bien au niveau de leur « *top-level* » qu'au niveau du traitement de leurs composants de base tels que les choses, les procès, les relations…

Une ontologie débute par une taxinomie, un arrangement structuré d'informations en classes qui catégorise un sujet et ses dépendants. La taxinomie est la partie centrale de la plupart des ontologies. Cependant, l'organisation taxinomique peut varier d'une façon très importante elle aussi.

Selon Gruber, (1993), la formalisation d'une ontologie se met en place grâce à 5 types de composants ainsi qu'un composant introduit par Sowa, (2000) (CHAMPLAUX, 2009): le rôle.

Les classes/ les concepts

Une classe ou un concept, représente un type d'objet dans l'univers. Les classes sont habituellement organisées en taxinomies auxquelles on applique des mécanismes d'héritage. Tous les concepts peuvent être organisés en une large taxinomie. Il peut aussi y avoir un grand nombre de hiérarchisations plus petites, ou bien pas de taxinomie explicite du tout.

Les concepts sont utilisés dans leur sens large. Ils peuvent être abstraits ou concrets, élémentaires (électrons) ou composés (atomes), réels ou fictifs. Il arrive que les définitions des ontologies aient été diluées, en ce sens que les taxinomies sont considérées comme des ontologies complètes (CAO, 2006).

En résumé, un concept peut être tout ce qui peut être évoqué et, partant de là, peut consister en la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc. Il est souvent fait référence aux concepts en tant qu'union de classes et d'instances, alors que chacun des constituants de l'ontologie est considéré comme un fragment de connaissance (knowledge piece).

Les relations :

Les relations représentent un type d'interaction entre les notions d'un domaine. Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensembles. Des exemples de relations binaires sont sous-classe-de, connecté-à.

Les rôles

Selon Sowa (2000), « *un rôle caractérise une entité par quelque rôle qu'elle joue dans sa relation à une autre entité. Le type* ***Humain****, par exemple, est un type de phénomène qui dépend de la forme interne de l'entité ; mais la même entité peut être caractérisée par des rôles du type, Mère, Employé ou Piéton*. ».

Les fonctions

Les fonctions sont aussi des cas particuliers de relations dans lesquelles le nième élément

de la relation est défini à partir des n-premiers. Comme exemple de fonctions binaires il y a la fonction mère-de ou carré-de, comme fonction ternaire, le prix d'une voiture neuve sur lequel on peut se baser pour calculer le prix d'une voiture d'occasion en fonction de son modèle, de sa date de construction et de son kilométrage.

Les axiomes

Les axiomes sont utiles à la structuration de phrases qui sont toujours vraies. Ils permettent de contraindre les valeurs de classes ou d'instances.

Les instances

Les instances sont utilisées pour représenter des éléments dans un domaine.

II.2.2. Processus de construction d'une ontologie

Le processus général de construction d'ontologies opérationnelles peut donc être découpé en 3 phases (VERDIN, 2006) :

La conceptualisation

Est l'identification des connaissances contenues dans un corpus représentatif du domaine considéré. Ce travail doit être mené par un expert du domaine (ou un groupe d'experts), assisté par un ingénieur de la connaissance, apportant son expertise des paradigmes de représentation des connaissances en machine pour aider à la structuration des connaissances ;

L'ontologisation :

Est la formalisation, autant que possible, du modèle conceptuel obtenu à l'étape précédente. Une part des connaissances du domaine peut, à ce niveau, être abandonnée, du fait de l'impossibilité de lever certaines ambiguïtés, ou du fait des limitations de l'expressivité du langage de représentation d'ontologie utilisé. Ce travail doit être mené par l'ingénieur de la connaissance, expert du modèle formel de représentation de l'ontologie, assisté de l'expert du domaine ;

L'opérationnalisation :

Est la transcription de l'ontologie dans un langage formel et opérationnel de représentation de connaissances. Ce travail doit être mené par l'ingénieur de la connaissance.

Il est à noter que ce processus n'est pas linéaire et que de nombreux aller-retour sont a priori nécessaires pour bâtir une ontologie opérationnelle adaptée aux besoins. Remarquons pour finir que ce modèle de construction d'ontologie est ascendant, c'est-à-dire que l'on part

des connaissances à représenter, pour aboutir à une représentation formelle. Mais une construction descendante est possible, qui consiste à choisir un modèle opérationnel de représentation, en fonction de l'objectif d'utilisation de l'ontologie, puis à instancier ce modèle avec les connaissances du domaine.

Cependant, certaines connaissances constituent à elles seules un domaine de connaissances. C'est le cas des notions générales telles que la causalité, le temps, l'espaces etc. et sont de ce fait utilisées dans tous les autres domaines. On parle alors de méta-connaissances, ou connaissances sur la connaissance. Elles ne prendront une dimension sémantique que si leur représentation est objet d'un consensus et portent une information active susceptible d'influencer le déroulement d'un processus, de produire de nouvelles informations ou permettre une prise de décision. Elles seront le cas échéant source d'ambigüité.

II.2.3. Classification des ontologies

Il existe de nombreuses sortes d'ontologies, destinées à des utilisations très variées. L'un des problèmes fréquemment rencontrés par les utilisateurs potentiels d'ontologies, est celui de la diversité des appellations de ces ontologies (et des finalités sous-jacentes). Les ontologies peuvent être alors classifiées selon divers critères. Nous présentons ici une classification basée sur le champ d'application des ontologies (ABROUK, 2006) :

Ontologie de haut niveau

Elle décrit des concepts très généraux comme l'espace, le temps, la matière, les objets, les événements, les actions, etc. Ces concepts ne dépendent pas d'un problème ou d'un domaine particulier, et doivent être, du moins en théorie, consensuels à de grandes communautés d'utilisateurs.

Ontologie de domaine

Elle décrit le vocabulaire ayant trait à un domaine générique (ex. : l'enseignement, la médecine…), notamment en spécialisant les concepts d'une ontologie de haut niveau.

Ontologie de tâche

Elle décrit le vocabulaire concernant une tâche générique (ex. : enseigner, diagnostiquer…), notamment en spécialisant les concepts d'une ontologie de haut niveau. Certains auteurs emploient le nom « *ontologie du domaine de la tâche* » pour faire référence à ce type d'ontologie.

Ontologie d'application

L'ontologie d'application contient des concepts dépendants d'un domaine et d'une tâche particuliers qui sont généralement subsumés par des concepts de ces deux ontologies. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité. Il s'agit donc ici de mettre en relation les concepts d'un domaine et les concepts liés à une tâche particulière, de manière à en décrire l'exécution.

Annexe III : Le parcours du web

Il est impossible de découvrir par une simple exploration toutes les pages Web accessibles sur Internet (BENOIT, 2006). Certaines pages sont protégées par un mot de passe ou bien possèdent une URL privée. En plus, de nombreuses pages sont dynamiques, par exemple générées par des formulaires donc ne possédant pas de liens.

Par ailleurs, étant donné la grande dynamique du Web et sa taille, pendant le temps nécessaire à la récupération des pages accessibles, de nombreuses autres pages auront été créées ou modifiées ou supprimées : une exploration exhaustive semble donc impossible (PICAROUGNE, 2004).

Un autre problème se pose au niveau de l'identification des pages Web. On pourrait définir une page par son URL, vu que toute page est accessible depuis son URL. Cependant, il se peut que la même page apparaisse sous plusieurs URL différentes, par exemple si un serveur possède plusieurs noms ou bien utilise des liens symboliques. On étudie alors le contenu des pages, et on considère généralement que 2 pages avec exactement le même contenu représentent la même page avec 2 URL différentes. Cela est acceptable vu que rien ne distingue les 2 pages, elles contiennent la même information et pointent sur les mêmes pages. Cela ne permet cependant pas d'identifier une page au cours du temps et de ses modifications.

III.1. LE GRAPHE DU WEB

Pour connaître la structure du Web et pouvoir étudier le graphe du Web, le premier problème consiste à explorer ce graphe. Ce processus d'exploration s'appelle un crawl : on considère un ensemble initial de pages, puis on suit les liens sortants, ce qui permet de découvrir de plus en plus de pages (BENOIT, 2006).

Rien ne garantit en revanche que l'on visite tout le Web. On s'intéresse alors à la structure du graphe, et notamment :

- le diamètre du graphe,
- la distribution des degrés sortant et entrant des pages,
- le niveau de connexité (composantes connexes du graphe)
- la structure macroscopique.

III.1.1. Structure du Web

Il existe deux niveaux d'appréhension d'Internet selon des paramètres différents :

- **Haut niveau** : plusieurs nœuds de communication s'interconnectent.
- **Bas niveau** : à l'intérieur de chaque nœud de communication, plusieurs serveurs de données sont fortement reliés entre eux.

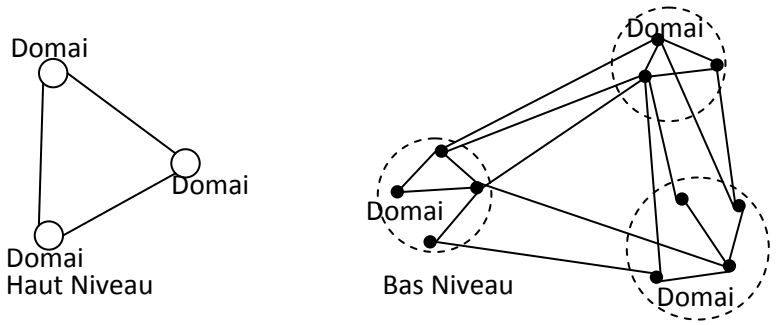


Figure 34: Les différents niveaux d'appréhension d'Internet.

Structure Macroscopique

Broder et al. (2000) cités dans (PICAROUGNE, 2004) ont effectué une étude topologique d'Internet portant sur 200 millions de pages et 2.5 milliards de liens à partir de plusieurs crawls du moteur de recherche Altavista. Cette étude a démontré que prés de 90% des pages forment un noyau fortement interconnecté si on ne tient pas compte de l'orientation des arcs. Seulement 17 millions (soit 8.5%) de pages restent déconnectés du noyau. La figure suivante représente la topologie réelle des pages Internet.

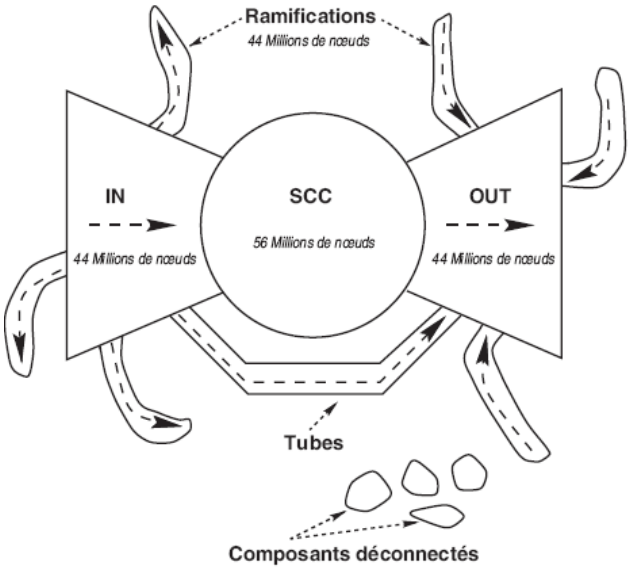


Figure 35: Graphe de connexion d'Internet

La zone principale SCC regroupe les pages fortement connectées, chacune peut atteindre une autre page en suivant un nombre minime de liens. Les deux autres familles IN et OUT représentent respectivement les pages qui peuvent atteindre le noyau mais qui ne peuvent pas être atteintes, et les pages pouvant être atteintes à partir du noyau sans pouvoir l'atteindre.

Enfin, les ramifications contiennent les pages qui ne peuvent atteindre le noyau et qui ne peuvent être atteintes par lui. Toutes ces familles contiennent le même nombre de pages avec un avantage pour le noyau.

Le diamètre maximal du noyau a été mesuré à au moins 28 liens, le diamètre maximal du graphe à 500 liens. Dans les sous graphes, la probabilité qu'il existe un chemin entre une source et une destination est de 24%. Si ce chemin existe, sa longueur est de 16 liens en moyenne si le graphe est orienté et de 6 liens seulement s'il ne l'est pas.

Structure Microscopique

Cette modélisation s'intéresse aux particularités locales propres au graphe du Web. On observe alors l'émergence de communautés. Un des problèmes dans l'analyse de la structure microscopique consiste à définir les communautés, un autre à les détecter automatiquement.

La première approche appelée « *fans/stars* » (KLEINBERG, 1998) considère une communauté comme un couple d'ensemble de pages Web telles que toutes les pages du premier ensemble pointent vers toutes les pages du second. De plus, les pages du second ensemble ne pointent pas les unes vers les autres.

Cette structure apparaît souvent : communauté centrée autour d'un sujet de prédilection. Les « *fans* » ont des pages qui mettent des liens vers leurs « *stars* ». En revanche, les stars n'ont aucune connexion entre elles car ce sont des concurrents. Le modèle de classification PageRank

se base sur cette approche, on voit que les pages des « *stars* » font autorité (fort PageRank), tout le monde s'y réfère.

Une deuxième approche, basée sur le modèle « *petit monde de kleinberg* » (KLEINBERG, 2000), (BENOIT, 2006), consiste à interpréter la notion de communauté par une collection de pages Web qui possèdent plus de liens vers les pages de la collection que vers les pages externes. Les communautés sont ensuite reliées entre elles par quelques liens.

Cette approche rejoint les problèmes de partitionnement de graphes. De telles communautés sont elles aussi centrées sur un sujet de prédilection, mais qui permet de construire des sous-graphes du Web. L'algorithme HITS de Kleinberg est basé sur ce modèle et a permis la construction de très puissants moteurs de recherche à des fins industrielles et professionnelles.

Propriétés statistiques du graphe du web

Les études sur le graphe du web ont permis de mettre en évidence les propriétés suivantes :

- La **distance moyenne** dans le graphe du Web est faible de l'ordre de 19.
- Le **coefficient de clusterisation** ou la probabilité pour que deux voisins d'un même sommet soient eux-mêmes voisins est élevé dans le cas du graphe du Web.
- La **distribution des degrés** en loi de puissance est notamment vérifiée pour le degré entrant (nombre de liens vers une page), ce qui est cohérent avec le fait que la plupart des pages sont très peu référencées, mais certaines le sont énormément.

Bien sûr, ces propriétés sont obtenues uniquement sur des observations partielles du graphe du Web, et biaisées par les méthodes d'exploration utilisées.

III.1.2. Modélisation du graphe

Modèle de Watts et Strogatz

Le modèle de Watts et Strogatz (1998) cherche à prendre en compte la forte clusterisation rencontrée en pratique. On part d'un anneau de sommets, où chaque sommet est relié à ses k plus proches voisins sur l'anneau, pour un k donné (anneau standard : k = 1). Pour chaque arête, une des extrémités est remplacée avec une probabilité p par une nouvelle extrémité choisie aléatoirement.

Ce modèle a fait un premier pas dans la direction de modèles réalistes de topologies. Cependant, il ne capture pas la distribution des degrés en loi de puissance.

Modèle d'Albert et Barabasi

Albert et Barabasi (1999) ont cherché à obtenir la distribution des degrés en loi de puissance. Pour cela, les sommets sont ajoutés un à un et reliés aléatoirement aux sommets préexistants. On parle d'attachement préférentiel : un nouveau sommet se lie beaucoup plus facilement avec un sommet déjà beaucoup lié qu'avec un sommet peu lié. En fait, la probabilité d'être relié à un sommet donné croît exponentiellement avec son degré.

La distribution des degrés suit bien une loi en puissance, et en plus la distance moyenne entre les sommets reste faible. En revanche, ce modèle ne respecte pas le coefficient de clusterisation observé dans le graphe du Web

III.2. PARCOURS A BASE DE CRAWLER

Commençons par distinguer le *crawl* du *crawler*. Un *crawl* est l'ensemble des pages du web parcouru. Un *crawler* est l'ensemble des moyens logiciels et matériels permettant de réaliser un crawl. Le crawler est chargé d'indexer les documents du web. Plusieurs problèmes sont posés à ce niveau : la fraîcheur de l'index en est sûrement le plus évident. Comment assurer une indexation « *fraîche* » alors que la taille du web ne cesse d'augmenter et que la bande passante réservée à cet effet n'a pas proportionnellement augmenté ? L'autre problème est celui du web caché : comment trouver et indexer les pages qui ne sont référencées nulle part et qui n'ont pas de liens entrants ?

III.2.1. Principaux Types de Crawlers

Il y a deux principaux types de crawlers : les crawlers statiques et les crawleurs dynamiques. Le principe de tous les crawleurs statiques est le même : à partir d'un ensemble initial de pages, ils analysent les hyperliens contenus dans ces pages, essaient de récupérer les pages pointées par ces hyperliens, et ainsi de suite, pour récupérer ainsi une partie sans cesse croissante des pages du Web accessible. Chaque page n'est a priori récupérée qu'une seule fois, et on arrête le procédé quand on estime le moment voulu atteint (PAGE & BRIN, 1998). Un crawleur dynamique, lui, n'est pas prévu pour s'arrêter, mais pour parcourir perpétuellement le Web et revisiter périodiquement les pages qu'il a parcourues.

Le Batch Crawl

La première technique utilisée pour parcourir le web était le batch crawling qui se déroulait en trois étapes : on détermine d'abord une liste d'URL comme point de départ du crawl ; ensuite des robots d'indexation vont aspirer ces pages en récupérant les URLs qu'elles contiennent. Ces nouvelles URLs sont placées dans des files d'attentes de liens à explorer.

Lorsque cette file est vide, on considère que le Web a entièrement été indexé. Ces URLs serviront de point de départ du prochain cycle du crawler.

Le batch crawling a de graves inconvénients. D'abord, le crawler indexe l'ensemble des pages à chaque cycle, y compris celles qui ne changent jamais. Le processus est donc forcément long et peut durer jusqu'à une semaine pour être déclaré opérationnel. Or, entre temps, certaines pages auraient évolué et même changé de contenu. La fraîcheur de l'index est fortement compromise entre deux *full crawl*.

Le Fresh crawl

La première solution est de lancer un robot différent, le *fresh crawl,* destiné à retrouver les nouvelles pages entre deux full-crawl. Une autre technique proposée par Cho et Garcia-Molina, 1999) consiste à réindexer uniquement les pages susceptibles de changer. En d'autres termes, seules les pages jugées « *importantes* » sont indexées. La fonction de calcul de l'importance d'une page dépend de plusieurs paramètres : le décompte des liens entrants et sortants, son *PageRank* actuel, sa localisation dans l'arborescence du site auquel elle appartient et même une évaluation sémantique de la requête. Une fois cette fonction définie, on indexe d'abord les pages les plus importantes. Des études statistiques ont démontré que le *PageRank* seul est suffisant à l'évaluation de l'importance d'une page.

Le Crawl incrémental

Enfin, (Cho et Garcia-Molina, 2000) ont proposé de remplacer le *full crawl* par un *crawl incrémental*, c'est-à-dire qu'au lieu de lancer l'opération périodiquement, une fois par mois, l'exploration du web se fait en continu. Le *crawler* incrémental détermine d'abord quelles sont les pages susceptibles de devenir obsolètes et de les mettre à jour. Bien que cette technique offre un gain de ressources considérable, elle nécessite un travail en amont assez conséquent : il faut séparer ici deux évolutions différentes : la disparition de pages et l'apparition de nouvelles d'une part, et les changements de contenu dans une page donnée d'autre part. Le moteur de recherche a donc besoin, pour que l'indexation soit pertinente, d'attendre un certain délai pour collecter des informations fiables sur la manière dont l'index évolue. Il convient par la suite de mettre à jour ces informations, et de lancer, périodiquement, un batch *crawling* général pour éviter une dérive de l'index.

III.2.2. Processus de crawl

Dans l'absolu, un crawl obéit à un processus relativement simple. Tout crawl commence par un ensemble fini de pages, appelé ensemble initial de parcours, *initial crawl set,* qui sont stockées dans une base. Pour chaque page, les liens sortants sont identifiés et les pages liées

sont insérées à leur tour dans la base, si elles n'y existent pas encore.

Ce processus est exécuté tant que les conditions d'arrêt (suffisamment de pages récupérées, limitation de la base, croissance devenue négligeable. . .) ne sont pas atteintes.

```
Variables
    Q : liste des pages à visiter ;
    V : Liste des pages visitées ;
    P : page en cours ;
    L : Liste de liens
Algorithme Crawl
    Initialiser Q = {p1, p2, ..., pn } ;
    V = {};
    tant que Q <> {} faire
        P:= choisirElément(Q) ;
        Charger(P) ;
        V:=V ∪ {P} ;
        L:=ExtraireLiensDans(P) ;

        Pour chaque  p' ∈ L faire
            Si (p'∉V) ∧ (p'∉Q) alors
            Q = Q ∪ {p'}

        Q := Q - {P}
    finTantque
Fin
```

Algorithme 10 : Algorithme typique d'un crawl

Quelques considérations et choix techniques peuvent améliorer considérablement la qualité du crawl :

Choix de l'ensemble initial

Bien choisir l'ensemble de départ semble une condition fondamentale à l'efficacité du crawl ; cet ensemble détermine le nœud central du papillon du web. Un choix mal fait pourrait focaliser le parcours dans les zones *out* ou les *ramifications* du web, ce qui diminuerait considérablement le nombre de pages explorées. Il est à noter aussi que le choix de grands sites est une mauvaise approche[1]. En effets, les liens contenus dans leurs pages font rarement références à d'autres sites ou d'autres serveurs.

Limitations matérielles

Même si en théorie, un moteur de recherche est capable d'indexer toutes les pages du web, des contraintes physiques, comme la bande passante, la taille de la base de données ou les délais d'exécution des requêtes, limitent généralement le nombre de pages indexables[2]. De plus certaines règles « *de politesse* » sont à observer : de plus en plus de sites contiennent des fichiers *robot.txt* dans leur racine. Ces fichiers énumèrent les seules pages qu'un moteur de recherche

[1] Les grands sites, comme les grands personnages, sont souvent égocentriques et suffisants dans la mesure qu'ils renvoient tout à eux et font rarement référence aux autres.

[2] Selon Google, le nombre maximum de pages qu'il peut indexer est de 5.8 milliards. Ce qui est théoriquement très loin de la réalité du web.

peut indexer. Les autres sont sensées rester privées.

Stratégie d'exploration

Quels liens suivre en premier ? Si certains crawlers suivent les liens dans l'ordre dans lequel ils figurent dans une page, d'autres estiment que les liens vers les pages « importantes » doivent être suivis en premier. De plus le parcours en largeur ou en profondeur de l'arborescence des pages dans un site semble être un critère à considérer. Aucune étude sérieuse n'a pu jusqu'à présent déterminer quelle stratégie est la plus efficace ; à chacune ses qualité et ses défauts.

Annexe IV :
Introduction aux métaheuristiques

La résolution des problèmes NP-Difficile a toujours été un challenge de taille pour les scientifiques. Pendant longtemps, toutes les recherches étaient orientées vers les algorithmes exacts pour des cas particuliers polynomiaux. L'émergence des métaheuristiques a permis de trouver des solutions de bonne qualité, généralement en des temps raisonnables.

Toutes les métaheuristiques s'appuient sur un équilibre entre l'intensification de la recherche et la diversification de celle-ci. L'intensification permet de rechercher des solutions de plus grande qualité en s'appuyant sur celles déjà trouvées. La diversification met en place une stratégie d'exploration d'un plus grand espace de solutions possibles et donc aller vers les minima locaux.

Toute la qualité des métaheuristiques réside dans un juste équilibre entre ces deux propriétés. Un facteur de diversification trop faible conduirait à une convergence trop rapide vers des minima locaux ; une faible intensification, par contre, produirait une exploration trop longue.

IV.1. LES METHODES DE RECHERCHE LOCALES

Face à un problème d'optimisation, la tâche la plus ardue est certainement celle de la modélisation. Quelle que soit la méthode de recherche, l'objectif est le même : faire évoluer une solution initiale S vers une solution optimale à un problème donné en effectuant des déplacements dans le voisinage de *S*.

De manière générale, ces méthodes de recherche s'arrêtent quand une solution localement optimale est trouvée.

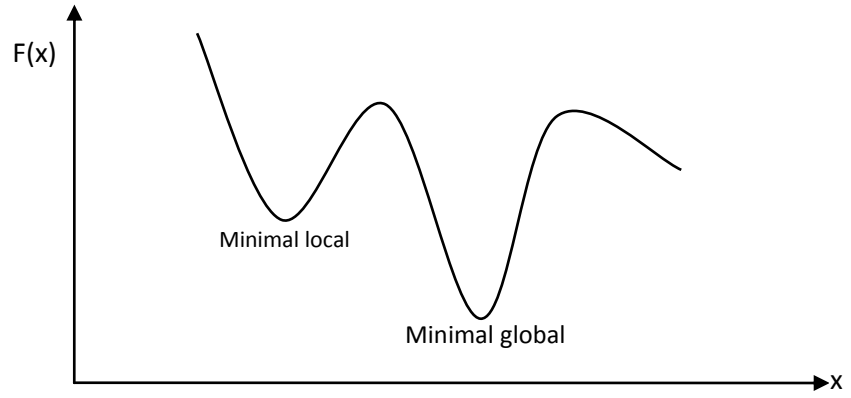


Figure 36 : minima local et global d'un ensemble de solutions

Dans l'idéal, il est préférable de toujours pouvoir échapper aux minima locaux. Il apparait clairement qu'il est possible d'opter pour des solutions moins bonnes mais qui élargiraient le voisinage permettant ainsi l'exploration de nouveaux espaces de recherche.

Différentes méthode de recherche existent :

IV.1.1. Méthode descente

Toutes les méthodes de recherche locale en descente s'appuient sur le même principe : partir d'une solution existante, chercher une solution « *meilleure* » dans le voisinage de la solution initiale.

Variables

S : Solution initiale
V : fonction de voisinage
F : fonction fitness

Algorithme simpleDescent

```
Trouver une solution initiale S
Repeter
    Choisir(S') dans V(S)
    si f(S')<f(S) alors
        S :=S'
tantque f(S)<f(X) ∀X ∈ V (S)
```

Fin

Algorithme 11 : l'algorithme simple descente.

Il est évident qu'un tel algorithme ne peut explorer que les solutions dans le voisinage de la solution initiale *S*, une des variations consiste à choisir uniquement la meilleure solution parmi toutes celles du voisinage.

```
Variables
    S : Solution initiale
    V : fonction de voisinage
    F : fonction fitness
Algorithme deepestDescent
    Trouver une solution initiale S
    Repeter
        Choisir(S') dans V(S)/F(S')<f(X)
            /∀X ∈ voisinage(S)
        si f(S')>f(S) alors
            S :=S'
    tantque f(S)>f(X) ∀X ∈ V(S)
fin
```

Algorithme 12 : l'algorithme plus grande descente

Le principal inconvénient de la méthode de descente est l'absence totale de diversification. L'intensification inhérente à cette méthode conduit trop vite à des blocages au niveau des solutions locales.

IV.1.2. Le recuit simulé

Introduite par kirkpatricl et al., 1981), la méthode du recuit simulé s'inspire du processus d'amélioration des solides (notamment les aciers) par recherche d'un état énergétique minimal correspondant à une structure stable. Plus formellement, on associe à une solution un état du métal, son équilibre thermodynamique est la valeur de la fonction objectif. Passer d'un état du métal à un autre correspond au passage d'une solution à une solution.

Le schéma de refroidissement, par analogie la fonction reproduction de la population, est la partie du problème qui requiert la plus grande attention.

Globalement, les étapes principales de l'algorithme de « *recuit simulé* » sont :

```
Variables
    S : Solution initiale
    T : cycle du recuit
    t : température
    V : fonction de voisinage
    F : Fonction de fitness
Algorithme recuitSimule
    Trouver une solution initiale S
    Fixer un cycle de recuit T
    Fixer la température Initiale t
    Repeter
        Choisir(S') dans V(S)
        ΔC = f(S')-f(S)
        Définir la probabilité d'accepter un mouvement P
        Si (ΔC<0) ou (exp(-ΔC/t)>P) alors
            S :=S'
        Mise à jour de t
    tantque non(Convergence)
Fin
```

Algorithme 13 : l'algorithme de recuit simulé

IV.1.3. La recherche *tabou*

Cette méthode a été baptisée recherche *tabou* pour exprimer l'interdiction de reprendre des solutions récemment visitées. La recherche « *tabou* » tente de modifier la structure du voisinage d'un point de l'espace de recherche afin de contraindre l'algorithme à explorer des régions de l'espace. D'un point de vue formel, le voisinage d'une solution s noté V(s) est modifié en V*(s) en fonction de la mémoire contenant les derniers pas parcourus dans l'espace des solutions.

La recherche « *tabou* » utilise une procédure locale à partir de V*(s) pour aboutir à une nouvelle solution s′. Le processus est ensuite réitéré à partir de s′ jusqu'à ce qu'un critère d'arrêt soit satisfait passant ainsi d'une méthode valide à une autre.

On dit alors que la nouvelle solution est un voisin de la précédente et on appelle voisinage l'ensemble des solutions que l'on peut atteindre à partir d'une solution.

Techniquement, chaque point exploré est enregistré dans une mémoire de taille fixe. À chaque itération, tous les mouvements possibles sont examinés et le moins mauvais est sélectionné parmi tous les points du voisinage direct de la position courante ne figurant pas dans la mémoire. Ceci permet d'éviter des mouvements cycliques, répétant indéfiniment la même suite de déplacements.

Quand la mémoire des points visités, notée *listeTabou*, est pleine, elle est gérée comme une liste circulaire en FIFO : on élimine le plus vieux point tabou et on insère la nouvelle solution.

```
Variables
    S : Solution initiale
    voisin : fonction de voisinage
    Sopt : solution optimale
    listeTabou : liste des nœuds tabous
Algorithme rechercheTabou
    Initialiser S de manière aléatoire
    Sopt := S
    listeTabou := {S}

    tantque (non Convergence) faire
        répéter
            S' = voisin(S)
        tantque S'∈ listeTabou

        si listeTabou est pleine alors
            Remplacer le dernier élément de listeTabou par S'
        Sinon listeTabou := listeTabou ∪ {S}
            si S' est meilleur que Sopt alors
                Sopt ← S'
        finSi
        S ← S'
    finTantque
Fin
```

Algorithme 14 : l'algorithme de base de la recherche « *tabou* ».

Cette méthode a par ailleurs l'avantage d'être facilement paramétrable. Il existe en effet au

maximum deux paramètres qui peuvent réellement influencer la recherche :

- **La taille de la listeTabou** qui est généralement déterminée empiriquement. Une taille trop petite peut amener l'algorithme à boucler sur une zone locale de l'espace de recherche tandis qu'une trop grande taille saturera rapidement les ressources disponibles.
- **le choix du critère d'arrêt** qui peut s'avérer difficile à déterminer pour éviter de prolonger trop longtemps la recherche ou de rater l'optimum global recherché.

IV.2. METAHEURISTIQUES A BASE DE POPULATION

Les méthodes de recherche à base de population, comme leur nom l'indique, travaillent sur une population de solutions et non pas sur une solution unique. Le principe général de toutes ces méthodes consiste à combiner des solutions entre elles pour en former de nouvelles en essayant d'hériter des « *bonnes* » caractéristiques des solutions parents. Un tel processus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait. Parmi ces algorithmes à base de population, on retrouve deux grandes classes qui sont les algorithmes génétiques et les colonies de fourmis.

IV.2.1. Les algorithmes génétiques

Proposé en 1975 par Holland, les algorithmes génétiques doivent leur popularité à Goldberg. De manière générale, les algorithmes génétiques utilisent un même principe. Une population d'individus (correspondant à des solutions) évolue en même temps comme dans l'évolution naturelle en biologie. Pour chacun des individus, on mesure sa faculté d'adaptation au milieu extérieur par le *fitness*. Les algorithmes génétiques s'appuient alors sur trois fonctionnalités :

- **la sélection** qui permet de favoriser les individus qui ont un meilleur fitness
- **le croisement** qui combine deux solutions parents pour former un ou deux enfants en conservant les bonnes caractéristiques des solutions parents.
- **la mutation** qui permet d'ajouter de la diversité à la population en mutant certaines caractéristiques (gènes) d'une solution.

IV.2.2. Les colonies de fourmis

Les algorithmes à base de colonies de fourmis ont été introduits par Dorigo et al., (1992). Une des applications principales de la méthode originale était le problème du voyageur de commerce et depuis elle a considérablement évolué.

Cette nouvelle métaheuristique imite le comportement de fourmis cherchant de la nourriture. A chaque fois qu'une fourmi se déplace, elle laisse sur la trace de son passage une odeur (la phéromone). Comme la fourmi est rarement une exploratrice isolée, avec plusieurs de ses congénères, elle explore une région en quête de nourriture. Face à un obstacle, le groupe des fourmis explorent les deux côtés de l'obstacle et se retrouvent, puis elles reviennent au nid avec de la nourriture.

Les autres fourmis qui veulent obtenir de la nourriture elles aussi vont emprunter le même chemin. Si celui-ci se sépare face à l'obstacle, les fourmis vont alors emprunter préférablement le chemin sur lequel la phéromone sera la plus forte.

Mais la phéromone étant une odeur, elle s'évapore. Si peu de fourmis empruntent une trace, il est possible que ce chemin ne soit plus valable au bout d'un moment, il en est de même si des fourmis exploratrices empruntent un chemin plus long (pour le contournement de l'obstacle par exemple). Par contre, si le chemin est fortement emprunté, chaque nouvelle fourmi qui passe redépose un peu de phéromone et renforce ainsi la trace, donnant alors à ce chemin une plus grande probabilité d'être emprunté.

On peut alors comprendre l'application directe du problème du voyageur de commerce. Chaque fourmi effectue un tour en empruntant des arêtes du graphe, puis au bout d'un moment les arêtes qui conduisent à une bonne solution deviendront les plus empruntées jusqu'à obtention d'une très bonne solution.

Même si le principe est simple, on peut voir apparaître des difficultés comme la quantité de phéromone à déposer, le coefficient d'évaporation, la définition du choix biaisé, etc.

L'algorithme de base du voyageur de commerce peut être écrit comme suit :

Algorithme PVC

```
Pour chaque Individu dans Groupe
  Initialiser Individu
FinPour

Tantque non critereArret
    pour chaque ville I

        pour chaque fourmi k faire
        choix de la prochaine ville I' pour la fourmi k
        dépot de la trace locale entre I et I'
        finpour
    finPour

    Choix de la fourmi ayant le plus court chemin
    dépot de la trace globale
finTantque
```

Fin

Algorithme 15 : résolution du Problème de voyageur de commerce par colonies de fourmis