Université de Montréal

MAGELLAN : UN AGENT POUR SIMPLIFIER LES ACHATS SUR INTERNET

par

Jonathan Paturel

Département d'informatique et de recherche opérationnelle

Faculté des Arts et des Sciences

Maître ès Sciences (M.Sc.) en informatique

Avril, 2002

Université de Montréal

Faculté des Arts et des Sciences

Ce mémoire intitulé :

MAGELLAN : UN AGENT POUR SIMPLIFIER LES ACHATS SUR INTERNET

présenté par :

Jonathan Paturel

a été évalué par un jury composé des personnes suivantes :

Jean Meunier	président-rapporteur
Claude Frasson	directeur de recherche
François Major	membre

Mémoire accepté le : 25 juillet 2002

Sommaire

Les engins de recherche traditionnels nous aident grandement pour trouver les informations pertinentes sur l'immensité du Web, mais leur efficacité pourrait être améliorée. Les agents intelligents de recherche nous permettent de pallier les failles que présentent les engins de recherche traditionnels. La recherche de produits sur le Web est une activité qui prend beaucoup plus de place dans notre emploi du temps avec la croissance des sites de commerce électronique sur le Web. Dans ce mémoire, nous allons donc proposer une solution pour rechercher des produits sur différents sites de vendeurs. Notre solution différera des engins de recherche traditionnels en ajoutant un filtre des résultats qui utilisera un rassemblement des goûts de l'usager, que nous appellerons le profil de l'usager. Ce profil bâtit au moyen des pages Web visionnées par l'usager ainsi que par l'usager, permettra de filtrer les résultats qui ne sont pas pertinents pour l'usager. Nous avons donc réussi à produire Magellan, un agent pour simplifier les achats sur le Web. L'ensemble du programme est fonctionnel, mais comme nous l'expliquons dans la discussion du mémoire, il y a place à l'amélioration.

Mots-Clés : agent, intelligent, recherche, Web, produit, commerce électronique, TFIDF.

Abstract

Traditional search engines help us a lot finding information on the Web, but they could be improved. Intelligent agents help us where traditional search engine did not, they provide a more personal way to search the Web thus improving the relevance of the documents returned. Searching products on the Web is an activity that is taking more and more of our time while e-commerce increases its popularity. In this thesis, we will propose a solution to search products on different sellers Web sites. Our solution will differ from the traditional search engines in that it will filter the results by taking accounts the documents relevance for the user. This filter will be implemented by using a user profile that will store the preferences of the user on different subjects. This user profile will be built with the Web pages the user view and choose to add to his profile or simply, by adding a sequence of words to the profile. We present you, Magellan, an agent built to simplify the search of products on the Web. The agent is functional but we did notice some improvements that could be implemented if the documentation and proper environment were available.

Key words: agent, intelligent, search, Web, product, e-commerce, TFIDF

Remerciements

Je tiens à remercier chaleureusement mon directeur de maîtrise, Claude Frasson, pour son dévouement et ses précieux conseils. Un merci tout particulier à Geneviève Bigras, pour son support et ses encouragements. Merci aussi à ma mère, Rose-Ange Bélanger, qui a été mon inspiration et qui m'a encouragé dans mon cheminement universitaire. Finalement, merci à mes amis, Dominic Ménard, Hugo-Christopher Piette, Nicolas Lamarre, Sylvain Gauthier, et les autres pour leur présence tout au long de mon travail.

À vivi...

Table des matières

Liste des tableaux

Liste des figures

Glossaire

ActiveX : c'est le nom que Microsoft a donné pour un ensemble de technologies de programmation et d'outils stratégiques orientés objet.

Bag-of-words : c'est la représentation d'un document, l'ensemble des mots d'un document réunis.

Client http : unité technologique qui peut traiter le protocole http reçu d'un serveur.

Crawler : unité technologique qui parcourt un ensemble de sites Web et les liens qui y sont présents pour rassembler des informations.

FAQ : Foire Aux Questions, rubrique présentant par sujets les questions les plus fréquemment posées par les utilisateurs, accompagnées des réponses correspondantes.

FTP : « File Transfer Protocol », protocole de transfert utilisé pour le partage de fichier informatique sur Internet.

GET : fonction du protocole HTML pour demander à un serveur Web d'envoyer une page Web à notre fureteur.

HTML : « Hypertext Markup Language », HTML est un langage de description de pages permettant de contrôler, par l'intermédiaire d'éléments appelés balises (tags), l'apparence qu'elles auront sur l'écran d'un utilisateur de serveurs Web.

HTTP : « Hypertext Transfert Protocol », protocole de communication utilisé pour le Web.

Hypertext : système de renvois permettant de passer directement d'une partie d'un document à une autre, ou d'un document à d'autres documents choisis comme pertinents par l'auteur.

Internet : réseau mondial associant des ressources de télécommunication et des ordinateurs serveurs et clients, destinés à l'échange de messages électroniques, d'informations multimédias et de fichiers. Il fonctionne en utilisant un protocole commun qui permet l'acheminement de proche en proche de messages découpés en paquets indépendants.

Java : langage de programmation.

JList : classe appartenant aux librairies de Swing qui représente les listes dans les formulaires.

LSI : « latent semantic indexing », méthode d'indexation et de recherche de documents basée sur l'analyse de la structure sémantique des documents.

Métarecherche : méthode de recherche effectuée sur un ensemble d'engins de recherche pour regrouper les différents résultats.

Métatags : type particulier de balise (tag). Voir tags.

NetNews : service permettant l'échange et la discussion sur un thème donné : Chaque utilisateur peut lire à tout moment les interventions de tous les autres et apporter sa propre contribution sous forme d'articles.

News : voir NetNews.

N-grams : groupement de mots ayant un sens commun.

Parser : en anglais, « parse » signifie diviser des phrases en éléments simples qui peuvent être analysés. Par exemple « parser » cette phrase reviendrait à la diviser

en mots et à identifier la nature de chaque élément (nom, verbe, adjectif,...). Le parsing est une étape très importante dans de nombreux domaines liés à l'informatique. Par exemple, les compilateurs doivent parser le code source afin de pouvoir le transformer en code exécutable. Le parsing est souvent divisé en analyse lexicale et sémantique. L'analyse lexicale consiste à diviser les chaînes de caractères en éléments appelés « tokens » (jetons) basés sur la ponctuation et d'autres éléments clés. L'analyse sémantique tente ensuite de déterminer le sens de la chaîne précédemment traitée. De la même manière, toute application qui traite des commandes complexes doit être capable de parser ces commandes. Cela concerne pratiquement toutes les applications destinées à l'utilisateur final.

POST : fonction du protocole HTML pour demander à un serveur Web d'envoyer une page Web à notre fureteur.

Prolog : langage de programmation logique.

Proxy : voir serveur-proxy.

Serveur-proxy : (serveur mandataire), dispositif informatique associé à un serveur et réalisant, pour des applications autorisées, des fonctions de médiation, telle que le stockage des documents le plus fréquemment demandés ou l'établissement de passerelles.

Socket : (connecteur logiciel), mécanisme logiciel de communication entre processus informatiques, souvent utilisé entre une application et un réseau.

SQL : (Structured Query Language, traduisez Langage de requêtes structuré) est un langage de définition de données (LDD, ou en anglais DDL Data Definition Language), un langage de manipulation de données (LMD, ou en anglais DML, Data Manipulation Language), et un langage de contrôle de données (LCD, ou en anglais DCL, Data Control Language), pour les bases de données relationnelles.

STC : « Suffix Tree Clustering », algorithme qui crée des regroupements basés sur les phrases partagées dans différents documents.

StopWord : (Mot Vide) mot qui, pris isolément n'est pas porteur d'information. Il n'est pas admis dans le langage documentaire et n'apparaît donc jamais à titre d'élément de classement ou de recherche.

SUBMIT : fonction du protocole HTML pour envoyer des informations à un serveur Web.

Swing : librairies des fonctions graphiques de Java.

TAG : balises HTML, voir HTML.

Telnet : utilitaire permettant l'utilisation de programmes sur des machines distantes, via un réseau de type Internet.

URL : « Uniform Ressource Locators », dénomination unique à caractère universel qui permet de localiser une ressource ou un document sur l'Internet, et qui indique la méthode pour y accéder, le nom du serveur et le chemin à l'intérieur du serveur.

Visual Basic : langage de programmation.

Web : dans l'Internet, système, réparti géographiquement et structurellement, de publication et de consultation de documents faisant appel aux techniques de l'hypertexte.

XML : (entendez eXtensible Markup Language et traduisez Langage à balises étendu, ou Langage à balises extensibles) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises (markup).

Introduction

Notre façon de nous informer a changé au cours de la dernière décennie. Bien que les journaux ou la télévision fassent toujours partie des sources d'informations privilégiées, plusieurs ont intégré l'Internet comme outil de communication dans les milieux de recherche et dans la vie de tous les jours. En se connectant à l'Internet, il est possible de parcourir le Web et ses innombrables sites afin de s'informer par le biais de textes, d'images, de sons ou de vidéos. L'Internet permet une interactivité nouvelle, parce que l'usager peut décider de s'informer sur ce qu'il lui plaît quand il lui plaît. Le Web lui offre des possibilités quasi infinies de sources d'informations. Les courriels, les News, le commerce électronique, le clavardage ne sont que quelques exemples des innovations qui sont utilisés sur l'Internet. Le Web est d'envergure mondial, et de plus en plus de sites Web font leur apparition chaque jour. La croissance du Web est exponentielle et cela apporte son lot d'avantages et de désavantages

La croissance du Web nous offre une quantité incroyable d'informations, mais en retour, comment l'usager peut-il retrouver aisément les informations recherchées ? Ce sont les engins de recherche qui facilitent la recherche d'informations et qui aident les usagers à ne pas se perdre dans la grande toile d'araignée qu'est le Web. Nous allons présenter les principaux engins de recherche au chapitre 2.

Cependant, bien que ceux-ci soient très utiles, les engins de recherche font un travail limité. L'usager doit souvent parcourir longuement les résultats avant de trouver un site pertinent. En effet, le monde virtuel qu'est le Web est développé par

une infinité de personnes ayant toutes et chacune différentes personnalités. Ce fait rend la structure du Web non homogène et très à l'image de l'ensemble de ses créateurs, mais malheureusement difficilement traitable par des systèmes informatiques. Comment pouvons-nous alors penser pouvoir mettre en place des engins de recherches qui retourneraient directement une information pertinente à 100 % à l'usager, alors qu'il est concrètement impossible d'analyser et d'interpréter par la science de l'informatique l'ensemble du Web ? Il semblerait alors que les engins de recherche ne soient pas la solution idéale. C'est pour remplir la tâche que les engins de recherche n'ont pas su remplir que les **agents de recherche intelligents** ont fait leur apparition.

Né de l'intersection entre la science de l'apprentissage machine et la science de la recherche d'information, les agents intelligents utilisent diverses méthodes pour essayer de répondre aux questions fondamentales que se posent les concepteurs d'agents intelligents face au Web :

- **Compréhension** : À quel niveau les agents intelligents peuvent-ils comprendre l'information proposée sur le Web ?

- **Utilité** : Est-ce que la compréhension de l'agent intelligent est assez bonne pour donner une efficacité supplémentaire à la solution proposée par les engins de recherche traditionnels ?

- **Adaptation** : Les agents existants dépendent d'une interface codée manuellement pour interagir avec l'Internet et les sites Web [1]. Est-il possible pour un agent intelligent d'approcher des sites non familiers et en extraire automatiquement les informations pertinentes à sa compréhension ?

- **Contraintes environnementales** : Quelles propriétés d'un site Web sont sous le contrôle des compétences d'un agent intelligent ? Est-ce que la compréhension du langage naturel est une complication nécessaire ? Combien de connaissances de base sont nécessaires pour assurer son utilité ?

Les travaux actuels sur les agents intelligents nous proposent plusieurs solutions pour les problèmes présentés ci-dessus. De même, les agents intelligents proposent différentes solutions pouvant combler les failles des engins de recherche, soient : la personnalisation des recherches d'information ainsi que le filtrage selon les goûts de l'usager. Ces deux solutions sont implantées par beaucoup d'interfaces intelligentes, ainsi que par des agents intelligents et elles ont pour but de permettre à l'usager d'obtenir des résultats plus liés à ses intérêts personnels. En effet, lors d'une recherche sur le Web, l'agent intelligent filtrera les résultats selon les besoins/désirs personnels de l'usager. En rendant l'ensemble des résultats plus pertinents pour l'usager, les agents intelligents permettent une économie de temps de recherche pour l'usager, ce qui est un net avantage par rapport aux engins de recherches traditionnels.

Dans ce présent mémoire, nous nous intéresserons au domaine de la recherche sur le Web par le biais d'agents. Puisque la vente et l'achat de biens sur Internet sont des domaines en pleine expansion, nous avons cru intéressant de créer un agent intelligent pouvant efficacement effectuer la recherche de biens sur les sites de marchands du Web. Nous concentrerons donc nos efforts à fournir à l'usager un agent de recherche interactif permettant de trouver plus rapidement et de manière personnalisée aux goûts de l'usager des produits sur le Web. Nous espérons ainsi faire économiser à l'usager le temps qu'il pourrait passer à chercher autrement sur le Web les différents sites de marchands et nous voulons maximiser les chances que l'usager trouve ce qui lui plaît. Nous visons à concevoir un agent intelligent qui aurait les outils nécessaires pour s'adapter aux différents sites de vendeurs et aux différents produits disponibles, mais qui limiterait les résultats aux produits présentant une certaine pertinence pour l'usager. Nous voulons que l'usager puisse ajouter différents marchands à une liste sans contrainte et puisse effectuer des recherches sans limitation quant au type de produit recherché. Ce mémoire introduit Magellan, un agent intelligent pour simplifier les achats sur Internet.

Voici quel sera le plan du présent mémoire : nous allons tout d'abord au chapitre 2 examiner les systèmes de filtrages et de recherches classiques. Au chapitre 3, nous allons analyser les systèmes à base d'agents intelligents actuellement utilisés pour la recherche sur le Web, puis au chapitre 4, nous définirons l'architecture de notre agent intelligent, Magellan. Le chapitre 5 nous présentera l'implantation et l'expérimentation de l'agent Magellan. Nous conclurons et discuterons des améliorations et des projets au chapitre 6.

Chapitre 2

Les systèmes de filtrage d'information

2.1 Introduction aux engins de recherche

La recherche sur le Web est une des tâches les plus utilisées par les usagers du Web. À cause de l'immense quantité d'information, c'est aussi une des tâches les plus frustrantes. Et cette situation ne va pas en s'améliorant, car le Web continue de grandir. Cette quantité d'information non homogène est difficilement traitable par n'importe quel moyen informatique. Les techniques traditionnelles pour rechercher l'information utilisent les mots clés. Un engin de recherche utilise les mots clés entrés par l'usager pour établir les liens vers les pages Web susceptibles de répondre selon certains algorithmes à la requête effectuée par l'usager. Cette approche retourne en général trop d'informations à l'usager dont seulement une petite partie est pertinente. La classification est aussi un problème avec les engins de recherche traditionnels. Les documents pertinents n'apparaissent pas nécessairement en début de liste et l'usager doit généralement fouiller par lui-même l'ensemble des résultats pour essayer de trouver le ou les document(s) recherché(s). Ce chapitre nous apporte un résumé des techniques de recherche disponible sur le Web.

Un système de recherche sur le Web est censé comporter certaines fonctions pour effectuer des recherches efficaces sur le Web. Selon [2], un système de recherche doit répondre par ordre d'importance, aux besoins suivants :

- efficacité à localiser et à ordonner les documents Web recueillis ;

- offrir une couverture complète du Web ;
- posséder des informations à jour sur le Web ;
- permettre un accès non biaisé aux pages Web ;
- posséder une interface facile à utiliser qui permet aux usagers de composer des requêtes raisonnables ;
- donner des résultats de recherche expressifs et utiles ;
- être un système qui s'adapte bien à la requête de l'usager.

2.2 La structure d'un engin de recherche

Les engins de recherches peuvent se classer dans deux catégories selon leur approche : les engins de recherche authentiques et les répertoires.

- Les **engins de recherche authentiques** créent eux-mêmes leurs listes automatiquement. L'engin de recherche HotBot (http://www.hotbot.com) est un exemple.

- Les **répertoires**, quant à eux, dépendent de l'action humaine pour créer leurs listes. Comme exemple, mentionnons Yahoo (http://www.yahoo.com).

Il existe aussi une forme hybride qui est un mélange entre les deux. Cette dernière nécessite une action humaine (comme les répertoires) ainsi qu'une partie automatique (comme les engins de recherche authentiques).

Les engins de recherche sont composés de trois éléments soit : le « crawler » ou araignée de balayage, le logiciel d'indexage, et le logiciel de recherche et de positionnement.

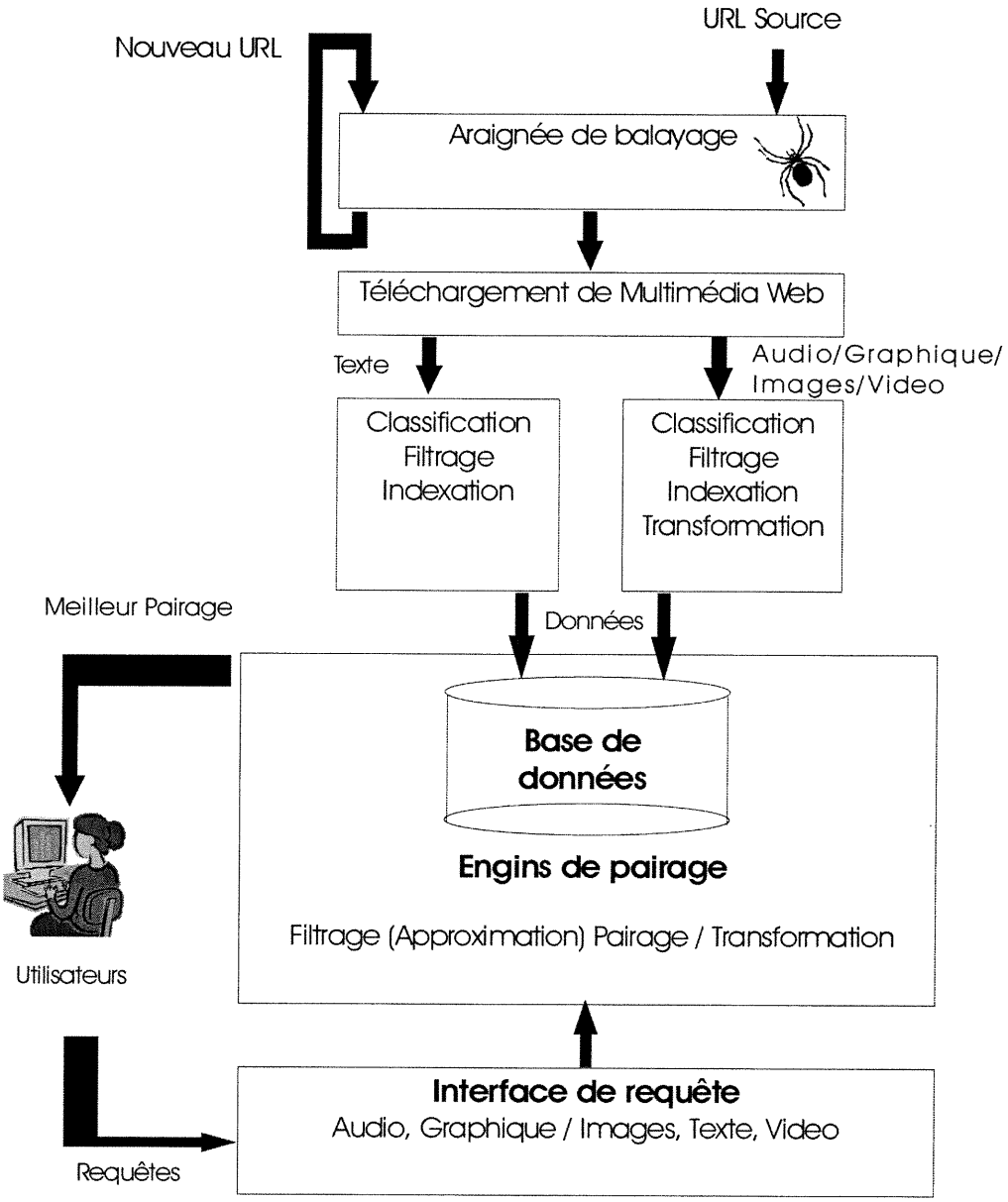

Figure 2.1 - Structure d'un engin de recherche [2].

2.2.1 Le « Crawler »

Le « crawler » est un programme qui parcourt automatiquement les différents sites Web et récolte des documents parmi ceux-ci. Il parcourt les pages Web comme un arbre dont chacun des liens URL (« Uniform Ressource Locators ») est une branche. Ce parcours est effectué selon un algorithme de recherche « largeur d'abord » ou « profondeur d'abord ». Les algorithmes de recherche « largeur d'abord » ou « profondeur d'abord » utilise une représentation en arbre sur lequel ils effectuent leur parcours en regardant tous les fils d'un nœud (« largeur d'abord ») ou en parcourant toute la profondeur jusqu'à atteindre une feuille avant de passer au fils suivant (« profondeur d'abord »). Chacun des éléments qui sont référés par les liens peut être des objets HTTP (« HyperText Transfert Protocols »), FTP (File Transfer Protocols), courriels (e-mail), news, telnet, ou autres. Le « crawler » est donc l'élément qui permet de bâtir la liste de l'engin de recherche. Le « crawler » doit parcourir plusieurs fois un même site pour effectuer une mise à jour des informations de l'engin de recherche sur une période de temps.

2.2.2 Le logiciel d'indexation

L'indexation consiste à traiter les informations et à les examiner pour en ressortir une structure qui sera facilement utilisable pour effectuer les recherches de l'engin de recherche. L'indexation débute d'abord par le filtrage, une étape cruciale. Le filtrage est nécessaire pour transformer les informations d'une manière standardisée et simplifiée pour permettre les manipulations et la recherche future sur l'index qui est généré. Ce n'est pas nécessairement l'ensemble du contenu de chaque

page Web qui est traité pour l'indexation. Dans la plupart des cas seulement certaines parties sont traitées pour minimiser le temps de traitement.

2.2.3 Le logiciel de recherche et de positionnement

Cette composante utilise l'index pour rechercher les informations qui satisfont à la requête de recherche entrée par l'usager. L'usager entre une série de mots séparés par des opérateurs booléens dans l'engin de recherche. Ensuite, le logiciel de recherche trouve les pages Web parmi l'index qui satisfont aux mots entrés. Le logiciel ordonne les résultats obtenus en fonction d'une valeur de pertinence déterminée par un algorithme de sélection ou de filtrage. Un exemple d'algorithme utilisé pour ordonner les sites Web serait un algorithme qui permettrait à l'engin de compter le nombre d'occurrences des mots entrés dans la page. L'exemple précédent peut même être grandement optimisé si l'engin ne fait qu'analyser les mots situés dans les « Métatags » (voir glossaire). Nous allons donc discuter dans les sous-chapitres suivants de quelques méthodes utilisées pour rechercher et ordonner des documents sur le Web.

2.3 L'exploration des hyperliens

Les hyperliens dans les documents Web sont une source d'information pour un engin de recherche. En effet, en analysant les liens des documents il est possible de déterminer ce que l'on pourrait appeler les « autorités » et les « hubs » [2].

Les **autorités** sont la meilleure source d'information sur un certain sujet (par exemple, le site Web officiel des Expos de Montréal est une autorité pour l'information sur les Expos).

Les « **Hubs** » sont les fournisseurs de liens qui mènent aux autorités (par exemple, les sites des fans des Expos de Montréal sur lesquels figurent un hyperlien vers le site Web officiel des Expos).

Si un même lien est présent sur plusieurs « Hubs », alors on en déduit que celui-ci mène à la page d'une autorité. Un calcul [2] est utilisé pour calculer la pertinence de l'autorité et des « hubs » pour un sujet donné. Par conséquent avec ce calcul, l'engin pourra, selon les mots recherchés, placer les pages Web les plus pertinentes en début de liste.

2.4 Récupération d'information

La récupération d'information est très utilisée dans la recherche de documents sur le Web. Parmi eux, les techniques les plus populaires selon [2] sont le « Relevance feedback » et le « data clustering ».

« Relevance Feedback »

Utilise les informations fournies par l'usager pour modifier la pertinence des pages Web retournées. Cette méthode peut ainsi éviter de retourner comme résultats les pages Web ayant été jugées, pour une certaine requête, moins pertinentes pour l'usager. Certaines autres méthodes modifient la requête effectuée par l'usager ou modifient l'index de l'engin de recherche. Ceci permet de modifier les résultats retournés par l'engin de recherche pour une série de requêtes similaires.

« Data clustering »

Cette technique permet de rendre les résultats plus pertinents à l'usager en classant les données des sites Web par catégorie. Chacune des catégories contient des éléments similaires. Ces catégories regroupent les documents Web qui sont pertinents pour les requêtes des usagers. Certains algorithmes tels que STC (« Suffix Tree Clustering ») [3] se basent sur les phrases partagées parmi les différents documents Web pour bâtir ses catégories. D'autres utilisent les requêtes des usagers pour bâtir les catégories selon les résultats pour plusieurs requêtes similaires. Plusieurs autres algorithmes existent pour catégoriser en paquets les informations.

2.5 Métarecherche

Les métarecherches comblent le besoin là où les engins de recherche classiques ne le peuvent. En effet, comme les engins de recherche ne partagent pas les informations entre eux, certaines informations pouvant être critiques ne seront pas offertes par tous. Par exemple, certains sites offerts sur les résultats d'un engin de recherche ne seront peut-être pas offerts sur un autre et si ces sites sont ceux recherchés par l'usager, il devra effectuer ses recherches sur plusieurs engins de recherche pour espérer en obtenir dans ses résultats. Donc, la solution proposée par les engins de métarecherche est d'effectuer la requête de recherche sur plusieurs engins de recherche simultanément, de regrouper les résultats, de les filtrer et de les trier selon différents algorithmes pour ensuite les présenter à l'usager. Ces systèmes n'utilisent pas en général leur propre index de recherche, mais se basent sur les systèmes existants pour recueillir leurs résultats. Ceci évite aux usagers d'utiliser plusieurs engins de recherche pour trouver l'information recherchée. La figure 2.2 de [2] montre la structure d'un système de métarecherche qui consiste en trois composantes selon [2]:

- **Expédition** (« Dispatch » sur la figure 2.2) : détermine à quels engins de recherche une requête sera envoyée. La sélection est généralement basée sur les disponibilités du réseau, des ressources locales et des performances à long terme de l'engin de recherche pour certains termes de la requête.

- **Interface** (« User Interface » sur la figure 2.2) : transforme la requête de l'usager en format pour satisfaire chaque engin de recherche où elle sera expédiée. Cette transformation varie d'un engin à l'autre.

- **Aperçu** (« Display » sur la figure 2.2) : les résultats bruts récoltés par les différents engins de recherche sont regroupés pour être présentés à l'usager. Chaque engin de recherche produit différents résultats des autres engins de recherche et ceux-ci doivent être combinés dans un format uniformisé pour faciliter son utilisation.

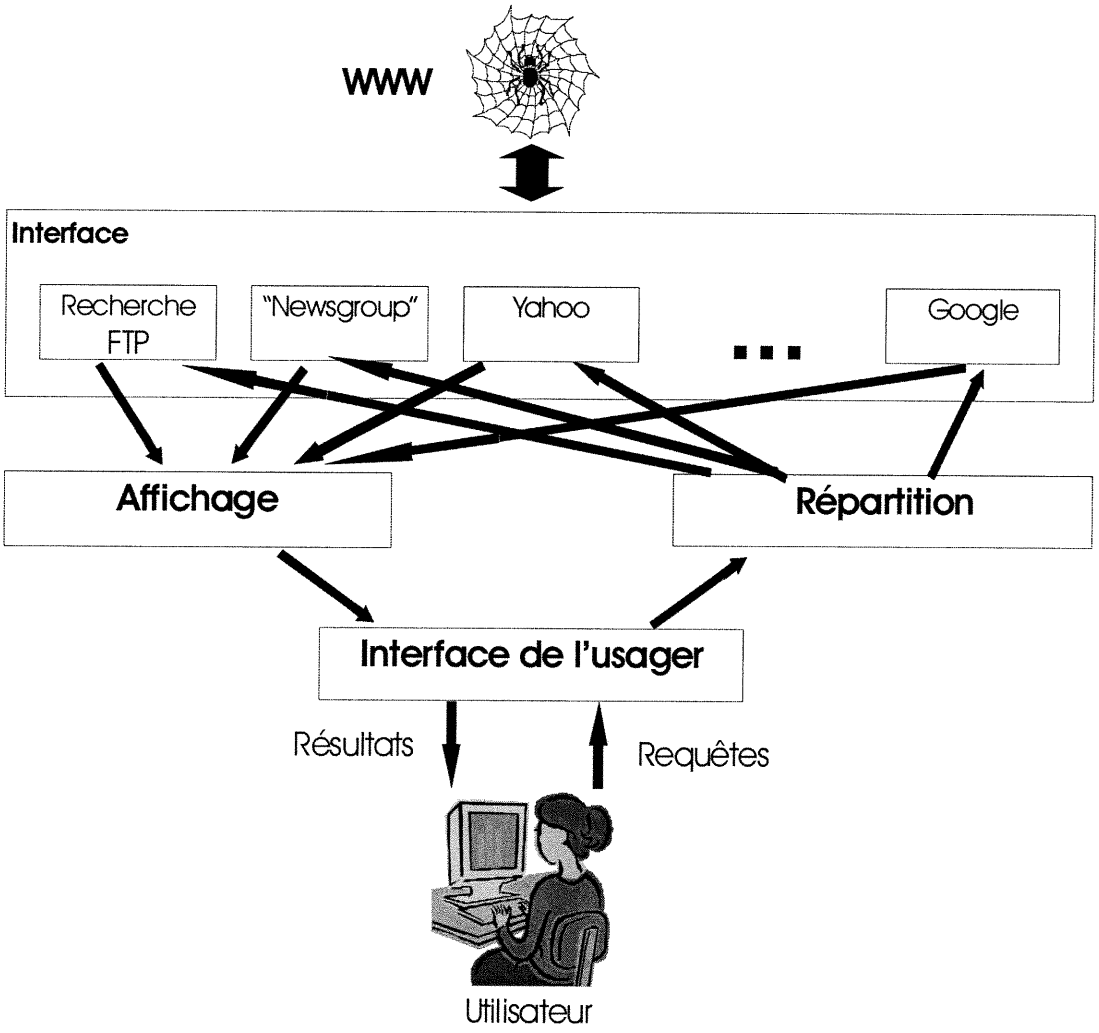

Figure 2.2 - Structure d'un engin de métarecherche [2].

2.6 L'approche SQL

Ce système utilise une forme de SQL (« Structured Query Language ») utilisée dans les bases de données. Le système considère le Web comme une énorme base de données et y effectue les requêtes SQL qui sont envoyées par l'usager. Cette

méthode est très compliquée à utiliser pour le commun des mortels et n'a pas reçu un grand succès.

2.7 La recherche de multimédia

Cette méthode de recherche est l'une des plus difficiles à effectuer. En effet, les différents supports vidéo, audio et image, sont difficilement traitables par les algorithmes de reconnaissance. Il est possible de catégoriser du texte mais c'est plus difficile de trouver des images, des vidéos ou des sons qui peuvent être reconnus et associés à une requête de l'usager sans contexte textuel. De nombreuses procédures et techniques ont été développées pour permettre une indexation sur les médias graphiques, de sons et de vidéo. Mais les nécessités en performances et en ressources dépassent de beaucoup les possibilités actuelles. Et le grand défi dans la recherche du multimédia réside dans l'indexation de supports sonores.

2.8 Les recherches de domaines spécifiques

Cette méthode a été développée dans le but d'améliorer les performances des engins de recherche. En effet, tel qu'il a été décrit dans [4], le « crawler » d'engins de recherches de domaines spécifiques explore seulement les documents et les liens qui peuvent être classifiés dans le domaine particulier de l'engin de recherche. Le « Crawler » évite ainsi de parcourir l'ensemble du Web et il se limite aux régions spécifiques à son domaine. Ceci permet de ménager beaucoup de ressources matérielles et réseautiques. Cette méthode permet aussi d'avoir des informations mises à jour plus rapidement.

2.9 Les autres

D'autres méthodes ont aussi été développées, autres que celles mentionnées ci-dessus [2] :

- une interface en langue naturelle pour rendre le système plus facile à utiliser ;

- des travaux ont été mis de l'avant pour rendre les composantes nécessaires à la recherche sur le Web plus efficaces, comme des algorithmes plus performants ou des « crawler » plus efficaces ;

- d'autres travaux plus axés vers la récupération d'information dans les documents XML (eXtensible Markup Language). Ils utilisent une technique de comparaison de structure approximative pour trouver les documents XML similaires en structure et en contenu.

2.10 Les principaux engins de recherche

Le tableau 2.1 pris de [2] nous montre les principaux engins de recherche commerciaux :

#	Nom	URL	Type	Sauvegarde	Méthode
1	AOL Search	http ://search.aol.com/	ER Hybride	Répertoire Ouvert	
2	AltaVista	http ://www.altavista.com/	ER	Répertoire Ouvert	
3	Ask Jeeves	http ://www.askjeeves.com/	SR		Langue Naturelle
4	Direct Hit	http ://www.directhit.com/	ER		Hyperlien
5	Excite	http ://www.excite.com/	ER		
6	FAST Search	http ://www.alltheweb.com/			Aptitude à passer à l'échelle
7	Go / Infoseek	http ://www.go.com/	ER Hybride		Recherche «ESP»
8	GoTo	http ://www.goto.com/		« Inktomi »	
9	Google	http ://www.google.com/	ER		Hyperlien
10	HotBot	http ://www.hotbot.com/	ER Hybride	« Direct Hit »	
11	IWon	http ://www.iwon.com/	ER Hybride	« Direct Hit »	
12	Inktomi	http ://www.inktomi.com/	ER		
13	LookSmart	http ://www.looksmart.com/	Répertoire	« Inktomi »	
14	Lycos	http ://www.lycos.com/	Répertoire	Répertoire Ouvert	
15	MSN Search	http ://search.msn.com/	Répertoire	« Looksmart »	
16	Netscpae Search	http ://search.netscape.com/	ER	Répertoire Ouvert	
17	Northern Light	http ://www.nothernlight.com/	ER		filtrage
18	Open Directory	http ://dmoz.org/	Répertoire		Volontaire
19	WebCrawler	http ://www.webcrawler.com/	ER		
20	Yahoo	http ://www.yahoo.com/	Répertoire	« Google »	

Tableau 2.1 - Les principaux engins de recherche commerciaux. ER : Engin de recherche, et SR : Service de réponses.

2.11 Conclusion

La grande expansion du Web laisse perplexe ses utilisateurs ainsi que les concepteurs d'engin de recherche. En effet, comme nous l'avons vu, les multiples solutions proposées pour améliorer les résultats de recherche ne peuvent démontrer qu'une chose : aucun des engins de recherche ne couvre efficacement l'immensité de l'information publiée sur le Web.

Le but de ce projet de maîtrise étant de trouver une solution permettant à un usager de rechercher des informations sur différents produits sur le Web, les liens

avec les systèmes existants nous apparaissent clairs et fondés. Il nous sera nécessaire de pouvoir rechercher les différents produits sur les différents sites Web des marchands et pour cela, les techniques des engins de recherche nous seront utiles.

Comme chaque marchand sur le Web dispose en général de son propre engin de recherche restreint à son catalogue seulement, la méthode de la métarecherche est toute indiquée pour nous permettre d'exécuter notre recherche sur plusieurs sites Web en même temps.

Évidemment, comme chacun des produits peut être classé par catégorie, ceci nous laisse entrevoir des possibilités pour le principe de la technique du « data clustering ». En effet, si l'usager ou l'agent arrivent à classer les produits en catégories nous pourrions limiter les recherches à certains marchands associés à la catégorie. Ceci aurait pour but d'améliorer les performances en se basant sur une technique de recherche éprouvée.

Quoique proches de notre but, les engins de recherche actuels ne satisfont pas entièrement nos besoins. Plusieurs autres moyens ont été inventés pour répondre aux lacunes des engins de recherche. Une idée née de la science de l'intelligence artificielle liée aux techniques de récupération de l'information a donné le jour aux systèmes à base d'agents [5]. Nous allons donc les explorer au chapitre 3.

Chapitre 3

Les systèmes à base d'agent

3.1 Introduction

Comme nous l'avons vu dans le chapitre 2, les engins de recherche ont quelques difficultés à traiter l'ensemble du Web. Mais comment les blâmer avec un monde virtuel où la quantité d'informations à traiter augmente de façon exponentielle chaque jour. Serait-il possible de rendre plus intelligents les systèmes de recherche classiques ? Si oui, il y aurait une possibilité d'améliorer les performances des systèmes de recherche classique. Pour essayer de palier aux différentes difficultés que les engins de recherche traditionnels ne pouvaient résoudre, l'intelligence artificielle a proposé plusieurs solutions. Ces solutions se présentent sous forme d'agents et d'interfaces intelligentes. Grâce à différentes techniques d'apprentissage machine et des algorithmes d'apprentissage, les agents et interfaces machines offrent un nouvel univers pour explorer le domaine de la recherche sur le Web. Nous allons donc discuter des travaux qui ont été effectués et des techniques utilisées dans le domaine de l'apprentissage machine pour les agents intelligents.

La prochaine section nous présente deux méthodes d'apprentissage machine utilisées par des agents intelligents. Ensuite, nous allons décrire différents agents existants qui aident les usagers à effectuer leurs recherches sur le Web.

3.2 L'apprentissage Machine et la recherche sur le Web

Les agents intelligents sont par définition des systèmes qui aident l'usager. La définition d'agent est, semblerait-il, sujette à discussion, mais tous s'accordent sur le point mentionné plus haut, l'usager doit en bénéficier. Notre problème consiste à aider l'usager à trouver l'information la plus pertinente possible sur le Web. Avec ce but en tête, les agents qui tentent de résoudre ce besoin utilisent pour la plupart deux approches qui sont : la collaboration entre agents et l'analyse de contenu.

3.2.1 L'analyse de contenu texte

C'est la science de la recherche d'information qui a inspiré les concepteurs de cette méthode. Cette méthode consiste à trouver les documents qui peuvent représenter un intérêt pour l'usager selon leurs contenus. En utilisant des méthodes de classification de textes, il est possible de trouver les similarités entre le contenu des documents que l'usager trouve pertinents et les autres. Cette méthode permet un filtrage des informations plus personnalisé aux goûts de l'usager.

Tel que décrit dans [5], cette méthode n'est pas sans inconvénient. Ce ne sont pas tous les différents aspects du contenu qui peuvent être traités : les films, les supports audio ou vidéos, les images, etc. sont en effet exclus. Et même, le contenu textuel n'est parfois traité que partiellement. Plusieurs méthodes d'apprentissage de texte peuvent être utilisées lorsqu'elles sont appliquées au contenu textuel.

Toutes les méthodes analysent le contenu d'un document et selon les différentes techniques, peuvent déterminer les documents qui sont similaires pour les

proposer à l'usager. Voici donc quelques agents qui utilisent la méthode d'analyse de contenu :

Webace, par Eui-Hong Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher et Jerome Moore [6] : un système qui explore et catégorise les documents sur le Web. Le système catégorise automatiquement un ensemble de documents, utilise une méthode pour générer de nouvelles requêtes qui seront utilisées pour rechercher de nouveaux documents. Ces nouveaux documents sont filtrés pour ne conserver que ceux qui sont le plus étroitement reliés à l'ensemble de départ.

WebWatcher, par ArmStron et cie.[7] : un système qui aide l'usager à trouver de l'information sur le Web en prenant des mots clés donnés par l'usager. Il suggère les hyperliens dans les pages Web. Il reçoit aussi des évaluations de la part de l'usager. Toutes les informations sont contenues dans une base de données (les mots clés, les liens parcourus et l'évaluation).

Lira, par Balabanovic and Shoham [8] : un système qui effectue des recherches sur le Web. Il recueille les meilleures pages pour les présenter à l'usager. L'usager évalue ensuite les résultats pour permettre au système de modifier ses critères de recherche et de sélection.

Musag, par Goldman et cie. [9] : un système qui utilise des mots clés pour effectuer des recherches sur le Web. Le système crée un thésaurus pour relier les

concepts qui sont similaires entre eux. Ce thésaurus est utilisé pour compléter les mots clés entrés par l'usager au moment d'effectuer une recherche.

Letizia, par Lieberman [10] : un agent d'interface qui analyse les documents visionnés par l'usager et suggère les liens qui pourraient être pertinent pour l'usager selon les documents Web qui ont été fréquentés par l'usager. Letizia recherche en largeur d'abord pour permettre à l'usager de cibler les liens ayant une pertinence directe avec son historique de documents visionnés.

Personal WebWatcher, par Mladenic [11] : un système qui marque de jaune les liens qui peuvent intéresser l'usager au cours de ses séances de furetage sur le Web. Le système bâtit un profil basé sur l'analyse du contenu des pages Web recherchées par l'usager.

Syskill & Webert, par Pazzani et cie. [12] Ackerman et al. [13] [14]: un système qui utilise les évaluations de l'usager sur les pages Web explorées pour construire le profil de l'usager. Chaque page Web est classée par catégorie tout comme les profils associés à celles-ci. Chaque profil est utilisé pour effectuer une métarecherche sur les engins de recherche connus pour y retirer d'autres documents pouvant potentiellement intéresser l'usager.

WAWA, par Shavlik et Eliassi [15]: un système qui utilise les préférences entrées par l'usager pour entraîner un réseau de neurones. Il utilise certaines

techniques de révision de théories pour raffiner les informations entrées par l'usager.

CiteSeer, par Bollaker et cie. [16] : Un système qui permet de trouver des articles scientifiques sur le Web. Ce système utilise les mots clés entrés par l'usager pour rechercher les documents pertinents. Il suggère ensuite des documents similaires selon les recherches effectuées par d'autres usagers et les citations contenues dans le document primaire.

News Weeder, par Lang [17] : un système qui utilise une méthode de classification de texte pour filtrer les nouvelles électroniques (NetNews) selon le profil de son usager. Il utilise sa propre interface qui ressemble à un fureteur pour permettre à l'usager de donner son évaluation des documents et ainsi donner un moyen de raffiner le profil.

ContactFinder agent, par Krulwich et Burkey [18] : un système qui assiste l'usager dans les babillards électroniques. L'agent lit et répond aux différents messages du babillard, recommande les usagers aux autres usagers qui peuvent les aider et classe les messages et en extrait le sujet.

FAQFinder, par Hammond et cie. [19] et Burke et al. [20] [21]: un système qui aide les usagers à trouver réponse à leurs questions. Les usagers utilisent une interface et y entrent les questions en langue naturelle. Les questions contenues dans les différentes FAQ (Foire aux questions ou « Frequently asked questions ») sont

comparées à la question de l'usager pour présenter à l'usager les cinq meilleures questions et réponses contenues dans les FAQ.

Antagomy, par Kamba et cie. [22]: un système qui construit un journal personnalisé à l'usager avec les différentes nouvelles du jour présentent sur le Web. Le système utilise des observations sur le comportement de l'usager sur différents articles de nouvelles pour bâtir un profil. Les différents articles sont ensuite comparés au profil pour trouver les articles ayant la plus grande similarité. Les articles sont ensuite placés par ordre : les plus pertinents en premier jusqu'aux moins pertinents.

Internet Fish, par LaMacchia [23]: un ensemble d'outils permettant à l'usager de trouver de l'information sur le Web. Le système permet la communication en langue naturelle avec des structures limitées. Le système permet d'effectuer des métarecherches sur le Web au moyen des engins de recherche existants et utilise un système d'évaluation des documents Web par l'usager.

Calendar Apprentice, par Mitchell et cie. [24]: un système qui aide l'usager à gérer l'emploi du temps de ses réunions. Le système est branché sur l'agenda électronique de l'usager et génère un profil de l'usager en fonction des préférences de l'usager sur ses réunions et les informations sur les individus qui assistent aux réunions. Le système propose des suggestions lors de nouveaux rendez vous.

3.2.2 Les agents de collaborations

Cette approche est en quelque sorte un apprentissage social de l'agent, basé sur l'information partagée entre les différents agents. Il est donc essentiel d'avoir une communauté d'agents qui communiquent entre eux pour que cette approche soit fonctionnelle. Les conseils donnés aux usagers se basent sur les réactions des autres usagers. Le système recherche les utilisateurs qui ont des préférences communes et recommande les articles que les autres ont appréciés. Les similarités, contrairement à l'approche d'analyse du contenu, ne se basent pas sur les ressemblances entre les différents documents, mais sur les ressemblances entre les usagers. Le contenu du document n'est aucunement analysé, seules les évaluations des différents usagers sont utilisées.

Cette approche n'est pas sans problèmes. Il est nécessaire d'avoir un bon nombre d'usagers pour que cette méthode soit efficace. Chaque usager doit évaluer un bon nombre de documents avant de pouvoir être joint à un groupe d'utilisateurs aux goûts similaires. Si un utilisateur se démarque par des préférences particulières, le système sera inutilisable par cet utilisateur car, il ne pourra être apparié avec d'autres utilisateurs.

Nous voyons les avantages à utiliser un système comme celui-ci. Les images, les sons, les films et tous les supports qui ne pouvaient être traités par la méthode d'analyse de contenu sont traitables aux agents de recherche qui se basent sur cette méthode. Voici quelques agents qui utilisent la méthode de collaboration.

Siteseer, par Rucker et Marcos [25]: Un système de recommandation de page Web qui utilise les « favoris » ou signets de l'usager comme base pour ses recommandations. Siteseer utilise les signets de chaque utilisateur comme une déclaration implicite de l'intérêt porté par l'usager pour le contenu du document lié ainsi que l'arrangement et le groupement des signets est indicateur de cohérence sémantique ou la pertinence des regroupements entre les sujets. De plus, Siteseer traite les répertoires comme un système de classification personnel qui permet de replacer dans son contexte les recommandations par classes définies.

PHROAK, par Terveen et cie. [26]: Un système qui reconnaît automatiquement et redistribue les recommandations des ressources Web saisies dans les nouvelles électroniques (« NetNews »).

GroupLens par Konstan et cie. [27]: Un système de filtrage pour les nouvelles électroniques (« NetNews de Usenet »). Le système utilise les évaluations des usagers pour les messages et établit une corrélation entre les usagers selon leurs valeurs attribuées par leurs évaluations des messages. Tel qu'il a été décrit par [5], la quantité de messages lus par usager est très faible comparée au nombre total de messages sur les « News ». Il est difficile, pour le système, de trouver les usagers compatibles, car une énorme quantité d'évaluations pour chaque usager seraient nécessaire pour couvrir tous les messages. Ce problème d'évaluations trop réparties est fréquent pour les agents de collaboration.

Referral Web par Kautz et cie. [28]: Un système interactif pour reconstruire, visualiser et rechercher des réseaux sociaux sur le Web. Ce système aide l'usager à rechercher des experts pour certaines catégories.

Lifestyle finder par Krulwich [29]: Un système qui combine l'analyse de contenu et la collaboration entre agents. Le système utilise un questionnaire pour regrouper les usagers selon des aspects démographiques. Une évaluation de la part de l'usager est utilisée pour évaluer les performances du système.

3.2.3 Les autres approches

Voici quelques autres travaux reliés à la recherche sur le Web qui ont été développés :

Un système pour **rechercher les librairies** de logiciel développé par Holte et Drummond [30] et [31]. Le système prend des mots clés tapés par l'usager et utilise un système de règles avec des inférences chaînées. Le système suppose que la librairie contient un seul type de produit et que le but de l'usager est de trouver un seul produit.

Une **interface de l'Internet** qui combine un « shell » de UNIX et le Web pour interagir avec les ressources de l'Internet. Ce système a été développé dans [32]. L'agent accepte des buts de l'usager et synthèse dynamiquement les séquences de commandes Internet pour réaliser ces buts.

Un **agent d'emploi** disponible sur le Web qui permet à l'usager de voir les informations et commander des courriels lorsque des informations sont intéressantes pour l'usager [33].

Un système développé dans [34] utilise un système qui s'adapte à l'usager pour lui permettre **d'automatiser la recherche d'information sur le Web**. Il utilise des algorithmes génétiques sur une population d'agents et les récompenses en « énergie » pour chaque document pertinent trouvé et retire de l' « énergie » pour l'utilisation des ressources réseautiques.

Shopbot par Robert B. Doorenbos, Oren Etzioni, Daniel S. Weld [35]: Un système qui effectue des recherches pour l'usager, dans le but de trouver des produits vendus sur le Web, pour permettre à l'usager de comparer les différentes offres. Les sites de vendeurs sont recherchés en parallèle et les résultats des recherches sont regroupés. Le système compare avec des modèles préprogrammés pour pouvoir distinguer les éléments de comparaisons entre un même produit (prix, description, modèle, année).

Référence	Nom de l'agent	Développé à	Le but de l'agent
Agents d'analyse de contenu			
[36]	WebWatcher	CMU	Fureter le Web
[8]	Lira	Stanford	Fureter le Web
[9]	Musag	Hebrew Uni.	Fureter le Web
[10]	Letizia	MIT	Fureter le Web
[11]	Personal WebWatcher	CMU, IJS	Fureter le Web
[12] [14]	Syskill & Webert	UCI	Fureter le Web
[15]	WAWA	Wisconsin Uni.	Fureter le Web
[16]	CiteSeer	TX, NEC corp., UMIACS	Trouver des publications sur le Web
[17]	NewsWeeder	CMU	Filtrer les News
[18]	ContactFinder	Andersen Consult.	Trouver des experts
[20] [21]	FAQFinder	Uni of Chicago	Répondre aux questions
[22]	Antagonomy	NEC corp.	Journal de nouvelles personnalisé
[23]	Internet Fish	MIT	Trouver de l'information sur le Web
[24]	Calendar Apprentice	CMU	Organisation des réunions
Agents de collaboration			
[25]	Siteseer	Imana, Inc.	Fureter le Web
[26]	PHROAKS	AT&T labs	Fureter le Web
[27]	GroupLens	Uni. Minnesota	Filtrer les nouvelles électroniques
[28] [37]	Referral Web	AT&T labs	Trouver des experts
[38]	Ringo Firefly	MIT MIT	Trouver de la musique Trouver de la musique, des films et des livres
Hybride entre analyse de contenu et collaboratif			
[39]	Fab	Stanford	Fureter le Web
[29]	Lifestyle Finder	AgentSoft Ltd.	Fureter le Web
[40]	webCobra	James Cook Uni.	Fureter le Web

Tableau 3.1 - Sommaire de quelques agents de recherche sur le Web [5].

3.3 Aperçu des méthodes d'apprentissage machine pour les données textuelles

Les agents qui utilisent les techniques d'apprentissage machine se servent de l'approche d'analyse de contenu pour apprendre. Ces techniques d'apprentissage machine ont été développées au départ pour l'analyse de document écrit et sans considération pour le Web. Elles ont trouvé une nouvelle vocation avec la création des agents de recherche sur le Web.

Le tableau 3.2 nous montre différents agents et les techniques d'apprentissage machine qui ont été utilisées pour chacun. Nous allons ensuite expliquer chaque composante du tableau.

Référence	Représentation des documents	Sélection des mots	Classification
[41]	Bag-of-words (frq)	Stop list + poids de fréquence	Règles de décision
[42]	Bag-of-words (frq)	Stemming + fréquence minimum	Arbre de décision enrichi
[36]	Bag-of-words	Informativité	TFIDF Winnow, WordStat
[8]	Bag-of-words (frq)	Stop-list + Stemming	TFIDF
[43]	Bag-of-words (frq)	LSI	-
[44] [45]	Bag-of-words (frq)	LSI (utilisant SVD)	TFIDF
[46]	Bag-of-words	_	Raisonnement basé sur la mémoire
[47]	Bag-of-words + position des mots	Fréquence minimum	Règles de decision ILP
[48]	Liste de mots ordonnés	_	Règles de décisions, Expert dormant
[49]	Bag-of-words + WordNet	Connectivité minimum	Graphe de relations sémantiques
[50]	Bag-of-words (frq)	Fréquence minimum + informativité	TFIDF, PrTFIDF, Algo. Naïf Bayes
[51]	Bag-of-words (frq)	Fréquence minimum	Machine de support vectoriel
[52]	Bag-of-words (frq)	Information mutual	Réseau Bayesien
[53]	Bag-of-words (frq)	Stop-list	Ensemble d'instance généralisé. K plus proches voisins
[54]	Bag-of-words	Stop-list + informativité	Algo. Naïf de Bayes, arbre de décisions
[55]	Bag-of-words	Trace du ratio de resemblance	Régression logistique avec algo. Naïf de Bayes
[56]	Bag-of-words (frq)	_	Windrow-Hoff, EG
[57]	Bag-of-words	_	Winnow
[38]	Bag-of-words + information d'entête	Sélection de mots clés	Résonnement base sur la mémoire
[11]	Bag-of-words (frq)	Informativité	Algo. Naïf de Bayes, Plus proche voisin
[58] [59]	Bag-of-words utilisant des n-grams (frq)	Stop-list + fréquence minimum + ratio curieu	Algo. Naïf de Bayes
[60]	Bag-of-words	Informativité	Règles de decision

[61]	Bag-of-words	Fréquence minimum	EM avec QBC
[12] [14]	Bag-of-words	Stop-list + informativité	TFIDF, Algo. Naïf de Bayes, Plus proche voisin, Réseau de neurones, Arbre de décision
[15]	Bag-of-words localisé	Stop-list + stemming	Théorie du raffinement sur le réseau de neurones
[62]	Bag-of-words + hyper-texte / graphe	Informativité	Algo. Naif de Bayes, ILP
[63]	Graphe de n-gram (seulement des bigrams)	Graphe de poids des frontières	Connexionniste combiné avec des algorithmes génétiques
[64]	Bag-of-words	Stop-list + fréquence minimum + stemming + relevance ou LSI	Réseau de neurones, Régression logistique
[65] [66]	Bag-of-words	Informativité, X^2 -stat	K-plus proche voisin, LLSF

Tableau 3.2 - Différents agents et les techniques d'apprentissage machine qui ont été utilisées pour chacun [5]. Noter que les « bag-of-words » dans le tableau 3.2 sont booléens sauf si « frq » (fréquence) est annexé.

Légendes du tableau 3.2 :

- LLSF : « Linear Least Square Fit » ou adaptation du plus petit carré linéaire,
- LSI : « Latent semantic indexing » ou indexation sémantique latente,
- ILP : « Inductive Logic Programming » ou programmation par logique inductive,
- EG: « Exponential Gradient » ou Gradient Exponentiel,
- EM: « Expectation Maximization » ou maximisation de l'espérance,
- QBC: « Query by committee » ou requête par comité,

- PrTFIDF: « Probalistic TFIDF » ou TFIDF avec probabilité et
- SVD: « Singular value decomposition » ou décomposition de valeur singulière.

La représentation des documents est la manière dont l'agent transforme un document pour pouvoir l'utiliser dans son traitement par l'algorithme d'apprentissage machine. Comme nous l'avons vu, cette représentation diffère d'un agent à l'autre. Voici donc les explications des termes utilisés dans la table :

« Bag-of-words » : cette représentation est la plus fréquemment utilisée. Tous les mots d'un document sont réunis en un vecteur sans ordre et aucune structure du texte n'est utilisée. Lorsque nous avons un ensemble de documents, le vecteur contient l'ensemble des mots de tous les documents. Quelques informations additionnelles peuvent être saisies dans le document comme : la structure des phrases, la position des mots ou les mots adjacents. Certains travaux utilisent des séquences de mots appelés « n-grams » (exemple : « intelligence artificielle » est un 2-gram, «World Wide Web » est un 3-gram).

Voici une illustration d'un « bag-of-words » selon [5] :

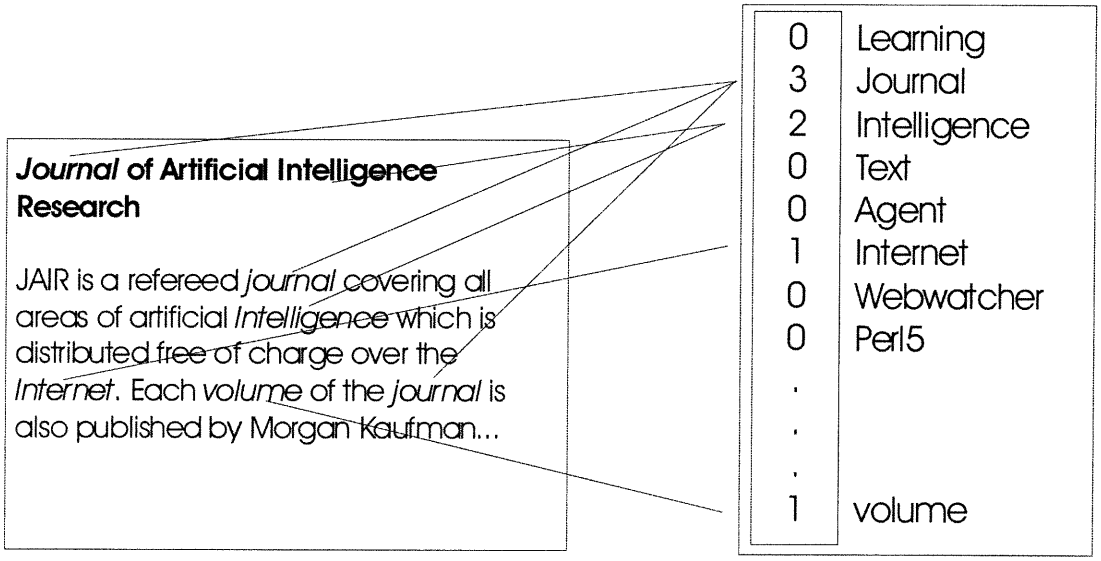

Figure 3.1 - La représentation du bag-of-words [5].

Les documents sont représentés en un vecteur ou chaque occurrence des mots y est associée à sa fréquence.

La plupart de l'utilisation du bag-of-words utilise soit des booléens pour indiquer qu'un mot est présent dans le document, soit la fréquence (frq dans le tableau 3.2) de ce mot dans le document (tel que montrée dans la figure 3.1).

Des travaux utilisent le n-gram, d'autres les structures hypertext et un graphe d'organisation des pages Web. Tel que mentionné dans l'article [5], il n'y a pas d'étude comparative ou directive sur la représentation des documents. Donc, aucune preuve ne nous indique si l'ajout d'information complémentaire à la représentation du bag-of-words apporte un gain en performance.

La quantité de mots à traiter dans le document peut être grandement réduite par certains moyens pour sélectionner les mots des documents :

- les mots qui sont utilisés le plus fréquemment dans le document et qui n'ont aucune signification pour la classification tel que « a », « the », « with » peuvent être retirés. Une liste de ces mots appelée en anglais « stop-list » est utilisée pour pouvoir facilement les reconnaître.

- les mots ayant la plus faible fréquence peuvent aussi être retirés du bag-of words. On fixe alors un seuil à franchir en terme de fréquence pour tronquer la liste du bag-of-words.

- en utilisant une langue en particulier il est possible de réduire la fréquence d'un même mot qui apparaît sous différentes formes. (par exemple, les verbes conjugués en différents temps ou différents nombres)

Plusieurs approches abordent aussi l'utilisation de méthodes indépendantes à la langue pour calculer la fréquence des mots. Certaines méthodes aussi utilisent LSI (« latent semantic indexing » [5]). La sélection des mots à inclure dans le bag-of-words est une opération critique. En effet, certaines expérimentations [5] indiquent que les meilleurs résultats sont obtenus en sélectionnant un petit pourcentage des mots (jusqu'à 10% de tous les mots) ou autrement, tous les mots.

Les algorithmes et les techniques pour la classification, tels que présentés dans le tableau 3.2 sont divers. L'utilisation de l'algorithme TFIDF (« term frequency-inverse document frequency weighting ») est très répandue. Diverses

modifications y ont été apportées pour en améliorer les performances. Mais peu importe la technique utilisée, Mladevic [5] mentionne qu'il n'existe aucune évidence concrète qui nous indique la supériorité d'un algorithme comparé à un autre. La plupart des expérimentations tente de démontrer la supériorité d'un algorithme sur TFIDF. Et les travaux décris dans [14] nous indiquent que la voie pour l'amélioration de la classification ne réside pas dans la recherche du meilleur algorithme, mais dans la méthode de sélection des mots.

3.4 Conclusion

Nous avons vu qu'une bonne quantité d'agents existent. Ils rendent la recherche sur le Web plus personnalisée pour l'usager et comble ainsi un besoin que les engins de recherche traditionnels ne pouvaient remplir. Ces agents semblent être la solution pour la recherche sur le Web.

Nous allons donc introduire notre solution pour la problématique énoncée en introduction, Magellan, un agent pour la recherche de produits sur le Web. Un agent qui va permettre à l'usager de trouver des produits sur les sites de vendeurs Web qui seront pertinents à l'usager.

Chapitre 4

Architecture de Magellan

4. 1 Introduction

Magellan est un agent connecté à l'Internet. Nous expliquerons ici comment et à quoi sert Magellan d'un point de vue global. Magellan permet à l'usager de trouver des produits qui sont susceptibles de lui plaire. Il n'est pas comme les engins de recherches connus (ex. AltaVista et Yahoo), car contrairement à ceux-ci il effectue non seulement des recherches sur le Web pour l'usager, mais il présente **les résultats filtrés selon les goûts de l'usager**. En effet, l'agent et l'usager construisent un profil ensemble. Ce profil permet de déterminer quels produits conviennent ou ne conviennent pas à l'usager. L'agent filtre donc les résultats obtenus et les trie pour simplifier la tâche de l'usager. Il réduit la quantité d'information en éliminant toutes les informations inutiles afin de ne présenter que les informations pertinentes. Nous supposons que ceci réduira le temps de recherche, car les pages n'ayant aucun lien avec le profil de l'usager ne seront pas présentées.

Imaginez que vous recherchiez une voiture sur le Web. Et bien, Magellan sait que vous aimez les voitures de marque Honda et BMW (choix hypothétiques), car vous avez déjà regardé des pages Web sur les Honda et les BMW alors que l'agent était en fonction. L'agent a aussi découvert que vous aimiez les voitures rouges. Vous donnez la commande à Magellan qui lui effectue une requête sur des sites de

marchands connus. Il vous retourne ensuite toutes les voitures susceptibles de vous plaire selon votre profil.

Nous allons maintenant faire un survol rapide des fonctions essentielles de Magellan. Ensuite, nous expliquerons chaque composante de l'architecture (Figure 4. 1) et leurs liens à ces fonctions.

Les fonctions principales de Magellan se divisent en deux parties : les **fonctions de recherches** et les **fonctions de création du profil**.

Ces deux fonctions principales se rattachent à quelques autres fonctions secondaires : la communication entre l'agent et l'usager, la communication entre les agents et la gestion des actions automatiques.

Voici l'architecture de Magellan (Figure 4. 1):

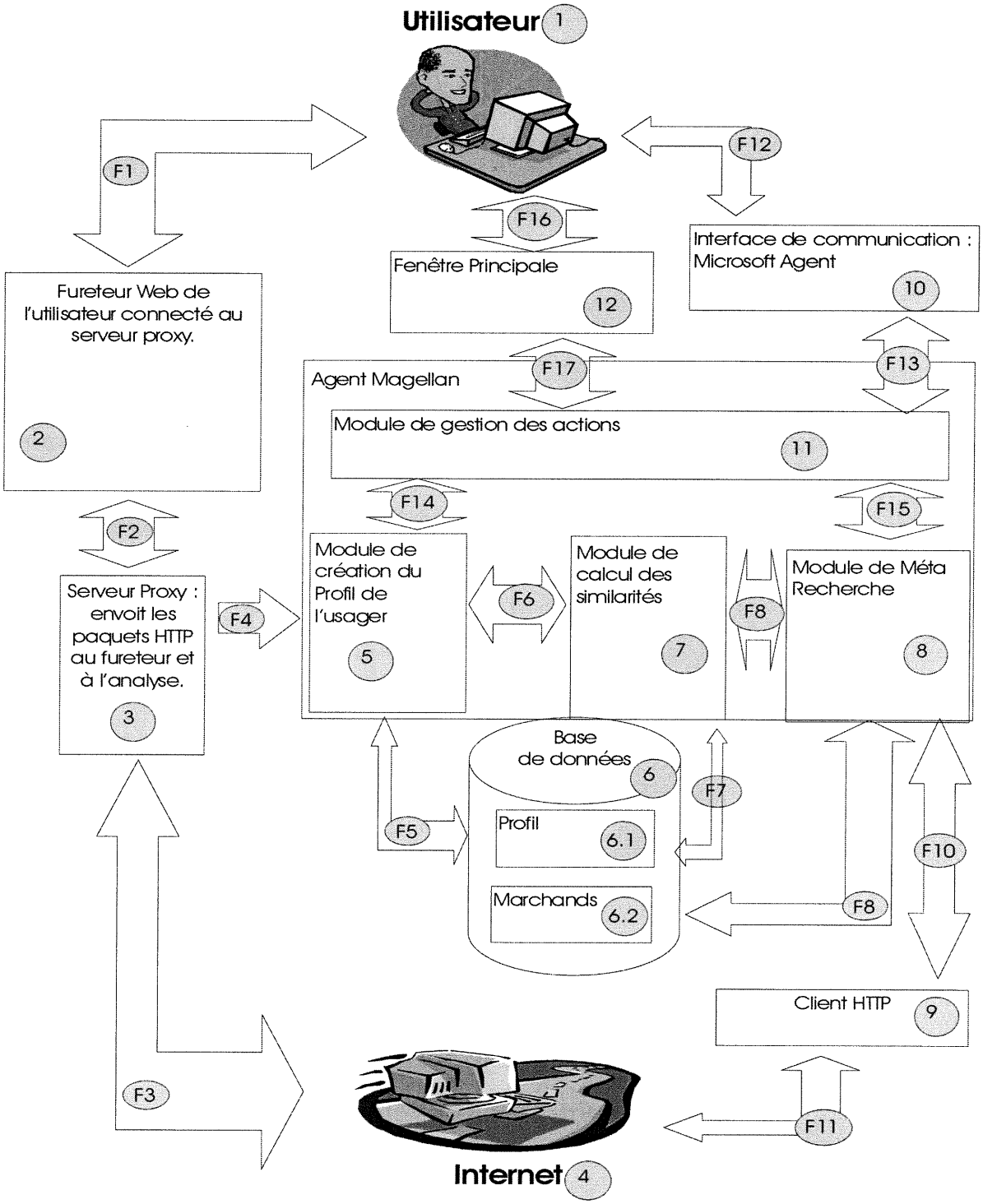

Figure 4.1 - Architecture de Magellan.

4. 1. 1 La fonction de recherche

Les **fonctions de recherche** utilisent des requêtes sur les sites de marchands afin de recueillir les résultats des recherches, les regrouper, les filtrer et les présenter à l'usager. Une liste de sites de marchands (6.2) est maintenue dans une base de données (6) et elle est mise à jour par l'agent et l'usager. L'épuration des résultats de recherches s'effectue grâce au profil de l'usager (6.1). Ce profil (6.1) permet de déterminer les informations inutiles. Cette fonction utilise le Client http (9) pour ses communications avec l'Internet (4). La fonction de recherche est effectuée par les composantes suivantes : le client http (9), le Module de métarecherche (8), la base de données (6) avec les tables de liste de marchands (6.2), le profil (6,1) et le module de gestion des actions (11).

4. 1. 2 La fonction de création du profil

La **fonction de création du profil** analyse les pages Web visionnées par l'usager afin de trouver le sujet de chaque document et de déterminer la préférence de l'usager pour ce sujet. L'usager se promène sur le Web et les pages visionnées sont analysées par l'agent. L'agent est en mesure de voir les pages Web grâce à un serveur-proxy (2) connecté entre le Web (4) et le fureteur (2) de l'usager. Ce serveur-proxy (2) intercepte les demandes du fureteur (2) et les réponses de l'Internet (4). Il copie toutes les pages Web et les envoie au module de création du profil de l'usager (5). Le module de création du profil de l'usager (5) détermine le sujet de la page Web et modifie le profil de l'usager si celui-ci lui indique qu'il porte un intérêt à ce sujet. La fonction de création de profil est effectuée avec les composantes suivantes : le serveur-proxy (2), le module de création du profil de l'usager (5), la

base de données avec la table du profil (6,1) et le module de calcul des similarités (7).

4. 1. 3 La fonction de communication

La **fonction de communication entre l'agent et l'usager** sert principalement à confirmer les modifications dans le profil et à permettre à l'usager de donner des commandes à l'agent. On définit par commande tous les ordres, les demandes et les confirmations que l'usager pourrait émettre à l'agent. La fonction de communication est effectuée par les composantes suivantes : le module de gestion des actions (11) et les interfaces de communication (10, 12).

4. 1. 4 La gestion des actions automatiques

La **gestion des actions automatiques**, quant à elle, est un moyen de faire en sorte que l'agent intervienne au bon moment. Cela signifie que l'agent n'interviendra pas n'importe quand afin de ne pas devenir une nuisance à son client (exemple classique d'un agent ennuyeux, le trombone de Microsoft). En effet, l'agent pour être utile doit être utilisé. Ennuyer l'usager signifie que l'usager va associer l'agent à une nuisance et donc en faire un usage moindre [67]. Cette fonction est effectuée par le module de gestion des actions (11).

Maintenant que nous avons présenté les fonctions principales et secondaires implémentées par Magellan, nous allons en décrire chacune des composantes. Commençons par l'essentiel, la construction du profil. Car sans le profil, la recherche ne pourrait pas être filtrée afin d'être personnalisée à l'usager.

4. 2 Le serveur-proxy (3)

Entrées : les requêtes (F2) du fureteur (2), les réponses (F3) de l'Internet (4)

Sorties :

- vers l'Internet (4) : les requêtes du fureteur (2)
- vers le fureteur (2) : les réponses reçues de l'Internet (4)
- vers le Module de création du profil de l'usager (5) : les requêtes du fureteur (2), les réponses reçues de l'Internet (4).

Le serveur-proxy (3) est un élément essentiel au fonctionnement de la création du profil. Il prend en entrée les requêtes du fureteur (2) et il les renvoie à l'Internet (4) et à l'analyseur (5). Il reçoit ensuite les réponses de l'Internet (4) et les envoie au fureteur (2) avec (F2) et à l'analyseur (5) avec (F4). Ses deux fonctions se résument à :

1) transmettre les requêtes du fureteur (2) vers l'Internet (4) et le module de création du profil de l'usager (5).

2) transmettre les réponses de l'Internet (4) au fureteur (2) et au module de création du profil de l'usager.

Voici un exemple du fonctionnement du serveur-proxy (3) (en supposant que les configurations du fureteur (2) permettent au serveur-proxy de recevoir les requêtes (F2) du fureteur, voir implantation du proxy dans le chapitre 5) : L'usager (1) exécute une requête pour une page sur le Web, il veut par exemple aller sur le site de l'Université de Montréal. Il tape http://www.umontreal.ca comme adresse dans son fureteur (2) et exécute la requête. Cette requête passe au serveur-proxy par (F2). Le serveur-proxy envoie la requête vers l'Internet (4) par (F3) et envoie en même temps une note signifiant qu'il a reçu une nouvelle requête au module de création du profil de l'usager (5) par (F4).

Une fois la requête reçue et interprétée par le serveur sur Internet (Ici umontreal.ca), ce serveur nous répond soit en envoyant l'information demandée (la page Web) ou en transmettant un message d'erreur. Cette réponse est reçue par le serveur-proxy (3) par (F3). Le serveur-proxy achemine la réponse au fureteur (2) et en même temps au Module de création du profil de l'usager (5).

4.3 Module de création du profil de l'usager (5)

Entrées : les notifications de requête du fureteur (2), les réponses de l'Internet (4), les messages du module de gestion des actions (11), le profil de la table du profil (6.1).

Sorties : les vecteurs à ajouter ou à enlever au profil de la table du profil (6.1), les vecteurs générés par le traitement des réponses de l'Internet, les messages vers le module de gestion des actions (11).

Le module de création du profil de l'usager (5) est une des deux moitiés essentielles dans le processus de la création du profil. Le serveur-proxy (3) retransmet tous les envois entre le fureteur (2) et l'Internet (4) et vice-versa. Si une réponse venant de l'Internet ne contient pas d'erreur, elle est traitée pour pouvoir ensuite possiblement modifier le profil de l'usager.

Une réponse venant de l'Internet est en général un document HTML. Magellan ne traite pas le XML, car le Parser qui nous était disponible au moment de la conception ne nous permettait que le traitement de HTML. Ce document est présenté en code HTML. Le module de création du profil de l'usager décortique le document en son ensemble pour créer un vecteur que nous appellerons TF, qui contient tous les mots texte, le titre et les entêtes de l'ensemble du document. Chaque mot a un nombre associé qui représente sa fréquence dans le document. Chaque mot n'apparaît qu'une seule fois dans le vecteur. Et le vecteur a une grandeur maximale prédéterminée qui sera appelé M.

Une fois le vecteur TF généré, le module de création du profil de l'usager envoie un message au module de gestion des actions (11) lui signalant que le vecteur attend pour être envoyé au module de calcul des similarités (7).

Le module attend un message de la part du module de gestion des actions lui signalant de continuer le traitement du vecteur TF.

Si entre temps le serveur-proxy (3) envoie une notification pour signaler la réception d'une nouvelle requête envoyée pas le fureteur (2), le vecteur TF est détruit.

Si un message provenant du module de gestion des actions (11) est reçu pour signaler de continuer le traitement, alors le module de création du profil de l'usager envoie le vecteur TF au module de calcul des similarités.

Si le nombre de vecteurs qui composent le profil est inférieur à un nombre prédéterminé N, alors le vecteur TF est ajouté au profil. Si le profil a atteint le nombre de vecteurs N (nombre maximum de vecteurs dans le profil), alors le vecteur TF est ajouté au profil. Nous calculons ensuite la valeur de similarité entre toutes les paires de vecteurs du profil. Les deux vecteurs ayant la valeur de similarité la plus élevée seront joints ensemble et les M (grandeur maximale des vecteurs) plus grandes valeurs seront conservés.

Les modifications de la table du profil sont envoyées par le module de gestion des actions et reçues par le module de création du profil de l'usager. Les modifications sur la table de profil sont traitées et appliquées sur la table du profil en conséquence.

4.3.1 Sous module : Parser HTML

Entrées : un flot de données (Data Stream).

Sorties : le document HTML décortiqué en éléments.

Le Parser HTML a une fonction très simple, il doit recevoir le flot de données du document HTML reçu et le présenter d'une manière traitable par le programme. Il décode le document HTML en éléments, chaque élément est accessible avec un énumérateur. Il est donc facile de traiter chaque TAG, paramètre, texte ou commentaire du document HTML.

4.3.2 Sous module : Module de création des vecteurs de document

Entrées : les mots à ajouter au vecteur ou les mots et leur fréquence.

Sorties : un vecteur TF traitable par le module de calcul des similarités.

Le module de création des vecteurs de document possède deux manières d'ajouter des éléments au vecteur TF. La première prend un mot en argument. Elle cherche dans le vecteur si se trouve le mot à ajouter, si oui, incrémente la fréquence de un, sinon l'ajoute au vecteur avec une fréquence de un. La seconde, prend un mot et une valeur de fréquence en arguments. La seconde manière fait la même fonction que la première, mais utilise la fréquence fournie dans ses calculs.

4. 4 Module de calcul des similarités (7)

Entrées : les vecteurs provenant du module de création du profil de l'usager (5), les vecteurs provenant du module de Méta recherche (8), les vecteurs qui composent la table du profil (6.1).

Sorties : les résultats du calcul de similarité vers le module de méta recherche (8), les vecteurs résultants du traitement.

Le module de calcul des similarités utilise l'algorithme **TFIDF** pour fournir une valeur qui indique le degré de similitude entre les deux vecteurs. Nous décrivons l'algorithme TFIDF au chapitre 5.4.1. Cet algorithme est un algorithme d'apprentissage machine utilisé pour la classification. Ceci nous permet donc de supposer que cette similitude est reflétée entre les deux documents d'où sont tirés les vecteurs. La valeur de similitude pour un document est calculée avec un algorithme, TFIDF.

Son fonctionnement de manière simplifiée est de représenter chaque document par un vecteur dans un espace de vecteurs, de sorte que deux documents semblables aient un vecteur similaire. La valeur d'un vecteur pour un document est calculée comme une combinaison statistique de la fréquence *TF(w,d)* (le nombre d'apparitions du mot *w* dans un document *d*) et de la fréquence des documents *DF(w)* (la fréquence où le mot *w* apparaît au moins une fois à l'intérieur d'un document). De la fréquence des documents, son inverse *IDF(w)* peut être calculé :

$$IDF(w) = \log \frac{|D|}{DF(w)}$$

|D| est le nombre total de documents. L'utilisation du logarithme est d'usage pour la normalisation. La valeur $d^{(i)}$ d'un élément dans le vecteur se situe entre 0 et 1 et elle est calculée comme le produit :

$$d^{(i)} = TF(w_i, d) \times IDF(w_i)$$

Un algorithme développé pour calculer les vecteurs TFIDF multiples a été développé dans [68] pour WebMate. Cet algorithme assume que l'usager a un nombre maximum de domaines d'intérêts *N*. Nous supposons que l'ensemble de profils initiaux est *V*, $|V| = 0$. Le nombre maximal de vecteurs TFIDF dans l'ensemble de profil est *N* et le nombre d'éléments dans un vecteur est *M*. Pour chaque exemple positif (un document qui intéresse l'usager) nous faisons :

1. début du processus : extraction des mots du document HTML. Nous enlevons les mots tels que : « a », « the », « is », « in », etc (« stop-words »). Nous rendons la forme des noms du pluriel à la forme simple et les verbes à leur forme originale. Le titre (<TITLE>) et les entêtes (<H1>, <H2> et <H3>) sont extraits, car plus de poids leur sera attribué ;

2. extraction du vecteur TFIDF pour ce document, appelons-le V_i ;

3. si $|V| < N$ ($|V|$ étant le nombre de vecteurs dans l'ensemble de profil V), alors $V = V \bigcup V_i$;

4. sinon, calculer le Cosinus similaire entre les deux vecteurs TFIDF en incluant le vecteur dans l'ensemble de profils V et le vecteur du nouveau document V_i. Nous présumons que l'ensemble de profils V est $\{V_1, V_2, ..., V_n\}(n = N)$.

$$Sim(V_j, V_k) = \frac{V_j \bullet V_k}{|V_j| \times |V_k|} \qquad j,k \in \{1,2,...,n,i\}$$

(V_j et V_k représente une paire de vecteurs pour l'ensemble V incluant V_i et $Sim(V_j, V_k)$ donne des valeurs entre 0 et 1)

5. combiner les deux vecteurs V_l et V_m avec la plus grande similarité :

$$V_l = V_l + V_m \quad (l,m) = \arg\max_{(x,y)}(Sim(V_x, V_y)) \quad x,y \in \{1,2,...,n,i\}$$

6. trier les poids du nouveau vecteur V_k en ordre décroissant et garder les M plus grands éléments du vecteur.

Cet algorithme est utilisé pour tous les vecteurs soumis par le module de création du profil de l'usager.

Lorsque le module de métarecherche envoie un vecteur au module de calcul des similarités, il est comparé avec les vecteurs du profil. Une fois que le vecteur du module de métarecherche a été comparé avec l'ensemble du profil, la valeur de similarité maximale est retournée au module de méta recherche. Ceci donne une valeur de similitude du vecteur avec certains éléments du profil. Cette valeur servira à déterminer si certains documents pourraient plaire à l'usager et à quel niveau.

4.4.1 Sous module : WordNet

Ce module n'est pas représenté dans la figure 4.1. Nous pouvons le considérer comme partie intégrale du module de calcul des similarités.

Le module WordNet prend en entrée les mots qui seront utilisés dans les calculs de similarité. Le module Wordnet utilise un dictionnaire et il retourne les mots à leur forme la plus simple. Par exemple, les verbes sont retournés à l'infinitif, les mots pluriels à leur forme simple. Il élimine aussi les « stop-words » tels que « the », « so », « what », qui n'ont aucune valeur et nuisent au calcul de similarité. Les mots qui ne figurent pas dans le dictionnaire Wordnet sont retournés sans changement (par exemple, les noms propres ne figurent pas dans le dictionnaire et devraient être traités). Wordnet ne peut traiter que les mots en anglais.

4. 5 Base de données (6)

Entrées : Des requêtes SQL.

Sorties : Des résultats de requêtes SQL.

La base de données doit fonctionner au moyen de requêtes SQL. Elle reçoit des requêtes du module de création du profil de l'usager, du module de calcul des similarités et du module de Méta recherche. Ses fonctions se résument à conserver le profil de l'usager et à garder la collection des marchands qui auront été entrés. Elle sera composée de deux tables : la table du profil et la table des marchands.

4.5.1 La table du profil (6.1)

La table du profil (6.1) contient les goûts de l'usager. C'est l'élément de comparaison pour trouver les produits qui peuvent intéresser l'usager. Elle est bâtie par l'usager au moyen de l'interface de communication et par l'agent, qui après l'analyse de documents Web, peut trouver des goûts à ajouter à la table du profil. Cette table consiste en une liste de produits ou de sujets avec certains éléments de fréquence et d'identification pour usage dans le module de calcul des similarités. La table du profil sert à conserver le profil de l'usager sous forme de vecteurs qui peuvent être directement utilisés par le module de calcul des similarités ainsi que par le module de création du profil de l'usager.

4.5.2 La table des marchands (6.2)

La table des marchands (6.2) contient la liste des sites de marchands qui peuvent être utilisés par le module de métarecherche de l'agent. Cette table contient tous les détails nécessaires pour effectuer une recherche dans le but de trouver des produits pouvant satisfaire l'usager. Nous y trouvons des informations telles que l'adresse Web et la requête pour effectuer une recherche.

4.6 Fenêtre principale

Entrées : les commandes de l'usager (communication avec l'agent, requête de recherche et ajout de marchand.).

Sorties : affichage des réponses et des messages pour que l'usager ait une interaction.

La fenêtre principale sert de point central dans la configuration de l'agent, dans la communication avec l'usager et dans les commandes de l'usager. C'est l'interface vers toutes les fonctions et les configurations de l'agent. La fenêtre principale se divise en quatre sections : le panneau d'édition du profil, le panneau d'édition des marchands, le panneau des configurations et le panneau de communication. La fenêtre présente ses différentes options avec un formulaire à onglets. L'usager sélectionne l'onglet qui représente le panneau qu'il désire utiliser.

4.6.1 Sous Module : panneau d'édition du profil

Le panneau d'édition du profil sert à visualiser le contenu de la table du profil (6.1). Ce contenu peut être modifié au loisir de l'usager. Si l'usager veut ajouter des valeurs à son profil, il n'a qu'à entrer les valeurs à ajouter à un vecteur du profil, ou simplement entrer un nouveau vecteur. Si l'usager veut retirer des valeurs de son profil, il n'aura qu'à sélectionner les éléments et les retirer. Les modifications faites par les différents modules au profil y sont affichées au même instant.

4.6.2 Sous Module : panneau d'édition des marchands

Le panneau d'édition des marchands sert à visualiser le contenu de la table des marchands. La liste des marchands y est présentée et l'usager peut les sélectionner pour les effacer ou en ajouter en entrant l'adresse Web.

4.6.3 Sous Module : panneau de configuration

Le panneau de configuration est utile pour régler les différentes options de l'agent comme les configurations de la métarecherche, de la grandeur des vecteurs du profil et de la grandeur du profil. Tous les détails nécessaires, surtout pour les tests et pour régler la vitesse de calcul de l'agent et la vitesse de la métarecherche, figurent sur le panneau de configuration. Les différentes variables sont accessibles par ce panneau.

4.6.4 Sous Module : panneau de communication

Le panneau de communication sert d'interface entre l'agent et l'usager. Il est composé de deux sections : la communication de l'agent et la section de l'usager. Dans la section de l'usager, différentes commandes peuvent être sélectionnées du menu déroulant par l'usager comme la commande pour effectuer une métarecherche, pour ajouter un marchand et pour indiquer à l'agent que l'usager veut ajouter une page Web à son profil. L'agent écrit dans la section de l'agent les messages qui indiquent à l'usager quoi faire et où en est l'exécution des commandes envoyées.

4.7 Interface de communication : Microsoft Agent (10)

Cette interface est une façon utile et intéressante d'interagir avec l'usager. Elle utilise la communication vocale et un interpréteur de langage vocal pour interagir. L'usager peut donc dire ses commandes verbalement et l'agent peut parler et faire des signaux visuels à l'usager. Cette interface, contrairement au panneau de communication de la fenêtre principale, peut toujours recevoir les commandes de l'usager, même lorsque celui-ci utilise son fureteur. En effet, lorsque l'usager sera en pleine séance de furetage, s'il désire ajouter une page Web à son profil, il ne lui suffira que de dire la commande pour que l'agent l'ajoute à son profil. L'agent peut aussi communiquer ses informations à l'usager de manière vocale et par écrit en même temps, car les paroles apparaissent dans des bulles (voir la figure 4.2).

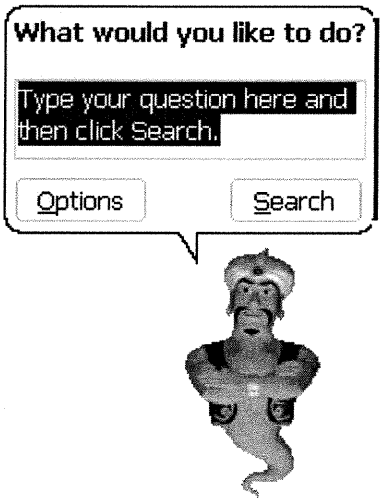

Figure 4.2 - Microsoft Agent en action.

4.8 Module de gestion des actions (11)

Entrées : commande de l'usager venant des interfaces, les différents messages à communiquer à l'usager venant de différents modules.

Sorties : les appels de fonctions du module de création du profil de l'usager et du module de métarecherche.

Le module de gestion des actions sert de contrôle entre les différents modules de l'agent. Il reçoit les instructions envoyées par l'usager aux interfaces, les traite et décide s'il doit soit les réacheminer aux modules respectifs, ne pas les réacheminer ou attendre avant de les faire exécuter. Il reçoit également les informations à faire parvenir à l'usager. Ces informations sont-elles aussi traitées de manière à décider si elles sont nécessaires à envoyer vers les interfaces de communications ou si elles

sont redondantes et risqueraient d'ennuyer l'usager. Ce module agit en tant qu'arbitre pour les interactions entre les différents modules de l'agent et de l'usager.

4.9 Module de Métarecherche (8)

Entrées : les requêtes de recherche, les nouveaux marchands à ajouter, les résultats de similitudes du module de calcul des similarités (7), les résultats des requêtes Web envoyées au client http (9) et la liste des marchands de la table des marchands (6.2).

Sorties : les modifications SQL à la table des marchands, les requêtes Web au client http (9), des vecteurs qui représentent des documents à analyser au module de calcul des similarités, les résultats de ses métarecherches à l'usager sous forme d'une page Html et les informations sur ses exécutions au module de gestion des actions (11).

Le module de métarecherche possède une des fonctions principales, soit celle d'effectuer les recherches dans les sites Web des marchands afin de trouver les produits qui pourraient plaire à l'usager. Ce module traite aussi l'ajout des nouveaux marchands, car ceci demande un procédé pour recueillir les informations qui pourront être réutilisées sur plusieurs recherches. (voir 4.9.1 pour les fonctions du sous-module du traitement des nouveaux marchands). Ainsi, le module reçoit la demande d'effectuer une recherche du module de gestion des actions. Il recueille alors la liste des marchands de la table des marchands, parcourt celle-ci et effectue une recherche Web sur chaque marchand en envoyant la requête de recherche au client http (9). Pour chaque résultat retourné par le client http, un explorateur de liens Web est généré (voir 4.9.2). Les résultats des explorateurs sont regroupés et

présentés à l'usager au moyen du sous-module de présentation des résultats (voir 4.9.3).

4.9.1 Sous-module : traitement des nouveaux marchands

Ce sous-module prend en entrée une adresse Web et produit un objet « marchand » qui contient toutes les informations pour pouvoir effectuer rapidement des métarecherches. Cet objet est ensuite envoyé pour être stocké dans la table des marchands. Nous nous basons sur l'espoir de pouvoir trouver et interagir avec un engin de recherche local sur le site du marchand. Nous essayons donc d'y trouver la commande http à envoyer au site pour qu'il nous retourne le résultat d'une recherche.

L'adresse Web envoyée en entrée est envoyée au Client http (9) pour en recueillir la page Web résultante. Le sous-module cherche dans la page Web un formulaire HTML où certains termes sont présents (par exemple le terme « Search »), pouvant indiquer que la commande de recherche y est présente. Le module extrait la commande de recherche et la mémorise avec l'adresse du site. Le sous-module effectue ensuite une métarecherche pour un produit impossible à trouver avec la nouvelle ligne de commande. La page Web recueillie est ensuite utilisée comme référence pour comparer avec les recherches futures et ainsi déterminer sans interpréter la page Web si la recherche a été fructueuse (les deux pages sont très différentes) ou infructueuse (les deux pages sont presque identiques). Nous tiendrons compte des diverses publicités et nous fixerons un pourcentage de ressemblance à respecter afin de déterminer si la recherche est réussie ou non.

4.9.2 Sous-module : explorateur de liens WEB

L'explorateur de liens Web sert à explorer les multiples liens des résultats d'une requête de recherche à un site Web de marchand. En effet, puisque le résultat des requêtes de recherche n'est pas personnalisé à l'usager, c'est donc l'explorateur Web qui permettra de discerner les pages Web qui pourront satisfaire et coïncider avec le profil de l'usager. L'explorateur de liens Web analyse le contenu de sa page avec le module de calcul de similarité, ce qui lui donne une valeur qui représente la similarité avec un élément du profil. Il explore de manière récursive tous les liens de chaque page et retourne une liste triée en ordre de valeur de similarité avec le profil de l'usager pour chaque page Web. C'est une recherche de type « profondeur d'abord» qui ne devrait pas inclure les liens vers d'autres serveurs, ni dépasser une certaine profondeur limite. Cette profondeur devrait se situer autour de trois ou quatre liens. Nous tenons compte aussi d'une valeur seuil pour filtrer les résultats. Cette valeur seuil est fournie par l'usager au moment d'envoyer la commande de recherche. Chaque mesure de similarité est comparée à ce seuil pour savoir si elle doit être incluse dans les résultats retournés ou non.

4.9.3 Sous-module : présentation des résultats de recherche

Le sous-module de présentation des résultats de recherche est parti intégrante du module de recherche. Il prend en entrée l'ensemble des pages Web et leurs valeurs de similarités retournées par l'explorateur de liens Web. Il bâtit ensuite une page Web avec ces résultats, en affichant un lien HTML pour chaque page Web de même que la valeur de similarité. Nous ouvrons ensuite une application de fureteur pour présenter la page Web générée. L'usager peut ainsi suivre les différents liens de la page Web pour visionner les produits sélectionnés par l'agent en ordre de similarité.

4.10 Client http (9)

Entrées : les requêtes Web à retransmettre à l'Internet (4), les réponses aux requêtes de l'Internet (4)

Sorties : les requêtes http et les réponses de l'Internet en forme de texte.

Le client http sert d'interface entre l'agent et l'Internet. Il fonctionne comme un fureteur, mais sans fenêtre et sans les autres options telles que l'interprétation des « applets », des scripts...etc. C'est un client pour le protocole http de l'Internet. Il procure à l'agent un moyen d'accéder au Web et une manière de recevoir les informations de celui-ci. Il effectue tous les transformations et traitements du protocole http et des connexions aux serveurs sur l'Internet pour fournir les pages Web en forme de texte traitable par l'agent.

Chapitre 5

Implantation et expérimentation

5.1 Implantation

En général, nous pouvons dire que l'implantation de Magellan a été une opération fastidieuse. L'ensemble a été réalisé en Java 1.3 sous une plate-forme Windows et ne peut être utilisé ailleurs. En effet, la plus grande partie du code a été réalisé en Java et pour certaines parties nous avons dû recourir à d'autres langages et souvent à des programmes précompilés pour le système d'exploitation Windows.

Nous allons décrire l'ensemble des méthodes et des techniques utilisées pour implanter les différentes composantes mentionnées au chapitre précédent.

Dans l'ensemble, nous avons utilisé le logiciel JBuilder 5 pour écrire tout le code Java. Ceci inclut la fenêtre principale (12), le module de gestion des actions, le module de création du profil de l'usager, le module de calcul des similarités (sauf le sous-module Wordnet), le module de métarecherche et le serveur-proxy.

Les composantes n'ayant pas été développées en Java sont l'interface de communication Microsoft Agent (développé en Visual Basic 6 avec l'utilisation de l'ActiveX Microsoft Agent) et le programme de dictionnaire Wordnet. Ce dernier a

été utilisé au moyen d'une interface Java qui accède directement aux bases de données de l'application.

5.2 Implantation du serveur-proxy

Le serveur-proxy utilise le code source de ProProxy. Nous avons utilisé le code source Java que nous avons modifié en certains endroits pour nos besoins et nous l'avons intégré au code source de l'agent. Le serveur ouvre un port sur l'ordinateur de l'usager pour que le fureteur y soit connecté. Nous avons choisi le port 80, mais ceci devrait être une des configurations modifiables par l'usager. Les modifications du ProProxy originale étaient pour les causes du module de création du profil de l'usager. En effet, certains problèmes ont été éprouvés :

1) comment déterminer la fin d'une page Web alors que nous savons qu'une page Web peut être composée de plusieurs autres éléments (ex. les «frames» de html, les images) ? Nous savons aussi que chacun de ces éléments génère une nouvelle requête au serveur pour obtenir les données.

2) les entêtes du protocole http sont inutiles à notre traitement. Serait-il possible de les extraire directement dans le serveur-proxy avant le traitement des informations ?

3) les informations utiles doivent être acheminées à une unité de décodage pour extraire facilement les éléments du code HTML qui nous seront utiles.

Nous avons donc modifié le code original afin de nous permettre d'apporter des solutions aux problèmes mentionnés ci-dessus.

1) le proxy tient un compte global de toutes les connexions et des requêtes effectuées par le fureteur (si celui-ci est connecté au serveur-proxy). Lorsqu'une page Web est transférée, chaque composante externe au document HTML principal nécessite une nouvelle requête et ceci, d'une manière récursive. Donc, si nous tenons compte du nombre de requêtes et du nombre de connexions, nous pouvons déduire que s'il ne reste plus de requêtes ni de connexions, le document Web a été transféré en son entier. Nous avons donc modifié le code afin d'indiquer au module de création du profil de l'usager (5) le moment où le serveur-proxy reçoit sa première requête pour un document et le moment où celui-ci est finalement reçu dans son entier.

2) l'entête d'un transfert http est reconnaissable, car il est toujours situé au début du transfert et il est séparé du reste du document par deux retours de chariot. Nous avons donc modifié le code de Pro-proxy [69] pour ne pas soumettre les entêtes au module de création du profil.

3) pour que le code HTML soit analysable, nous avons utilisé un «parser» HTML. Celui-ci requiert un flot de données. Ainsi, pour chacune des connexions nous créons un Parser HTML avec une connexion tunnel («Piped Stream») au serveur-proxy. Le serveur-proxy, au moment de recevoir la connexion de l'Internet, crée son «parser» HTML, fait un prétraitement des informations reçues et il les réachemine au «parser» HTML au moyen de la «PipedOutputStream».

5.3 Implantation du module de création des profils

Pour ce qui est du module de création du profil, le tout a été programmé en Java. Il prend les informations des Parser HTML et recueille les TAG HTML pour le titre, les entêtes et le texte des éléments qui forment le document. Le titre et les entêtes sont conservés afin de leur donner un poids supplémentaire dans la création du profil. Chaque mot des éléments conservés est traité pour enlever les caractères qui ne sont pas alphanumériques. Nous générons ainsi un vecteur appelé « vecteur TF », dont chaque élément est le mot et sa fréquence dans le document. Évidemment, chaque mot y est représenté qu'une seule fois, si le même mot est ajouté, la fréquence du mot correspondant dans le vecteur est incrémentée de un.

Ce vecteur est conservé jusqu'à la réception d'un signal du Module de gestion des actions (11) ou du proxy. Si le proxy signale la réception d'une nouvelle requête, le vecteur est détruit et le module se prépare à en bâtir un nouveau. Si le module de gestion des actions reçoit de l'usager l'ordre d'ajouter à son profil la page qu'il visionne en ce moment (celle dont nous avons créé le vecteur TF), le module envoie le vecteur au module de calcul des similarités pour être ajouté au profil.

5.3.1 Le « parser » HTML

Pour l'implantation du « parser » HTML, nous avons utilisé le code source de Arthur Do, ADC Parser [70]. Le code source étant en Java, aucune modification n'a été nécessaire. Le code source a simplement été intégré au code de l'agent. Son utilisation est très simple et nous permet d'extraire chaque élément d'une page HTML selon les « TAG » qui nous intéressent.

5.4 Implantation du module de calcul des similarités

Le module de calcul de similarité utilise l'algorithme TFIDF [71] pour calculer la similarité entre deux vecteurs. Cet algorithme sera décrit en 5.4.1. Le calcul est basé sur l'algorithme développé par [68]. Cet algorithme est basé sur TFIDF normalisé avec un cosinus. L'algorithme est prouvé comme étant très fiable pour la création de profil (voir 5.4.1). Comme le sous-chapitre suivant est réservé aux détails de l'algorithme, nous allons discuter des détails d'implantation du module de calcul des similarités. Nous utilisons deux fonctions principales ; la première dans le but de créer le profil (utilisé par le module de création du profil), l'autre pour évaluer la similarité entre les éléments du profil et un document représenté par un vecteur TF. Le fonctionnement de la première fonction va comme suit :

1) nous récupérons le profil, appelons le V;

2) nous ajoutons le nouveau vecteur TF au profil V;

3) si le nombre de vecteurs dans le profil dépasse une limite préétablie appelée N, nous exécutons les prochaines étapes, sinon nous arrêtons.

4) nous comparons toutes les paires de vecteurs possibles à l'intérieur du profil (à l'exception d'un vecteur avec lui-même) et nous trouvons la paire avec la plus grande similitude.

5) cette paire de vecteurs est jointe en un seul vecteur. Nous tronquons ensuite le nouveau vecteur au nombre maximum d'éléments dans un vecteur. Cette valeur est préétablie et est appelée M.

6) nous ajoutons le nouveau vecteur généré au profil.

Cette procédure peut devenir très longue si M ou N sont établis à des valeurs extrêmes. Comme nous pouvons le constater, cette procédure est d'ordre $N \times M$. Avec un M à 100 unités par vecteur et un N de 500, le test nous a menés à un manque de mémoire. Nous avons fixé la valeur de N à 200 et de M à 100, ce qui donne un temps de traitement raisonnable et des résultats très appréciables. Ceci est une évaluation personnelle, mais les configurations peuvent être changées selon le désir de l'usager au moyen du panneau de configuration de la fenêtre principale (12).

5.4.1 L'algorithme TFIDF

La technique adoptée pour apprendre le modèle de préférence d'un usager, est une norme bien testée de la Recherche d'information (information retrieval) appelée en anglais « term frequency-inverse document frequency weighting » (tfidf) [71]. C'est une technique simple qui est difficile à battre [17].

Cette technique suppose un vecteur dictionnaire $\vec{D}$ où chaque élément d_i est un mot. Chaque document a alors une représentation vectorielle $\vec{V}$, où l'élément v_i est le poids du mot d_i pour ce document. Si le document ne contient pas d_i, alors $v_i = 0$. La caractéristique réfère à un élément du vecteur $\vec{V}$ pour un document et est notée v_i. Plus un mot *t* apparaît souvent dans un document *d*, meilleure est la chance que *t* est relié au sujet du document *d*. Le nombre de fois que *t* apparaît parmi tous les documents est appelé la fréquence des documents de *t* (« document frequency of *t* ») ou df_t. Plus df_t est grand, moins grande est la chance que *t* puisse se distinguer entre les documents. Par conséquent, pour un document donné, la pertinence du document

est basée sur un terme qui est directement proportionnel à $tf_{t,d}$ (le nombre de fois que le mot *t* apparaît dans le document *d*) et inversement proportionnel à df_t. Le poids de chaque mot rencontré dans une collection de document est calculé comme suit :

$$w(t,d) = tf_{t,d} \log\left(\left|N\right| / df_t\right)$$

où *N* représente l'ensemble des documents .

Le pouvoir de classification d'un mot est représenté par son poids. Chaque élément v_i du vecteur $\vec{V}$ qui représente un document contient le poids TFIDF qui est calculé avec la formule indiquée plus haut pour le mot d_i dans le dictionnaire $\vec{D}$. La similarité (ou la non similarité de deux documents) peut être mesurée en utilisant le cosinus de l'angle entre les deux vecteurs représentants les deux documents à comparer. L'apprentissage du profil de l'usager nécessite un ensemble de documents qui seront utilisés pour entraîner l'algorithme. Chaque vecteur sera comparé parmi les autres vecteurs de l'ensemble $\vec{D}$. Les deux vecteurs ayant la plus grande similarité seront unis en un seul. Ce vecteur est réduit à la grandeur limite d'un vecteur établi *M* et il est joint à l'ensemble $\vec{D}$.

La vérification de la pertinence d'un document s'effectue comme suit : le document pour lequel nous voulons obtenir la pertinence du document est transformé en vecteur $\vec{V}$. Nous comparons ensuite le vecteur $\vec{V}$ avec le dictionnaire $\vec{D}$ utilisant

la formule mentionnée plus haut. La valeur de similarité obtenue est ce qui nous sert de base pour évaluer la pertinence du document. Cette valeur est ensuite comparée avec le seuil établi par l'usager pour savoir si nous devons le considérer comme pertinent pour l'usager.

L'algorithme est en continuel entraînement, car l'usager peut ajouter à sa guise les nouvelles pages Web à analyser et ainsi pour modifier son profil. Les résultats devraient en théorie s'approcher de plus en plus des goûts de l'usager.

Il est possible d'améliorer l'algorithme TFIDF. Plusieurs méthodes ont été testées et sembleraient améliorer la performance de TFIDF. La plupart de ces méthodes impliquent une analyse linguistique plus poussée et utilisent les liens linguistiques pour améliorer le simple algorithme statistique qu'est TFIDF [72] [73] [74]. Ces liens et données linguistiques sont ensuite utilisés en complément de TFIDF au moyen de différents algorithmes d'intelligence artificielle comme : les réseaux de neurones, des modèles de Markov cachés [74] ou d'autres modèles statistiques. Nous ne pouvions nous permettre l'exploration de ces voies, car aucun outil d'analyse linguiste n'a pu être utilisé pour des raisons monétaires et de disponibilité. Par contre, il est très probable qu'inclure les analyses linguistes des pages Web pourrait nous permettre d'améliorer grandement la création et la recherche de notre agent.

5.4.2 Wordnet

Wordnet [75] est un logiciel de dictionnaire anglophone disponible sur le Web. Sa plus grande utilité dans notre projet a été de pourvoir réduire à leur forme la

plus simple les mots qui sont recueillis dans les pages Web. Nous nous servons de l'interface JWordNet [76] pour utiliser les bases de données de Wordnet. Ceci nous permet d'effectuer des requêtes sur les différents fichiers pour trouver la forme la plus simple des mots à traiter. Nous utilisons aussi la liste des « Stopword » fournis dans Wordnet, afin d'éliminer ceux-ci des différents traitements du module de calcul des similarités. Par contre, l'interface JWordNet ne nous permet pas d'utiliser la table des « StopWord ». Nous avons donc importé cette table dans notre base de données principale afin de pouvoir l'utiliser et effectuer une requête pour voir si le mot que nous traitons est un « stopword ».

5.5 Implantation de la base de données

Nous avons utilisé la base de données fournie par JBuilder Enterprise, JDataStore version 4.030. N'importe quelle base de données SQL aurait pu et pourrait être utilisée moyennant quelques modifications mineures dans les lignes de code pour modifier le «driver» et l'emplacement utilisé. La simplicité de JDataStore a été un facteur décisif pour l'acceptation de son utilisation. Mais comme tout le code utilise SQL, ceci favorise sa compatibilité avec plusieurs autres bases de données. Nous avons remarqué au cours des tests d'utilisation que la base de données se corrompt facilement. Nous avons du conserver des reproductions de sécurité de notre base une fois le profil établi pour ne pas le perdre. Les outils de réparation ont été infructueux à accomplir leur tâche. Ceci nous pousse donc à choisir un autre type de base de données pour les travaux futurs.

5.6 Implantation de la fenêtre principale

Nous avons créé la fenêtre principale au début, comme moyen premier de communication et de configuration de l'agent. Cette fenêtre contient tout ce que nous jugions essentiel pour interagir et configurer l'agent. La fenêtre principale est composée de différents panneaux (ou onglets) qui mènent aux différentes options. C'est un formulaire Swing qui contient un panneau à onglets. Nous pouvons le voir sur la figure 5.1.

5.6.1 Panneau d'édition du profil

Voici un aperçu du panneau et de la fenêtre principale :

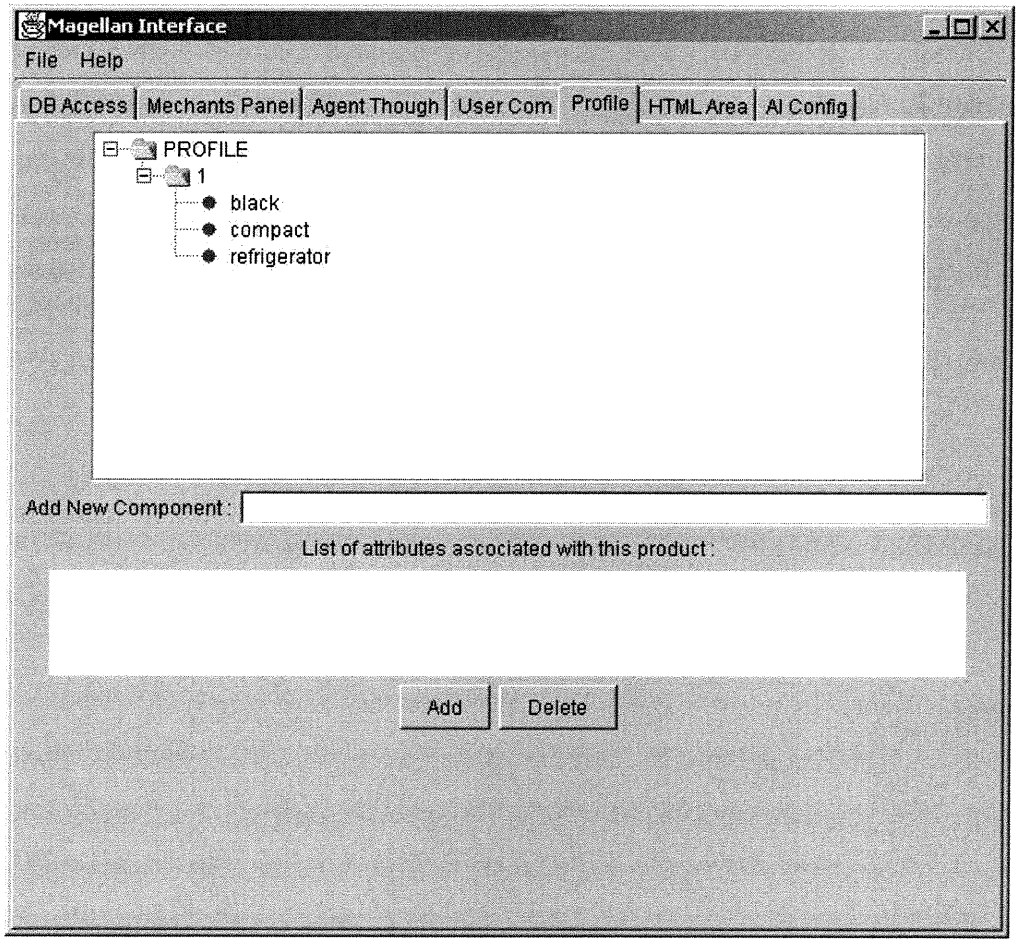

Figure 5.1 - Panneau d'édition du profil.

Comme nous pouvons le voir, le profil y est représenté sous forme d'arbre. Chaque vecteur du profil est identifié par un entier et chaque mot contenu dans le vecteur y est inscrit comme une feuille. En sélectionnant un niveau de l'arbre, nous pouvons y ajouter ou y enlever un vecteur ou des éléments du vecteur. L'espace de texte « Add new Component » sert pour insérer les nouvelles valeurs à un vecteur ou pour insérer un nouveau vecteur, tout en fonction du niveau de l'arbre sélectionné par l'usager. Donc, si l'usager sélectionne la racine « Profile » (niveau 0), l'action du

bouton « Add » vise à ajouter un nouveau vecteur. Si l'usager sélectionne l'identificateur d'un vecteur (niveau 1), l'action du bouton « Add » ajoute les éléments inscrits dans la zone de texte séparé par des espaces. L'action du bouton « Delete » effacera un vecteur en entier du profil. Si les feuilles sont sélectionnées, alors le bouton « Delete » effacera l'élément du vecteur dont l'identification est indiquée par le parent de la feuille.

5.6.2 Panneau d'édition des marchands

Ce panneau est très intuitif. Il affiche les serveurs des différents marchands dans une JList Java. Chaque élément peut être sélectionné pour être effacé avec le bouton « Delete ». Nous ne pouvons ajouter les marchands par ce panneau, car le panneau de communication offre cette fonction. La figure 5.2 illustre la fenêtre principale à l'onglet du panneau de configuration.

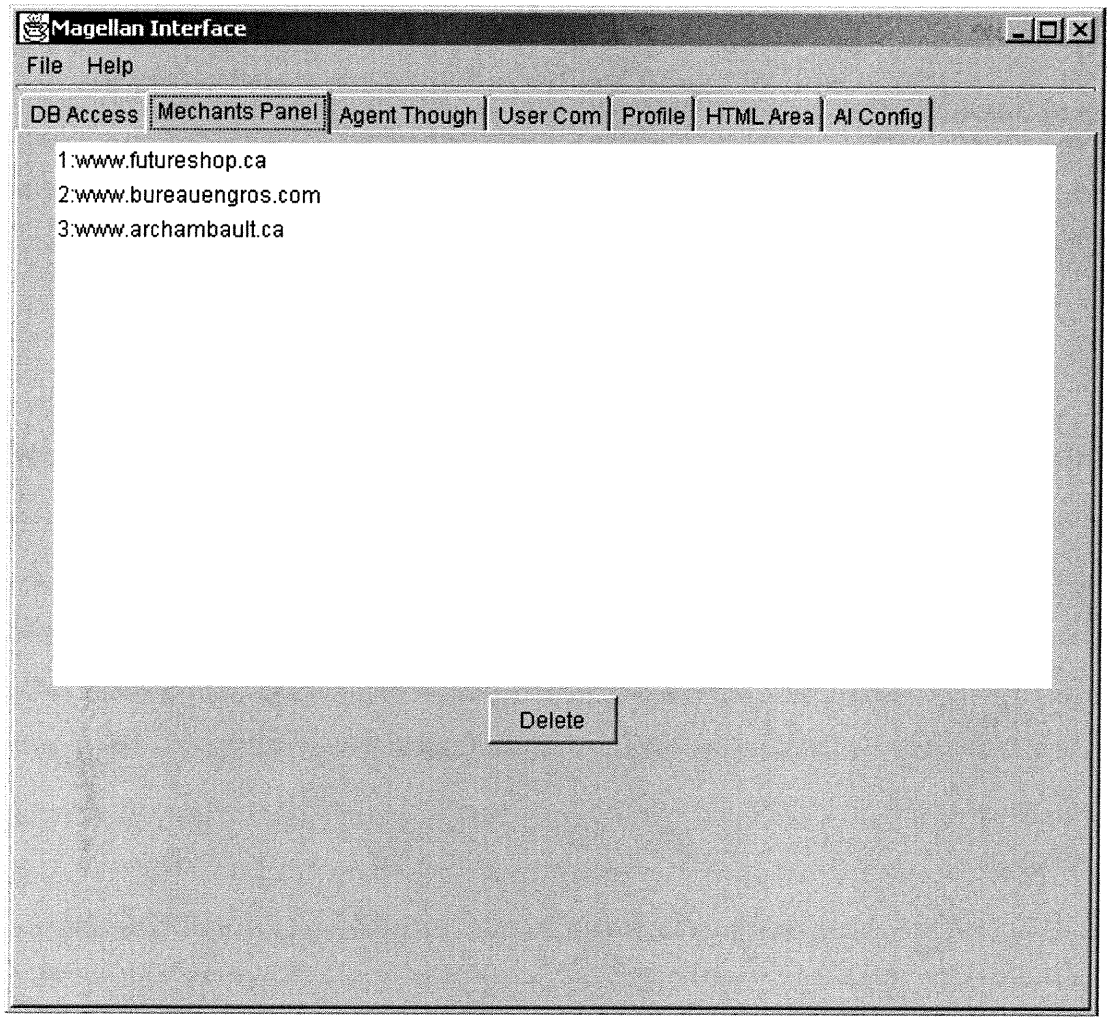

Figure 5.2 - Le panneau d'édition des marchands.

5.6.3 Panneau des configurations

Le panneau de configuration est là où se retrouvent toutes les variables pouvant changer les performances ou le mode de fonctionnement de l'agent. Pour l'instant, seul l'emplacement de la base de données, le port local où le proxy se connecte sur l'ordinateur, la variable N et la variable M de l'algorithme TFIDF y

sont reproduites. Mais si jamais d'autres variables nécessitent d'être ajoutées à l'agent, c'est dans ce panneau que nous les retrouverons. La figure 5.3 illustre la fenêtre principale ouverte sur l'onglet du panneau des configurations.

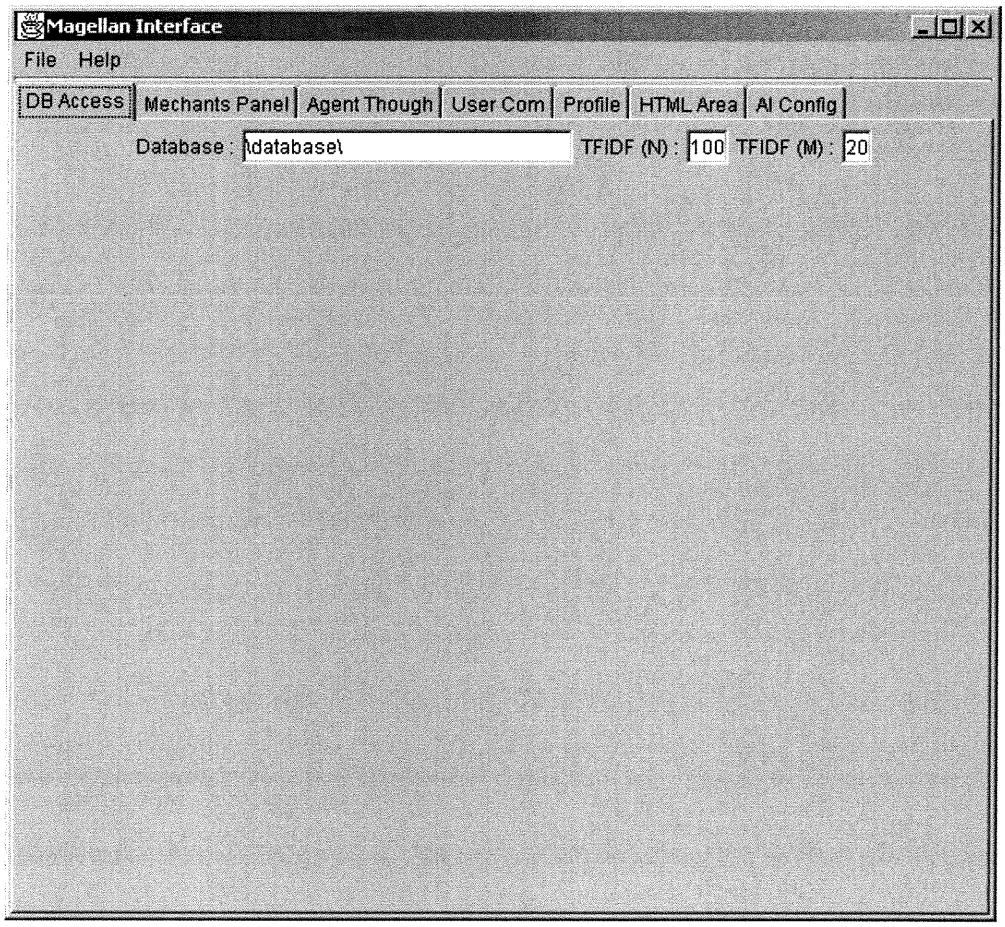

Figure 5.3 - Le panneau des configurations.

5.6.4 Panneau de communication

Le panneau de communication sert d'interface pour entrer les commandes de l'usager. La figure 5.4 illustre les différentes composantes de ce panneau :

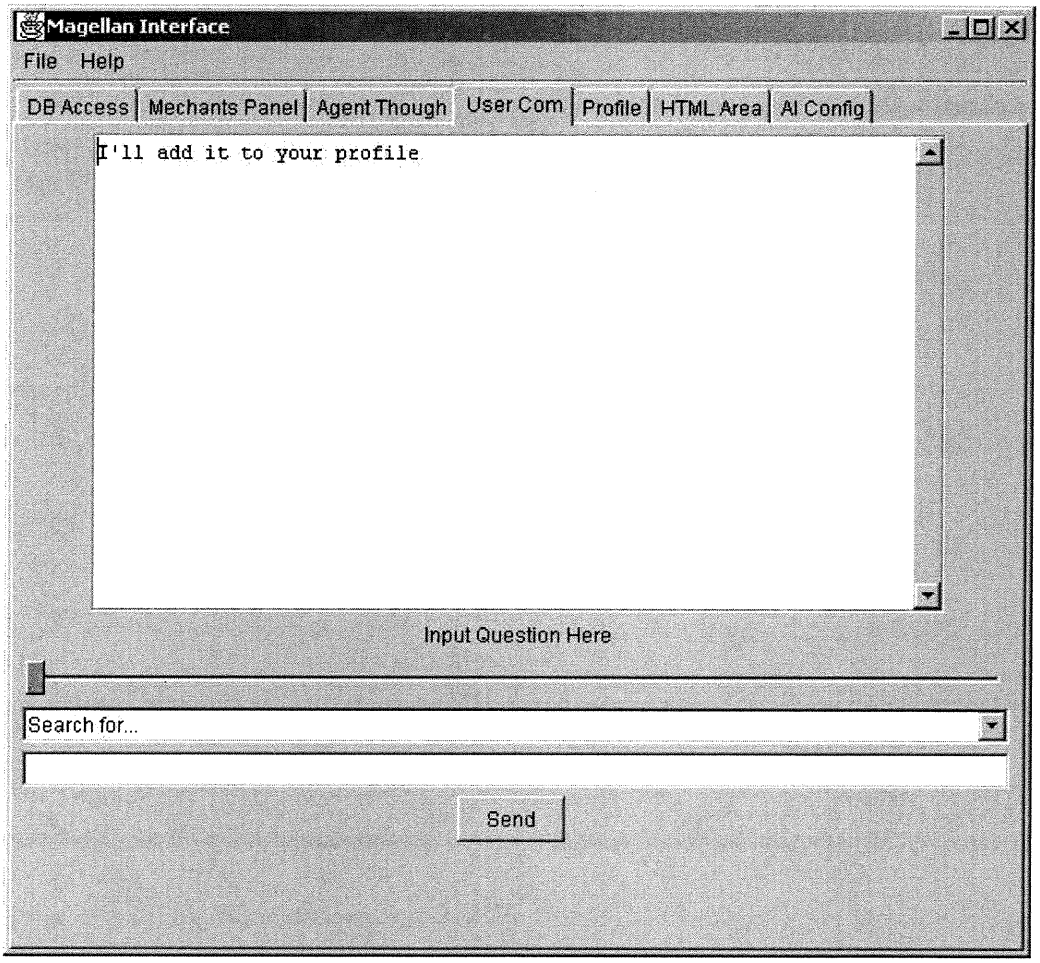

Figure 5.4 - Panneau de communication.

Le grand espace de texte sert pour afficher les commandes de l'usager et les réponses de l'agent. La barre de défilement est utilisée pour choisir le seuil dans les recherches. L'endroit où nous voyons « Search for » est appelé le contrôle des commandes. C'est l'endroit où l'usager peut choisir parmi les différentes commandes qu'il peut passer à l'agent :

« **Search for...** » : exécute une recherche pour les éléments inscrits dans l'espace texte en dessous du contrôle des commandes. L'utilisation de la barre de défilement est utilisée ici pour fixer une valeur de seuil du calcul ;

« **I like this page** » : signale à l'agent que l'usager aime la page qu'il est en train de visualiser ;

« **Add merchant address...** » : ajoute le marchand à la liste selon l'adresse Web indiquée dans l'espace de texte en dessous du contrôle des commandes ;

« **Find me something I would like** » : recherche l'ensemble des marchands pour les produits présents dans le profil. La valeur de seuil qui est représentée par la valeur de la barre de défilement est utilisée pour filtrer les résultats.

5.7 Implantation de l'interface de communication Microsoft Agent

Nous avons utilisé l'ActiveX Microsoft Agent [77] pour cette interface. Cette interface supplémentaire est nécessaire pour pouvoir interagir avec l'usager avec la parole sur la simple pression d'une touche (« ScrollLock » par défaut). L'interface parle grâce au logiciel « Lernout & Hauspie® TruVoice Text-To-Speech Engine ». La reconnaissance de la voix est possible avec « Microsoft Speech Recognition Engine for speech input ». Évidemment, des boîtes de son et un microphone sont nécessaires pour entendre l'agent ou pour lui parler. La figure 5.5 nous présente la forme de l'agent, un génie, que nous avons choisi pour l'interface Microsoft Agent :

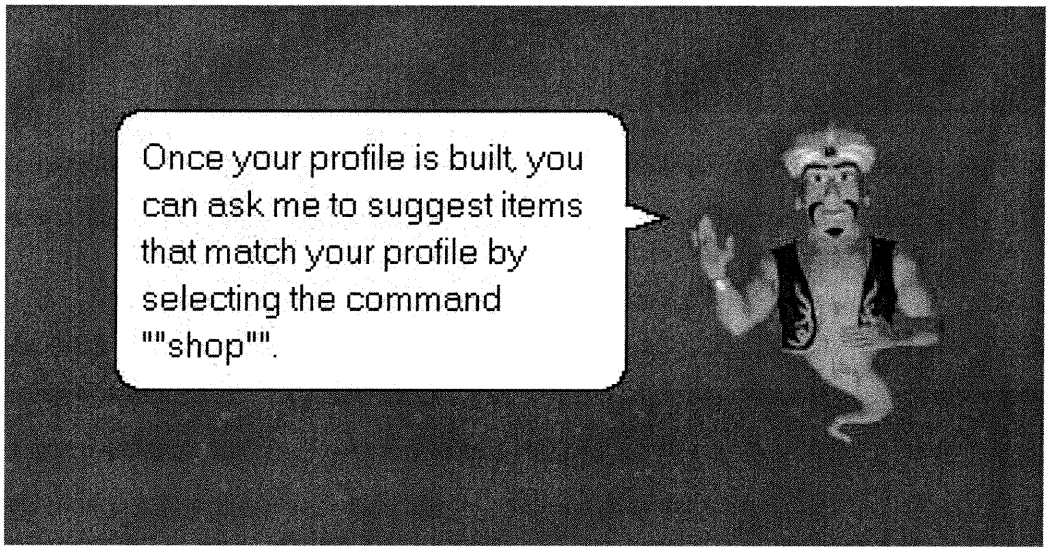

Figure 5.5 - Microsoft agent communique à l'usager.

Cette interface est présente dans l'écran de façon modale et indépendante des autres fenêtres. L'usager peut donc communiquer la commande pour ajouter une page Web à son profil soit :

1) en disant verbalement la commande « I like it » ;

2) en sélectionnant la commande, au moyen de la souris dans le «pop-up» de l'agent qui affiche les commandes offertes ;

3) en choisissant la commande dans la fenêtre de Commandes Vocales de l'agent tel qu'il est montré dans la figure 5.6.

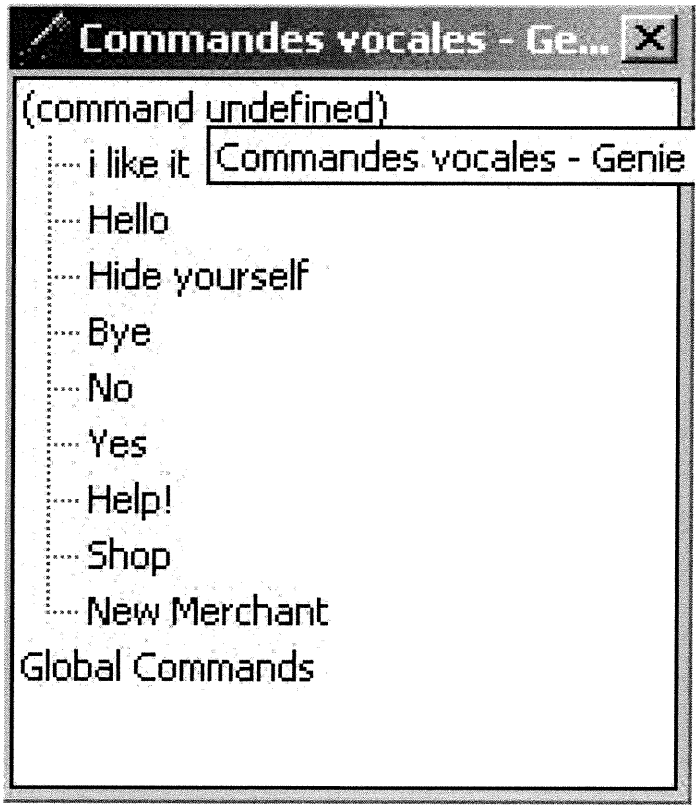

Figure 5.6 - Fenêtre de commandes vocales de l'agent.

L'implantation de l'ActiveX s'est avérée une tâche difficile. Comme n'importe quel ActiveX utilisé avec Java, un pont est nécessaire, car aucune classe de

base n'offre le support des ActiveX. Microsoft Agent ne répondait pas bien aux différents ponts utilisés pour joindre les ActiveX et Java. Nous avons donc conçu un programme en Visual Basic pour utiliser Microsoft Agent. Les commandes pour Microsoft Agent sont transmises entre le programme principal en Java et le programme Visual Basic par connexion «socket» sur l'ordinateur. Toutes les fonctions de Microsoft agent ont été implantées au moyen d'un simple protocole textuel transmis entre les deux programmes par une connexion «socket» sur le même ordinateur. De la même façon, un ensemble de commandes a été implanté pour les différentes réponses du programme Visual Basic vers le programme Java. Cette implantation fonctionne à merveille. Le programme Java ouvre le programme Visual Basic et le programme Visual Basic se termine lorsque ses «sockets» se ferment.

5.8 Implantation du module de gestion des actions

Le module de gestion des actions est le centre de gestion des interactions avec l'usager. Nous n'avons pas développé à fond ce module, car le temps était un élément qui nous restreignait. Il y a place à de nombreuses améliorations, car nous avons concentré notre développement aux fonctions principales et non à l'amélioration des interactions de l'agent. Ce module, qui agit comme un répartiteur de messages, reçoit les messages des modules de communications et les transmet aux différents modules en dessous. Les messages reçus du programme externe en Visual Basic pour la communication, sont interprétés avant d'être traités. Voici donc les messages et comment ils sont traités :

« Search for...» : appelle l'exécution de la fonction de recherche avec paramètre à rechercher du module de métarecherche.

« **I like this page**» : envoie la notification au module de création du profil de traiter le vecteur TF qui a été créé à partir de la page Web récemment visionnée par l'usager.

« **Add merchant address...** » : Appelle la création d'un nouvel objet marchand avec comme paramètre l'adresse Web du marchand.

« **Find me something I would like**» : Appelle l'exécution de la fonction de recherche sans paramètres.

Lors de la réception d'une page Web, l'agent envoie une notification à l'usager pour lui signaler comment il peut ajouter la page Web à son profil. Cet avis est envoyé pour la première page Web reçue. Ce ne sera qu'après la réception de 10 autres pages Web que l'agent renverra cet avis, s'il n'a pas reçu de message « I like it » de la part de l'usager.

L'addition de logique floue en Prolog pour la gestion des actions et des interactions est un projet qui est envisageable avec l'utilisation d'interpréteur Prolog en Java.

5.9 Implantation du module de métarecherche

Ce module utilise l'adresse Web fournie pour aller chercher les informations utilisées pour les recherches sur le site Web du marchand et crée un objet Java appelé marchand. Ce marchand fait partie de la collection de marchands stockée dans la table des marchands. Une fois qu'un ou plusieurs marchands sont présents dans la collection, l'usager peut demander à l'agent d'effectuer une recherche pour certains produits compatibles avec le profil. L'usager peut aussi demander à l'agent de rechercher la collection pour n'importe quel produit qui pourrait l'intéresser, sans avoir à spécifier des produits à rechercher. Le module utilise beaucoup le client http pour effectuer son furetage et ses analyses à l'insu de l'usager. Je vais utiliser l'adresse du site de « Future Shop» (www.futureshop.ca) comme exemple tout au long de ce chapitre et dans l'expérimentation. Nous nous sommes inspirés de la méthode utilisée par Shopbot [1]. Notre agent crée ses marchands d'une manière similaire à Shopbot. L'idée étant très générale est très intuitive, il est difficile de faire autrement. La différence est dans la recherche. En effet, Shopbot utilise une méthode de comparaison avec des patrons préétablis pour recueillir les prix et autres détails sur les produits pour permettre à l'usager de comparer. Magellan parcoure la liste résultante de la recherche et la filtre (d'où le terme de métarecherche) selon le profil de l'usager pour ensuite lui présenter les résultats par ordre de similarité avec son profil. Plusieurs améliorations ont été envisagées pour les différentes tâches accomplies par le module de recherche et celles-ci seront discutées dans les points suivants.

Lorsqu'une commande de recherche est reçue par le module de métarecherche, celui-ci parcourt chaque objet marchand dans sa collection et leur envoie à chacun une commande de recherche.

Lorsqu'un objet marchand reçoit une commande de recherche, il envoie la requête de recherche (qui a été recueillie lors de la création de l'objet marchand) au client http. Avec les résultats retournés par le client HTTP, l'objet marchand crée un explorateur de liens pour parcourir les différents liens des pages reçus sur une certaine profondeur préétablie. Chaque page est analysée et un vecteur TF est créé. Ce vecteur est envoyé au module de calcul des similarités et le résultat de similitude est conservé. Une fois tous les liens explorés, le marchand envoie la liste des pages explorées et leur résultat de similitude respectif.

Le module de métarecherche recueille les listes des pages explorées retournées par chacun des marchands et trie les résultats en ordre décroissant. Une page Web locale est ensuite créée et Internet Explorer (Microsoft) est ouvert avec la page Web en paramètre d'ouverture pour qu'une nouvelle fenêtre d'Internet Explorer permette à l'usager de visualiser les résultats de la métarecherche.

Nous pourrions améliorer la technique proposée de la manière suivante : au lieu d'explorer l'ensemble de la page retournée après avoir envoyé la requête de recherche par le marchand, il serait beaucoup plus économique et plus précis d'explorer la liste des produits retournés par la recherche sur le site. Nous éviterions ainsi d'explorer les liens de publicité multiples ou tous ceux n'ayant aucun rapport direct avec le résultat de la recherche. La méthode utilisée par Shopbot [1] est trop contraignante. Shopbot reconnaît ses informations à traiter avec des patrons préétablis pour lesquels il essaye d'extraire les données de la page Web. Si le marchand modifie moindrement sa présentation ou le design de son site et qu'aucun patron ne permet l'extraction des données, alors le marchand ne pourra pas être utilisé.

Nous désirons en effet ne pas limiter l'ajout des marchands à seulement ceux dont le site Web satisfait aux exigences de nos patrons. Nous voulons permettre l'interprétation de la manière la plus générale possible.

Une technique proposée par [78] est celle de reconnaître les structures dans les pages Web. La liste de résultats de produits est généralement structurée et répétitive. Il serait envisageable de permettre une reconnaissance de la structure de la liste par notre agent afin de permettre une amélioration sans trop restreindre les types de marchands qui pourront être traités par l'agent. Malheureusement, le programme WHIRL qui est associé au document mentionné [78] n'est plus disponible et les informations sur les algorithmes utilisés et leur implantation nous étaient inconnues au moment d'écrire ce mémoire. L'implantation de la reconnaissance des structures dans une page Web par l'agent Magellan est dans nos projets.

Nous avons aussi testé une nouvelle façon d'organiser le profil de l'usager. Comme les produits recherchés peuvent être organisés en catégories par l'usager (exemple : la catégorie voitures pour les BMW, la catégorie musique pour les disques compacts de Céline Dion...etc.), il y a moyen d'optimiser la performance de l'algorithme TFIDF. Avec ces catégories, l'algorithme procède tel qu'il a été décrit plus haut, mais pour un ensemble de différents profils qui auront été séparés en catégories. Donc, au lieu de procéder pour un seul profil, il parcourt les différents profils. S'il s'agit d'ajouter un nouveau vecteur TF à l'ensemble des profils, c'est la catégorie définie par l'usager qui fixe dans quel profil l'agent ajoute le vecteur. Pour trouver la similarité d'un document, l'agent parcourt l'ensemble des profils, il effectue la comparaison décrite plus haut pour trouver la valeur maximale de

similarité dans ce profil et c'est la valeur maximale pour l'ensemble des profils qui est retournée.

Cette méthode offre une plus grande précision pour déterminer la valeur de similarité et de pertinence aux documents.

5.9.1 Implantation de la création des marchands

L'usager nous fournit une adresse Web. Nous allons ensuite effectuer une série de tâches afin de recueillir les informations nécessaires pour effectuer une métarecherche sur le site du marchand. Si une de ces opérations échoue, le marchand n'est pas ajouté à la collection.

1) l'agent va chercher la page Web qui correspond à l'adresse fournie par l'agent (Exemple : www.futureshop.ca)

2) l'agent parcourt la page et les liens de celle-ci récursivement, mais en étant restreint au serveur de départ. Cette récursivité est continue jusqu'à ce que l'agent rencontre un formulaire HTML (« FORM ») où se trouvent certains mots clés tel que : « search », « find », « look », etc.

(Exemple : <TR>

<FORM method="POST" name="sitesearch" onsubmit="return CheckInputValue()"
ACTION=http://www.futureshop.ca/catalog/subclass.asp?logon=&langid= ….)

3) une fois le formulaire trouvé, nous prenons le paramètre ACTION du formulaire en note. Ce paramètre sert d'adresse de référence pour envoyer la commande POST ou GET lorsque l'action SUBMIT est initialisée dans le formulaire.

(Exemple:
ACTION=http://www.futureshop.ca/catalog/subclass.asp?logon=&langid= ….)

4) nous recherchons ensuite dans le formulaire le champ texte visible qui contient un paramètre HTML où nous pourrons retrouver l'un des mots clés suivant : « search », « key », « word », etc. Si aucun champ texte ne comporte un des mots clés recherchés, nous prendrons alors le champ le plus susceptible d'être celui où un usager entrerait ses paramètres de recherche. S'il y a un seul champ texte visible, celui-ci est choisi. Si plusieurs champs de texte sont présents, alors le premier visible est choisi.

(Exemple : <INPUT SIZE="17" VALUE="mot(s)-clé(s) ici" NAME="keyword" maxlength="100" ...)

5) une fois le champ texte choisi, l'agent prend en note le paramètre « NAME » de ce champ. (Exemple : NAME="keyword")

6) si toutes les étapes ont été fructueuses jusqu'ici, nous avons obtenu la requête à envoyer au serveur Web du marchand. Si nous voulons effectuer une recherche, cette requête ressemble à ceci :

S'il y a présence du caractère « ? » dans la ligne ACTION alors :

REQUETE = ACTION + NAME + les produits à rechercher séparés par des '&'

Sinon

REQUETE = ACTION + ? + NAME + les produits à rechercher séparés par des '&'

Donc, nous obtenons la ligne de requête à envoyer au serveur pour les différents produits que l'agent pourrait vouloir rechercher.

7) une fois la requête établie, nous envoyons cette requête avec un produit impossible à trouver (Exemple : « asdsdsdjfsereiias »). Le résultat retourné sera conservé pour des fins de comparaisons. En effet, si une recherche retourne un résultat négatif, le seul moyen de l'agent de le vérifier est de comparer ce résultat avec celui de la recherche impossible. Ceci permet à l'agent de ne pas s'attarder sur le site pour cette recherche.

Une fois les informations obtenues (soit la ligne de commande pour effectuer une requête de recherche et le résultat négatif d'une recherche), l'agent ajoute ce

marchand à la collection de marchands et à la table des marchands. Si une des informations ne peut être obtenue, le marchand ne sera pas accepté.

5.9.2 Implantation de l'explorateur de liens

L'explorateur de liens reçoit une page Web et un entier de profondeur comme paramètre. Il décrémente la profondeur de un. L'explorateur de liens crée un vecteur TF avec la page Web qui lui est donnée en paramètre et l'envoie au module de calcul de similarité. Si la valeur de similarité dépasse un certain seuil (choisi par l'usager), il ajoute à la liste des résultats retournés (une liste qui est partagée par l'ensemble des explorateurs de liens pour un marchand) l'adresse de la page Web et sa valeur de similitude qui a été retournée par le module de calcul des similarités. Il crée ensuite un nouvel explorateur de liens pour chacun des liens sur la page Web (seulement ceux, dont le serveur est identique à celui de la page Web reçue en paramètre). Une fois l'exploration des liens terminée sur la profondeur requise, la liste totale des résultats (les adresses et leur similitude avec le profil) est retournée au module de métarecherche.

5.9.3 Implantation de la création des résultats

Lorsque la liste contenant l'ensemble des adresses de pages Web et leur valeur de similitude a été retournée par tous les marchands, ils sont regroupés et triés de nouveau.

L'agent crée ensuite un fichier texte nommé HtmlResults.htm dans le répertoire HtmlResults de son répertoire principal. Le contenu du fichier top.htm est

ensuite recopié dans le nouveau fichier. Top.htm contient le format et le début de la page Web. Nous bouclons ensuite sur la liste retournée par les marchands et nous inscrivons un lien avec la valeur de similarité. Nous terminons en écrivant une fin de page commune. Une fois le fichier créé et rempli, nous utilisons Jacob pour créer un objet COM. Cet objet est associé à Internet Explorer (Microsoft) et nous lui donnons la commande qui permettra d'ouvrir notre page Web des résultats que nous venons de créer. Voici le résultat de ces étapes :

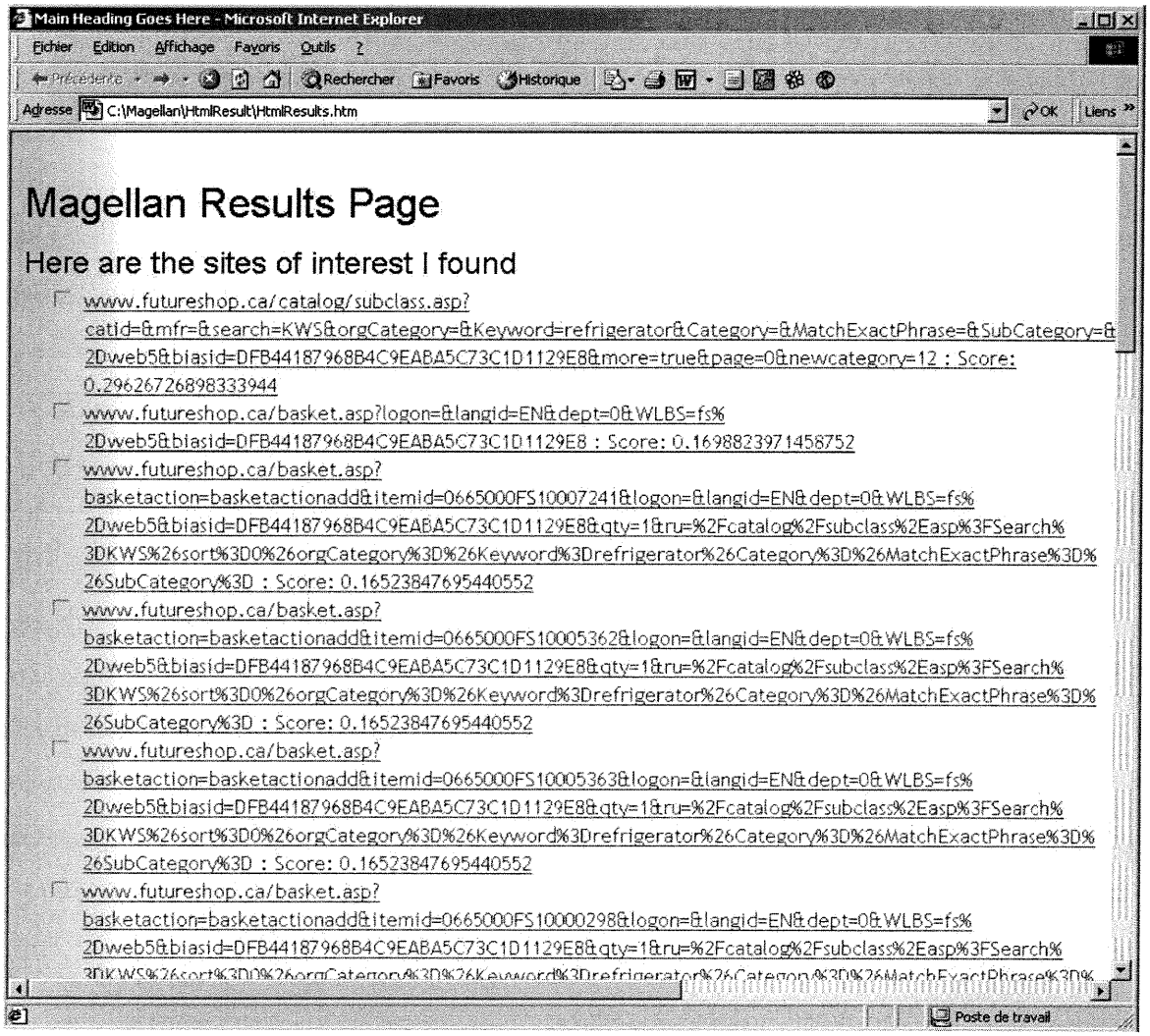

Figure 5.7 - Page des résultats de Magellan.

Chaque lien représenté dans la page des résultats permet à l'usager de cliquer dessus pour voir la page correspondante.

5.10 Implantation du client http

Le client http utilise le programme de Tschalär [79]. Nous n'avons utilisé que la section principale de HTTPClient 0.3-3. Les autres sections (dans les sous-répertoires) ont été effacées du code, car elles nous donnaient des erreurs et elles n'étaient pas utilisées. Nous n'utilisons que les connexions http (HTTPConnection) et la fonction getFile pour nous procurer les fichiers Web.

5.11 Expérimentation

Les expérimentations ont été concluantes. La création du profil a été vérifiée à la main et elle correspond à notre vérification. La recherche nous a donné d'impressionnants résultats. Nous allons vous présenter la création à la main d'un profil et la recherche sur un site de marchands d'électroménagers.

Nous avons tout d'abord créé un profil pour différents produits afin de vérifier la fonction de création du profil de l'usager. Nous avons ensuite ajouté à la main le vecteur : « refrigerator compact black ». Voici les résultats du panneau de profil :

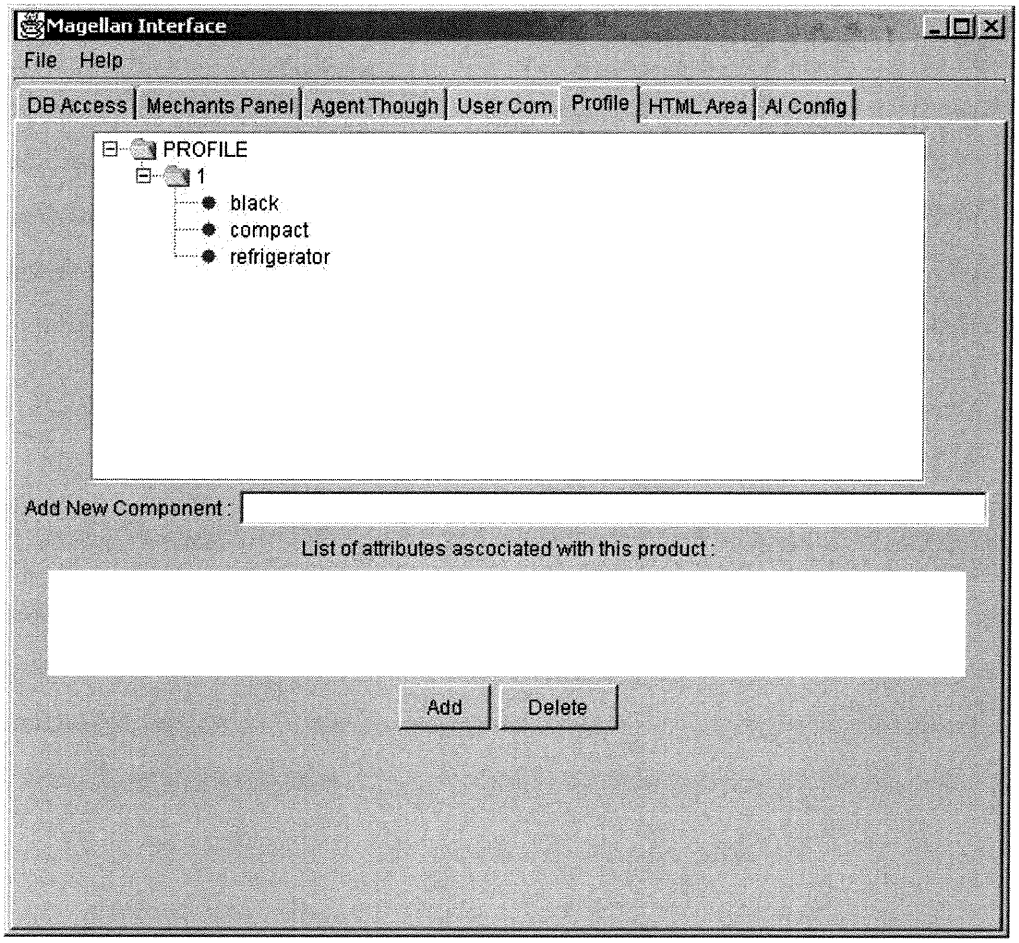

Figure 5.8 - Résultats du panneau de profil.

Une fois le profil créé, nous avons ajouté le site Web du magasin Futureshop à notre liste de marchands en utilisant le panneau de communication et en lui donnant comme paramètre l'adresse : www.futureshop.ca.

Voici le panneau d'édition des marchands après l'ajout :

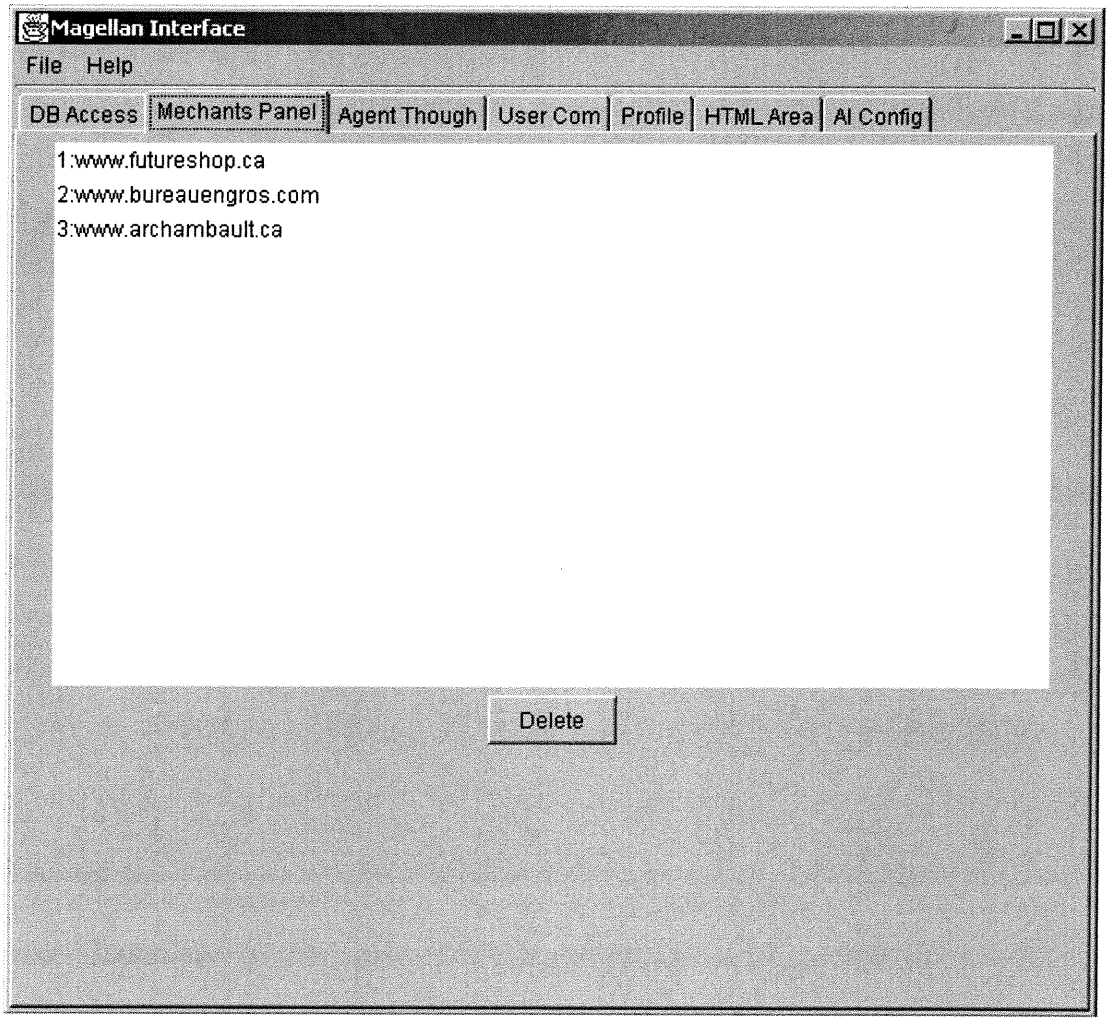

Figure 5.9 - Résultat du panneau d'édition des marchands après ajout de quelques marchands.

Évidemment, les valeurs dans les variables de l'objet marchand ont été vérifiées et correspondent aux bonnes valeurs, soit : la requête à envoyer pour effectuer des recherches et la page pour une recherche invalide.

Malgré le grand nombre de marchands qui a été ajouté avec succès dans notre liste par l'agent Magellan, certains sites de marchands n'ont pas pu être ajoutés à la liste des marchands. Par exemple, pour l'adresse de « www.canadiantire.com », notre agent n'a pas pu trouver la commande pour effectuer les recherches sur le site. Nous devrons améliorer la technique pour trouver la commande de recherche pour que l'agent soit encore plus général qu'en ce moment.

Ensuite, nous avons commandé à notre agent de nous rechercher des réfrigérateurs. Nous avons donc sélectionné la commande « Search for…» et nous avons tapé comme argument « refrigerator » et en fixant le seuil à 0. L'agent a alors exploré sa liste de marchands pour nous présenter les résultats affichés dans la figure qui suit

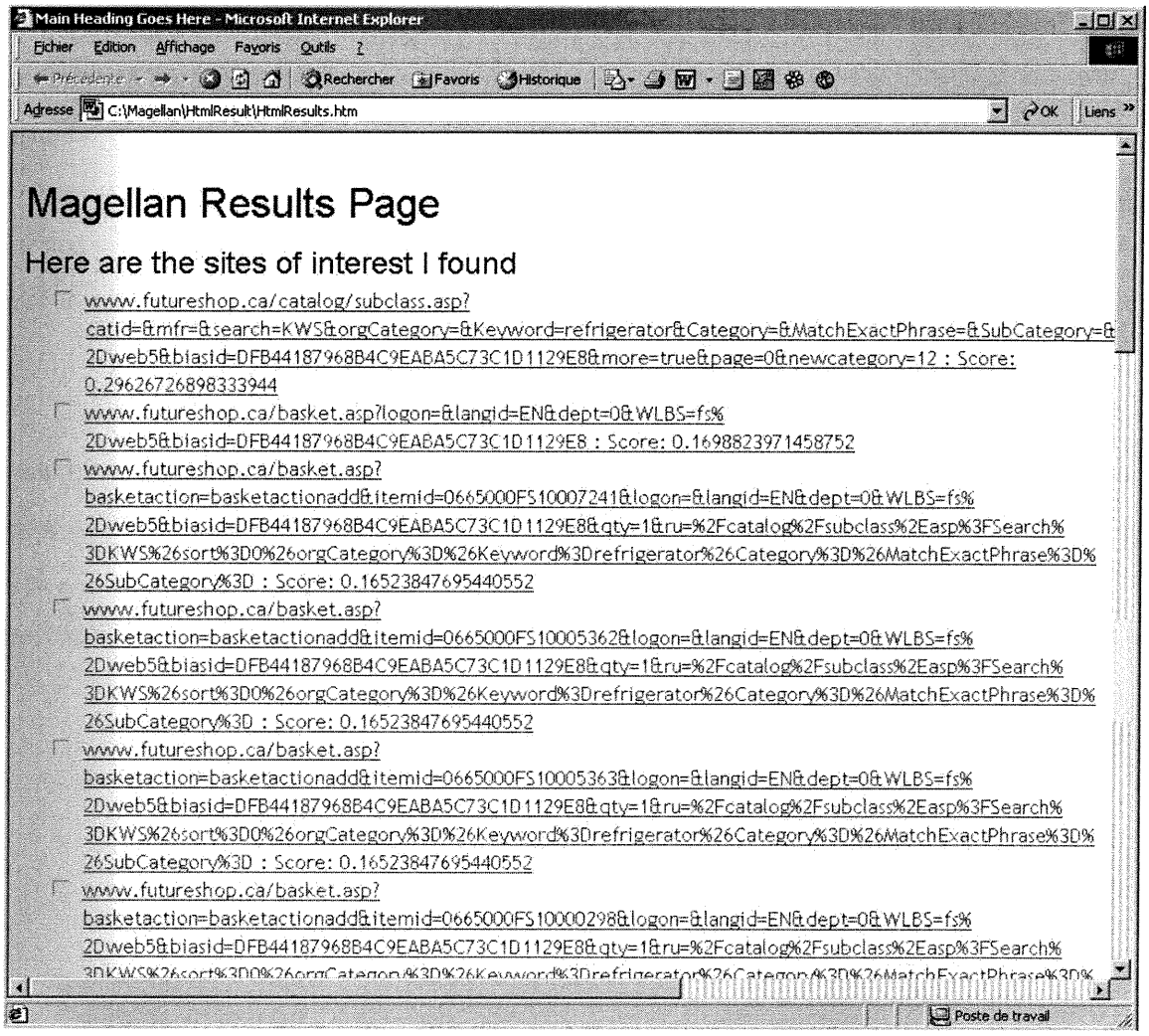

Figure 5.10 - Le résultat de recherche affiché dans Internet Explorer.

Le premier lien nous emmenait directement à une liste de réfrigérateurs compacts où plusieurs étaient de couleur noire.

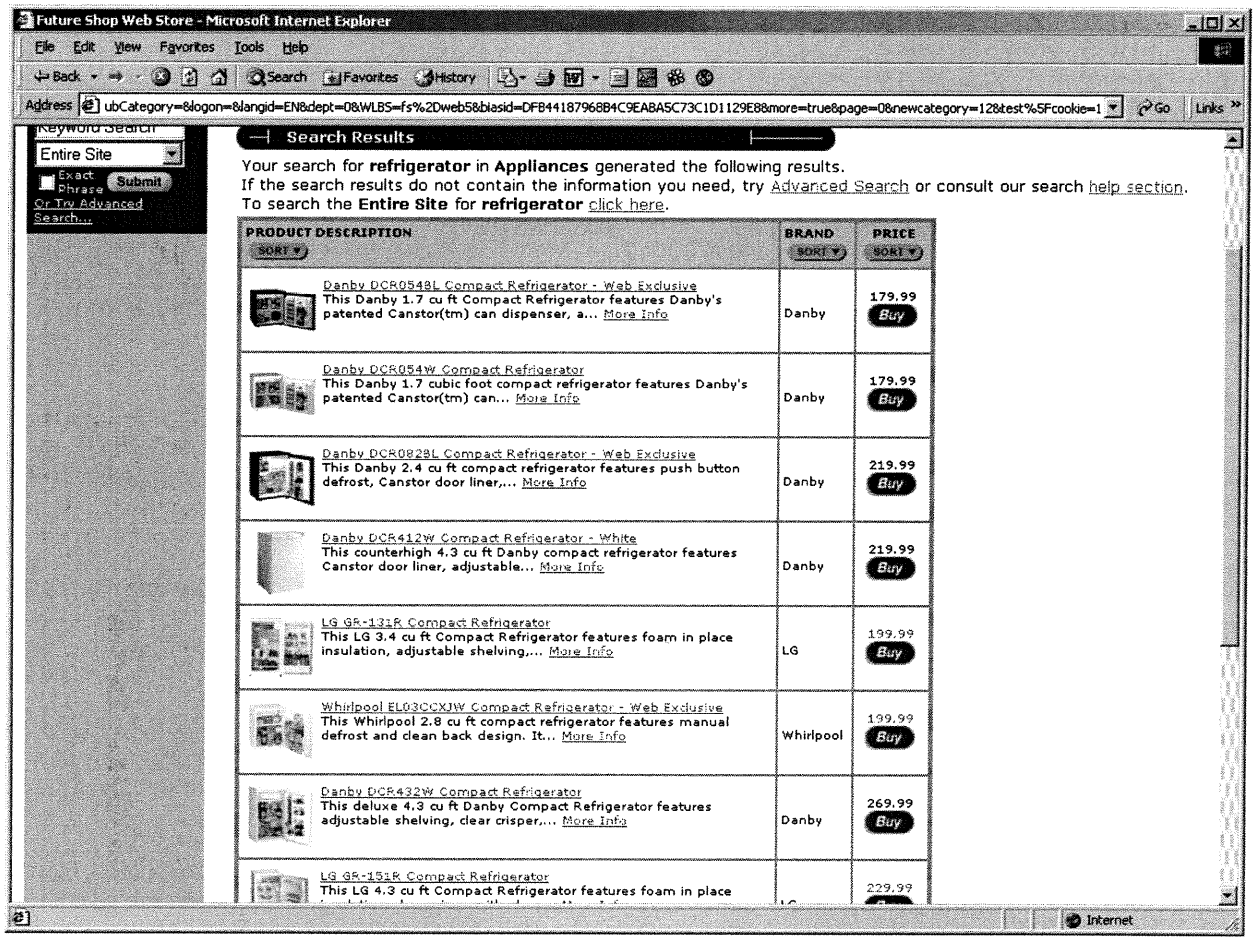

Figure 5.11 - Le premier lien de notre recherche.

Les autres liens avaient des valeurs de similarité (les valeurs de similarité sont entre 0 et 1) moindre (en dessous de 0.3). Après avoir effectué plusieurs tests, nous en avons conclu que la valeur qui nous donnait le plus de résultats, tout en conservant les principaux documents pertinents, devrait se situer au-dessus de 0.3.

Plusieurs liens nous menaient à notre « Panier », comme si nous avions effectué des sélections pour des achats futurs. Cela nous indique que l'exploration a

parcouru certains liens, comme les boutons « Buy » ou « Your Shopping Cart ». Ceci peut être corrigé si l'agent pouvait reconnaître la structure de liste et n'explorer que celle-ci. Pour cela, nous allons utiliser dans de futurs travaux, un moyen de reconnaître les structures pour améliorer les performances de notre agent.

Voici un autre exemple : nous avons utilisé le site de « Hudson Bay Company » http://www.hbc.com (La Baie). Dans notre profil, nous avons inséré un vecteur contenant les mots Parker, Brother, Monopoly, Players. Le seuil a été élevé à 0,5 pour mieux filtrer les liens pertinents. La commande de recherche pour le mot « game » a été donnée à l'agent qui nous a trouvé ceci :

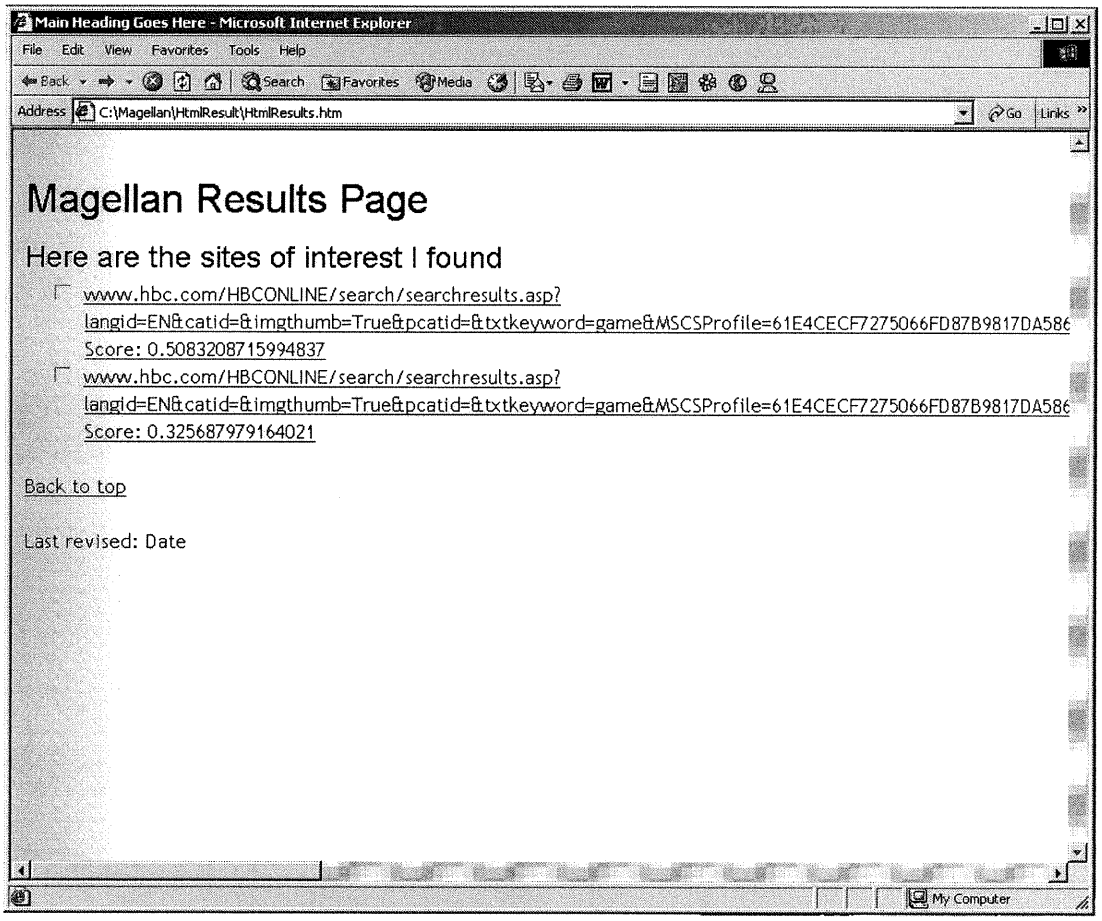

Figure 5.12 – Les résultats pour www.hbc.com.

Le premier lien nous a amené sur la neuvième page de l'engin de recherche, où nous y trouvons les jeux de Monopoly et Monopoly Junior. Le deuxième lien nous a dirigé sur une page où certains jeux de Parker Brother étaient disponibles, mais pas celui de Monopoly.

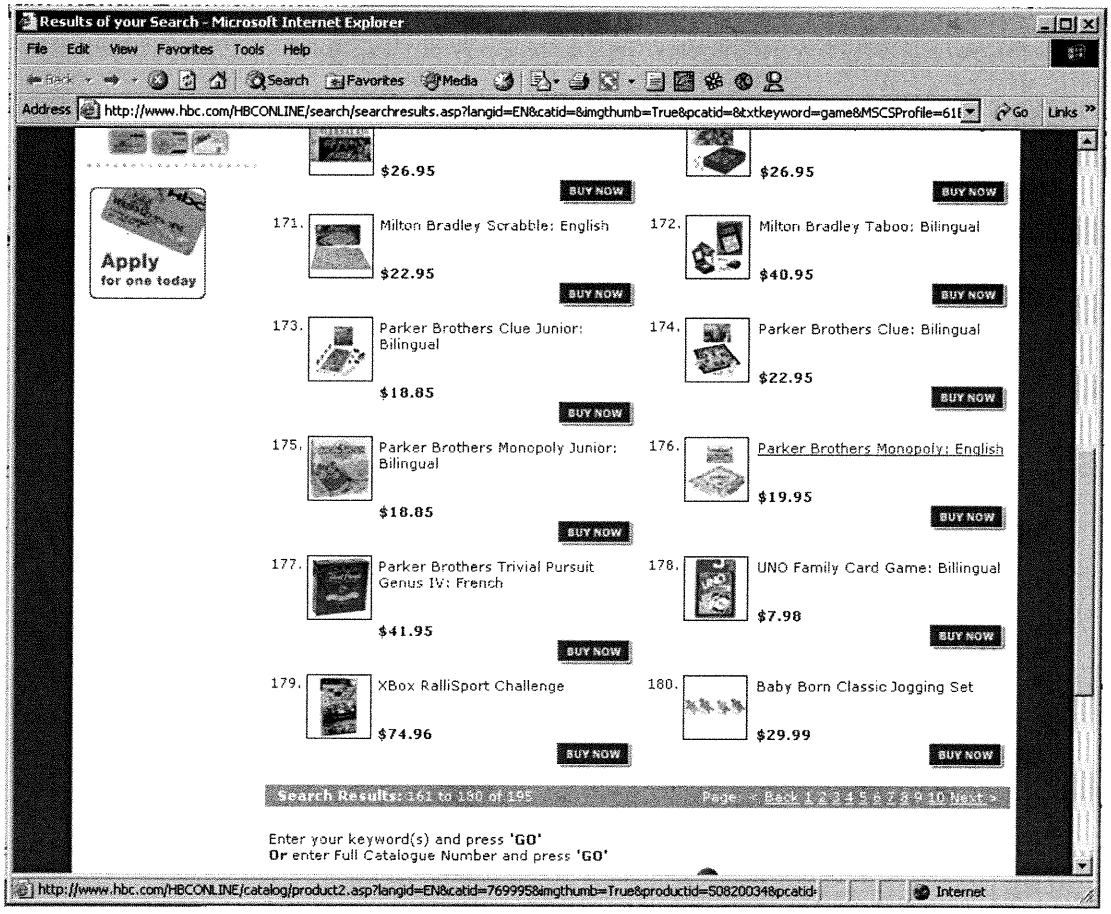

Figure 5.13 – Le premier lien nous amène directement au but.

Discussion et conclusion

Nous considérons avoir atteint notre but, créer un agent de recherche intelligent qui permet à l'usager de rechercher des produits qui pourraient lui plaire sur les différents sites des marchands sur le Web. Magellan répond bien aux attentes, comme nous l'avons vu dans l'expérimentation. Il est efficace et il permet en des temps raisonnables de rechercher une série de sites Web de marchands. Nous avons donc produit un agent qui répond aux exigences que nous avions énoncées dans l'introduction.

- **Compréhension** : notre agent comprend les termes utilisés en anglais et sa compréhension s'étend aux autres langages aussi. Notre algorithme utilise le dictionnaire Wordnet pour trouver la forme simple des mots en anglais et les autres mots sont traités sans modifications. Les mots en anglais sont donc traités avec plus de compréhension de la part de l'agent, mais sans en être restreint.

- **Utilité** : notre solution apporte une nouvelle utilité au domaine de la recherche sur le Web. Aucun des engins de recherche traditionnels n'est capable de rechercher des produits sur une liste de sites de marchands de manière générale et sans limitation selon les goûts de l'usager. Magellan permet d'ajouter des sites de marchands sans tenir compte de structure prédéterminée. Magellan bâtit un profil de l'usager qui lui permet de filtrer les résultats selon les goûts de l'usager.

- **Adaptation** : notre agent à une certaine puissance adaptative, car la méthode pour décortiquer les sites des marchands lui permet d'ajouter de nombreux sites de marchands à sa liste sans que ceux-ci doivent se conformer à une forme ou une structure prédéterminée. Tous les sites sont analysés de la même manière, peu importe la langue utilisée dans le site. Les sites en anglais sont traités un peu plus en

profondeur grâce au dictionnaire Wordnet. L'efficacité de l'agent est donc supérieure pour les sites en anglais.

- **Contraintes environnementales** : seul le dictionnaire Wordnet est utilisé pour ajouter un niveau d'efficacité (pour les sites Web en anglais seulement), mais aucune compréhension du langage naturel n'est nécessaire pour effectuer l'analyse des sites Web. L'algorithme TFIDF nous fournit un outil d'analyse statistique sur l'ensemble des mots peut importe le langage des sites Web analysés.

Améliorations et projets

Évidemment, Magellan n'est pas parfait. Telles qu'il a été énoncé dans les chapitres précédents, la diversité et l'absence de normes sur le Web font en sorte qu'il est pratiquement impossible de créer l'outil de recherche parfait. Cependant, nous avons su explorer une voie que les engins de recherche traditionnels et les agents n'ont pas su explorer avec beaucoup d'efficacité ni de succès. Plusieurs améliorations peuvent être apportées à Magellan, je vais donc vous les expliquer, car celles-ci font partie des projets de Magellan.

Reconnaissance des structures

Il serait possible de reconnaître la structure de liste qui nous est renvoyée par les engins de recherche sur les sites des marchands [78]. Nous pourrions ainsi restreindre notre exploration aux liens qui figurent dans cette structure. Malheureusement, les informations dans le texte de Cohen ne sont pas assez pertinentes pour nous permettre de réaliser cette fonction avec Magellan. Nous

n'avons pas pu en trouver de meilleure et l'auteur a enlevé l'accès public aux informations qui auraient pu nous aider.

Donc, si l'agent peut reconnaître les structures de liste, il est possible de penser qu'il peut reconnaître les structures de la présentation des données pour chaque produit (les détails, le prix, la marque, etc.). Il serait ensuite possible de retirer les informations que l'usager pourrait nécessiter pour lui permettre de comparer les différents produits sans avoir à parcourir les liens des résultats de recherche. Par exemple, l'usager pourrait trouver parmi une liste de réfrigérateurs les moins dispendieux. Les résultats de recherches présenteraient le nom, le prix, la description ou d'autres détails sur chacun des produits.

Une autre base de données

Après nos expérimentations, nous nous sommes rendu compte que la base de données de JDataStore (marque de commerce Borland) n'est pas très efficace et se corrompt très facilement. Nous devrions changer pour Ms SQL Server, mais le manque de budget nous a contraints à utiliser JDataStore. Une autre option a été tentée lors des expérimentations, nous avons utilisé une base de données Ms Access au moyen de JDBC, mais les objets Java ne peuvent être stockés au moyen de Access. Pour faire fonctionner la base Access, une bonne quantité du code aurait dû être changée et cela aurait occasionné une grande perte de temps et d'efficacité.

Reconnaissance des « paniers d'achat»

Une bonne amélioration dans l'efficacité de notre agent pourrait être apporté si celui-ci ne parcourait pas, lors de son exploration, les liens des « paniers d'achat ». Le « panier d'achats » est le lien utilisé pour ajouter des produits à la facture de l'usager. Ces paniers ne contiennent aucune information utile. Ce lien nuit à la performance de l'agent car :

- la première fois que l'agent parcourt ce lien, le site du marchand ajoute un produit à la liste d'achats de l'usager ;
- la seconde fois que l'agent parcourt ce lien, il ouvre la liste des achats où peuvent figuré des produits pouvant intéresser l'usager, donc l'agent ajoute le lien aux résultats qui seront présentés à l'usager.

Donc, le lien vers le panier d'achats est ajouté souvent plus d'une fois aux résultats. Il serait nécessaire de reconnaître ce genre de lien pour ne pas les traités.

Améliorer le temps de recherche

Bien que nous trouvions le temps de recherche très raisonnable, il y a toujours possibilité à amélioration. C'est d'ailleurs un atout que les engins de recherche tentent perpétuellement d'améliorer. Diverses hypothèses nous permettraient d'améliorer le temps de recherche. Certaines des améliorations mentionnées plus haut, telles qu'une reconnaissance de structure et l'amélioration de la performance de la base de données nous permettraient d'améliorer le temps de recherche en général, car ils influencent directement la vitesse de l'algorithme TFIDF.

Traitement du XML

Le « parser » que nous avons utilisé ne permet pas le traitement du code XML. Il suffirait de trouver un nouveau « parser » qui combinerait le traitement de HTML et du XML pour permettre à Magellan d'utiliser les deux langages. Notre programme étant très modulaire, ce changement nécessiterait que très peu de temps.

Comparaison avec Shopbot

Shopbot, un agent d'assistance pour les achats sur le Web est très similaire à Magellan. Ils ont en effet plusieurs points en commun :

- les deux agents traitent surtout les informations codées en HTML, les javascripts, ainsi que les autres supports ne sont pas traités ;
- les deux agents fonctionnent exclusivement sur le Web;
- les marchands doivent posséder un répertoire qui se recherche.

Chaque agent a ses avantages et ses inconvénients :

Shopbot	Magellan
Shopbot distingue les éléments des descriptions de produits fournis par le site du marchand ce qui permet à l'usager de comparer les produits.	La distinction des éléments dans les descriptions des produits fait partie des projets d'amélioration (voir reconnaissance des structures).
Shopbot utilise des patrons pour reconnaître les éléments de la description des produits. Ces patrons sont programmés et donc limitent la recherche de Shopbot à un domaine à la fois.	Aucune restriction de domaine.
N'utilise aucune information sur l'usager (à l'exception de la requête de recherche)	Utilise un profil de l'usager bâti avec les informations fournies par l'usager ainsi que par certaines des pages Web parcourues par l'usager.

Tableau 6.1 – Comparaison de Magellan avec Shopbot.

De plus en plus, le développement des techniques modernes de recherche sur le Web tend à se diriger vers des recherches où l'usager joue une part plus importante que celle de procurer des mots clés. C'est lui qui sait ce qui lui est nécessaire comme information et il connaît le contexte ou ces informations doivent s'appliquer il est donc essentiel d'essayer de trouver le contexte et les informations qu'il ne fournies pas aux systèmes de recherche. Ces informations sont obtenues par différent moyen et le profil de l'usager en est un. Il décrit l'usager par ses fréquentations sur le Web et ses goûts qu'il aura entrés.

Bibliographie

1. Doorenbos, R. B., Etzioni, O., Weld, D.S., *A Scalable Comparison-Shopping Agent for the World-Wide Web*, W. L. J. a. B. Hayes-Roth, Editor. 1997, ACM Press: Marina del Rey, CA, USA: p. 39--48; http://citeseer.nj.nec.com/doorenbos97scalable.html.
2. Hu, W.-C., *An overview of the World Wide Web search technologies*, in *proceedings of 5 th World Multi-conference on System, Cybernetics and Informatics SCI2001*. 2001: Orlando, Florida: p. 22-25; http://citeseer.nj.nec.com/hu01overview.html.
3. Zamir, O., Etzioni, O., *Web document clustering: A feasibility demonstration.*, in *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1998, ACM Press: Melbourne, Australia: p. 46-54; http://citeseer.nj.nec.com/zamir98web.html.
4. Chakrabarti, S., van den Berg, M., Dom, B., *Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery*, in *8th World Wide Web Conference*. 1999: Toronto, Canada; http://citeseer.nj.nec.com/chakrabarti99focused.html.
5. Mladenic, D., *Text-learning and related intelligent agents: a survey*, in *IEEE Intelligent Systems*. 1999. **14**(4): p. 44-54; http://citeseer.nj.nec.com/article/mladenic99textlearning.html.
6. Han, E.-H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J., *WebACE: a Web agent for document categorization and exploration*, in *AGENTS '98, Proceedings of the second international conference on Autonomous agents*, K. P. S. a. M. Wooldridge, Editor. 1998, ACM Press: New York. **9-13**: p. 408 - 415; http://www.acm.org/pubs/articles/proceedings/ai/280765/p408-han/p408-han.pdf.
7. Joachims, T., Freitag, D., Mitchell, T.M., *Web Watcher: A Tour Guide for the World Wide Web*, in *IJCAI (1)*. 1997: p. 770-777; http://citeseer.nj.nec.com/50326.html.
8. Balabanovic, M., Shoham, Y., *Learning Information Retrieval Agents: Experiments with Automated Web Browsing*, in *Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments*. 1995: Stanford, CA: p. 13-18; http://citeseer.nj.nec.com/balabanovic95learning.html.
9. Goldman, C., Langer,A., Rosenschein J., *Musag: an agent that learns what you mean*, in *PAAM 96*. 1996; citeseer.nj.nec.com/goldman96musag.html.

10. Lieberman, H., *Letizia: An Agent That Assists Web Browsing*, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence IJCAI95*, C. S. Mellish, Editor. 1995, Morgan Kaufmann publishers Inc.: Montreal, QC, CAN: p. 924-929; http://citeseer.nj.nec.com/lieberman95letizia.html.
11. Mladenic, D., *Personal WebWatcher: Design and Implementation.* 1996, School of Computer Science, Carnegie-Mellon University; http://citeseer.nj.nec.com/mladenic96personal.html.
12. Pazzani, D., Muramatsu, J., Billsus D., *Syskill Webert: Identifying Interesting Web Sites*, in *AAAI Spring Symposium on Machine Learning in Information Access*. 1996: Stanford: p. 54-61; http://citeseer.nj.nec.com/pazzani98syskill.html.
13. Ackerman, M., Billsus, D., Gaffney, S., Hettich, S., Khoo, G., Kim, D.J., Klefstad, R., Lowe, C., Ludeman, A., Muramatsu, J., Omori, K., Pazzani, M.J., Semler, D., Starr, B., Yap, P., *Learning Probalistic User Profiles*, in *AI magazine*. 1997. **18**(2): p. 47-56.
14. Pazzani, M., Billsus, D., *Learning and Revising User Profiles: The Identification of interesting Web Sites*, in *Machine Learning*. 1997. **27**(3): p. 313-331; http://citeseer.nj.nec.com/pazzani97learning.html.
15. Shavlik, J., Eliassi-Rad, T., *Building intelligent agents for Web-based tasks: A theoryrefinement approach*, in *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98.* 1998: Stanford, CA; http://citeseer.nj.nec.com/shavlik98building.html.
16. Bollacker, K., Lawrence, S., Giles, L., *CiteSeer : An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications*, in *Proceedings of the Second International Conference on Autonomous Agents*, K. P. S. a. M. Wooldridge, Editor. 1998, ACM Press: Minneapolis MN, USA: p. 116-123; http://citeseer.nj.nec.com/bollacker98citeseer.html.
17. Lang, K., *News Weeder: Learning to filter Net-news*, in *In Proc. of the 12th International Conference on Machine Learning ICML95*. 1995, Morgan Kaufmann: San Francisco: p. 331-339.
18. Krulwich, B., Burkey, C., *The ContactFinder agent: Answering buletin board questions with referrals*, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI 96*. 1996: Portland, Oregon: p. 10-15.
19. Hammond, K., Burke, R., Schmitt, K., *A Case-Based Approach to Knowledge Navigation*, in *AAAI Workshop on Indexing and Reuse in Multimedia Systems*. 1994, American Association for Artificial Intelligence: p. 46-57; http://citeseer.nj.nec.com/hammond94casebased.html.
20. Burke, R., Hammond, K., Kozlovsky, J., *Knowledge-based Information Retrieval for Semi-Structured Text*, in *Working Notes from AAAI Fall Symposium on AI Application in Knowledge Navigation and Retrieval*. 1995, American Association for Artificial Intelligence: p. 19-24; http://citeseer.nj.nec.com/burke96knowledgebased.html.

21. Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., Schoenberg, S., *Question Answering from Frequently Asked Question Files: Experiences with the {FAQ} Finder System*, in *AI Magazine*. 1997. **18**(2): p. 57-66; http://citeseer.nj.nec.com/article/burke97question.html.
22. Kamba, T., Sakagami, H., Koseki, Y., *ANATAGONOMY: a personalized newspaper on the World Wide Web*, in *International Journal Human-Computer Studies*. 1997. **46**: p. 789-803.
23. LaMacchia, B. A., *Internet Fish*, in *A revised version of a thesis proposal.* 1996, MIT; http://citeseer.nj.nec.com/lamacchia96internet.html.
24. Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J.P., Zabowski, D., *Experience with a Learning Personal Assistant*, in *Communications of the ACM*. 1994. **37**(7): p. 81-91; http://citeseer.nj.nec.com/mitchell94experience.html.
25. Rucker, J., Polanco, M.J., *Siteseer: personalized navigation for the Web*, in *Communication of the ACM*. 1997. **40**(3): p. 73-75; http://doi.acm.org/10.1145/245108.245125.
26. Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J., *PHOAKS: a system for sharing recommendations*, in *Communication of the ACM*. 1997. **40**(3): p. 59-62; http://doi.acm.org/10.1145/245108.245122.
27. Konstan, J. A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J., *GroupLens: applying collaborative filtering to Usenet news*, in *Communication of the ACM*. 1997. **40**(3): p. 77-87; http://doi.acm.org/10.1145/245108.245126.
28. Kautz, H., Selman, B., Mehul, S., *Referral Web: combining social networks and collaborative filtering*, in *Communication of the ACM*. 1997. **40**(3): p. 63-65; http://doi.acm.org/10.1145/245108.245123.
29. Krulwich, B., *Lifestyle Finder*, in *AI Magazine*. 1997. **18**(2): p. 37-46.
30. Holte, R. C., Drummond, C., *A Learning Apprentice For Browsing*, in *AAAI Spring Symposium on Software Agents*. 1994.
31. Drummond, C., Ionescu, D., Holte, R., *A Learning Agent that Assists the Browsing of Software Libraries*, in *Software Engineering*. 1995, Computer Science Dept., University of Ottawa; http://citeseer.nj.nec.com/drummond95learning.html.
32. Etzioni, O., Weld, D., *A softbot-based interface to the Internet*, in *Communication of the ACM*. 1994. **37**(7): p. 72-76; http://doi.acm.org/10.1145/176789.176797.
33. Gams, M., Grobelnik, M., *Intelligent agents in information society*, in *Proceedings of the 6th Electrotechnical and Computer Science Conference ERK'97*. 1997: Ljubljana, Slovenia : Slovenia Section IEEE: p. 125-128.
34. Menczer, F., *ARACHNID: Adaptaive Retrieval Agents Choosing Heuristic Neighborhood for Information Discovery*, in *Proceedings of the 14th International Conference on Machine Learning ICML'97*. 1997: p. 227-235; http://citeseer.nj.nec.com/menczer97arachnid.html.

35. Doorenbos, R. B., Etzioni, O., Weld, D.S., *A Scalable Comparison-Shopping Agent for the World-Wide Web (1997)*, W. L. J. a. B. Hayes-Roth, Editor. 1997, University of Washington.
36. Armstrong, R., Freitag, D., Joachims, T., Mitchell, T., *WebWatcher: A Learning Apprentice for the World Wide Web*, in *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*. 1995: Stanford, CA: p. 6-12; http://citeseer.nj.nec.com/armstrong97webwatcher.html.
37. Kautz, H., Selman, B., Shah, M., *The Hidden Web*, in *AI Magazine*. 1997. **18**(2): p. 27-36; http://citeseer.nj.nec.com/kautz97hidden.html.
38. Maes, P., *Agents that reduce work and information overload*, in *Communication of the ACM*. 1994. **37**(7): p. 30-40; http://doi.acm.org/10.1145/176789.176792.
39. Balabanovic, M., Shoham Y., *Fab: content-based, collaborative recommendation*, in *Communications of the ACM*. 1997, ACM Press. **40**(3): p. 66-72; http://doi.acm.org/10.1145/245108.245124.
40. de Vel, O., Nesbitt, S., *A collaborative filtering agent system for dynamic virtual communities on the Web*, in *Working notes of learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. 1998; citeseer.nj.nec.com/158278.html.
41. Apté, C., Damereau, F., Weiss, S.M., *Toward Language Independent Automated Learning of Text Categorization Models*, in *Proc. of the 7th Annual International ACM-SIGIR COnference on Research and Development in Information Retrieval*. 1994: Dubline.
42. Apté, C., Damereau, F., Weiss, S.M., *Text Mining with Decision Rules and Decision Trees*, in *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. 1998; http://citeseer.nj.nec.com/apte98text.html.
43. Bartell, B. T., Cottrell, G.W., Belew, R.K., *Latent semantic indexing is an optimal special case of multidimensional scaling*, in *Proceedings of the Fifteenth Annual International ACM SIGIR conference on Research and development in information retrieval*. 1992, ACM Press: Copenhagen, Denmark: p. 161-167; http://doi.acm.org/10.1145/133160.133191.
44. Berry, M. W., Dumais, S.T., O'Brien, G.W., *Using Linear Algebra for Intelligent Information Retrieval*, in *SIAM Review*. 1994. **37**(4): p. 573-595; http://citeseer.nj.nec.com/berry95using.html.
45. Foltz, P. W., Dumais, S.T., *Peronalized information delivery: An analysis of information filtering methods*, in *Communication of the ACM*. 1992. **35**(12): p. 51-60; http://doi.acm.org/10.1145/138859.138866.
46. Creecy, R. H., Masans, B.M., Smith, S.J., Waltz, D.L., *Trading MIPS and memory for knowledge engineering*, in *Communication of the ACM*. 1992. **35**(8): p. 48-64; http://doi.acm.org/10.1145/135226.135228.
47. Cohen, W. W., *Learning to Classify English Text with ILP Methods*, in *Proceedings of the 5th International Workshop on Inductive Logic*

Programming, L. D. Raedt, Editor. 1995, Department of Computer Science, Katholieke Universiteit Leuven: Leuven: p. 3-24; http://citeseer.nj.nec.com/cohen95learning.html.

48. Cohen, W. W., Singer, Y., *Context-sensitive learning methods for text categorization*, in *ACM Transactions on Information Systems (TOIS)*. 1999, ACM Press. **17**(2): p. 141-173; http://doi.acm.org/10.1145/306686.306688.
49. Gelfand, B., Wulfekuhler, M., Punch III, W.F., *Automated concept extraction from plain text*, in *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. 1998; http://citeseer.nj.nec.com/147859.html.
50. Joachims, T., *A Probabilistic Analysis of the Rocchio Algorithm with {TFIDF} for Text Categorization*, in *Proc. of the 14th International Conference on Machine Learning ICML97*. 1997: p. 143-151; http://citeseer.nj.nec.com/107422.html.
51. Joachims, T., *Text categorization with support vector machines: learning with many relevant features*, in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. N. e. d. a. C. e. l. Rouveirol, Editor. 1998, Springer Verlag, Heidelberg, DE: p. 137-142; http://citeseer.nj.nec.com/joachims98text.html.
52. Lam, W., Low, K.F., Ho, C.Y., *Using Bayesian Network Induction Approach for text Categorization*, in *15th International Joint Conference on Artificial intelligence IJCAI97*. 1997: p. 745-750.
53. Lam, W., Ho, C.Y., *Using a generalized instance set for automatic text categorization*, in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 1998, ACM Press: Melbourne, Australia: p. 81-89; http://doi.acm.org/10.1145/290941.290961.
54. Lewis, D. D., Ringuette, M., *Comparaison of two learning algorithms for text categorization*, in *Proc. of the 3rd Annual Symposium on Document Analysis and Information Retrieval SDAIR-94*. 1994: Las Vegas, TX, USA: p. 81-93; http://citeseer.nj.nec.com/lewis94comparison.html.
55. Lewis, D. D., Gale, W.A., *A sequential algorithm for training text classifiers*, in *Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval*. 1994, Springer-Verlag New York, Inc.: Dublin, Ireland: p. 3-12.
56. Lewis, D. D., Schapire R.E., Callan, J.P., Ron Papka, R., *Training Algorithms for Linear Text Classifiers*, in *Proc. of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, H.-P. F. a. D. H. a. P. S. a. R. Wilkinson, Editor. 1996, ACM Press, New York, US: Zurich, CH: p. 298-306; http://citeseer.nj.nec.com/lewis96training.html.
57. Liere, R., Tadepalli, P., *Active Learning with Committees: Preliminary Results in Comparing Winnow and Perceptron in Text Categorization*, in

Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98. 1998.

58. Mladenic, D., Grobelnik, M., *Feature selection for classification based on text hierarchy*, in *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. 1998; http://citeseer.nj.nec.com/10425.html.
59. Mladenic, D., Globelnik, M., *Word sequences as features in text learning*, in *Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK98)*. 1998: Ljubljana, Slovenia : IEEE section: p. 145-148; http://citeseer.nj.nec.com/mladenic98word.html.
60. Moulinier, I., Ganascia, J.-G., *Applying an existing machine learning algorithm to text categorization*, in *Connectionist, statistical, and symbolic approaches to learning for natural language processing*, S. W. a. E. R. a. G. Scheler, Editor. 1996, Springer Verlag: p. 343-354; http://citeseer.nj.nec.com/moulinier96applying.html.
61. Nigam, K., McCallum, A., *Pool-Based Active Learning for Text Classification*, in *Conference on Automated Learning and Discovery CONALD-98*. 1998; http://citeseer.nj.nec.com/23143.html.
62. Slattery, S., Craven, M., *Learning to Exploit Document Relationships ans Structure: The Case for Relational Learning on the Web*, in *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery CONALD-98*. 1998.
63. Sorensen, H., McElligott, M., *PSUN: A Profiling System for Usenet News*, in *CIKM'95 Intelligent Information Agents Workshop*. 1995: Baltimore; http://citeseer.nj.nec.com/118055.html.
64. Wiener, E. D., Pedersen, J.O., Weigend, A.S., *A neural network approach to topic spotting*, in *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*. 1995: Las Vegas, US: p. 317-332; http://citeseer.nj.nec.com/wiener95neural.html.
65. Yang, Y., *Expert network: effective and efficient learning from human decisions in text categorization and retrieval*, in *Proceedings of the seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval*. 1994, Springer-Verlag New York, Inc.: p. 13-22.
66. Yang, Y., *An evaluation of statistical approaches to text categorization*, in *Journal of Information Retrieval*. 1998. **1**(1/2): p. 69-90; http://citeseer.nj.nec.com/90507.html.
67. Bradshaw, J. M., *Software agents*, J. M. Bradshaw, Editor. 1997.
68. Chen, L., Sycara, K., *WebMate: A Personal Agent for Browsing and Searching*, in *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)"*, K. P. S. a. M. Wooldridge, Editor. 1998, ACM Press: New York. **9-13**: p. 132-139; http://citeseer.nj.nec.com/chen98webmate.html.

69. Mohapatra, T. K., *ProProxy*. 2001; http://members.tripod.com/tanmaykm/work/javawork.html.
70. Do, A., *ADC Parser*. 1998; http://www.do.org/products/parser/.
71. McGill, M. J., Salton, G., *Introduction to Modern Information Retrieval*. 1983.
72. Matsumura A., T. A., Adachi J.:, *The Effect of Information Retrieval Method Using Dependency Relationship Between Words*, in *Conference Proceedings of RIAL2000*. 2000: Paris, France. **April 12-14**: p. pp.1043-1058; citeseer.nj.nec.com/414155.html.
73. Tomek, S., *Building Effective Queries In Natural Language Information Retrieval*, in *Proceedings of the 5th Applied Natural Language Conference (ANLP-97)*. 1997: Washington, DC: p. 299-306; http://citeseer.nj.nec.com/tomek97building.html.
74. Miller, D. R. H., Leek, T., Schwartz, R.M., *A Hidden Markov Model Information Retrieval System*, in *Research and Development in Information Retrieval (SIGIR'99)*. 1999: Berkeley, US: p. 214-221; http://citeseer.nj.nec.com/miller99hidden.html.
75. Miller, G. A., *Wordnet*. 2001, Princeton University; http://www.cogsci.princeton.edu/~wn/.
76. Steele, O., *JWordNet*. 1998; http://jwn.sourceforge.net/.
77. Microsoft, *Microsoft Agent*. 1998; http://www.msdn.microsoft.com/msagent/.
78. Cohen, W. W., *Recognizing Structure in Web Pages using Similarity Queries*, in *American Association for Artificial Intelligence*. 1999: p. 59-66; http://citeseer.nj.nec.com/cohen99recognizing.html.
79. Tschalär, R., *HTTPClient*. 1996-2001; http://www.innovation.ch/java/HTTPClient/.