Université de Montréal

Faculté des études supérieures

WHITE RABBIT - Agents intelligents d'analyse de discussion pour la reconnaissance de profils d'usagers compatibles

Par :

Marc-André Thibodeau

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures

en vue de l'obtention du grade de

Maître ès Sciences (M.Sc.) en informatique

Décembre, 2000

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

WHITE RABBIT - Agents intelligents d'analyse de discussion pour la reconnaissance de profils d'usagers compatibles

Présenté par :

Marc-André Thibodeau

A été évalué par un jury composé des personnes suivantes :

Président-rapporteur :	Jean Vaucher
Directeur de recherche :	Claude Frasson
Co-directrice de recherche :	Esma Aïmeur
Membre du Jury :	Marc Kaltenbach

Mémoire accepté le : 16 février 2001

Sommaire

Ce mémoire aborde la problématique de la découverte de personnes compatibles à l'intérieur d'une organisation ou d'une entreprise afin d'en améliorer la coordination. L'approche que nous avons choisie emploie des agents qui apprennent sur les intérêts de leurs utilisateurs par l'analyse des messages que ceux-ci envoient à travers une interface de discussion en temps réel (*chat room*). Les agents construisent ainsi des profils des utilisateurs qui sont ensuite soumis à un réseau de neurones non-supervisé, de type Kohonen Maps, pour en faire la catégorisation. Finalement, des présentations sont effectuées entre les personnes se trouvant classées dans une même catégorie et possédant par conséquent des profils potentiellement compatibles.

White Rabbit, le système construit durant cette recherche, est inspiré d'autres systèmes de jumelage (de l'anglais, *matchmaking*) utilisant des agents intelligents, en particulier Yenta, Butterfly et Arcadia. Toutefois, notre système se démarque par l'utilisation du réseau de neurones pour faire la catégorisation, une technique plus sophistiquée que les algorithmes de catégorisation utilisés par ces autres systèmes.

Notre système a été évalué selon plusieurs aspects dans un environnement contrôlé. Les résultats des tests ainsi effectués sont encourageants et nous permettent de croire qu'un transfert de notre approche à un réel environnement industriel serait possible dans l'avenir.

Table des matières

Table des figures

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé à mener à bien cette recherche. Plus particulièrement, je remercie mon directeur Claude Frasson pour m'avoir finement dirigé dans mes travaux et mes amis Simon et Amal pour des discussions philosophiques enlevantes et un climat de travail inoubliable.

Introduction

Dans une grande entreprise ou un centre de recherche de grande envergure, il est fréquent de voir surgir des problèmes de mauvaise coordination et de manque de coopération. Les dédoublements d'efforts deviennent monnaie courante et il s'ensuit généralement une importante baisse de productivité par rapport à celle escomptée. Le fait est que lorsque beaucoup de gens travaillent au sein d'une même organisation, tous ne sont pas au courant de toutes les ressources disponibles ou des autres projets en cours au sein de celle-ci. Parfois, des groupes de personnes travailleront sur des projets similaires, voire identiques, réinventant des outils déjà réalisés, ou développant plusieurs fois des composantes semblables plutôt que de conjuguer leurs efforts et de partager leurs connaissances. De la même façon, des ressources comme des experts ou des réalisations passées pourraient être mises à profit, mais ne le sont pas en raison de l'ignorance de leur existence.

La gestion des connaissances et des ressources des entreprises est un problème complexe sur lequel se penche de nombreux théoriciens depuis plusieurs années. Mais d'où provient cette complexité ? Selon Peter Senge, fondateur du "Center of Organization Learning" au MIT, elle provient des relations humaines à l'intérieur d'une grande organisation. [20]

> "Dans un monde humain, vous devez négocier avec la complexité humaine. Vous devez négocier avec le fait que les gens sont différents. (...) Or, les personnes, comme individus, ne créent rien. La création, ou la réalisation

> de quelques chose de nouveau, est toujours le produit d'une communauté humaine. (...) La qualité de la pensée dans les organisations est très, très fortement influencée par la qualité des relations à l'intérieur de celles-ci."

Larry Prusack, gérant principal du groupe de consultants d'IBM et directeur du "Knowledge Management Institute", parle, quant à lui, du problème de la confiance. Selon lui, la confiance mutuelle entre les gens à l'intérieur d'une organisation est la clé de son succès. Il ajoute qu'on peut faire quelque chose pour améliorer la coopération. [20]

> "La chose la plus utile que vous puissiez faire si vous voulez optimiser ce qu'une organisation "connaît", c'est de rendre ses connaissances visibles de telle sorte que chacun soit en mesure d'utiliser ce que les autres connaissent."

Rendre visible les connaissances des personnes faisant partie d'un groupe aux autres personnes du groupe est une chose à la base assez simple. Il suffit d'afficher publiquement leurs compétences et leurs besoins, sur un babillard étectronique par exemple. Toutefois, cette dernière option :

- facilite peu la découverte d'un expert puisqu'elle ne réduit pas l'espace de recherche : une personne doit parcourir elle-même toute la liste de profils et juger elle-même lesquels sont intéressants pour elle.
- ne facilite pas l'établissement de relation de confiance entre les individus. L'anonymat ne peut être ainsi préservé et on doit afficher publiquement nos compétences et nos intérêts. Or, l'être humain préfère généralement ne divulguer ces informations que de façon graduelle et réciproque avec les individus qui l'entourent.
- encourage peu l'entraide en ne présentant pas de manière automatique des personnes pouvant être utiles. Une personne doit faire l'effort d'aller elle-même consulter la liste publique. Et puisqu'il s'agit d'une charge de travail supplémentaire, peu de gens se tourneraient vers cette solution lorsqu'ils ont besoin d'aide.

Regrouper les personnes compatibles de façon automatisée est une autre paire de manches. En effet, construire des profils des personnes, regrouper les profils similaires et finalement introduire les personnes les unes aux autres, tout ceci de manière automatique, est beaucoup plus complexe que de simplement publier une liste des compétences de tous et chacun sur un babillard électronique. Ce travail implique l'utilisation de techniques de l'Intelligence Artificielle qui permettont à un programme d'apprendre sur les intérêts et compétences de ses usagers de façon autonome, de catégoriser ces usagers et d'intervenir auprès d'eux pour les encourager à coopérer. Certains systèmes ont déjà tentés d'aborder ce problème [13] ou encore des variantes de celui-ci [9], [5]. Cependant, ils ont rencontrés des difficultés importantes :

- l'accès à des données personnelles (documents électroniques, courriels) des individus pour construire leur profil pose un important problème de confidentialité.
- les méthodes de catégorisation qu'ils utilisent sont rudimentaires.
- les méthodes d'apprentissage sont limitées par l'absence de structures ou de représentation du domaine de connaissances.

C'est donc cette problématique que nous tenterons d'aborder et de résoudre. L'approche que nous avons choisie est de favoriser l'établissement de relations entre les employés d'une entreprise à l'aide d'un système de jumelage.

L'objectif premier de ce système sera de mettre en relief les ressemblances et les complémentarités des intérêts et des compétences de ses utilisateurs pour que ceux-ci puissent plus facilement établir des liens les uns avec les autres. Pour ce faire, nous tenterons de découvrir les gens compatibles en analysant la conversation qu'ils tiennent à travers une interface de discussion en temps réel, de sorte à éviter le problème de la confidentialité posé par l'analyse des courriels et documents électroniques. L'analyse sera de plus basée sur une structure de connaissances portant sur un domaine choisi afin d'en augmenter la précision. Les résultats de cette analyse pourront ensuite servir à catégoriser les usagers, et ce à l'aide de techniques avancées, pour finalement

les mettre en communication les uns avec autres lorsqu'ils feront partie d'une même catégorie.

Ceci, croyons-nous, pourra d'une part faciliter la "rencontre" entre les employés et donc les aider à collaborer en permettant l'établissement graduel de liens entre eux, et d'autre part rendre plus visible les connaissances disponibles à l'intérieur de l'entreprise.

Notre second objectif sera d'obtenir une solution adaptée au monde réel, ou en d'autres termes, qui pourrait facilement être transférée à une entreprise ou appliquée à d'autres domaines. Ceci implique l'utilisation de technologies récentes et solides et le développement d'une architecture souple et pouvant supporter un nombre assez grand d'utilisateurs.

Ce mémoire fera d'abord le point au chapitre 1 sur l'état de la recherche dans le domaine des systèmes de jumelage en présentant plusieurs de ceux-ci, dont quelques-uns traitent d'une problématique semblable à la nôtre. Il s'agira d'une étude qui fera ressortir les caractéristiques et lacunes de ces systèmes et qui nous permettra ensuite de faire des choix éclairés quant à l'architecture et aux méthodes que nous implanterons. Le chapitre 2 fera donc un survol des techniques d'explicitation des connaissances et de catégorisation, deux concepts clés des systèmes de jumelage. Les premières servent à extraire et formaliser les connaissances des gens. Les secondes sont les méthodes qui sont utilisées pour faire des regroupements d'objets (ou entités) présentant des traits communs. Nous serons à ce point-ci en mesure de sélectionner les méthodes les plus appropriées pour résoudre notre problème. Suite à cette étude théorique, nous pourrons enfin donner au chapitre 3 les détails de l'architecture et de l'implantation de notre système, en décortiquant chacun de ses modules. Ce chapitre présentera également le système ALICE, fruit d'un projet de maîtrise conjoint. Notre système fera ensuite l'objet d'une évaluation, décrite au chapitre 4 visant à déterminer s'il présente des performances acceptables et si nos objectifs ont été atteints ou non.

Ce dernier chapitre ouvrira finalement la porte sur les possibilités futures de notre architecture et sur les améliorations qui pourraient lui être apportées.

Chapitre 1

Les systèmes de jumelage

Le jumelage (en anglais, "matchmaking"), est le processus qui consiste à découvrir un donneur approprié pour un demandeur via un agent faisant l'intermédiaire entre les deux parties [30]. Ainsi donc, le fonctionnement classique des systèmes de jumelage s'établit comme suit : (1)les agents des fournisseurs informent un ou plusieurs agents intermédiaires de leurs capacités, (2)le ou les agents intermédiaires emmagasinent ce profil, (3)un requérant demande à un agent intermédiaire s'il connaît un ou des fournisseurs possédant certaines capacités, puis (4)l'agent intermédiaire jumelle la requête avec les profils emmagasinés et retourne le résultat, un sous-ensemble de tous les profils.

Les systèmes de jumelage sont généralement des systèmes centralisés, i.e. où toute l'information est stockée sur une machine unique. Une telle organisation est attrayante pour sa simplicité, en particulier pour le programmeur. En effet, les systèmes centralisés [13] :

- sont faciles à administrer, puisque presque toutes les ressources logicielles se retrouvent sur une ou des machines qui sont directement sous le contrôle du programmeur,

- s'ils sont utilisés pour le commerce, il est généralement trivial de les structurer de telle sorte que les usagers puissent être facturés pour leur utilisation, ou encore de telle sorte que de la publicité leur soit transmise durant l'utilisation,
- s'ils requièrent également que les informations personnelles des usagers soient ré-utilisées, par exemple à des fins de marketing, alors centraliser les données sur les serveurs même de la compagnie rend cette tâche facile.

Ce chapitre fait la présentation de plusieurs systèmes de jumelage ayant inspiré la conception de notre solution. Les systèmes Yenta, Butterfly et Arcadia sont décrits plus en détails, puisque ceux-ci touchent à une problématique semblable à la nôtre. Finalement, cette discussion nous amènera vers un survol des caractéristiques que possédera notre système.

1.1 Yenta

Yenta est un système de jumelage conçu pour trouver des personnes ayant des intérêts similaires et encourager la communication entre ces personnes [12]. Ce système est composé d'un ensemble d'agents[1] décentralisés se regroupant en catégories (ou *clusters*) pour représenter les similarités d'intérêts des usagers. Ces catégories peuvent ensuite être utilisées pour faire les présentations et permettre aux usagers d'envoyer et recevoir des messages aux personnes faisant partie de la même catégorie, i.e. ayant des intérêts semblables. L'algorithme utilisé se sert de références entre les agents et simule la communication "de bouche à oreille" entre les gens lorsque ceux-ci cherchent un expert.

[1]Tout au long de ce document, le terme agent employé seul désigne un agent intelligent, une entité logicielle définie entre autre dans [24]

Comme nous l'avons déjà mentionné, plusieurs systèmes d'agents implantés actuellement utilisent une architecture centralisée, par laquelle un agent sert un ou plusieurs utilisateurs. Toutefois, une telle architecture présente des désavantages :

- il est difficile d'appliquer une architecture centralisée à des systèmes de grande taille. Shardanand et Maes [26] ont montré que le problème des communications à travers un tel système est généralement d'une complexité d'ordre quadratique ;
- un serveur centralisé procure un seul point où un panne accidentelle peut avoir des conséquences catastrophiques pour un système qui doit être fiable et disponible à tout moment.

Un problème important des systèmes décentralisés est celui, pour les agents, de trouver les autres agents et de communiquer avec eux. Une solution simple à ce problème est d'organiser les entités du système en une hiérarchie, comme c'est le cas pour les enregistrements de ressources dans le DNS (domain-name system) de l'Internet. Cette solution permet à Yenta de réduire la complexité du problème de recherche de quadratique à logarithmique. [14]

Toutefois, un système possédant un grand nombre d'agents présente de façon inhérente des difficultés de coordination. Par exemple, si tous les agents du système doivent communiquer avec chacun des autres agents du système, il s'ensuivra une croissance exponentielle du nombre de messages transmis en fonction du nombre d'agents et par conséquent, le système ne pourra pas accueillir un grand nombre d'utilisateurs.

Les idées centrales utilisées par Yenta pour la découverte des bonnes catégories pour les agents se résument comme suit :

- comparer l'information des agents d'une manière décentralisée, c'est-à-dire, un à un,

- utiliser les références entre les agents et un algorithme semblable au "hill-climbing"[17] pour trouver les autres agents plus appropriés lors de la recherche de pairs,
- construire des catégories d'agents ayant des affinités,
- utiliser ces catégories pour faire les présentations entre les usagers aux intérêts similaires et pour permettre la messagerie entre eux,
- utiliser des agents persistants, i.e. qui s'exécutent sur de longues périodes de temps, en arrière-plan. Les agents profitent de ce temps pour communiquer avec les autres agents.

Les détails de l'algorithme de catégorisation sont fournis par Foner et Crabtree [14] et sont résumés à la section 1.1.3.

Une fois les catégories construites, elles peuvent être utilisées de différentes façons. D'abord, pour faire la messagerie à l'intérieur d'un groupe. Un usager peut par exemple envoyer un message à un agent particulier de la même catégorie ou encore à tous les agents du groupe par un algorithme d'inondation (*flooding*) en utilisant l'information contenue dans la "cache de cluster" (voir section 1.1.2). Ensuite, les présentations entre usagers peuvent être faites à l'intérieur de la catégorie de façon graduelle et symétrique, afin de préserver la sécurité et l'établissement d'une relation de confiance entre les usagers qui sont présentés. Les usagers doivent pouvoir demander explicitement à être présentés à une certaine personne. Ainsi, par la messagerie et les présentations, le système aide l'usager à trouver l'expert dont il a besoin, puisque les intérêts de celui-ci, s'il est représenté par un agent du système, se retrouvent également groupés par catégories.

1.1.1 Trouver les intérêts d'un usager

L'agent d'un usager doit d'abord déterminer ses intérêts, et ensuite mettre ce profil (d'intérêts) à jour de temps à autres afin de refléter l'évolution des intérêts de

l'usager dans le temps. Ceci peut être fait de plusieurs façons. Par exemple, l'agent peut demander périodiquement des mots clés ou des phrases de l'usager qui résument ses intérêts. Ou encore, l'agent peut le faire de façon semi-automatique par l'analyse des courriels ou des fichiers de texte de l'usager. Cette dernière approche est celle utilisée par Yenta.

Les informations recueillies sur les intérêts des usagers doivent être convenablement regroupées en différents domaines d'intérêts de façon précise afin de pouvoir être utilisées par les agents. Il s'agit de l'étape de la "pré-catégorisation". Afin de faire une pré-catégorisation efficace, Yenta nécessite un traitement préalable des courriels afin d'éliminer les informations superflues comme les signatures et de diviser leur contenu en particules plus petites afin de ne pas perdre des informations des messages longs.

1.1.2 Les structures de données utilisées pour la catégorisation

Dans Yenta, chaque agent comporte différentes structures de données lui permettant de faire ses recherches et de former avec les autres agents des catégories efficaces. Ces structures de données sont :

- une *cache de cluster*, qui contient les noms de tous les agents connus présentement par l'agent comme étant dans la même catégorie que lui ;
- une *cache de rumeur*, qui contient les noms et d'autres informations (un sous-ensemble des textes analysés par les agents correspondants) sur les r derniers agents avec lesquels l'agent a communiqué ;
- une liste de contacts qui est une liste prioritaire des agents qui ont été découverts et que l'agent n'a pas encore contactés.

1.1.3 La catégorisation

Une fois la pré-catégorisation complétée, la catégorisation des agents doit être réalisée. Voici comment cela se produit pour Yenta : Un certain agent A trouve un autre agent B. L'agent A fait alors la comparaison de ses informations locales (groupes de fichiers textes) avec celles de l'agent B. Une fois cette comparaison complétée, l'agent A peut avoir trouvé des ressemblances acceptables. Si c'est le cas, les caches sont mises à jour. L'agent B effectue le même processus, symétriquement, sur A. Ensuite, que des ressemblances aient été trouvées ou non, l'étape suivante consiste à acquérir des références vers des agents pouvant constituer des meilleurs jumelages. Pour ce faire, l'agent A fait alors les mêmes comparaisons qu'il avait faites sur B, mais cette fois-ci sur les informations contenues dans la cache de rumeur de B. Encore une fois, les bons jumelages sont ajoutés aux caches et les noms des agents concernés sont ajoutés à la liste de contacts de A. Ces agents seront donc contactés une fois que A en aura fini avec B et les autres agents déjà présents dans sa liste. Encore une fois, B effectue le même processus sur A. De cette façon, les agents A et B augmentent leurs connaissances sur les autres agents du systèmes et par le fait même, sur les autres usagers du système pouvant être compatibles avec les intérêts des usagers qu'ils représentent.

Cette technique s'inspire du bouche à oreille utilisé dans la vie de tous les jours pour trouver les personnes qui peuvent nous aider. Il s'agit d'une variante de l'algorithme de "*hill-climbing*" qui consiste à commencer une recherche à un point donné, avec un objectif fixé, et de remonter à travers le réseau de contacts jusqu'à un expert qui peut répondre au besoin défini par l'objectif.

1.2 Butterfly

Butterfly est un système d'agents intelligents qui fait l'analyse de la conversation sur IRC (*Internet Relay Chat*) afin de suggérer aux utilisateurs des canaux de discussion susceptibles de les intéresser [9]. Pour ce faire, il échantillonne les conversations ayant lieu dans les divers groupes de discussion disponibles et construit, pour chaque utilisateur, un profil de ses intérêts. Nous verrons en détails plus loin la méthode adoptée par Butterfly pour la construction des profils, l'analyse des conversations, et les suggestions qui s'ensuivent.

1.2.1 IRC

Trois moyens populaires d'interaction de groupe basés sur l'Internet existent à l'heure actuelle. Les listes de courrier électronique (*mailing lists*), les forums de discussion (*newsgroups*), et la discussion en temps réel (*chat*). IRC est le réseau de discussion de la troisième catégorie qui est le plus répandu et le plus populaire. Les groupes de conversation sur IRC sont définis par canaux (ou "chat rooms"). Les utilisateurs se joignent de manière explicite aux canaux auxquels ils désirent participer et tout message que ceux-ci envoient sur un canal est vu par tous les autres utilisateurs branchés sur celui-ci. Les canaux sont créés sur demande par n'importe quel utilisateur IRC et ils demeurent actifs tant qu'il reste au moins une personne l'utilisant. Il y a en moyenne 10 000 canaux actifs à tout moment, chacun d'eux étant identifié par un court nom qui souvent ne donne que peu d'indications sur leur contenu. Sur IRC, aucune hiérarchie ou aucun mécanisme d'organisation n'existe afin d'aider les utilisateurs à trouver des canaux intéressants. Trouver un canal approprié devient donc un jeu de devinette pour les utilisateurs.

C'est ce problème précis qui est abordé par Butterfly. En effet, il tente de faciliter la tâche de trouver des canaux d'intérêts par l'échantillonnage du contenu de tous les canaux et en gardant un profil d'intérêts des utilisateurs basé sur des mots clés.

Le fait que Butterfly ait ainsi été construit en utilisant une infrastructure de discussion existante et de grande taille, plutôt qu'un "modèle réduit" de laboratoire, a grandement influencé la conception du système, comme nous le verrons maintenant.

1.2.2 Les profils des utilisateurs et du contenu des canaux

Le profil d'intérêts utilisé par Butterfly est basé sur un simple vecteur de termes associés à des poids positifs et négatifs. La version actuelle de Butterfly utilise des constantes fixes pour un intérêt "normal", "élevé" et "mitigé" (exemple : 100, 200, -50). Toutefois, des poids plus variés pourraient être utilisés si les profils étaient appris.

Dans Butterfly, le contenu des canaux est également représenté par des vecteurs de termes avec des poids correspondants à la fréquence d'occurrences des termes dans les conversations. Ainsi, il devient possible de déterminer le degré d'intérêt d'un canal pour un utilisateur en calculant le produit scalaire des deux vecteurs. Si le produit obtenu est supérieur ou égal à un certain seuil, alors le canal est recommandé à l'utilisateur. La construction et l'évolution des vecteurs représentant le contenu des canaux dépend directement de la méthode utilisée pour l'échantillonnage des canaux.

1.2.3 Échantillonnage des canaux

Une première contrainte rencontrée par Butterfly dans sa tâche d'échantillonnage des canaux d'IRC est celle de la confidentialité. En effet, bien que les discussions soient publiques, les utilisateurs d'un canal croient que seuls les autres utilisateurs du même canal sont en mesure de lire ce qu'ils écrivent. Ainsi, un agent qui "espionnerait" ces conversations, à l'insu des personnes en cause, violerait cette règle de confidentialité. C'est pourquoi Butterfly se manifeste sur les canaux comme un utilisateur normal, c'est-à-dire qu'il est visible et identifiable par les autres utilisateurs du même canal.

Une seconde contrainte est posée par IRC pour l'échantillonnage des canaux : les utilisateurs, à tous moment, ne peuvent se joindre à plus d'une dizaine de canaux simultanément. Par conséquent, Butterfly ne survole qu'une dizaine de canaux à la fois en utilisant un "horaire de visite" donnant la priorité aux canaux n'ayant jamais été visités.

En visitant un canal, Butterfly construit le vecteur de termes représentant son contenu, en comptant les occurrences de mots clés dans la conversation. Au moment de quitter le canal, il diminue de moitié les poids de l'ancien vecteur pour ce canal et additionne les poids du vecteur de cette visite. Butterfly permet ainsi au vecteur de contenu d'évoluer avec la conversation et de donner plus d'importance aux éléments récents de celle-ci. Butterfly détermine finalement si le canal doit être recommandé à des utilisateurs.

Un autre choix important ayant été fait dans la conception de Butterfly est celui d'une architecture centralisée, contrairement à Yenta. C'est-à-dire qu'un agent représente plusieurs utilisateurs plutôt qu'un seul. Ce choix a été fait afin de ne pas surcharger le réseau IRC, mais entraîne certaines limites desquelles il a déjà été discuté précédemment, à la section (1.1).

1.2.4 L'interaction avec l'usager

Les agents de Butterfly envoient des messages IRC comme s'ils étaient des utilisateurs humains connectés normalement, pour communiquer avec ses usagers. L'avantage pour ces derniers est de ne pas avoir à installer de logiciels supplémentaires pour utiliser Butterfly. Tout se passe dans la fenêtre de discussion habituelle. De plus, les commandes utilisées sont en pseudo-anglais comme par exemple : "Who am I?", "I am interested in *something*", "Where have you been?", "I am very interested in *something*", etc. Le but d'utiliser un langage pseudo-naturel n'est pas d'humaniser le comportement de l'agent ou encore de mystifier l'utilisateur en lui faisant croire qu'il

s'adresse à un être humain, mais plutôt de lui offrir une syntaxe facile à retenir et significative.

1.2.5 L'avancement du projet

Comme Yenta, Butterfly est un prototype opérationnel. Il a été soumis à des tests limités. L'un des problèmes anticipés est la courte durée des périodes d'échantillonnage sur chaque canal, ce qui implique que le système doit s'exécuter longtemps avant que les vecteurs de contenu ne soient représentatifs des conversations en cours dans les canaux. Un autre problème de taille est celui de l'existence de canaux "secrets" dans IRC qui sont souvent, selon des habitués du réseau, les canaux ayant la plus grande valeur. Or, Butterfly ne peut pas en pratique trouver tous les canaux secrets et ainsi peut ne jamais rencontrer quelques-uns des meilleurs canaux.

1.2.6 Les perspectives futures

On explore présentement l'idée de fournir à Butterfly une architecture multi-agent distribuée, i.e. où chaque utilisateur possède un agent. Celle-ci utiliserait aussi des agents de groupe connaissant les propriétés des canaux et des nouveaux participants désirés. Les propriétés d'un groupe sont :

- son rôle,
- les sujets qu'il traite,
- son format de discussion (questions/réponses, discussion ouverte, etc.),
- le ton de l'interaction (politesse, agressivité etc.),
- ses règles sociales,
- Ses dynamiques

Ces informations sur les groupes pourraient être en parties obtenues de manière explicite par les utilisateurs de ces groupes, et en parties déduites de l'observation des interactions.

1.3 Arcadia

Arcadia est un projet conjoint ayant uni les efforts de trois grands laboratoires : France Télécom/CENT, IRIT et ONERAA/CERT [5]. Le but de ce projet était de démontrer que les principes des systèmes à multiples agents étaient applicables au développement de services d'accès à l'information innovateurs. Le résultat fut la création d'une architecture à agents multiples servant à faciliter à ses utilisateurs la localisation d'information digne d'intérêt, ce qui se rapproche beaucoup du travail d'un fureteur. Le problème des fureteurs actuels est qu'ils nécessitent une quantité incommensurable de travail pour réaliser l'indexation et l'entreposage d'une masse d'information disponible sur l'Internet toujours grandissante. Donc, l'idée d'utiliser un ensemble d'agents coopératifs pour se partager le travail, ayant donné naissance à Arcadia, semble être toute naturelle. L'architecture ainsi développée a ensuite été expérimentée par l'implantation d'un système effectuant un jumelage entre acheteurs et vendeurs de véhicules usagers.

1.3.1 L'architecture d'Arcadia

Les agents d'Arcadia possèdent une architecture décentralisée à trois niveaux : le niveau médiation, le niveau transaction et le niveau croyances. Chacune de ces couche est en fait constituée d'agents communiquant entre eux et reliés à l'un des agents de la couche supérieure, procurant une structure arborescente. Les agents de plus haut niveau, les médiateurs, distribués sur différentes machines, ont pour rôle premier de permettre l'ajout transparent de nouveaux services et de nouveaux usagers provenant de n'importe où sur la planète. Leur second rôle est de structurer le domaine de recherche afin de faciliter la localisation rapide de services intéressants pour un usager. La seconde couche, quant à elle, fait correspondre à chaque usager et à chaque service un agent de transaction. Finalement, chacun des agents des deux premières couches possède des croyances sur les autres agents composant la "société".

Il s'agit de la troisième et dernière couche de l'architecture Arcadia. Ces croyances sont continuellement mises à jour suivant le flot de transactions ayant lieu à travers le système.

1.4 Autres systèmes

Webhound/Webdoggie [19] et HOMR/Ringo/Firefly [18] sont des exemples typiques de systèmes de jumelage centralisés. Dans leur cas, un serveur centralisé maintien l'information sur les intérêts des usagers et ceux-ci se connectent au système (via un fureteur) pour découvrir si un "mariage" a été réussi. La faiblesse de ces deux systèmes est qu'ils requièrent tous deux une participation importante de l'usager pour établir et maintenir leur profil d'intérêts.

Kuokka et Harada [16] ont décrit un système procurant un service de vente (brokering service) qui jumelle des offres de ventes et des demandes provenant des utilisateurs. Utilisant aussi une architecture centralisée, il nécessite une représentation des intérêts des usagers hautement complexe et structurée.

Par ailleurs, le système Sixdegrees [28] offre une idée intéressante de jumelage. En effet, ce site garde l'information sur les personnes connues par les usagers et tente ensuite, à partir de cette information, de trouver les arbres de recouvrement minimaux (les chemins les plus courts) vers les autres personnes que voudraient connaître ces usagers. Il procède en demandant les adresses de courrier électronique des personnes connues par l'usager ainsi que le lien que ceux-ci possèdent avec lui (ainsi que quelques autres informations comme la profession, etc.). Le système peut alors contacter les personnes qu'un usager veut connaître, et si la relation n'est pas refusée par la personne contactée, alors le lien est établit. Ainsi, Sixdegrees peut par exemple répondre à des questions comme "Qui est-ce que je connais, connaît un médecin ?".

Bien entendu, ce système est également centralisé et requiert certains efforts pour rassurer ses utilisateurs sur le respect de la confidentialité des informations fournies.

PlanetAll [22] est un système similaire au précédent, cependant, celui-ci cherche plutôt à trouver des personnes connues jadis par un usager. Comme Sixdegrees, il s'agit d'un système centralisé basé sur le WWW. Toutefois, le principe clé derrière PlanetAll est le "groupe d'affinités", i.e. les organisations du monde réel desquelles l'usager a déjà été membre durant sa vie (Écoles, clubs, organisations religieuses, etc.). L'usager fournit donc une liste des organisations desquelles il a fait partie, et le système peut ensuite permettre la correspondance entre les personnes d'un même groupe. Malheureusement, bien que l'objectif du système soit encore une fois noble, il constitue une proie de choix pour des pirates en quête de fausses identités et représente un intérêt commercial important pour des compagnies désireuses d'accroître leur visibilité auprès de clients potentiels.

D'autres systèmes à ne pas négliger sont les systèmes de jumelage romantiques. En effet, un très grand nombre de systèmes tentent d'aider leurs usagers à rencontrer l'âme soeur. Ces systèmes semblent être invariablement centralisés et n'utiliser que des méthodes d'apprentissage et de catégorisation rudimentaires. Par exemple, Match.com [10] acquiert de l'usager une description de lui-même et de la personne qu'il désire rencontrer (sexe, âge, localisation géographique, etc.), puis utilise un simple algorithme de filtrage pour sélectionner les annonces compatibles avec les descriptions obtenues.

Bien sûr, il existe aussi des système de jumelage décentralisés autres que Yenta et Arcadia. Une grande partie de ces systèmes sont utilisés pour le commerce électronique et simulent par exemple des centres commerciaux. Ceux-ci tendent à utiliser des agents qui se déplacent d'un endroit à l'autre, négocient, achètent, vendent des produits, des actions, etc. Le système Harvest [15] en est un bon exemple. Harvest utilise

un ensemble décentralisé d'agents fouineurs, négociateurs, collecteurs et autres pour conclure des ventes sur le WWW.

1.5 Notre système

Cette discussion nous permet maintenant de faire ressortir les caractéristiques que possédera notre système. Nous nommerons ce dernier White Rabbit.

Tout d'abord, nous avons choisi d'implanter une architecture que nous pourrions qualifiée de semi-centralisée. En effet, notre système sera décentralisé du point de vue des agents personnels, comme c'est le cas pour Yenta. En d'autres mots, chaque utilisateur sera représenté par un agent intelligent qui lui sera dédié et qui sera chargé de construire un profil de ses intérêts. Cette architecture, comme nous l'avons souligné à la section 1.1, possède le désavantage de poser des risques de congestion lorsque le nombre d'agents devient grand. Or dans notre cas, ceci n'est que peu préoccupant puisqu'à l'instar de Yenta, White Rabbit s'adresse à une communauté limitée de chercheurs.

Maintenant, l'architecture de White Rabbit sera centralisée du point de vue de la catégorisation qui sera exécutée par un agent "médiateur" unique, semblable à celui de l'architecture Arcadia (section 1.3). Cet agent médiateur sera aussi en charge d'aiguiller les messages entre les agents personnels du système.

On peut dès lors noter les similitudes entre notre architecture et celle d'Arcadia. En effet, la correspondance entre les deux architectures est simple à établir. Comme Arcadia, la nôtre possédera à son sommet la couche de médiation, facilitant la localisation des agents et par conséquent la localisation des usagers et services intéressants. La couche transactionnelle d'Arcadia sera dans White Rabbit constituée d'agents personnels, représentant chacun des usagers du système et, finalement, la couche de croyances sera représentée par les profils des intérêts des usagers. La force

d'une telle architecture est sa grande flexibilité. Elle est suffisamment générale pour pouvoir être appliquée à un grand nombre de domaines. Toutefois, White Rabbit possédera un avantage majeur sur Arcadia : sa représentation des connaissances. Dans le cas d'Arcadia, chaque mot d'une requête soumise au système par un usager est traité individuellement. Or, notre système possédera un graphe de connaissances (un type de réseau sémantique), lui fournissant une grande quantité d'information sur les différents liens existant entre les mots clés. Ceci lui permettra donc de découvrir des profils d'usagers ou de services intéressants, même dans le cas d'une requête inexacte de la part de l'usager. Par exemple, si la requête contient des synonymes de mots clés représentant des services ou des usagers, ceux-ci doivent également être considérés comme intéressants, ce qu'Arcadia est incapable de faire. White Rabbit sera donc pourvu de capacités d'analyse de la langue naturelle de base qui pourraient être poussé plus loin lors de futurs développements. (voir section 4.2)

L'utilisation de mots clés présentera par ailleurs un autre avantage important. En effet, cela permettra à White Rabbit de ne considérer, dans les textes analysés, que les mots clés contenus dans son graphe de connaissance, ou encore les mots similaires à ceux-ci. Par conséquent, toute information superflue sera ignorée et ce, sans recourir à un traitement préalable de l'information. Ceci élimine par exemple le besoin de pré-traiter les messages analysés, comme c'est le cas pour Yenta.

Une autre caractéristique de notre système sera qu'il analysera la discussion en temps réel (*chat*) entre les usagers plutôt que de leurs emails et autres documents personnels, contrairement à Yenta. Analyser des documents privés poserait de graves problèmes de confidentialité dans un environnement industriel ou même universitaire. C'est donc un problème important que White Rabbit évitera en utilisant plutôt les discussions supervisées. L'usager sachant que ses propos seront observés, celui-ci pourra se contenter de ne dévoiler que l'information qu'il juge pertinente à ses recherches et pouvant être diffusées publiquement.

Pour la construction de White Rabbit, l'aspect de confidentialité sera fortement pris en compte. Les agents déclareront explicitement leur présence et manifesteront clairement leur intention d'écouter et d'analyser les conversations. De plus, la permission d'un usager sera requise pour que ses informations personnelles puissent être divulguées à une autre personne par son agent et l'usager pourra également garder secrète une partie de ses informations et ne divulguer que les informations qu'il désire partager.

L'analyse du "chat" par des agents est, comme nous l'avons vu, l'une des fonctions de Butterfly (voir section 1.2), toutefois le rôle des deux systèmes est différent. White Rabbit doit former lui-même des groupes (ou catégories) d'utilisateurs ayant des intérêts communs, comme le fait Yenta alors que Butterfly suggère aux utilisateurs des groupes qui sont déjà existants et qui ont été créés explicitement par ces mêmes utilisateurs. Ce dernier n'a donc pas de fonction de catégorisation à remplir.

Autre différence avec Butterfly, nous n'utiliserons pas IRC comme réseau de discussion. L'utilisation d'IRC comme médium de discussion pour White Rabbit a été une possibilité envisagée. Toutefois, comme son objectif est plutôt de construire des groupes à travers un réseau d'utilisateurs de taille réduite et puisqu'aucun mécanisme de contrôle de la conversation n'existe sur IRC, son utilisation nous aurait obligée à nous pencher sur des problèmes autres que ceux que le projet essaie d'aborder. Nous construirons donc plutôt un système de discussion de taille réduite qui possédera les capacités de contrôle de la conversation et l'organisation dont nous avons besoin. Ceci nous permettra du même coup d'éviter le problème des canaux cachés d'IRC.

Pour ce qui est maintenant de la catégorisation, nous avons choisi une approche très différente de celle des systèmes de jumelage étudiés précédemment. En effet, notre algorithme de catégorisation sera un type réseau de neurones déjà bien connu et très utilisé à cette fin. Les détails de cet algorithme seront fournis à la section portant sur la catégorisation (section 2.2).

1.6 Conclusion

Ce chapitre a présenté plusieurs systèmes de jumelage. En analysant ces systèmes, nous avons pu déterminer plusieurs caractéristiques que possédera notre logiciel. Il faut maintenant se pencher sur les techniques utilisées pour acquérir l'informations sur les usagers et réaliser le jumelage. Ce sera l'objet du prochain chapitre qui présentera les techniques d'explicitation de connaissances et de catégorisation, deux étapes importantes des systèmes de jumelage.

Chapitre 2

Construction et catégorisation des profils

Deux étapes sont essentielles à la réalisation du jumelage : extraire des profils représentatifs des intérêts des personnes ou des entités à jumeler, puis catégoriser ces profils. La qualité du jumelage dépendra donc de la qualité des méthodes utilisées pour réaliser ces deux étapes. Pour construire les profils, nous utiliserons des méthodes d'explicitation (ou extraction) de connaissances. Pour faire la catégorisation, bon nombre d'algorithmes existent. Nous exposerons ici ces problématiques et nous mettrons en relief les techniques qui pourront le mieux nous aider à atteindre notre objectif.

2.1 L'explicitation des connaissances

Les techniques d'explicitation des connaissances (en anglais, *knowledge elicitation*) sont des méthodes servant à extraire de l'information des experts, i.e., à découvrir ce qu'ils connaissent et à les formaliser de manière à pouvoir être compréhensibles par les humains et les machines. L'explicitation peut être faite par des techniques manuelles, semi-automatiques (interactives) ou automatiques (apprentissage machine) [1].

Les techniques manuelles et semi-automatiques d'explicitation de connaissances exigent une participation active de l'expert et consistent en général à le questionner ou à observer son comportement. En voici quelques exemples [1] :

L'interview consiste à poser directement des questions à l'expert pour obtenir des informations sur ses connaissances.

L'analyse de protocole consiste à enregistrer les propos de l'expert qui est invité à penser tout haut pendant la résolution d'un problème réel, puis à analyser cet enregistrement.

L'observation directe consiste à faire un enregistrement vidéo de l'expert dans son environnement de travail réel.

Les grilles-répertoires consistent à répertorier les connaissances de l'expert puis de l'inviter à donner une évaluation numérique de l'importance accordée à celles-ci dans les diverses expériences qu'il a mené.

Le "laddering" consiste à pousser l'expert à découvrir des liens entre ses connaissances et à les préciser. On lui demande par exemple de fournir des exemples, des synonymes, d'organiser ses connaissances hiérarchiquement, etc.

Les techniques automatiques d'explicitation des connaissances sont quant à elles des techniques permettant de découvrir, sans l'intervention de l'expert, des connaissances, des liens, des similarités, des poids, des règles, etc. à travers l'analyse des informations sur l'expert et de son comportement. Il s'agit de méthodes d'apprentissage machine de l'intelligence artificielle.

2.1.1 La construction des profils

Pour notre système, la méthode des grille-répertoires servira à construire pour chaque chercheur un profil de base, à partir duquel un profil plus complet pourra

ensuite être établit. Ceci contribuera à réduire le temps requis pour bâtir un profil d'intérêts représentatif d'une manière automatique.

Une fois les profil d'intérêts de base construits, les agents personnels des utilisateurs se chargeront de les enrichir automatiquement en analysant les messages que ceux-ci soumettront lors de la discussion. La méthode utilisée sera semblable à celle utilisée par Yenta et décrite dans [13]. Il s'agira donc de comptabiliser à l'intérieur des messages les occurrences de mots clés ou de mots similaires à des mots clés faisant partie de la base de connaissance du système. Mais la ressemblance avec la méthode de Yenta s'arrêtera ici. En effet, nous irons plus loin en utilisant également un graphe sémantique[1] spécifiant des liens sémantiques entre les différents mots clés de la base. De plus, nous utiliserons une fonction non-linéaire pour la représentation des intérêts, plutôt que des constantes fixes comme c'est le cas dans Butterly [9] ou qu'une fonction linéaire, comme pour Yenta [13]. Ceci devrait selon nous fournir une image plus réaliste des divers degrés d'intérêts des utilisateurs.

2.2 La Catégorisation

La tâche principale à réaliser pour l'atteinte de nos objectifs consiste à regrouper des profils d'utilisateurs ayant des intérêts compatibles, et implique la classification de ces profils en catégories. Le problème de la catégorisation débute avec un ensemble d'objets non classés, et nécessite un moyen de mesurer la similarité entre ceux-ci. Son but est de regrouper les objets, donc de les classifier, de sorte à maximiser un quelconque standard de qualité [17]. Plusieurs algorithmes classiques existent pour résoudre ce problème. La *taxonomie numérique*, l'un des plus anciens algorithmes de catégorisation, traite chaque objet à classer comme un vecteur de nombres, et la similarité entre deux objets est alors déterminée par la distance euclidienne entre ces deux

[1] nous nous référerons à ce graphe sémantique par l'expression "graphe de connaissances" dans le reste de ce document.

points. Une seconde méthode basée sur la similarité, souvent appelée *catégorisation par agglomération*, construit les catégories de manière récursive. Mais le problème des stratégies basées sur la similarité est qu'elles sont incapables de capturer la signification *sémantique* d'une catégorie. Ces méthodes traditionnelles ne peuvent tout simplement pas tenir compte des connaissances sur le domaine pour fournir une explication sémantique aux classes obtenues. On dit dans ce cas que la définition obtenue est "extentionnelle", i.e. qu'elle n'est que l'énumération des membres de la classe. Or, l'idéal est d'obtenir une définition "intentionnelle" des classes, i.e. des règles générales qui peuvent ensuite être utilisées pour classifier à la fois les membres déjà connus, et les membres futurs de la catégories. Par exemple, l'ensemble {Clinton, Bush, Reagan, Kennedy, ...} est une définition extentionnelle d'une classe, alors que l'ensemble $\{x : x$ est un président des États-Unis$\}$ est une définition intentionnelle de la même classe.

Suite à nos lectures, nous avons donc retenu deux méthodes de catégorisations très documentées dans la littératures, et ayant déjà fait dans l'histoire de l'informatique maintes fois la preuve de leur efficacité. Il s'agit d'ID3 et des Kohonen Maps qui seront maintenant décrites. Nous expliquerons ensuite pourquoi notre choix s'est arrêté sur le deuxième algorithme.

2.2.1 ID3

ID3 est un algorithme d'arbre décisionnel qui induit des concepts à partir d'exemples, et produisant un outil de classification "intentionnelle" [23]. ID3 permet de déterminer la classification d'un objet, suivant la valeur qu'il possède pour un ensemble de propriétés. Il pourrait, par exemple, classifier des profils financiers parmi l'une de plusieurs catégories (Haut Risque, Risque Modéré, etc.) selon un ensemble de critères (Dettes, Revenus, etc.) pour aussi peu qu'on lui fournisse un nombre suffisant d'exemples.

L'algorithme construit un arbre de décision à partir des exemples fournis en évaluant, pour chacun des critère, la quantité d'information qu'il procure pour la classification.

L'évaluation du gain d'information pour chacun des critères est basé sur une théorie mathématique appelée la théorie de l'Information, développée par Shannon en 1948 [25] et encore aujourd'hui grandement utilisée en informatique et en télécommunications. Cette théorie permet de mesurer la quantité d'information contenue dans un message (équation (2.1)). De cette façon, l'ordre d'importance des différents critères est déterminé et l'arbre de décision est construit de manière récursive selon cet ordre. Ainsi, les critères fournissant le plus d'informations sont choisis comme les premiers critères de l'arbre (partant de la racine).

$$I(M) = \sum_{i=1}^{n}(-p(m_i)\log_2(p(m_i))) \tag{2.1}$$

où $M = \{m_1, m_2, m_3, \ldots, m_n\}$ est l'ensemble de tous les messages possibles et $p(m_i)$ est la probabilité d'occurrence de chacun des messages. Cette équation est le résultat de la formalisation d'une intuition. En effet, si l'on prend par exemple deux sous dont l'un est truqué et tombe pile 3 fois sur 4, alors un message nous prédisant le résultat est plus intéressant dans le premier cas que dans le deuxième, et donc le premier cas contient plus d'information que le second. Ainsi, en utilisant l'équation (2.1), on obtient

$$I(sousnormal) = -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1$$

$$I(soustruque) = -\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) = 0.811$$

Ensuite, on peut calculer l'espérance de la quantité d'information qui serait requise pour compléter l'arbre si on utilisait chacun des critères comme racine de l'arbre. Cette mesure est calculée à l'aide de l'équation (2.2).

$$E(P) = \sum_{i=1}^{n} \left(\left(\frac{|C_i|}{|C|} \right) I(C_i) \right) \tag{2.2}$$

où P est une propriété (ou critère), C est un ensemble d'exemples d'entraînement divisé en sous-ensembles $\{C_1, C_2, \ldots, C_n\}$ correspondants aux exemples possédant la i^e valeur pour la propriété P (n étant le nombre de valeurs possibles pour cette propriété).

Finalement, à partir de la quantité d'information fournie par l'ensemble d'entraînement et de l'espérance de la quantité d'information requise pour compléter l'arbre selon chacun des critères, on peut calculer le gain d'information obtenu pour chacun des critères par l'équation (2.3).

$$gain(P) = I(C) - E(P) \tag{2.3}$$

En calculant le gain d'information pour chaque critère, on peut alors choisir le critère donnant le gain le plus élevé et en faire la racine de l'arbre, puis appliquer de manière récursive cet algorithme sur les sous-arbres produits par ce choix.

ID3 a été évalué dans des tests contrôlés ainsi que dans de nombreuses applications et a ainsi maintes fois prouvé son efficacité en pratique [17]. Globalement, les tests effectués ont révélé une importante amélioration de la catégorisation effectuée par ID3, i.e. une diminution du taux d'erreur de classification, en fonction de l'augmentation du nombre d'exemples fournis au système.

2.2.2 Self-Organizing Maps

Cet algorithme, aussi appelé Kohonen Maps, fut inventé en 1982 par Teuvo Kohonen, professeur de l'université de Helsinki, en Finlande, et a depuis longtemps fait ses preuves en étant appliqué à de nombreux domaines comme la médecine [27], la

physique [4] et la séismologie [21]. Il s'agit d'un type de réseau de neurones utilisant un algorithme d'entraînement non supervisé, et qui, à travers un processus d'auto-organisation, dispose ses unités de sortie en une représentation topologique des données d'origines [8].

Contrairement aux réseaux de neurones habituels, utilisant la propagation arrière (backpropagation), les Self-Organizing Maps (ou SOMs, ou cartes auto-organisées) ne sont constituées que de deux couches d'unités (ou neurones) : les unités d'entrée et les unités de sortie. Les unités d'entrée forment un vecteur à n éléments auxquels seront affectées les valeurs des vecteurs de données d'entrée. Une donnée d'entrée est donc un vecteur de nombres réels et peut être représenté par

$$x(t) = x_1, x_2, \ldots, x_n$$

où t est l'indice du vecteur d'entrée (faisant partie d'un ensemble de vecteurs d'entrées). Les unités de sortie sont quant à elles disposées sur un plan et sont complètement reliées aux unités d'entrée, i.e. que chacune des unités de sortie est reliée à toutes les unités d'entrée. Chacun de ces liens est associé à un poids (un nombre réel) et donc, chaque unité de sortie possède un vecteur de dimension n pouvant être comparé au vecteur d'entrée. On peut définir une unité de sortie comme suit :

$$m_i(t) = (m_{i1}, m_{i2}, \ldots, m_{in})$$

où i est l'indice du neurone et t représente le temps, puisque les valeurs des poids contenus dans les vecteurs de sortie évoluent en fonction du temps. Initialement, les poids peuvent être déterminés de manière aléatoire, mais il est possible d'accélérer l'entraînement en choisissant des poids plus appropriés. La figure 2.1 donne une image conceptuelle du réseau obtenu.

L'algorithme des SOMs, comme les autres algorithmes de réseau de neurones, comprend deux phases. La première phase est l'entraînement du réseau qui lui permet

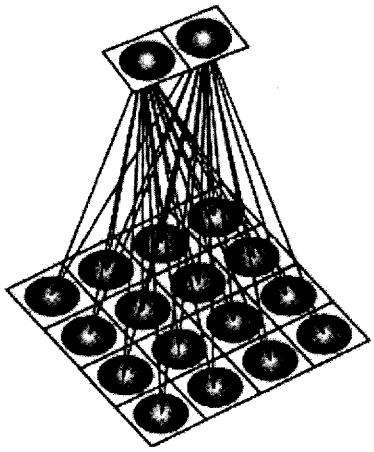


FIG. 2.1 – Carte de Kohonen

de former et de stabiliser sa carte, i.e. sa couche de sortie. Cette étape entraîne la mise à jours des poids contenus dans les unités de sortie et nécessite des vecteurs de données d'entrée. La seconde phase est l'utilisation du réseau pour catégoriser des vecteurs de données fournis en entrées, sans qu'il n'y ait de mise à jour des poids du réseau.

Dans la phase d'entraînement, des vecteurs d'entrée sont fournis un par un au réseau qui, pour chacun d'eux, réalise les étapes suivantes :

1) Une unité de sortie est déclarée vainqueur, celle dont le vecteur de poids est le plus similaire au vecteur d'entrée. Pour déterminer laquelle est la gagnante, on calcule la distance euclidienne entre chacun des vecteurs de poids et le vecteur d'entrée. Le vainqueur est celui dont la distance calculée est la plus courte. La distance euclidienne entre deux vecteurs est définie comme suit :

$$a = \sqrt{\sum_{i=0}^{n} (poids_i - entree_i)^2} \tag{2.4}$$

L'unité la plus près, notée $m_c(t)$ est donc identifiée selon la condition suivante :

$$|x(t) - m_c(t)| = \min\{|x(t) - m_i(t)|\}$$

2) Ensuite, les poids de l'unité gagnante et des unités voisines sont mis à jour selon la formule de récurrence suivante :

$$m_i(t+1) = m_i(t) + \alpha(t)\, h_{ci}(t)\, [x(t) - m_i(t)] \tag{2.5}$$

où α est le taux d'apprentissage et h est la fonction de voisinage, déterminant à quel point le neurone i et le neurone vainqueur c sont proches voisins. Pour que le réseau converge et se stabilise, i.e. que les poids de ses unités de sorties se stabilisent, le taux d'apprentissage doit être choisi entre 0 et 1 exclusivement et diminuer avec le temps (diminuant ainsi l'ampleur de la mise à jour avec le temps) et de même, la fonction de voisinage h doit également diminuer avec le temps. Cette dernière est définie par

$$h_{ij} = exp\left[\frac{-|r_i - r_j|^2}{2\sigma(t)^2}\right] \tag{2.6}$$

où i et j sont les indices des neurones comparés, r_i et r_j sont les position des neurones i et j sur le plan de la couche de sortie et $\sigma(t)$ est une fonction qui décroît de façon linéaire en fonction du temps. Il s'agit d'une fonction de forme Gaussienne qui, on peut le remarquer, diminue lorsque la distance entre les neurones i et j sur le plan augmente. Ceci implique donc de garder pour chaque neurone de sortie, en plus du vecteur de poids, sa position en deux dimensions, sur le plan.

Ce processus d'entraînement est répété jusqu'à ce qu'il y ait convergence (stabilisation) des vecteurs de poids. Une fois cette convergence obtenue, si on associe chacun des neurone de sortie à une catégorie, le réseau peut alors être utilisé pour catégoriser des vecteurs de données fournis en entrée en déterminant simplement quel est le neurone vainqueur. Conséquemment, le réseau entraîné constitue une définition intentionnelle des catégories représentées par les unités de sortie, puisqu'il permet de catégoriser des profils nouveaux, encore jamais rencontrés par le réseau. Le nombre

d'unités de sortie doit être choisi selon les besoins. Il doit simplement correspondre au nombre de catégories qui doivent être créées. On peut dès lors voir facilement l'application de cet algorithme à la catégorisation de profils d'usagers lorsque ceux-ci sont codés sous la forme vecteurs de nombres réels.

2.2.3 Choix de l'algorithme

Nous avons choisi les Kohonen Maps pour réaliser la catégorisation dans le système White Rabbit, et ce pour des raisons simples. ID3 fait un apprentissage supervisé, i.e. qu'il requiert qu'on lui fournisse des exemples de classification correcte des usagers selon leurs intérêts. Ceci implique qu'il soit possible de déterminer une classification correcte de profils d'usager. Or, le problème que tente de résoudre White Rabbit est précisément de déterminer une telle classification pour des profils, pour laquelle nous n'avons à prime abord aucun indice, aucun critère nous permettant de qualifier une classification comme étant bonne ou mauvaise. Par conséquent, aucun exemple ne peut être fourni au système et donc, le choix des Kohonen Maps s'est imposé de lui-même. En effet, grâce à son apprentissage non supervisé, cette technique permettra à White Rabbit de catégoriser des profils, sans que des exemples de "bons résultats" n'aient à lui être fournis.

De plus, les Kohonen Maps accomplissent exactement le travail requis, i.e. qu'ils prennent en entrée des données à n dimensions, et les font correspondre à des vecteurs de données à deux dimensions. On appelle les méthodes de réduction de la dimension de vecteurs *méthodes de projection*[8]. Or, nos profils d'utilisateurs seront précisément des vecteur à n dimensions, où n est le nombre de mots clés se trouvant dans la base de connaissances. Les Kohonen Maps nous permettent donc d'associer un profil à une catégorie se trouvant sur une carte à deux dimensions.

2.3 Conclusion

Les profils des usagers seront donc formés d'un ensemble de mots clés, chacun d'eux associé à un poids indiquant le degré d'intérêt de l'usager pour le concept représenté par le mot clé. Ces profils pourront alors être catégorisés à l'aide d'un réseau de neurones non-supervisé de type SOM, capable de diviser un ensemble de vecteurs de même dimensions en un nombre fini de catégories.

Ceci termine la présentation des concepts théoriques sous-jacents à notre solution. Nous pouvons maintenant exposer les détails de l'implantation du système que nous avons réalisé suite à cette recherche.

Chapitre 3

White Rabbit

Le projet a mené au développement d'un système d'agents intelligents d'analyse de discussion [31]. Ce chapitre fait état de l'architecture de ce prototype et du fonctionnement de ses différentes composantes. Nous avons entièrement conçu et réalisé tous les modules décrit dans ce chapitre. De plus, ce projet a été réalisé conjointement avec le projet ALICE développé dans le cadre d'un autre projet de maîtrise [2]. ALICE est un éditeur permettant de modéliser facilement et graphiquement une entreprise sous la forme de différentes entités (Départements, Employés, Projets, Publications, etc.) et de construire une base de connaissances qui permet à White Rabbit de faire une analyse poussée et efficace des intérêts de ses utilisateurs. La représentation des connaissances ainsi développée a été nommée PC^2 (pour Publication, Chercheur, Projet et Connaissance, les entités centrales de la représentation). La section 3.9 est consacrée à la présentation d'ALICE.

3.1 Architecture du système

Comme nous l'avons mentionné à la section 1.3, l'architecture de White Rabbit (figure 3.1) est comparable à celle d'Arcadia. Les agents du systèmes, responsables de l'analyse et de la classification des usagers forment trois couches. Il y a un agent

médiateur, se trouvant sur la même machine que le serveur de discussion et constituant la première couche. Celui-ci possède une base de profils de toutes les entités modélisées, une base de connaissances sur laquelle sont basés ces profils, un réseau de neurones de type Self-Organizing Maps (présenté à la section 2.2.2) pour réaliser la catégorisation des profils et finalement une base d'adresses, constituée des adresses IP des machines de tous les utilisateurs connectés au serveur de discussion. La seconde couche est formée d'agents de communication, basée sur Java RMI. Ces agents font le lien entre les usagers et l'agent médiateur en faisant le transport du profil et des résultats de la catégorisation. Ce sont également ces agents qui sont responsables des présentations entre les usagers, i.e. qu'ils doivent communiquer avec les autres agents homologues du système pour quérir les informations personnelles d'utilisateurs ayant des profils similaires, tout en respectant les contraintes de confidentialité posées par chacun d'eux. Finalement, des agents personnels, répartis sur les machines des utilisateurs, font le filtrage des messages envoyés par leur usager et mettent à jour leur profil. L'agent personnel doit aussi interagir avec l'usager qu'il représente, en lui posant des questions afin de préciser ses intérêts tout au long de la discussion.

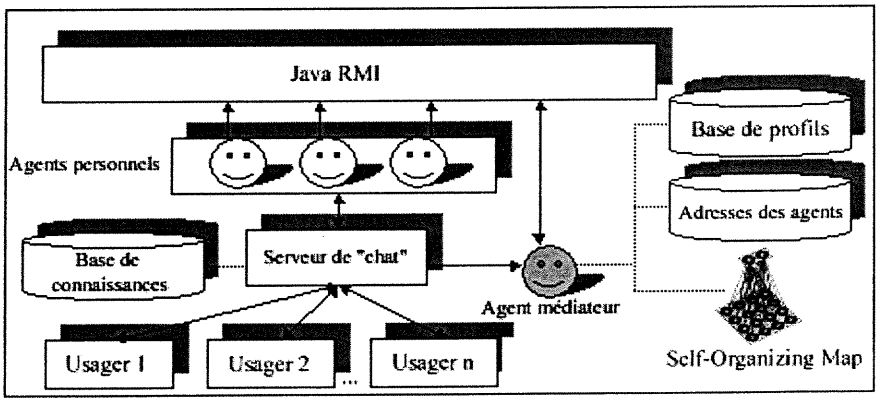


FIG. 3.1 – Architecture du système White Rabbit

Outre les agents, le système est composé de :

- un serveur de discussion ("chat server") qui organise le flux de messages des utilisateurs à travers le réseau ;
- une interface dédiée au client lui permettant d'envoyer et de recevoir des messages ;
- une interface dédiée à l'administrateur permettant à ce dernier de gérer les utilisateurs et leurs suggestions quant à l'accroissement de la base de connaissance. Elle permet de plus la visualisation des résultats de la catégorisation par le réseau de neurones et de modifier les paramètres de ce dernier ;
- d'une couche Java RMI nécessaire à la communication entre les agents. Cette couche est expliquée à la section 3.8.

La figure 3.2 montre l'architecture d'un agent personnel dans son environnement. Dans ce schéma, les agents de communication forment le module de communication, faisant le lien entre un agent personnel et l'agent médiateur. Les différentes parties de cette figure seront maintenant expliquées en détails dans les sections suivantes.

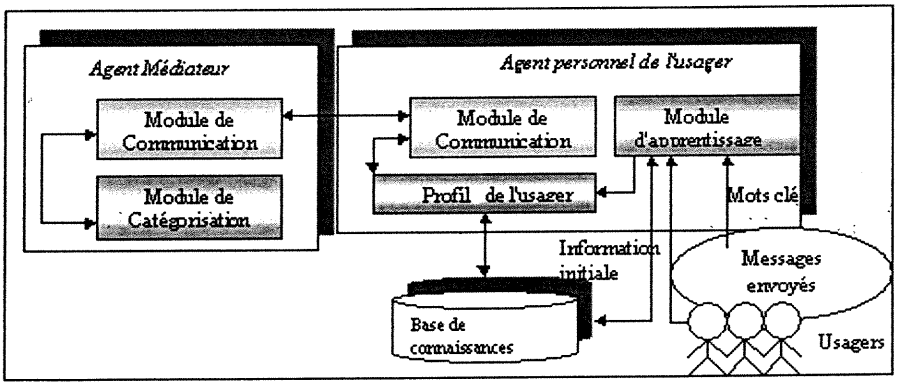


FIG. 3.2 – Architecture d'un agent personnel

3.2 Le profil de l'usager

Tout le système fonctionne autour de cette composante essentielle. Jusqu'ici, nous avons parlé du profil de l'usager seulement en termes d'intérêts. Cependant, pour qu'un tel système soit réellement utile et profitable pour une entreprise, il ne s'agit pas seulement de jumeler les personnes ayant des intérêts communs, mais bien de jumeler les personnes ayant des besoins particuliers avec d'autres personnes possédant les connaissances pouvant y répondre. Ainsi, le profil de l'usager tel que nous l'avons implanté est divisé en deux parties : l'une contenant toute l'information pertinente sur les intérêts (besoins) et l'autre sur les capacités (compétences) de l'usager, dans le domaine choisi. Cette information permet aux agents de découvrir les similarités et complémentarités et donc de faire une catégorisation appropriée des usagers. Le profil d'intérêts représente les besoins de l'usager alors que le profil de capacités représente son expertise.

Il importe donc que l'agent trace à travers le profil un portrait le plus fidèle possible de ces deux facettes de l'usager qu'il représente. Notez que White Rabbit ne fait qu'extraire un profil d'**intérêts**, et non pas un profil de capacités. En effet, nous supposons que les personnes qui discutent en utilisant le système cherchent des gens pouvant les aider. C'est ALICE qui est chargé de construire les profils de capacités des employés, et ce, à partir de leurs rapports, publications, curriculum vitae, etc. ALICE procure également à White Rabbit une base de profil d'intérêts pour les usagers à partir de la description des projets auxquels ils participent au moment actuel (figure 3.3).

La différenciation entre intérêts et capacités est selon nous l'une des principales forces de notre système, et à la fois l'une des principales lacunes des systèmes de jumelage qui ne font souvent que jumeler les personnes aux intérêts semblables, comme Yenta par exemple [13], alors qu'il est généralement certainement plus profitable pour

ces personnes de découvrir des personnes répondant à leurs besoins, des experts, des personnes complémentaires.

Comme nous l'avons déjà mentionné, l'approche que nous avons adoptée pour la construction des profils est basée sur la représentation de connaissances PC^2 développée dans le cadre du projet ALICE. Le modèle PC^2 consiste à former un graphe de connaissances (des mots clés) auquel sont reliés les usagers via les publications et les projets auxquels ils participent. Les connaissances sont reliées les unes aux autres par des liens portant une signification sémantique. Par exemple, des connaissances peuvent être liées entre elles par des liens de similarité, ou encore de spécialisation/spécialisation.

Ainsi, tout le domaine de connaissances choisi est représenté par un graphe. Le profil construit pour chacun des usagers est donc un ensemble de liens aux différentes connaissances de ce graphe avec pour chaque lien un poids associé. Ces poids varient de 0 à 1 et représentent le degré d'intérêt (ou de compétence) de l'usager pour chacune des connaissances de la base, 1 signifiant un intérêt très élevé et 0 un intérêt nul. Au cours de la discussion et de son analyse, ces poids sont constamment modifiés et mis à jour par le module d'apprentissage. Ce sera au moment de faire cet apprentissage que les bénéfices de l'utilisation du modèle PC^2 pour la représentation des connaissances deviendront évidents (voir la section 3.3 sur le module d'apprentissage).

En plus de contenir un vecteur de poids, le profil contient deux choses. Tout d'abord, un degré de certitude pour chacun des poids, reflétant le niveau de certitude que possède l'agent personnel pour ces poids. La certitude d'un agent personnel pour un poids augmente lorsqu'il pose directement des questions à l'usager à propos de son intérêts sur la connaissance correspondante et que celui-ci lui donne une réponse. Par ailleurs, cette certitude diminue lorsqu'un usager se contredit au cours de la discussion, ce qui entraînera plus de questions de la part de l'agent concernant cette connaissance. Donc, plus la certitude de l'agent est élevée pour un poids donné, moins

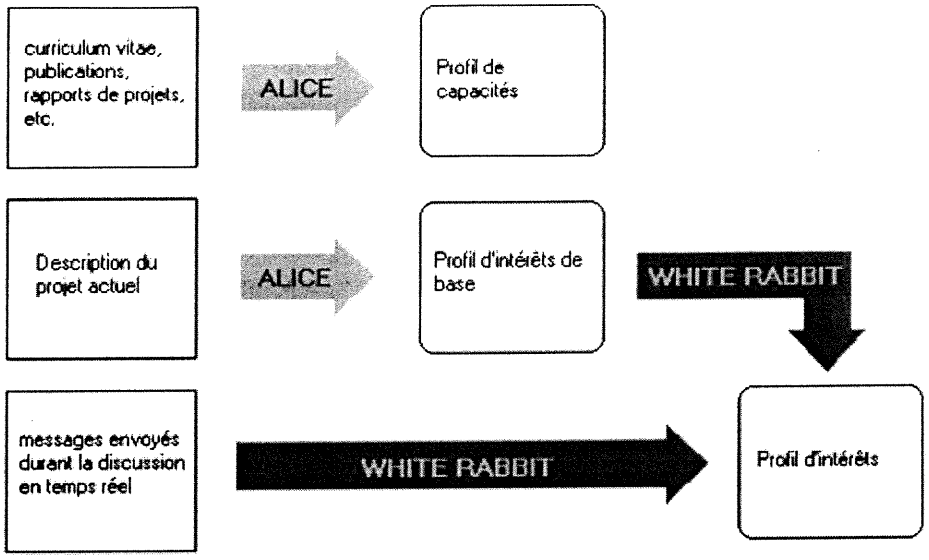


FIG. 3.3 – La construction des deux composantes du profil de l'usager

il posera de questions à l'usager concernant la connaissance correspondante, et vice versa. Ce mécanisme permettant à l'agent de vérifier et de corriger le profil est décrit à la prochaine section 3.3.

Finalement, le profil contient les informations personnelles de l'usager, celles qui seront utilisées pour faire les présentations par les agents. Il s'agit des nom, adresse de courrier électronique, numéro de téléphone, photo, département, position dans l'entreprise, description de l'expertise (curriculum vitae), liste des publications, et description du projet en cours. De plus, chacun de ces éléments d'information est accompagné d'un marqueur de confidentialité, indiquant si cet élément peut être divulgué ou non par son agent personnel à d'autres personnes. L'usager peut donc lui-même marquer comme confidentielles les informations qu'il désire garder secrètes.

3.3 Le module d'apprentissage

Comme nous l'avons mentionné à la section précédente, le module d'apprentissage est celui qui modifie les poids du profil de l'usager pour le rendre plus conforme à la représentation des intérêts de ce dernier. Ce processus se fait en trois étapes. La première est une acquisition préliminaire d'information sur l'usager par l'intermédiaire d'un questionnaire, une forme de grille-répertoire (voir section 2.1). La première fois qu'il utilise le système, l'usager est invité par ce questionnaire à lui fournir des mots clés reflétant ses intérêts de recherches. Cette information est utilisée par White Rabbit pour augmenter le profil d'intérêts de base pour l'usager généré avec ALICE et est stockée dans la base de profils de l'agent médiateur.

La seconde étape est celle de l'analyse de la discussion (algorithmes 1 et 2). Celle-ci consiste à extraire les mots clés du domaine des messages envoyés par l'usager et ensuite de mettre à jour le profil d'intérêts en augmentant le poids des connaissances associées en suivant une fonction sigmoïde (équation (3.1))

$$f(x) = \frac{1}{1+e^{-\lambda x}} \tag{3.1}$$

où $f(x)$ est le nouveau poids, x est "l'importance" du mot clé et λ est un paramètre réglant la pente de la fonction, et donc la vitesse avec laquelle elle converge vers 1 suivant l'augmentation de x. L'importance, comme nous l'avons définie, est le nombre d'occurrences du mot clé modifié selon certains facteurs comme le degré de similarité entre les connaissances, les déclarations d'intérêts de l'utilisateur, etc. Inversement, on peut donc trouver l'importance d'un mot clé à partir de son poids par l'équation (3.2) obtenue en isolant x dans l'équation (3.1).

$$x = \frac{\ln\left(\frac{1}{f(x)} - 1\right)}{-\lambda} \tag{3.2}$$

La fonction sigmoïde est fréquemment utilisée dans les réseaux de neurones pour la mise à jour des poids des liens entre les unités. Elle possède la propriété de varier strictement entre 0 et 1 et sous une forme donnant une augmentation lente au départ, puis rapide après quelques occurrences, et lente à nouveau pour les valeurs élevées de x. La figure 3.4 montre l'allure de cette fonction avec $\lambda = 1$.

Algorithme 1 Analyser (*message*)

Require: $graphe = motsCles \cup liens$

Ensure: $graphe.noeudsVisites = \emptyset$

for all $mot \in message$ **do**

if $mot \in \{motsCles\}$ **then**

$MettreAJourPoids(mot, graphe, 1)$

else if $similaire(mot, mot2) \wedge mot2 \in \{motsCles\}$ **then**

$increment \leftarrow maxSousChaine(mot, mot2)/mot.longueur$

$MettreAJourPoids(mot2, graphe, increment)$ $\{increment < 1\}$

else

ignorer le mot

end if

end for

De plus, grâce à la représentation des connaissances sous forme PC^2, il devient possible de mettre à jour les poids de toutes les connaissances reliées au mot clé découvert en suivant des heuristiques simples (algorithme 2). Par exemple, toutes les connaissances similaires à celle trouvée voient leur poids modifié dans la même direction et suivant la même fonction, de façon proportionnelle au degré de similarité inscrit dans les liens du graphe de connaissances. Conséquemment, suite à la découverte de quelques mots clés seulement, le module d'apprentissage de l'agent ar-

Algorithme 2 MettreAJourPoids (*mot*, *G*, *increment*)

$G.noeudsVisites \leftarrow G.noeudsVisites \cup mot$

$importance \leftarrow reverseSigmoide(mot.poids)$ (équation 3.2)

$importance \leftarrow importance + increment$

$nouveauPoids \leftarrow sigmoide(importance)$ (équation 3.1)

$mot.poids \leftarrow nouveauPoids$ {changement du poids du mot dans le profil de l'usager}

for all $motSuivant \in \{mot.connaissancesUtilisees\} \wedge \notin \{G.noeudsVisites\}$ **do**

$MettreAJourPoids(motSuivant, 0.8 * increment)$

end for

for all $motSuivant \in \{mot.connaissancesSimilaires\} \wedge \notin \{G.noeudsVisites\}$ **do**

$nouvelIncrement \leftarrow niveauSimilarite/3.2$ {$niveauSimilarite$ est une donnée contenue dans le lien de similarité entre deux connaissances et est inclue entre 1 et 3}

$MettreAJourPoids(motSuivant, nouvelIncrement)$

end for

rive à mettre à jour la quasi totalité des poids des connaissances du profil d'intérêts. Comme nous le verrons plus loin, ceci est très important pour le bon fonctionnement de l'algorithme de catégorisation que nous utilisons : les Self-Organizing Maps [8].

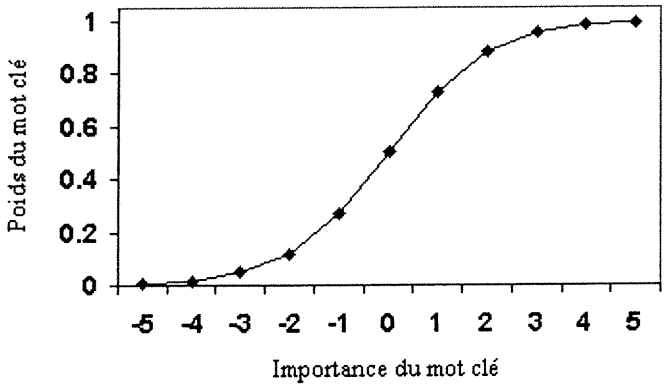


FIG. 3.4 – La fonction sigmoïde

Durant la troisième phase de l'apprentissage, l'agent personnel effectue occasionnellement une vérification du profil pour permettre à l'usager de confirmer ou d'infirmer un ou plusieurs intérêts du profil, afin d'augmenter son niveau de certitude concernant ceux-ci. Lorsque la certitude de l'agent pour un poids du profil est faible, l'agent aura une forte tendance à questionner l'usager afin de savoir s'il s'intéresse ou non à la connaissance associée (figure 3.5).

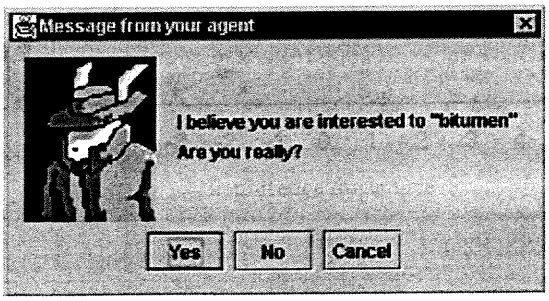


FIG. 3.5 – Interrogation de l'usager par son agent personnel

Si la réponse de l'usager est "Yes" ou "No", alors le poids du mot clé est ajusté en conséquence et la certitude de l'agent pour ce poids est augmenté. Si la certitude de l'agent pour un poids dépasse un certain seuil déterminé par l'administrateur, alors il ne questionne plus l'usager pour cette connaissance. Toutefois, si l'usager contredit ce qu'il avait préalablement affirmé en déclarant explicitement ne pas être intéressé à une connaissance pour laquelle il avait montré de l'intérêt par le passé (i.e. dont le poids est élevé), alors le poids de celle-ci est diminué, et la certitude de l'agent l'est également. Ce mécanisme permet à l'agent de corriger dynamiquement des erreurs pouvant s'être introduites dans le profil, et de bien représenter les changements d'intérêts des usagers dans le temps.

De plus, nous avons introduit un mécanisme de "vieillissement" des profils. En d'autres termes, la certitude de l'agent pour chacun des poids de son profil d'intérêt diminue de manière lente et constante de sorte à tenir compte du fait que les intérêts de la personne représentée puisse changer avec le temps. Ainsi, après un certain temps, l'agent peut poser à nouveau des questions sur un concept afin de s'assurer que son utilisateur est toujours en accord avec le poids qui lui est accordé. La vitesse du vieillissement peut être réglée par l'administrateur.

Bref, la seconde étape d'apprentissage est faite de manière totalement automatique et la troisième ne demande qu'une faible participation de l'usager qui n'a qu'à discuter avec ses confrères et répondre aux questions périodiques posées par son agent. Ceci constitue un avantage important sur le système Butterfly présenté à la section 1.2. En effet, ce dernier force l'usager à déclarer de manière explicite ses intérêts et les poids obtenus sont rigides, n'offrant aucune flexibilité au système pour s'adapter et corriger ses erreurs. C'est l'usager qui doit lui-même corriger les erreurs du système en modifiant explicitement son profil en cours de discussion. L'interface de discussion de White Rabbit permet tout de même à l'usager d'accélérer l'apprentissage du système en faisant de telles déclarations explicites, mais il n'est nullement dans l'obligation

de le faire pour que son profil soit mis à jour. De la même façon, la première étape de l'apprentissage par l'intermédiaire d'un questionnaire n'a pour but que de faciliter la tâche de l'agent pour parvenir plus rapidement à la construction d'un profil représentatif et l'usager peut très bien choisir de n'offrir aucune information initiale et ainsi de laisser à White Rabbit tout le travail de déterminer ses intérêts en cours de discussion.

3.4 Le module de catégorisation

Comme mentionné plus haut, l'algorithme de catégorisation que nous utilisons est celui des Self-Organizing Maps [8] et constitue un aspect clé de notre système. Dans notre application, le réseau de neurones se trouve dans l'agent médiateur. C'est donc lui qui reçoit les profils des usagers produits par les agents personnels et qui a pour rôle de les classer dans une catégorie.

Tout d'abord, comme nous l'avons préalablement expliqué (section 2.2.2), le réseau de neurones doit être entraîné, afin qu'il converge, i.e. que les valeurs de ses poids se stabilisent, et qu'il produise, par le fait même, une définition intentionnelle des catégories qui seront ensuite attribuées. Cet entraînement est fait à partir de l'ensemble des profils de capacités des employés, contenus dans la base de données et extraits par ALICE (Notez que les profils de capacités sont utilisés pour l'entraînement, puisque notre but est ici de trouver les employés possédant des compétences similaires au besoins, ou intérêts, des personnes qui discutent). Ceci implique que les profils soient fournis comme valeurs d'entrée pour la couche d'entrée du réseau de neurone, ce qui est fait de manière triviale, puisqu'un profil est en fait constitué d'une liste de poids (des nombres réels entre 0 et 1), correspondant à chaque mot clé de la base de connaissance. Par conséquent, convertir un profil d'usager en donnée d'entrée du SOM signifie simplement attribuer de manière ordonnée les poids du profil aux unités de la couche d'entrée. Par "manière ordonnée", nous entendons que les unités d'entrée

du réseau correspondent aux mots clés ordonnés de la base de connaissance, et donc que les poids doivent être attribués selon cet ordre, sans quoi le i^e poids d'un profil ne correspondra pas nécessairement au même mot clé que le i^e poids d'un autre profil.

À travers l'interface qui lui est fournie par le système (voir section 3.7), l'administrateur du système peut gérer les paramètres importants du réseau de neurones : le taux d'apprentissage, le nombre de passes d'entraînement à travers l'ensemble de profils, et le nombre de neurones dans la couche de sortie, définissant le nombre de catégories voulues.

Ensuite, l'agent médiateur reçoit périodiquement les profils d'intérêts des usagers connectés au serveur de discussion. À chaque fois qu'un profil d'intérêts est ainsi reçu, il est traduit en donnée d'entrée pour le réseau de neurones, de la même façon que les profils de capacités utilisés pour l'entraînement. Cette fois-ci par contre, les poids des liens dans le réseau ne seront pas modifiés. Un neurone vainqueur de la couche de sortie sera simplement déterminé (tel que décrit dans la section 2.2.2) et le nom de ce dernier (ici, le numéro du neurone) devient le nom de la catégorie du profil d'intérêts reçu. Ce dernier se retrouve ainsi dans l'une des catégorie, où se trouve potentiellement quelques-uns des profils de capacités des employés. Si c'est le cas, les noms de ces employés sont renvoyés à l'usager connecté, en plus du nom de la catégorie dans laquelle il a été classifié. Ces informations sont affichées sur son interface de discussion (voir l'interface de discussion, section 3.6).

Donc, à partir de ces nouvelles informations, l'usager est d'abord en mesure de consulter la liste des employés ayant un profil complémentaire au sien, puis d'en sélectionner un à l'aide d'une liste de choix ("*combo box*") et finalement de demander à en savoir plus sur cette personne en cliquant simplement sur le bouton adjacent à la liste. Le processus de présentation est décrit dans la section 3.5 sur le module de communication.

On peut donc voir comment le système réduit l'espace de recherche pour un usager désirant contacter des personnes possédant des compétences pouvant répondre à ses besoins. Et il est facile d'imaginer l'économie de temps et d'énergie que cela représente si la compagnie modélisée possède plusieurs centaines, voire plusieurs milliers d'employés.

3.5 Le module de communication

Le module de communication est, comme nous l'avons décrit dans l'architecture, formé d'agents de communication faisant la passerelle entre les agents personnels et l'agent médiateur en plus d'effectuer les présentations entre les usagers. Cette couche d'agent repose entièrement sur la technologie de Java RMI, brièvement décrite à la section 3.8.

Deux aspects du système White Rabbit démontrent l'autonomie et la mobilité[1] de ses agents. Le premier est l'étape de catégorisation du profil d'un client, réalisée par l'agent médiateur. En effet, après avoir analysé et mis à jour le profil de son client, l'agent personnel doit l'envoyer vers le médiateur pour qu'il le catégorise. L'agent personnel confie donc le profil de son usager à un agent de communication qui transmet le profil à l'agent médiateur à travers le réseau. La localisation (adresse IP) de ce dernier, situé sur la même machine que le serveur de discussion, est connu de tous les agents de communication du système au moment où leur client établit une connexion au serveur. L'agent médiateur met alors sa base de profils à jour et attribue au profil de l'usager un numéro de catégorie (voir section 3.5, le module

[1] la mobilité de nos agents n'est que conceptuelle en ce sens que l'agent ne se transporte pas pour s'exécuter sur une autre machine (voir section 3.8). Il n'y a en fait que les profils qui sont effectivement sérialisés et transmis d'une machine à l'autre. Par conséquent, les personnes qui considèrent la mobilité comme étant une condition sine qua non à l'appellation "agent" d'un programme peuvent s'objecter à notre nomenclature. Ceci dit, le débat peut continuer...

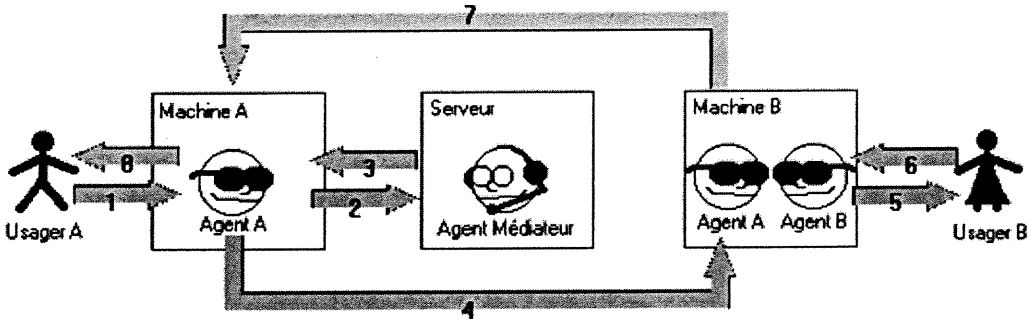
de catégorisation). Puis, il retourne à l'agent de communication les résultats de la catégorisation. Finalement, ce dernier retransmet à l'agent personnel de l'usager les résultats obtenus.

La seconde manifestation de mobilité se produit lorsque l'usager demande qu'on lui présente un second usager faisant partie de la même catégorie, telle que déterminée par l'agent médiateur (figure 3.6). À cet instant, l'agent du client (A) faisant la requête entre en contact avec l'agent (B) associé au client à présenter. Ainsi l'agent (A) est en mesure de demander de plus amples informations à l'agent (B) pour ensuite les dévoiler à son client ayant fait la requête. Pour ce faire, l'agent (B) demande à l'usager (B) s'il accepte d'être présenté à l'usager (A). Si celui-ci accepte, alors l'agent (B) confie à l'agent (A) les informations sur son client, puis l'agent (A) présente à son usager les informations obtenues, tout en respectant les contraintes de confidentialité (figure 3.7) posées par l'usager (B). Cependant, si le client (B) refuse d'être présenté, alors l'agent (A) reste bredouille et indique à son client que la présentation a été refusée.

Cette deuxième situation permet de voir l'avantage principal apporté par l'architecture partiellement décentralisée de notre système d'agents. En effet, ceci permet de réduire considérablement le nombre de messages transmis au serveur puisqu'une fois que les adresses IP des agents sont fournies par le médiateur, toutes les transactions entre agents personnels peuvent être effectuées indépendamment du serveur. Ainsi, les risques de congestion ou de manque de ressources dans le réseau sont grandement réduits même si le nombre d'agents personnels est élevé.

3.6 L'environnement de discussion

Nous avons développé nous-mêmes l'environnement de discussion fourni à l'usager par White Rabbit plutôt que d'utiliser une interface existante comme IRC, et ce pour



1. L'usager A demande à son agent que l'usager B lui soit présenté ;
2. L'agent A demande à l'agent médiateur l'adresse IP de la machine sur laquelle l'agent B se trouve ;
3. L'agent médiateur fournit à l'agent A l'adresse IP si l'usager est connecté. S'il ne l'est pas, sauter à l'étape 8;
4. L'agent A entre en contact avec l'agent B et lui demande les infos de son client, grâce à Java RMI ;
5. L'agent B demande à l'usager B si il ou elle désire que ses informations personnelles non-confidentielles soient divulguées à l'usager A ;
6. L'usager B rend une réponse affirmative ou négative ;
7. L'agent A amène sur sa machine les informations sur l'usager B si la réponse de l'étape 6 est positive. Dans le cas contraire, il reste les mains vides ;
8. L'agent A présente à l'usager A les informations obtenues, s'il y a lieu, à travers une fenêtre de présentation. Dans le cas contraire, il l'avertit que soit l'usager B n'est pas connecté, soit celui-ci a refusé la présentation.

FIG. 3.6 – Mécanisme des présentations

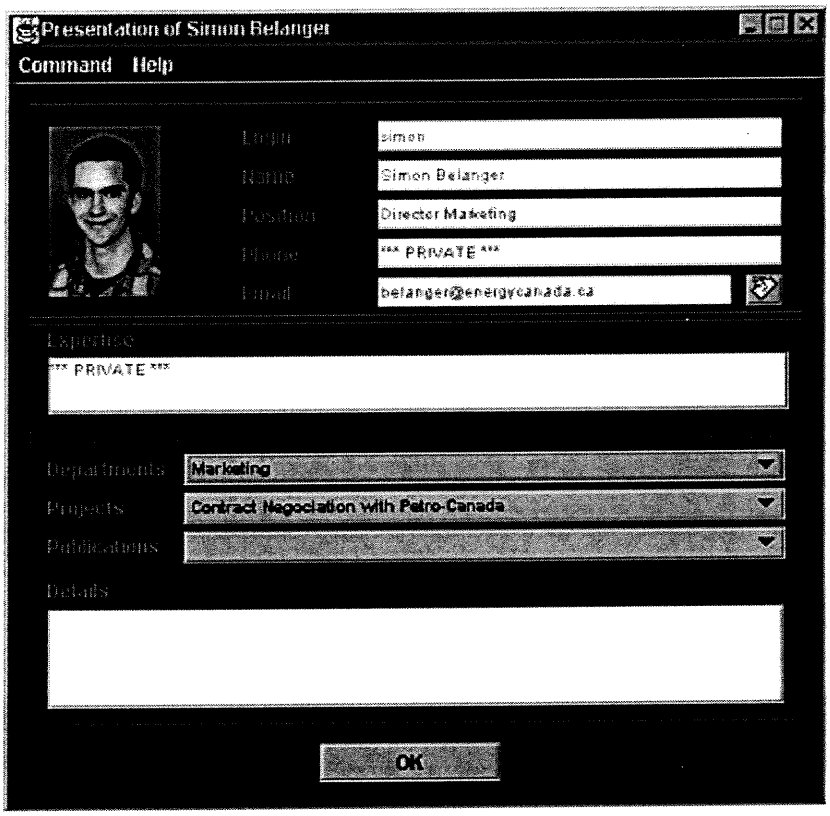


FIG. 3.7 – Fenêtre de présentation

plusieurs raisons. Nous voulions avoir un environnement de discussion nous permettant facilement d'intégrer les fonctions d'analyse de la discussion et de catégorisation. Nous voulions aussi, à travers l'interface, pouvoir restreindre d'une certaine façon la discussion au domaine d'intérêts choisi. Nous avons donc réalisé notre propre environnement de discussion simple permettant à plusieurs personnes de dialoguer et aux agents de faire leur travail de manière efficace (Figures 3.8 et 3.9).

L'interface permet à un usager :

- de créer et de maintenir un profil d'intérêts,
- de le visualiser,
- de discuter en spécifiant, s'il le juge nécessaire, le type des messages qu'il envoie (déclaration d'intérêts explicite, formulation de restrictions, etc.),
- d'ajuster le niveau d'activité de son agent,
- de connaître la liste des usagers jugés comme étant compatibles par son agent (suite à la catégorisation),
- d'être présenté à un ou plusieurs de ceux-ci,
- de recevoir et répondre aux messages que lui envoie son agent personnel,
- de sauvegarder son profil pour le récupérer lors d'une session ultérieure.

La figure 3.8 donne un exemple de conversation entre deux usagers connectés au système sous les pseudonymes de Marc et Simon. La liste des usagers connectés est située en haut à droite. Les messages à envoyer sont écrits dans la zone de texte située au bas de l'interface, puis sont envoyés aux destinataires (sélectionnés dans la liste de choix "*send to*") en cliquant sur le bouton "*send*". Lorsqu'un message est ainsi transmis, plusieurs événements se produisent. Tout d'abord, le message est traité par l'agent personnel qui tente d'y trouver des occurrences de mots clés ou encore des mots similaires à des mots clés (voir le module d'apprentissage, section 3.3). Si de tels mots sont trouvés dans le message, le profil de l'usager est alors mis à jour, en tenant

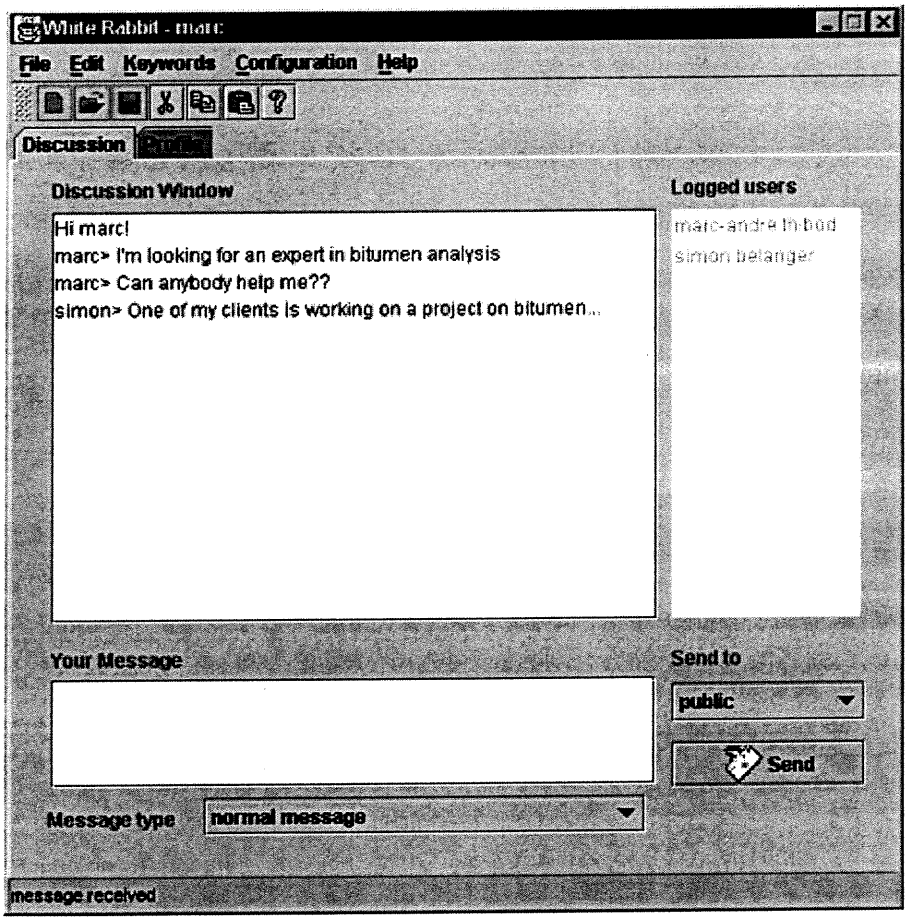


FIG. 3.8 – Interface de discussion de White Rabbit

compte du type du message (déterminé par l'usager dans la liste de choix "*message type*"). Le type du message est l'un des trois choix suivant :

1. message normal, généralement utilisé durant la conversation,

2. déclaration d'intérêts, utilisé pour signifier à l'agent que le message envoyé décrit un ou plusieurs de ses intérêts.

3. restriction, utilisé pour signifier à l'agent que le message envoyé contient des termes ne faisant pas partie de ses intérêts.

Le type du message n'est destiné qu'à accélérer l'apprentissage de l'agent et à permettre à l'utilisateur de corriger manuellement le profil construit par l'agent. Par conséquent, il est tout à fait possible de n'utiliser que des messages de type "normal" et de parvenir à de bons résultats (voir l'évaluation du système à la section 4.1.4).

Suite à l'analyse du message par l'agent personnel, le message est transmis au serveur de discussion à travers la connexion. Celui-ci reçoit en même temps l'information sur le ou les destinataires du messages. Le serveur sélectionne alors les connexions des personnes auxquelles le message est destiné, puis leur transmet le message qui sera affiché sur leur fenêtre de discussion (dans la zone de texte "*Discussion Window*"). Les personnes recevant le message peuvent alors le lire et y répondre, donnant lieu à la discussion, et permettant à tous les agents personnels du système d'apprendre sur les intérêts de leur usager.

Le profil appris par l'agent peut être consulté par l'utilisateur en cliquant sur l'onglet "*Profile*" de son interface de discussion. En sélectionnant cet onglet, il obtient l'interface de la figure 3.9. Comme on peut le voir sur la figure, le profil est affiché sous la forme d'une table à deux colonnes. La colonne de gauche est la liste des mots clés de la base, alors que la colonne de droite indique un poids pour chacun des mots clés, sous la forme d'un histogramme. Les mots clés possédant un poids plus important ont une barre (de l'histogramme) plus longue. Une couleur plus foncée quant à elle indique un niveau de certitude plus élevé par rapport au poids déterminé. Finalement,

l'usager peut trier la table, par ordre alphabétique, ou par ordre de poids, croissant ou décroissant, en cliquant sur l'entête de celle-ci.

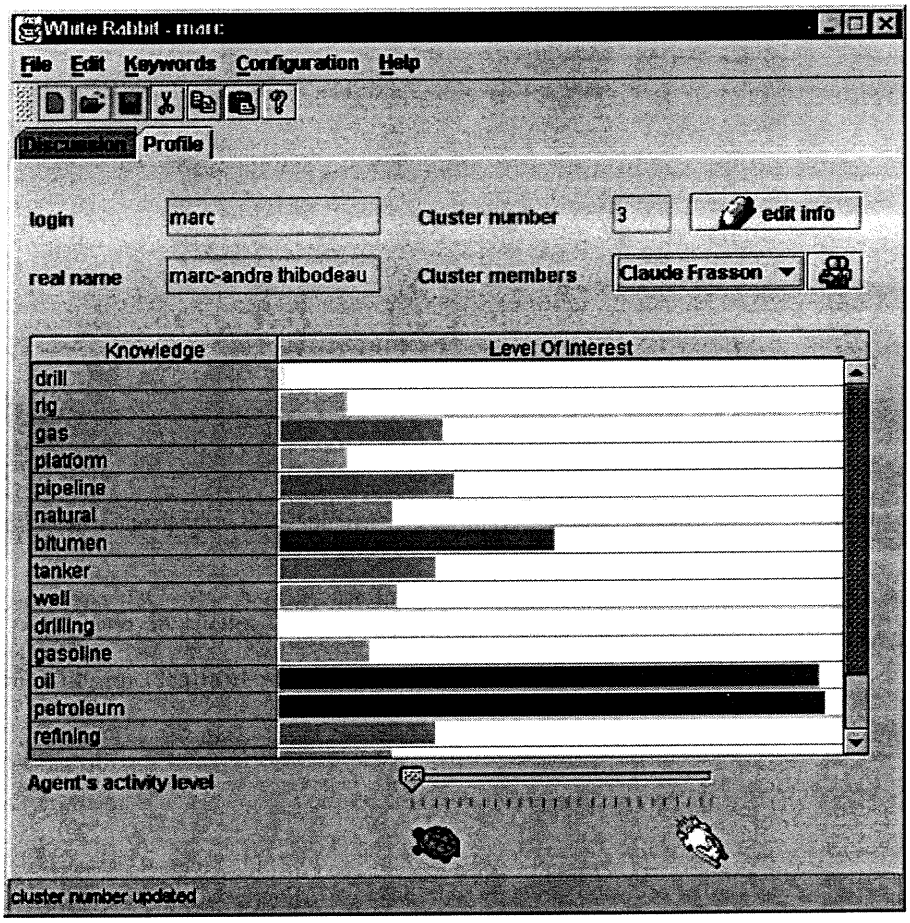


FIG. 3.9 – Onglet "Profile" de la fenêtre de discussion servant à la gestion du profil de l'usager

Sur ce même onglet, l'usager peut contrôler la vitesse d'activité de son agent (au bas, "*Agent activity level*"). Un agent très actif pose plus souvent des questions à l'utilisateur et construit donc plus vite le profil, mais peut, par le fait même, devenir trop présent au goût de l'utilisateur. Celui-ci peut donc régler l'activité de son agent selon ses besoins.

C'est finalement sur cet onglet que l'usager peut consulter la liste des employés classés dans la même catégorie que lui par le réseau de neurones, et demander à ce que l'un de ceux-ci lui soit présenté (voir le module de communication, section 3.5).

Tout ceci implique préalablement qu'une connexion soit établie entre le client et le serveur de discussion et ensuite que des informations initiales soient transmises au client (les usagers connectés au système, les informations personnelles sur l'usager sauvegardées lors des sessions précédentes, la liste des mots clés de la base ainsi que les poids et certitudes correspondants). Pour établir une connexion, un usager peut soit ouvrir son profil, s'il a déjà utilisé le système, soit créer un nouveau profil, s'il s'agit de sa première session. Dans le dernier cas, l'usager doit fournir un pseudonyme, un mot de passe, ainsi qu'un numéro d'employé, servant à l'identifier au niveau du serveur. Chacun des employé ne peut avoir qu'un seul compte ouvert, et chaque pseudonyme ne peut être utilisé qu'une seule fois. Le pseudonyme permet de conserver l'anonymat durant la discussion, et permet de ne pas surcharger la fenêtre de discussion. Une fois la connexion établie, les informations initiales sont transmises au client et la discussion peut avoir lieu.

3.7 L'interface de l'administrateur

White Rabbit offre également à l'administrateur du système une interface lui permettant de gérer les usagers du système, de connaître l'adresse de leur machine, de visualiser la répartition des utilisateurs dans les différentes catégories, de modifier les paramètres du réseau de neurones et de modifier la base de connaissances (figure 3.10). Trois paramètres du réseau de neurones peuvent être modifiés, soient le taux d'apprentissage, le nombre de passes d'entraînement effectuées et le nombre de neurones de sorties (le nombre de catégories produites). Notez que si (1) la base de connaissances est modifiée ou si (2) le nombre d'unités de sorties est changé, alors le réseau de neurone doit être initialisé et entraîné à nouveau, car les catégories pro-

duites deviennent alors invalides et la taille des données d'entrées incompatibles avec la taille de la couche d'entrée du réseau. Mais ceci ne diminue en rien les performances du système, puisque ce processus est relativement rapide (voir section 4.1.3) et tout à fait transparent pour les usagers.

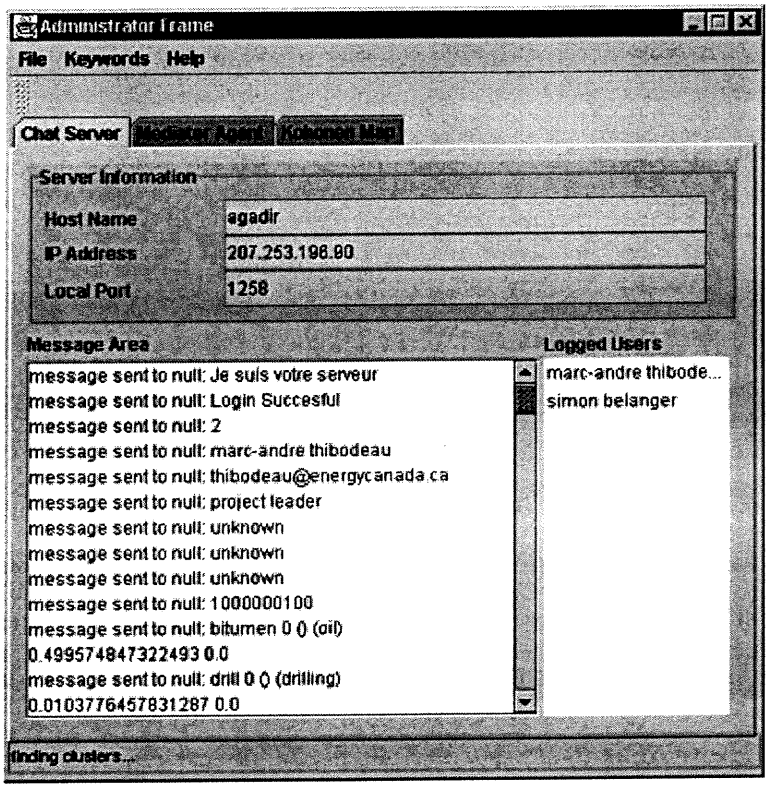


FIG. 3.10 – Interface de l'administrateur

De cette façon, l'administrateur de White Rabbit est en mesure d'ajuster le nombre de catégories au nombre d'employés dont les profils sont dans le système.

La seconde fonction importante de cette interface est de permettre la modification de la base de connaissances. En réalité, toutes ces modifications sont faites par l'intermédiaire d'ALICE qui sera présenté brièvement au chapitre 3.9. Toutefois, l'interface de l'administrateur lui permet de recevoir de la part des usagers des sug-

gestions de nouveaux mots clés qu'il peut alors accepter ou refuser. S'il accepte une proposition, White Rabbit communique avec ALICE qui ajoute le mot clé à la base et qui retransmet ensuite la modification à tous les clients connectés au système afin que ceux-ci mettent à jour leur copie locale du graphe de connaissances.

3.8 Implantation

Nous avons entièrement développé le système White Rabbit en Java 2 (JDK 1.2). L'un des avantages majeurs de ce choix est la portabilité apportée par Java, caractéristique indispensable d'un système qui serait implanté dans un environnement potentiellement hétérogène. De plus, Java possède tous les outils nécessaires pour la construction d'agents intelligents et autonomes [6] et ce dans une trousse de développement complète, gratuite, et éprouvée. L'un de ces outils est Java RMI qui a été utilisé comme méthode de communication.

Le système "Remote Method Invocation" (RMI) de Java permet à un objet exécuté sur une machine virtuelle Java (JVM) d'invoquer des méthodes d'un objet exécuté sur une autre JVM. RMI procure donc un moyen de communication dans un système d'objets distribués utilisant le langage Java. Du point de vue du programmeur, RMI permet par exemple à des objets d'une application client d'invoquer des méthodes d'objets situés sur une application serveur tout comme s'il s'agissait d'invocations de méthodes locales, en camouflant toute la mécanique nécessaire au transfert des données (paramètres et résultats) à travers le réseau .

Une habileté unique de RMI, constituant sa principale force, est de permettre à une application de télécharger le code d'un objet d'une autre application en cours d'exécution [32]. Ainsi, les types de données et les comportements, au départ disponibles seulement sur une application, peuvent être transmises à une autre application (pouvant se trouver sur une autre machine virtuelle, distante) et de cette façon, en

accroître les capacités dynamiquement. La "mobilité" des agents de notre système est donc réalisée de manière conceptuelle, les agents étant des applications distantes pouvant, grâce à RMI, s'échanger des fragments de code (les profils) et ainsi modifier leur comportement dynamiquement.

Finalement, la base de données utilisée pour stocker toutes les informations sur l'entreprise, les employés, les utilisateurs, les connaissances, les liens entre les connaissances, et les profils est une base de données relationnelle MS-Access, reliée à notre application par le driver JDataConnect de la compagnie Softsyn, préféré à la passerelle JDBC-ODBC fournie à même la trousse de développement de Java pour sa plus grande stabilité.

3.9 ALICE

ALICE est un système d'extraction de connaissances construit parallèlement au système White Rabbit par Simon Bélanger et décrit dans [2] et [3]. Les deux applications sont complémentaires et forment ensemble un solide système de recherche d'expert et d'aide à la coordination d'une entreprise.

ALICE est tout d'abord constitué d'un éditeur permettant de facilement modéliser une entreprise et de créer et modifier la base de connaissances sur celle-ci. Une entreprise est modélisée par différents types d'entités définies dans le système. Les types d'entités de base pour la modélisation sont le Département, l'Employé, le Projet et la Publication (Document, Rapport, etc.). Il est important de noter que la conception d'ALICE permet d'ajouter facilement de nouveaux types d'entités, afin d'étoffer et de préciser le modèle de la compagnie. Ainsi, l'administrateur (celui qui fait la modélisation et qui construit la base de connaissances) peut ajouter, enlever ou modifier des entités à travers une interface graphique simple. De même, il peut ajouter, enlever ou modifier des liens entre ces entités afin de montrer l'appartenance,

par exemple, de certains employés à un départements, ou encore la participation d'un employé à un projet, etc. Il construit donc un graphe représentant la compagnie. Les entités de ce graphe sont ensuite reliées au graphe de connaissances par l'intermédiaire de leurs profils d'intérêts et de capacités. L'administrateur doit par conséquent construire également un graphe de connaissances constitué de mots clés reliés entre eux par différents types de liens indiquant par exemple la similarité entre deux mots clés, ou encore qu'un mot clé est une généralisation d'un autre (figure 3.11). Ce second graphe permet de cerner le vocabulaire d'un domaine quelconque et c'est à partir de ce vocabulaire que se fait l'extraction des profils des utilisateurs.

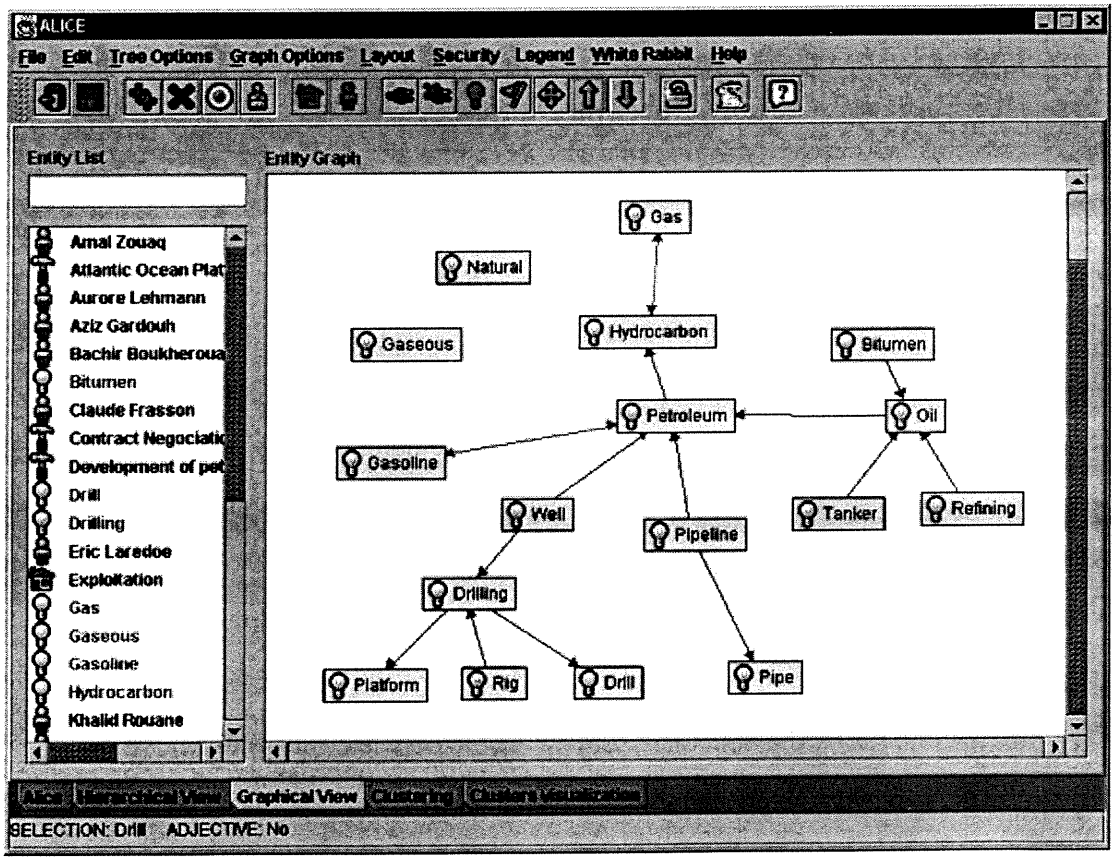


FIG. 3.11 – Visualisation et édition du graphe de connaissance PC^2 à l'aide d'ALICE

Une fois que les deux graphes sont construits, le système peut analyser les publications du graphe pour y découvrir les mots clés et ainsi en extraire un profil de capacités pour les employés qui en sont les auteurs. Ces profils sont complétés par l'analyse des curriculum vitae. Ce processus d'analyse est fait automatiquement par ALICE et est similaire à celui utilisé par les agents de White Rabbit lorsque ceux-ci surveillent les messages envoyés par leur usager (algorithme 1 décrit à la section 3.3). De la même façon, ALICE extraie un profil d'intérêts de base pour les employés à partir des descriptions des projets sur lesquels ils travaillent, (information fournie par le graphe de modélisation de l'entreprise). Ce dernier profil sera complété à l'aide de l'analyse de la conversation entre ces employés par White Rabbit. Finalement, un profil est évalué pour les projets et les départements à partir des profils des employés qui y sont reliés et par conséquent, toutes les entités de l'entreprises se retrouvent "profilées". Tous ces profils peuvent être visualisés par des histogrammes dont chaque barre représente le poids d'un mot clé dans un profil (figure 3.12). On peut voir sur la figure le profil d'un employé, constitué d'un profil de ses intérêts (à droite) et de ces capacités (à gauche).

Cette représentation sous la forme de deux graphes est la représentation PC^2 dont nous avons parlé précédemment. Il faut voir cette représentation comme deux couches complètement reliées entre elles. C'est grâce à cette représentation qu'il est possible pour nos agents de créer un profil représentatif des intérêts et capacités des employés et de découvrir des liens entre ceux-ci. De plus, puisque toutes les entités de l'entreprise possèdent un profil, il est aussi simple de comparer n'importe quelles deux entités de l'entreprise que de le faire avec des employés comme c'est le cas dans White Rabbit. Il s'agit là de la seconde tâche importante d'ALICE. En effet, ALICE possède aussi un moteur de recherche permettant à un utilisateur de faire des requêtes pour trouver, par exemple, des publications répondant à ses besoins, ou

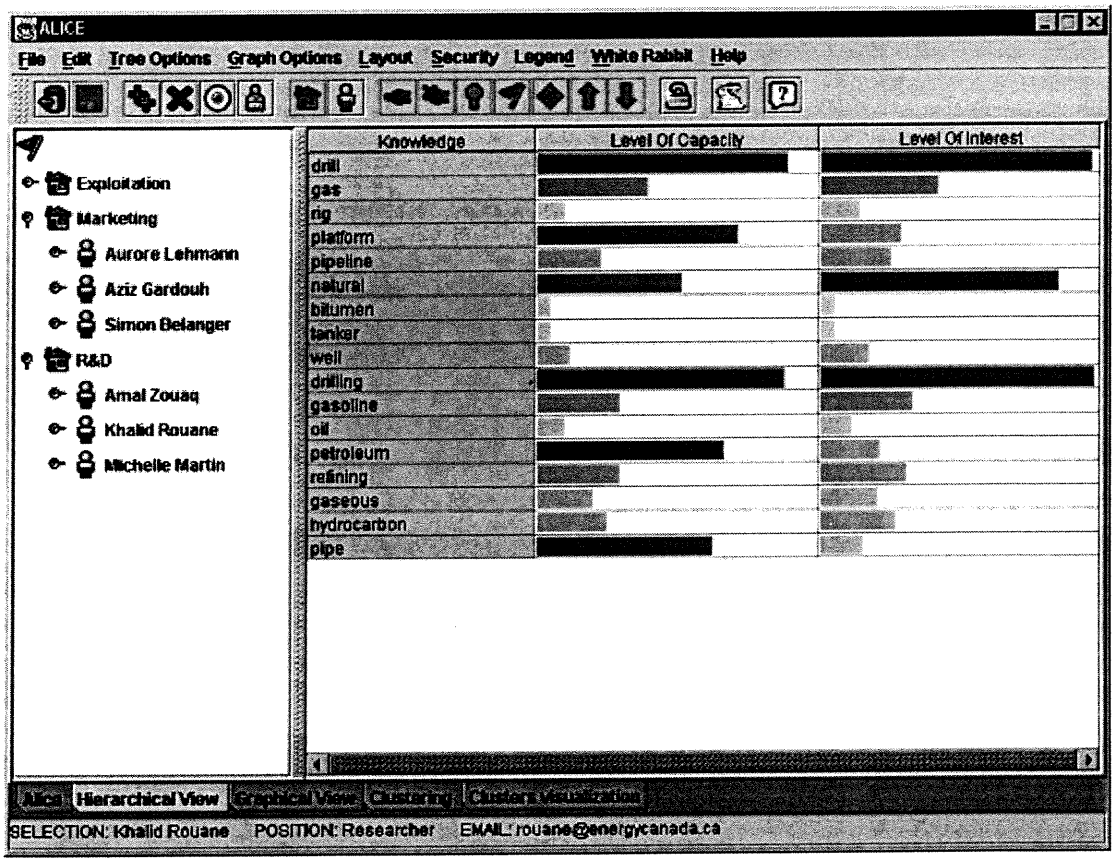


FIG. 3.12 – Visualisation des profils d'intérêts et de capacités des entités de l'entreprise à l'aide d'ALICE

encore à un gestionnaire de réunir dans un même projet tous les employés possédant les connaissances requises pour mener celui-ci à terme, etc.

Bref, une fois ALICE et White Rabbit réunis, nous obtenons un système capable d'aider, croyons-nous, un administrateur à visualiser sa compagnie et à en améliorer sa coordination, en plus d'aider ses employés à trouver les ressources dont ils ont besoins.

3.9.1 Découverte de nouveaux mots clés

L'un des plus important problème qu'ALICE a tenté de résoudre est celui de la découverte automatique de nouveaux mots clés, i.e. de mots clés se trouvant dans les textes analysés, mais qui ne sont pas dans la base de connaissances. Il s'agit d'un problème complexe qui est abordé par la recherche d'information. ALICE utilise une technique classique pour y parvenir. Il construit d'abord un dictionnaire contenant tous les mots rencontrés qu'il stocke dans sa base de données. Le nombre d'occurrences par document, et le nombre total d'occurrences dans tous les documents sont calculés pendant une phase de pré-traitement, précédant la phase d'analyse. À partir de ces statistiques, le système peut alors déterminer un degré de *discrimination* pour chacun des mots du dictionnaire. Un mot possède un haut degré de discrimination s'il revient souvent, mais dans peu de documents. Ces mots sont généralement de bons candidats pour devenir des mots clés. Par conséquent, lorsque le degré de discrimination d'un mot dépasse un certain seuil, ALICE demande à l'administrateur si ce mot devrait être ajouté ou pas dans la base de connaissance.

3.10 Conclusion

Nous avons présenté et expliqué en détails les différentes composantes de notre système. Nous avons présenté quelques algorithmes clés et décortiqué les principaux

modules : apprentissage, communication, catégorisation et environnement de discussion. Nous avons finalement présenté brièvement le logiciel ALICE, complémentaire à White Rabbit. Le prochain chapitre discute des résultats des tests que nous avons fait subir à notre système pour en faire l'évaluation sous différents aspects.

Chapitre 4

Résultats et Discussions

White Rabbit est à la base un système conçu pour répondre aux besoins d'une communauté de taille restreinte comme celle d'un centre de recherche, d'une organisation ou d'une entreprise. Par conséquent, les problèmes que pourraient poser son élargissement à une communauté de grande taille n'ont pas été notre préoccupation première. L'emphase de nos efforts a plutôt été mis sur l'efficacité du travail des agents, c'est-à-dire le choix réfléchi d'une représentation de connaissances, l'implantation d'algorithmes d'apprentissages, de catégorisation et de communication efficaces, la convivialité de l'interface graphique, l'utilisation de technologies de pointe comme Java RMI et la version la plus récente de Java. Le logiciel obtenu est portable sur toutes les plates-formes et vise le développement d'une application utile et facilement transférable, tant aux industriels qu'aux différentes communautés scientifiques. C'est également cet aspect pratique qui nous a fait choisir l'analyse de la discussion en temps réel plutôt que l'analyse du courrier électronique des usagers par exemple. En effet, cette dernière alternative pose de sérieux problème d'accès à l'information et de confidentialité dans un milieu de travail réel comme il en est fait mention dans la littérature [12].

Des tests de base ont été menés pour vérifier l'efficacité de notre système. La première constatation est l'importance prépondérante de la base de connaissances dans la qualité des résultats de la catégorisation des usagers. En effet, la qualité de ce processus dépend directement des liens entre les connaissances permettant aux agents de construire une représentation juste des intérêts de l'usager. Si le graphe de connaissances est incomplet et donc peu représentatif de la réalité, alors l'apprentissage de l'agent est inefficace et donc l'étape de la catégorisation perd tout son sens. La section 4.1 présente les résultats des expériences que nous avons menées pour l'évaluation de notre système. Ces expériences consistent en collectes de données statistiques pour l'évaluation de la performance du système et en collectes de commentaires pour une évaluation qualitative de son utilité. La section 4.2 ouvre la porte à des recherches futures en évoquant quelques améliorations possibles et envisagées, et finalement la section 4.3 présente d'autres applications possibles de notre architecture.

4.1 Évaluation du système

Cette section est divisée en cinq parties, chacune évaluant un aspect particulier du système. Il s'agit d'évaluations quantitatives et qualitatives visant à vérifier si l'objectif premier a été atteint et ce, à quel degré, en plus de fournir une appréciation des performances globales du système construit.

4.1.1 Validation de la représentation des connaissances

Comme nous l'avons précédemment mentionné, la représentation des connaissances que nous avons choisie est un ensemble de mots clés reliés les uns aux autres par des liens possédant différentes valeurs sémantiques. La structure ainsi obtenue est un graphe où les noeuds sont des mots clés et où les liens servent à établir les relations entre ces mots et de ce fait définissant un domaine. Nous avons déjà intuitivement supposé l'importance de la présence des liens dans l'apprentissage d'un

profil représentatif et réaliste des usagers et, par conséquent, dans la qualité de l'étape de la catégorisation. Nous avons tenté de vérifier cette affirmation de manière plus rigoureuse que par notre simple intuition.

Hypothèse : Sans les liens du graphe de connaissances, la catégorisation est inefficace.

Méthodologie : D'abord, créer deux graphes de connaissances contenant le même vocabulaire (et dans ce vocabulaire, des synonymes), l'un avec des liens sémantiques entre les mots clés, l'autre sans aucun lien. Ensuite, créer deux individus dont l'un possède une publication traitant de l'un des mots du vocabulaire, et l'autre qui se connectera à l'environnement de discussion et parlera en utilisant des mots autres que le mot se trouvant dans la publication du premier, mais reliés sémantiquement à celui-ci. Les deux individus devraient donc être compatibles et par conséquent, le système devrait les classer dans la même catégorie. Faire cette expérience avec les deux graphes créés et observer les résultats de la catégorisation.

Résultats : Le domaine choisi a été la musique. À l'aide d'ALICE, nous avons construit une base de connaissances possédant un graphe de 26 mots clés, puis nous avons copié cette base. Dans l'une des copies (Base A) nous avons ajouté des liens sémantiques de deux types : généralisation et similarité. Par exemple, les mots "Violoncelle" et "Instrument" sont reliés par une généralisation allant de "Violoncelle" à "Instrument", alors que les mots "Orchestre" et "Fanfare" sont reliés comme étant des synonymes. Le graphe contient en tout 40 liens dont 8 liens de similarité et 32 de généralisation. Nous avons ensuite créé deux individus et une publication. La publication est un texte contenant quatre fois le mot "Gospel" intitulé "History of Gospel" qui ne contient aucun autre mot clé du graphe. Nous avons relié la publication à l'un des deux individus, disons l'individu 1. Conséquemment, le profil de capacités de l'individu 1 est mis à jour en fonction de la base de connaissance utilisée. Voici le résultat avec chacune des bases :

Avec liens (Base A)	Sans liens (Base B)
Gospel : 99%	Gospel : 99%
Spiritual : 94%	Spiritual : 5%
Musique : 34%	Musique : 5%
...	...

On voit que dans le cas de la base A, les mots reliés à "Gospel" dans le graphe de connaissances sont aussi mis à jour à la hausse. Ainsi le profile de capacités de l'individu 1 reflète le fait que, puisqu'il a publié sur le sujet du Gospel, alors il doit aussi posséder des connaissances sur la musique, le Spiritual (relié au Gospel dans la base A par un fort lien de similarité), etc. Dans le cas de la base B, seulement le poids du Gospel est augmenté. Les autres poids demeurent à leur valeur initiale (5%).

Ensuite, le deuxième individu se connecte au serveur de discussion via White Rabbit. Dans les deux cas, l'individu 2 écrit et envoie les phrases suivantes sur le "chat" (en spécifiant qu'il s'agit de déclarations d'intérêts) :

"Je m'intéresse au spiritual et aux styles qui y sont apparentés."

"Je m'y connaît très peu dans le style spiritual."

Le comportement du système suivant cette courte discussion (disons ici un court monologue) diffère selon la base de connaissances utilisée. Avec la base A, l'agent personnel de l'individu 2 lui demande s'il est "intéressé au *Gospel*", puisque l'agent découvre l'intérêt de cet individu pour le *spiritual* et qu'il sait (grâce aux liens sémantiques du graphe) qu'il existe un lien très fort entre *gospel* et *spiritual*. Supposons que l'individu 2 réponde par l'affirmative à la question posée par son agent. Le profil d'intérêts obtenu pour l'individu 2 est alors :

Spiritual : 99%

Gospel : 97%

Musique : 14%

...

Par contre, en utilisant la base B, l'agent personnel de l'individu 2 ne lui pose aucune question, ne faisant aucun lien entre son intérêt pour le *spiritual* et quelque autre terme contenu dans le vocabulaire. Le profil d'intérêts alors obtenu pour l'individu 2 est le suivant :

Spiritual : 99%

Gospel : 5%

Musique : 5%

...

On peut dès lors remarquer la similitude entre le profil de capacités de l'individu 1 et le profil d'intérêts de l'individu 2 en utilisant la base A. Maintenant, si on observe les profils obtenus avec la base B, on remarque que l'individu 1 ne possède des connaissance que dans le Gospel principalement, et que l'individu 2 ne s'intéresse qu'au spiritual, mais pas au gospel, puisqu'il n'a aucunement parlé de Gospel. Les profils des deux individus ne sont donc pas compatibles avec la base B, mais semblent l'être avec la base A.

Dans les faits, avec la base A, les individus 1 et 2 se retrouvent catégorisés ensemble par le réseau de neurones lorsqu'il y a jusqu'à 6 unités dans la couche de sortie, donc 6 catégories produites (valeur déterminée expérimentalement). Avec la base B, les individus se retrouvent dans la même catégorie si et seulement s'il y a 2 unités ou moins dans la couche de sortie.

Conclusions : On peut par conséquent constater premièrement que les profils des usagers sont mieux cernés avec une base contenant des liens entre les mots clés qu'avec une base n'en contenant pas. En effet, le système parvient à relier deux individus compatibles, dans le cas d'une base avec liens, lorsque le réseau de neurones divise en six catégories les profils qu'on lui soumet, alors que s'il n'y a pas de liens dans

la base, le réseau de neurones ne parvient pas à réunir ces deux individus à moins qu'il n'y ait que deux catégories produites ou moins. Les liens fournissent donc aux Self-Organizing Maps plus d'information, leur permettant de mieux discriminer les entrées qui lui sont fournies.

De plus, les liens permettent de trouver des similarités entre des profils même lorsqu'elles ne sont pas explicitement exprimées, comme ce fut le cas dans notre expérience. En effet, à aucun moment le premier individu n'a évoqué le mot spiritual, et à aucun moment le second individu n'a mentionné le mot gospel et pourtant, puisque ces deux mots sont reliés par la sémantique, dans le domaine particulier de la musique, ces individus devraient être mis en relation. Il s'agit là d'une lacune de certains moteurs de recherches disponibles sur l'Internet qui ne peuvent généralement pas découvrir des documents intéressant pour une personne si cette dernière ne spécifie pas les mots clés exacts ou des sous-mots de ces mots clés (ex : La Toile du Québec).

Il faut remarquer l'importance de la façon dont est construite la base de connaissances pour l'obtention de résultats satisfaisants. En effet, une base dont les liens seraient établis de manière peu représentative procurerait sans doute des résultats encore moins bons qu'une base sans liens, puisque des associations seraient faites là où il n'en existe pas. Les liens sont donc une arme à deux tranchants qu'il incombe à l'administrateur de manipuler avec soin.

4.1.2 Évaluation qualitative de l'effet de la taille de la base de connaissances

La taille de la base de connaissances, i.e. le nombre de mots clés et de liens la composant, influe nécessairement sur la qualité des résultats. Nous faisons ici une analyse de l'effet de la taille de la base de connaissances sur l'efficacité du système.

Hypothèse : Un graphe de connaissances petit réduit l'efficacité du système

Discussion : Il apparaît logique et évident que, si le vocabulaire de la base n'est pas suffisamment élaboré et ne couvre pas tout le domaine traité, alors des intérêts et des capacités ne pourront pas ou seront mal représentées dans les profils des usagers. Il s'ensuivra nécessairement une perte d'efficacité du système puisque celui-ci ignorera alors des mots importants durant la conversation.

Hypothèse : La taille du graphe de connaissances n'est pas directement proportionnelle à la qualité de la catégorisation.

Discussion : Il serait erroné, à partir de la première hypothèse, de conclure qu'un graphe de grande taille entraînerait invariablement une amélioration de la catégorisation. En effet, si nous définissons la taille du graphe de connaissances comme suit :

$$t(g) = v(g) + e(g) \tag{4.1}$$

où g est un graphe de connaissances, $v(x)$ est le nombre de mots clés d'un graphe x, et $e(x)$ est le nombre de liens du graphe x. Et comme nous l'avons conclu suite à l'expérience 4.1.1, les liens jouent un rôle prépondérant dans la qualité de la catégorisation. Donc, il ne suffit pas qu'un graphe de connaissances soit de grande taille, il faut également que le ratio $e(g)/v(g)$ soit suffisamment élevé.

Conclusion :La qualité d'un graphe de connaissances n'est pas quantifiable, mais elle est proportionnelle :

1. au nombre de mots clés qu'elle contient,
2. à son ratio liens/mots clés.

4.1.3 Analyse de la performance

Cette section fait l'évaluation des performances du logiciel en termes de vitesse d'exécution. Les points particuliers du système que nous avons évalués sont la vitesse

de chargement de la base (i.e. le temps que prend le programme à charger toutes les entités et tous les liens en mémoire principale à partir de la base de données), le temps d'apprentissage du réseau de neurones et la vitesse d'analyse des messages par les agents, les trois algorithmes clés du système.

À la lumière de l'expérience décrite en 4.1.2, nous avons affirmé en toute logique qu'une base de connaissances est plus représentative de la réalité lorsqu'elle est de grande taille. Toutefois, la taille de cette base influe nécessairement sur les temps mentionnés. La présente expérience nous a permis de vérifier dans quelle mesure ces temps d'exécution varient en fonction de la taille de la base de connaissances.

Hypothèse : Les temps de chargements de la base, d'apprentissage du réseau de neurones et d'analyse des messages augmentent augmentent de façon linéaire en fonction de la taille de la base.

Méthodologie : Cette expérience est faite en trois parties. La première consiste à calculer les temps de chargement de la base avec différentes bases de connaissances en faisant varier le nombre d'employés et le nombre de mots clés dans la base (l'entité employé a été choisi arbitrairement. Nous aurions pu utiliser n'importe lequel des type d'entités). La seconde consiste à calculer les temps d'apprentissage du réseau de neurones en utilisant les mêmes bases de connaissances utilisées dans la première partie de cette expérience. La dernière partie de l'expérience est une analyse théorique qui nous permet d'évaluer le temps d'analyse des messages échangés par les usagers à l'intérieur de l'environnement de discussion.

Ceci signifie créer deux séries de bases de connaissances, la première ayant un nombre de mots clés constant et dont chacune contient un nombre d'employés différent, et la seconde possédant un nombre d'employés constant et dont chacune possède un nombre de mots clés différent. Puis, construire ces bases à l'aide de l'éditeur de graphe de connaissances d'ALICE. Ensuite, ajouter le code nécessaire pour faire le calcul des

temps voulus, et finalement faire des graphiques à partir des données recueillies qui pourront alors être analysés.

Chacun des temps est calculé cinq fois et le temps moyen obtenu est considéré. Le tout a été fait sur une machine de type pentium-III de 500 MHz, avec une mémoire vive de 128 Megs et sur la plate-forme Windows NT 4.0.

Résultats : 15 bases de connaissances ont été créées. La première série est composée de bases contenant 100 mots clés et un nombre variable d'employés, soient : 0, 10, 20, 30, 40 et 50 employés. La seconde série est composée de bases contenant 50 employés et un nombre variable de mots clés, soient 10, 20, 30, 40, 50, 60, 70, 80 et 90 connaissances (Il n'y a évidemment aucune base ne contenant aucun mot clé, car dans ce cas, l'apprentissage du réseau de neurones ne saurait être effectué). La première partie de l'expérience consistait donc à calculer les temps de chargements de la base de données cinq fois avec chacune des bases contenues dans les deux séries et d'en déduire un temps moyen de chargement pour chacune des bases. Le tableau 1 présente les données obtenues. Dans cette table, les tc_i sont les différentes mesures prises pour les temps de chargements de la base. le tc_{moyen} est la moyenne des tc_i, utilisée pour construire les graphiques de cette section.

Nb Employés (100 mots)	tc_1	tc_2	tc_3	tc_4	tc_5	tc_{moyen}	***écart-type***
0	491	481	421	421	420	**446.8**	36.0
10	1162	1181	1172	1192	1162	**1173.8**	12.9
20	1753	1813	1802	1843	1893	**1820.8**	51.8
30	2744	2673	2673	2764	2674	**2705.6**	44.7
40	3215	3245	3204	3215	3145	**3204.8**	36.7
50	4105	4056	4066	4056	4086	**4073.8**	21.3

Nb Mots Clés (50 emp.)							
10	751	721	731	711	701	**723.0**	19.2
20	1022	981	1002	1012	1001	**1003.6**	15.2
30	1302	1321	1291	1292	1302	**1301.6**	12.1
40	1582	1562	1563	1592	1532	**1566.2**	23.0
50	2203	2193	2193	2173	2194	**2191.2**	11.0
60	2504	2463	2483	2484	2433	**2473.4**	26.8
70	2774	2774	2774	2784	2864	**2794.0**	39.4
80	3224	3075	3125	3134	3155	**3142.6**	54.2
90	3585	3555	3535	3575	3555	**3561.0**	19.5
100	4105	4056	4066	4056	4086	**4073.8**	21.3

Tableau 1 - Tableau des temps de chargement selon différentes tailles de base. Les temps sont en milli-secondes.

Les graphiques 1 et 2 montrent une croissance linéaire du temps de chargement des bases de connaissances dans le système en fonction du nombre d'entités (ici, des employés) qu'elles contiennent, avec un nombre fixe de mots clés. À partir de l'équation de la droite se trouvant sur le graphique 2, qui fait l'interpolation des points obtenus par une fonction du premier degré, on peut estimer le temps de chargement d'une entité à 71,8 milli-secondes.

Les graphiques 3 et 4 montrent également une croissance linéaire du temps de chargement des bases de connaissances, mais cette fois-ci en fonction du nombre de mots clés de la base, avec un nombre d'employés constant. Cette fois-ci, l'accroissement du temps de chargement est moins élevé, comme le montre la dérivée de

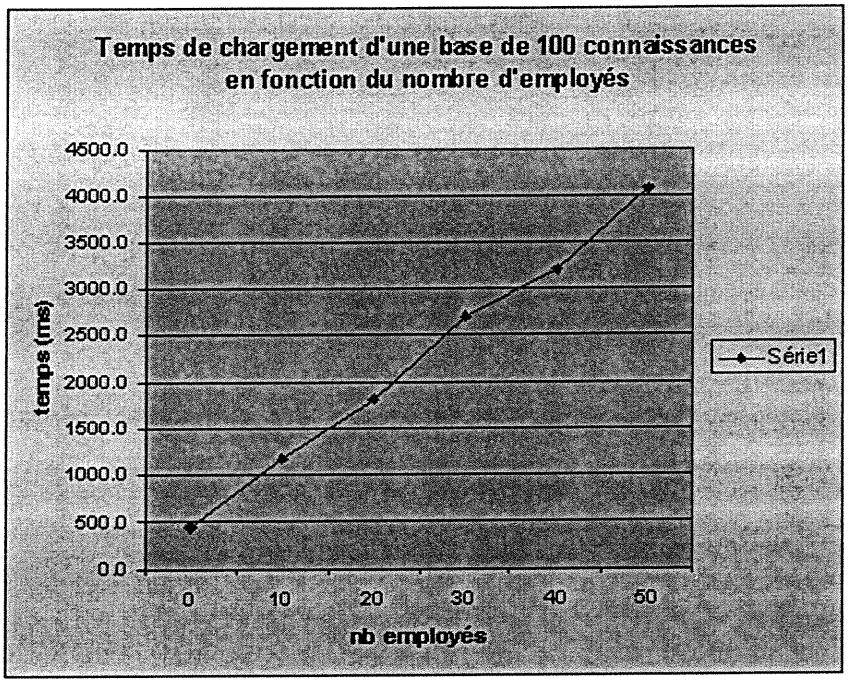


FIG. 4.1 – Graphique 1

l'équation d'interpolation du graphique 4. En effet, l'accroissement de temps obtenu pour chaque connaissance est cette fois-ci d'environ 37,1 milli-secondes.

Maintenant, la seconde partie de l'expérience consistait à observer l'effet de la taille de la base de connaissances sur le temps d'apprentissage du réseau de neurones. Nous avons procédé de la même façon que pour les mesures de temps de chargements. Les données obtenues sont présentées dans le tableau 2. Cette fois, la colonne ta_{moyen} donne les temps d'apprentissage moyens pour les différentes tailles de base, calculés à partir des cinq mesures se trouvant dans les colonnes ta_i. Les graphiques 5, 6 et 7 de ont été obtenus à partir de ces données.

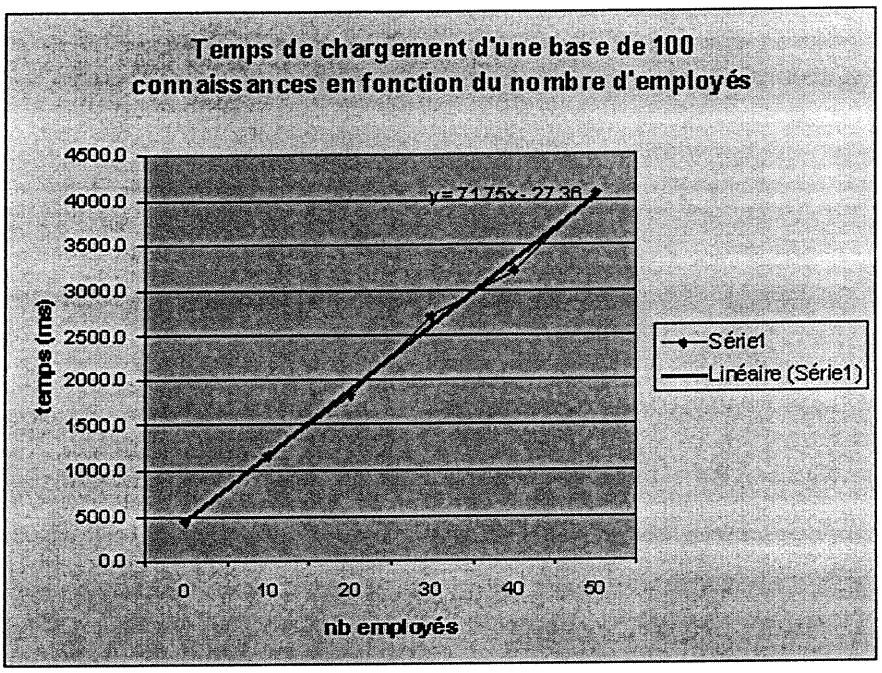


FIG. 4.2 – Graphique 2

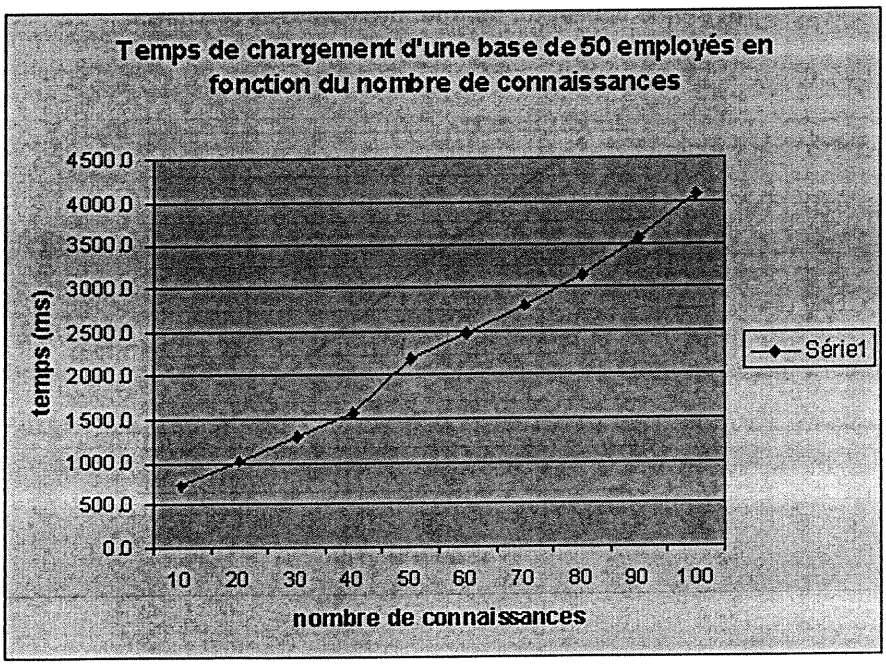


FIG. 4.3 – Graphique 3

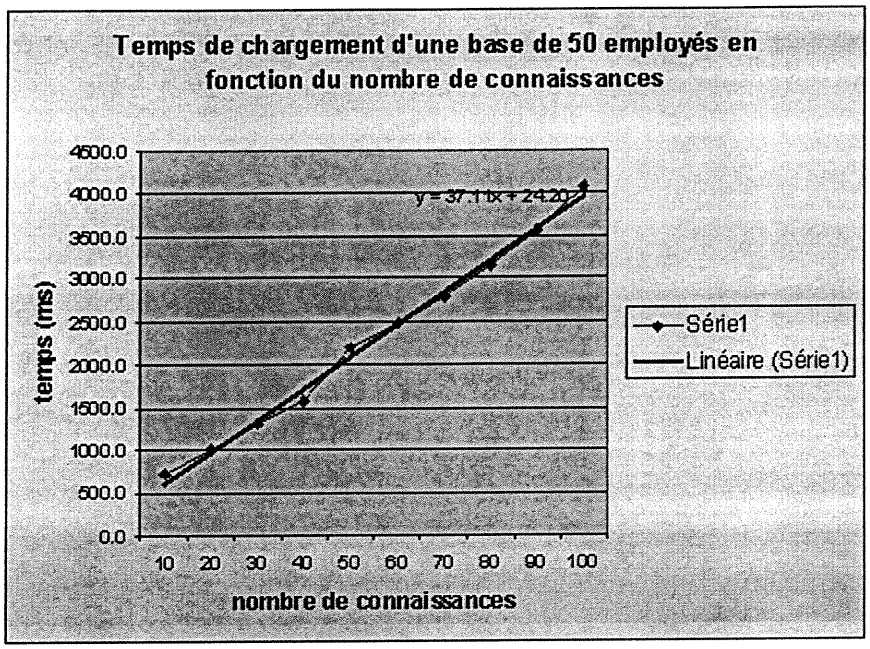


FIG. 4.4 – Graphique 4

Nb Employés (100 mots)	ta_1	ta_2	ta_3	ta_4	ta_5	ta_{moyen}	*écart-type*
0	x	x	x	x	x	x	x
10	31606	32166	32477	31735	31795	**31955.8**	358.1
20	32507	32447	31725	33017	32276	**32394.4**	465.1
30	31285	31816	32397	31526	31786	**31762**	415.3
40	30474	30153	29783	30784	29993	**30237.4**	396.3
50	29583	29803	30013	29903	30454	**29951.2**	322.7
Nb Mots clés (50 emp.)							
10	5708	6350	5859	6540	5939	**6079.2**	350.6
20	8963	8732	8803	8783	9253	**8906.8**	211.9

30	11176	10936	11617	11547	11146	**11284.4**	288
40	14411	14711	13670	14211	14060	**14212.6**	389.1
50	16614	16885	16434	16924	16945	**16760.4**	226
60	19588	19438	19508	19458	19188	**19436**	150.2
70	22572	21501	22833	21350	22372	**22125.6**	661.8
80	25126	24815	25137	25427	24856	**25072.2**	247.9
90	27570	27459	27670	27069	26999	**27353.4**	302
100	29583	29803	30013	29903	30454	**29951.2**	322.7

Tableau 2 - Tableau des temps d'apprentissage du réseau de neurones selon différentes tailles de bases. Les temps sont en milli-secondes.

Le graphique 5 montre une courbe en apparence illogique. En effet, le temps d'apprentissage diminue avec l'augmentation du nombre d'employés dans la base. En fait, il faut noter que dans tous les cas, 10 000 passes d'entraînement sont effectuées. Ceci signifie que peu importe le nombre d'entités, le système utilise 10 000 profils pour entraîner le réseau de neurones. Par conséquent, s'il existe 1000 profils dans le système, celui-ci passera 10 fois à travers cet ensemble de profil pour l'entraînement. Et donc le nombre d'employés qui varie ne devrait pas influer sur le temps d'apprentissage. Et d'ailleurs, en observant les données, on peut voir que le temps d'apprentissage oscille entre 29,95 secondes et 32,39 secondes, ce qui est une variation relativement faible de 2,44 secondes. On peut peut-être expliquer cette variation par le nombre décroissant du nombre de passes qui doivent être exécutées à travers tout l'ensemble de profils à mesure que celui-ci s'accroît, ou encore par des erreurs lors de la prise de données pouvant avoir été causées par des variations de la charge du microprocesseur en raison par exemple de l'exécution de processus extérieurs à notre système.

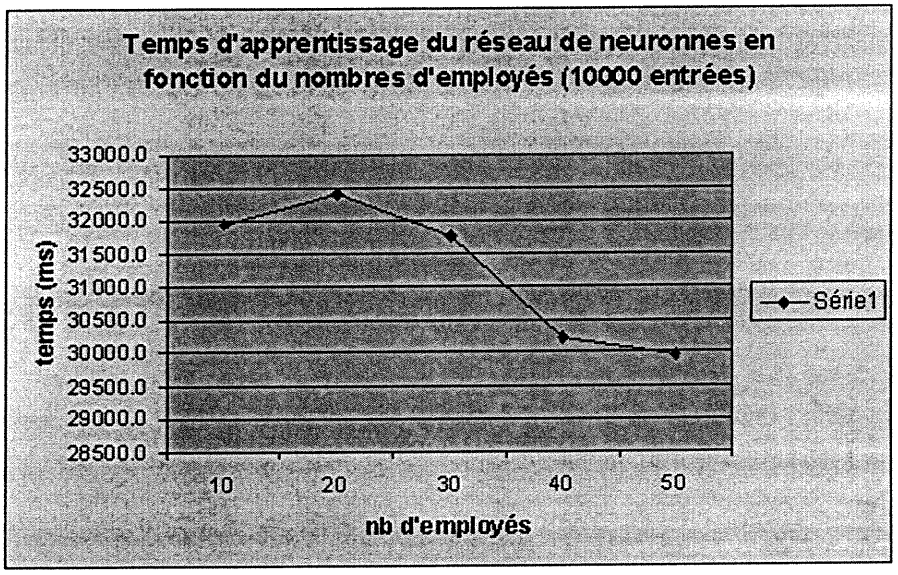


FIG. 4.5 – Graphique 5

Les graphiques 6 et 7, quant à eux, montrent un accroissement très stable du temps d'apprentissage en fonction du nombre de connaissances, avec un nombre constant d'employés. Ceci correspond parfaitement à ce que nous nous attendions. En effet, le nombre de mots clés fait varier la **taille** des profils. Donc, même si le nombre de profils utilisé demeure le même, le nombre de données utilisées pour faire les calculs requis pour l'apprentissage, lui, augmente avec le nombre de mots clés. Et cette augmentation du temps d'apprentissage est constante puisque le nombre de profils utilisés ne varie pas. La dérivée de l'équation d'interpolation du graphique 7 nous donne une augmentation du temps d'apprentissage de 266,2 milli-secondes par mot clé. Il est toutefois important de remarquer que ce temps d'apprentissage dépend aussi des paramètres d'apprentissage du réseau de neurones, que nous avons considérés ici comme constants, i.e. le nombres de passes d'entraînements à 10 000, le taux d'apprentissage à 0.9 et une couche de sortie de quatre unités par quatre unités. Ces derniers peuvent être modifiés par l'administrateur pour réduire le temps d'apprentissage dans le cas où celui-ci deviendrait trop lourd.

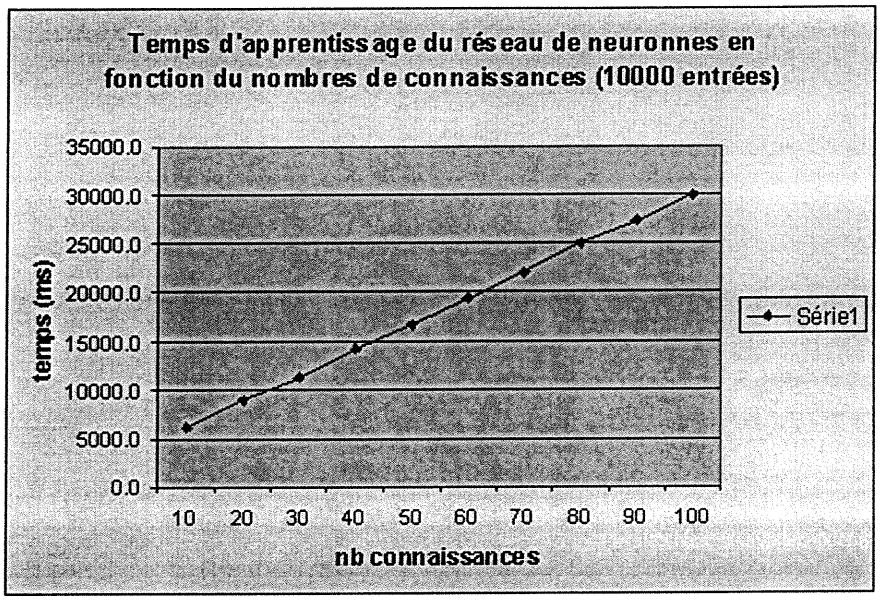


FIG. 4.6 – Graphique 6

Finalement, l'algorithme d'analyse des messages n'a été évalué que théoriquement, puisque sa rapidité n'a que très peu d'influence sur l'efficacité du système en comparaison des deux algorithmes évalués précédemment. En effet, les messages étant analysés beaucoup plus rapidement par les agents qu'ils ne sont écrits par les usagers, ces derniers ne sont en aucun cas contraints par la vitesse de cet algorithme, à la condition bien sûr d'utiliser une implantation raisonnable. Il est tout de même intéressant de jeter un coup d'oeil à cet algorithme et aux facteurs qui influencent sa performance.

L'algorithme d'analyse des messages est décrit à la section 3.3. Décortiquons-le étape par étape. Tout d'abord, un message est traité mot par mot par l'agent. Par conséquent, le temps d'analyse sera directement proportionnel au nombre de mot contenu dans le message. Ensuite, pour chacun des mot, l'agent doit déterminer s'il s'agit ou non d'un mot clé de la base de connaissances. Ceci est effectué en temps constant, puisque le graphe de connaissances est contenue dans une table de hachage

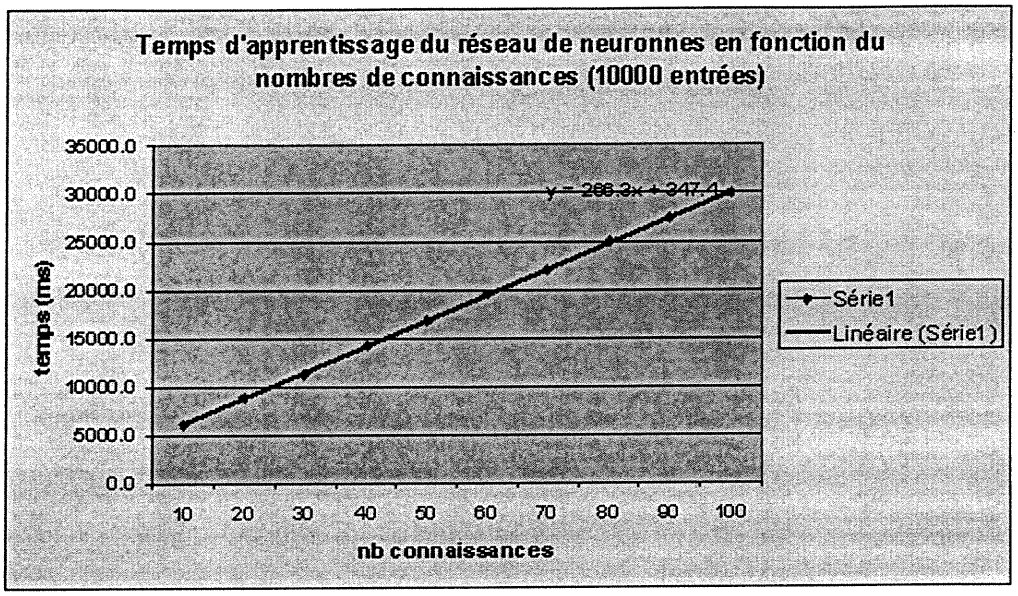


FIG. 4.7 – Graphique 7

[29]. Ensuite, si le mot est bel et bien un mot clé, l'agent doit alors en trouver le poids dans le profil de son usager, ce qui est également réalisé en temps constant, le profil étant une extension de la table de hachage de Java faisant correspondre un poids à chaque mot. Le nouveau poids est alors calculé en utilisant simplement la formule sigmoïde (équation (3.1)) donnée à la section 3.3 puis les poids de tous les mots clés reliés à celui trouvé dans le message sont mis à jour, ceci impliquant une fouille du graphe. La fouille d'un graphe, dans le pire des cas, s'effectue en $O(n)$ où n est le nombre de noeuds du graphe, ici les mots clés. Dans le cas où le mot du message n'est pas un mot clé, l'agent essaie de trouver un mot clé similaire, c'est-à-dire, ayant une région de 3 lettres identiques ou plus (les mots de moins de 3 lettres qui ne sont pas dans la base sont ignorés). Il s'agit de la partie la plus complexe de l'algorithme, car à ce moment, l'agent doit parcourir tous les mots clé de la base et vérifier si chacun possède une région identique à une région du mot du message. Si un tel mot similaire est trouvé, son poids est mis à jour tel que décrit plus haut.

Conclusions : Soient n le nombre d'entités dans la base de données et m le nombre de mots clés. D'après nos résultats, nous pouvons estimer le temps de chargement de la base par (4.2)

$$t_c = 71.8n + 37.1m + c \tag{4.2}$$

où c est une constante. Par conséquent, la complexité de cet algorithme est en $O(m+n)$ et on peut noter que le temps de chargement d'une entité est environ deux fois supérieur au temps de chargement d'un mot clé.

Ensuite, comme nous l'avons déjà mentionné, le temps d'apprentissage du réseau de neurones varie essentiellement et de façon linéaire en fonction de la taille des profils, i.e. du nombre de connaissances dans la base (en supposant l'ensemble des paramètres d'apprentissage du réseau constants). L'apprentissage est donc en $O(m)$.

Que pouvons-nous dire maintenant sur la capacité du système à supporter un environnement réel de grande taille? Prenons un exemple. Soit une compagnie de 1000 employés, avec 20 départements, 50 projets, et 5000 publications qu'on voudrait modéliser. Un ingénieur de connaissances détermine qu'un ensemble de 1000 concepts (mots clés) est suffisant pour bien cerner le domaine d'activité de la compagnie et de ses employés. Alors le temps de chargement de la base se situerait en appliquant l'équation (4.2) autour de 6070 x 71.8 + 1000 x 37.1 = 472 926 ms $\approx$ 7 minutes 53 secondes. Le temps d'apprentissage du réseau de neurones (en utilisant les paramètres d'apprentissage par défaut) d'environ 266,2 x 1000 = 266 200 ms $\approx$ 4 minutes 26 secondes. Ce sont donc des temps acceptables dans la mesure où le système n'est pas régulièrement interrompu (ce qui nécessite de recharger la base de connaissances à chaque nouveau démarrage du système) et où l'ensemble de mots clés demeure relativement stable (chaque sauvegarde de la base entraîne une recréation du réseau de neurones si l'ensemble de mots clés a été modifié).

L'analyse des messages, quant à elle, peut devenir lente si le nombre de connaissances devient très grand, en raison de la recherche de mots similaires lorsqu'un mot d'un message n'est pas un mot clé. Cependant, les messages échangés sur un environnement de discussion sont généralement courts et la conversation qui s'ensuit est ponctuée de nombreuses pauses (pour la lecture des messages des autres usagers et pour l'écriture des messages à envoyer), ce qui donne tout le temps nécessaire aux agents (qui se distribuent la charge de travail) pour analyser ces messages sans que les usagers n'en soient importunés ou que le fil de la conversation n'en soit perturbé.

4.1.4 Appréciation du système

La dernière partie de l'évaluation que nous avons menée consistait à recueillir des commentaires de plusieurs personnes suite à l'utilisation du système. Bien entendu, il s'agit là d'une évaluation purement qualitative, mais aussi la plus importante selon nous. En effet, le but de cette expérience était de vérifier le logiciel sous trois aspects : la convivialité, l'utilité et l'éthique. Trois thèmes étroitement reliés dont chacun peut déterminer si le système pourrait être utile ou utilisable dans un environnement réel. Le tout visant à vérifier si l'objectif du projet est somme toute atteint et ce dans quelle mesure. Rappelons ici notre objectif initial : faciliter la coopération d'individus travaillant au sein d'une organisation ou d'une entreprise par la découverte de leurs intérêts communs et de leurs connaissances complémentaires.

Notons que le but ici n'est pas de vérifier le caractère optimal du logiciel ou de l'algorithme de catégorisation par exemple. Pourquoi n'est-t-il pas approprié de poser ici cette question ? Premièrement, parce qu'il n'existe aucune métrique pour déterminer si deux êtres humains partagent ou non des intérêts similaires dans un domaine donné, pas plus qu'il n'en existe pour déterminer si deux personnes ont des connaissances complémentaires. Et deuxièmement, les solutions optimales prennent généralement beaucoup de temps à converger ; la plupart des problèmes de ce type

sont NP-complets. Il est toujours préférable que ce genre de système soit d'une vitesse acceptable et approximativement correct qu'insuffisamment rapide et parfait. [13]

But : Vérifier si l'interface du système est conviviale, si le système est utile, c'est-à-dire s'il effectue un bon travail de catégorisation et de présentation et si, de fait, il facilite la découverte de personnes ayant des intérêts similaires ou des connaissances complémentaires et finalement, si le système est acceptable du point de vue éthique dans un milieu de travail réel.

Méthodologie : Nous avons choisi le domaine des rencontres pour réaliser cette expérience. La raison de ce choix est simple : il nous fallait avoir accès à une grande quantité de données décrivant d'une part les "intérêts" de plusieurs personnes, et d'autre part les "compétences" de ces mêmes personnes. Or, il est très facile de trouver sur l'Internet des agences de rencontres fournissant gratuitement des annonces décrivant des personnes cherchant l'âme soeur (leurs compétences) ainsi que ce que ces personnes cherchent chez leur éventuel partenaire (leurs intérêts).

Nous avons donc premièrement recueilli sur un site Internet dédié à la rencontre 197 annonces de personnes cherchant l'âme soeur. Ces annonces nous ont alors permis de construire une base de connaissances en utilisant des mots clés s'y trouvant. Nous avons ensuite choisi 30 annonces bien étoffées parmi celles-ci, c'est-à-dire avec suffisamment de contenu pour qu'elles soient catégorisées de façon significative (éliminant les annonces ne contenant qu'une phrase ou deux, trop peu caractérisées pour être intéressantes), et décrivant des femmes âgées de 20 à 30 ans. Pourquoi avoir défini ainsi une catégorie particulière de personnes ? Simplement parce que notre système n'est pas conçu pour éliminer des personnes selon des critères précis, comme le sexe ou l'âge. Le domaine des rencontre est particulier sur ce point.

Les 30 annonces sélectionnées ont ensuite été entrées dans le système qui les a catégorisées à l'aide du réseau de neurones. Il fallait ensuite inviter quelques personnes

de sexe masculin à discuter du profil de la personne recherchée à travers l'interface de discussion. La conversation était dirigée, i.e. que nous posions des questions aux sujets de l'expérience susceptibles de leur faire décrire avec le plus de détails possible les caractéristiques qu'ils recherchent chez une compagne. Après quelques minutes, nous les invitions à consulter la liste des personnes classées dans la même catégorie qu'eux et à lire les annonces décrivant ces personnes en leur demandant d'évaluer qualitativement la valeur des annonces lues, i.e. avec quelle exactitude ces annonces correspondaient à leurs intérêts. Nous leur avons distribué un questionnaire visant à faciliter leur réflexion et recueillir leurs commentaires touchant aux trois aspects que nous cherchions ici à évaluer. Huit personnes se sont volontairement prêtées à cette évaluation.

Résultats : Au sujet de l'interface graphique, la grande majorité des participants (7 sur 8) ont eu de la facilité à maîtriser le fonctionnement de l'interface de discussion, lui attribuant une note générale de 4 sur 5 ou de 5 sur 5. Notons toutefois que sur ces personnes, deux seulement n'avait jamais utilisé une interface de discussion. Les commentaires récoltés sur cet aspect du logiciel suggèrent l'implantation de certaines fonctions, comme par exemple la possibilité de discuter sur des canaux privés. Ce qui ressort de ces commentaires est la nécessité de "standardiser" notre interface avec celle des interfaces de discussion les plus populaires, comme ICQ et mIRC. Toutefois, l'esthétique et la disposition des éléments de l'interface semble faire ou presque l'unanimité.

Les avis sont plus partagés quand à l'aspect éthique de l'utilisation d'un tel logiciel dans une entreprise. Bien que la majorité des personnes questionnées (5 sur 8) aient approuvé sans réserve l'utilisation d'un système d'analyse des conversations, en invoquant qu'il s'agirait d'un outil efficace pour réduire les recoupements de tâches et d'aide à la décision et à la gestion du personnels et des ressources d'une entreprise, une bonne partie des personnes (3 sur 8) sont beaucoup moins enthousiastes.

Une personne refuserait catégoriquement l'utilisation d'un tel système, affirmant qu'il s'agirait d'une atteinte à la confidentialité. Les deux autres évoquent la possibilité que le logiciel puisse être utilisé par l'employeur pour "trier" les employés selon leur compétences et par conséquent leur porter préjudice si elles sont jugées par le système ne pas correspondre aux besoins de la compagnie. Ces derniers mettent l'emphase sur l'importance de préserver l'anonymat et sur le fait que cet outil devrait demeurer une aide aux employés, en les aidant à coopérer, plutôt qu'une aide à la gestion du personnel.

Nous sommes conscients des risques qu'entraînerait l'utilisation de notre système dans une entreprise quant à la classification des ressources humaines. Mais il nous paraît évident que ce système ne saurait être utilisé pour "décider de la compétence" des employés qui l'utilise, de la même façon qu'un système expert comme MYCIN [7], par exemple, ne peut être utilisé pour prodiguer des diagnostics à des patients. Ce ne sont que des outils permettant d'aider leurs utilisateurs à mieux évaluer les paramètres d'un environnement complexe. Par conséquent, nous ne croyons pas qu'un employeur responsable pourrait utiliser notre système pour distribuer des postes à ses employés sans qu'il ne tienne compte de sa propre connaissance de ceux-ci ou des opinions même de ceux-ci. White Rabbit et ALICE ne peuvent qu'orienter un employeur vers certaines personnes ou certains documents lorsque celui-ci tente, par exemple, de combler un manque d'expertise à l'intérieur d'un projet, mais pas rendre des décisions. Ce serait à l'employeur de rechercher, parmi les ressources mises en relief par notre système ou non, celles qui apporteraient une aide réelle à la productivité. *Mettre en relief* les compétences et intérêts compatibles afin d'aider la coopération, voilà le rôle du système que nous avons conçu, rien de plus.

Finalement, nous avons demandé aux sujets de notre expérience d'évaluer l'utilité du logiciel. Lorsque nous leur demandons s'ils croient que White Rabbit faciliterait la rencontre de gens intéressants, la moitié répond que oui. Quatre personnes sur

huit utiliseraient occasionnellement le logiciel, deux ne l'utiliseraient pas et deux l'utiliseraient souvent. Ces chiffres montrent un niveau de confiance moyen quant à la capacité du système à proposer des gens intéressants et qu'ils devraient être convaincus avec l'usage. Ce qui est très intéressant toutefois est que six personnes sur huit jugent avoir été classées par le système avec des gens correspondant bien à leurs intérêts suite à la discussion. Ce résultat tend à démontrer que notre objectif est atteint. En effet, cela signifie que notre système est bel et bien capable de découvrir parmi plusieurs personnes celles ou quelques-unes de celles qui sont intéressantes et ce, à travers l'analyse des messages échangés sur une interface de discussion en temps réel. Il ne s'agit pas d'une preuve, bien sûr, mais il s'agit tout de même d'un résultat très encourageant, confirmant que la méthode de catégorisation et la représentation des connaissances fournissent de bons résultats.

4.2 Améliorations possibles

Bien entendu, plusieurs améliorations pourraient être apportées à White Rabbit. La première d'entre elles (qui a été lue maintes fois dans les commentaires recueillis à l'expérience présentée en 4.1.4) est la nécessité d'implanter des fonctions plus avancées de reconnaissance de langue naturelle afin de former les profils avec plus de précisions. Nous savons que ceci constitue la principale lacune de notre système qui est incapable de détecter, lors des conversations, qu'une personne, par exemple, utilise une négation, ou d'évaluer l'importance d'un mot clé en fonction des qualificatifs utilisés, ou encore de découvrir un mot clé lorsque l'usager l'utilise dans un autre genre (masculin plutôt que féminin ou vice versa) que celui se trouvant dans la base de connaissances. Tout ceci nécessiterait l'utilisation d'un dictionnaire complet et de fonctions d'analyse de la langue naturelle. Bien que ceci limiterait l'usage aux langues définies dans les dictionnaires utilisés, les avantages encourus seraient énormes. Par exemple, en utilisant WordNet [11], un dictionnaire doublé d'une base de connaissances syn-

taxique et sémantique très étoffée sur la langue anglaise et disponible gratuitement sur l'Internet, il deviendrait possible pour White Rabbit de faire une analyse très poussée en découvrant de manière automatique de nouveaux liens de généralisation, ou de similarité entre des mots clés ou encore de détecter, par exemple, qu'un mot utilisé par l'usager est une forme conjuguée d'un verbe faisant partie de la base de connaissances. Il s'agirait d'une amélioration majeure.

Une autre possibilité d'amélioration serait de transformer l'application en "applet" Java pour qu'il puisse automatiquement être utilisé de n'importe où, n'importe quand. Le système prendrait alors une envergure plus globale et pourrait éventuellement aider à trouver l'expertise n'importe où sur la planète, mais tel n'est pas l'objectif de White Rabbit. Ceci poserait de nombreuses contraintes de performance et d'esthétique qu'une application nous permet beaucoup mieux d'éviter. On pourrait donc plutôt parler ici de façon plus juste de changement de vocation que d'amélioration possible.

4.3 Autres applications de l'architecture

L'une des particularités de l'architecture utilisée pour la réalisation de notre système est qu'elle est générique et facilement transposable à d'autres domaines. En effet, le domaine d'application du système est principalement défini par le contenu de sa base de connaissances. Changer cette base, par exemple, pour une base contenant des mots clés et des liens couvrant le domaine de l'industrie pétrolière rend le système capable de découvrir les profils compatibles dans ce domaine particulier. Et ce changement de base de connaissances est rendu rapide et facile par l'éditeur de graphe de connaissances fournit par ALICE. Il serait donc simple d'apporter des modifications à White Rabbit pour l'appliquer à une foule d'autres choses que l'amélioration de la coopération chez des employés d'une même entreprise. Cette section fait un survol

de quelques-unes des autres applications possibles et utiles que pourrait avoir notre système.

4.3.1 Utiliser les sites Web

Il serait possible de modifier légèrement notre architecture pour que les profils d'usagers soient construits non pas à partir de la conversation et des fichiers personnels de l'usager, mais plutôt à partir des sites Web qu'il consulte fréquemment. Ceci donnerait un système similaire à Webhound/Webdoggie [19], à la différence près que White Rabbit est (partiellement) décentralisé et qu'il offre à ses utilisateurs des outils de communications interpersonnelles, ce que Webhound ne possède pas.

Construire une telle application est relativement simple. D'une part, il serait possible à l'aide d'un programme externe, de trouver et d'importer toutes les pages web référencées dans la liste de signets ("bookmarks") de l'utilisateur, puis de fournir les fichiers ainsi acquis en entrée à White Rabbit qu'il pourrait alors traiter comme s'il s'agissait de messages envoyés par l'utilisateur. Ce serait la façon la plus simple d'atteindre le nouvel objectif. D'autre part, une méthode plus simple pour l'utilisateur, mais légèrement plus complexe pour le programmeur consisterait plutôt à modifier notre système pour le rendre capable de quérir lui-même les pages web. Mais quoi qu'il en soit, une telle application est sans équivoque une application réaliste de notre système.

4.3.2 Analyser les requêtes à une base de données

Une autre application intéressante de notre architecture serait de faire l'analyse des requêtes soumises à la base de données d'une compagnie par ses employés afin d'en extraire les similarités. En effet, il serait de cette façon possible de trouver quels employés, ou encore quels groupes de travail ou départements s'intéressent aux mêmes choses. Ceci permettrait au gestionnaires de la compagnie de découvrir

les duplications d'efforts et donc ensuite d'optimiser le travail de ses employés en encourageant la collaboration entre les groupes ou les employés partageant des intérêts similaires. Ceci impliquerait seulement de formuler une métrique pouvant calculer la similarité entre deux requêtes. Il faudrait également modifier la représentation des intérêts en donnant par exemple des poids à des éléments de requêtes plutôt qu'à des mots clés, comme c'est le cas dans le système actuel.

4.3.3 Trouver l'âme soeur

Bien que cette application puisse en faire sourire plusieurs, il n'en reste pas moins qu'il s'agit sans doute de celle qui est, et de loin, la plus utilisée pour les systèmes de jumelage. En fait, nous croyons que White Rabbit, tel qu'il est actuellement, pourrait très bien atteindre cet objectif. Il possède du moins tous les outils requis : emmagasinage d'informations personnelles sur les utilisateurs, conservation de la confidentialité d'une partie ou de la totalité de ces informations, environnement de discussion pouvant être anonyme, et, surtout, construction du profil de l'utilisateur en deux volets, intérêts et capacités. Dans ce cas particulier, les intérêts représenteraient le profil de la personne recherchée, et les capacités le profil de la personne qui cherche (comme nous l'avons fait dans l'expérience décrite à la section 4.1.4). Ceci est le point le plus important, car pour faire de tels regroupements, il ne s'agit pas de réunir des personnes ayant des intérêts similaires pour la simple raison qu'une personne peut être attirée plutôt par le contraire que par la similitude. Bref, il s'agit plutôt de trouver des profils complémentaires, ce que White Rabbit peut faire, contrairement à bon nombre de systèmes de jumelage. Par exemple, Yenta est incapable de tenir compte de cet aspect [13]. Il ne jumelle ensemble que les personnes ayant des profils d'intérêts similaires.

D'autre part, les systèmes existants spécialisés dans ce domaine forcent généralement l'utilisateur à remplir préalablement de très longs questionnaires visant à cerner à la

fois leur profil et le profil recherché et ne font aucun apprentissage subséquent des intérêts de leurs usagers. Or, White Rabbit comme nous l'avons déjà expliqué, n'a besoin que de conversation et de réponses à quelques questions, demandant aussi peu d'effort que possible de la part de ses utilisateurs et continue sans arrêt son apprentissage de leurs intérêts. Et certainement, l'utilisation de réseaux de neurones pour faire les regroupements constitue un avantage sur beaucoup de systèmes de jumelage romantique, généralement peu intelligents.

4.3.4 Commerce électronique

Pour une telle application, nous voudrions construire un système capable de jumeler les produits ou les services de vendeurs avec des consommateurs potentiels. Ceci implique tout d'abord de découvrir les "profils" des produits et services offerts, ce qui peut être fait en utilisant une base de mots clés appropriée. Ensuite, tout comme le fait déjà White Rabbit, il suffirait de former les catégories et de présenter les vendeurs aux acheteurs. Une amélioration pourrait cependant être apportée en ajoutant aux agents de communications du systèmes les capacités pour négocier (protocoles, algorithmes de négociation).

4.3.5 Autres applications

On peut facilement imaginer une foule d'autres applications au système que nous avons développé. Par exemple, jumeler des chômeurs avec des entreprises cherchant à employer de nouvelles personnes, jumeler des personnes désirant partir en voyages avec des villes ou des régions du monde, etc. Cette diversité découle de l'aspect générique du problème du jumelage et est très bien représentée par la grande quantité des systèmes déjà existant, comme ceux présentés au chapitre 1 sur les systèmes de jumelage.

4.4 Conclusion

Ce chapitre a fait l'évaluation du système sur plusieurs points, comme la performance, la convivialité et l'efficacité. Il ressort de cette discussion que White Rabbit pourrait s'adapter à un environnement de plus grande envergure et qu'il accomplit le travail souhaité, c'est-à-dire catégoriser efficacement les profils d'intérêts de ses utilisateurs et favoriser la rencontre, ou la communication entre eux. Toutefois, l'utilisation de tel logiciel dans une réelle entreprise pourrait poser des problèmes éthiques par rapport à la catégorisation des personnes en cause.

L'architecture générale et flexible de notre système nous permettrait de l'adapter à de nombreux domaines où il est nécessaire de découvrir des jumelages. Nous pourrions finalement envisager plusieurs améliorations pour l'avenir, en particulier du côté de la compréhension de la langue naturelle par les agents personnels du système.

Conclusion

L'un de nos objectifs était de trouver une solution réaliste à un problème réel. Or, White Rabbit se veut d'abord et avant tout un système applicable à la résolution de problèmes concrets de l'industrie. Il est réalisé dans la plus récente version de Java disponible actuellement et utilise des techniques connues et efficaces de l'intelligence artificielle pour parvenir à cette fin, en particulier une carte de Kohonen, un réseau de neurones non-supervisé. Nous n'avons trouvé aucun système de jumelage utilisant ce type de réseaux de neurones. Les algorithmes de catégorisation utilisés dans ces systèmes sont généralement beaucoup plus simples, comme nous l'avons vu au chapitre 1. White Rabbit évite en plus le problème de confidentialité rencontré entre autres par Yenta en analysant une conversation publique plutôt que des documents personnels des usagers.

L'étude que nous avons réalisée au chapitre 4 tend à démontrer que notre objectif premier qui, rappelons-le, était de favoriser l'établissement de relations entre les employés d'une entreprise afin d'en améliorer la coordination et d'en augmenter le niveau de coopération, a été atteint. Bien sûr, notre système n'a pas été testé dans un environnement réel et il nous est donc impossible d'affirmer hors de tout doute et de prouver formellement que l'objectif a bel et bien été atteint. Toutefois, les résultats obtenus par les tests de performance et sur l'appréciation du système nous permettent d'être positifs.

Le principal problème envisageable proviendrait plus des employés que du système lui-même. En effet, il n'est pas évident que les employés d'une grande compagnie veuillent collaborer. Il existe souvent une très grande compétition au sein d'une même entreprise qui pourrait à elle seule rendre inutile une telle application du système. Mais ceci relève plus de la sociologie que de l'informatique. Nous croyons toutefois que les gens communiquent plus facilement sous le couvert de l'anonymat que leur procure la discussion sur un environnement de *chat*. Le second problème important à surmonter est un problème d'éthique. En effet, selon la culture et les valeurs de chacun, l'utilisation d'un tel système pourrait être perçu comme une ingérence dans sa vie privée, une réduction des individus à un vecteur de nombres, et une porte ouverte à des injustices de la part de l'employeur. Mais comme n'importe quel outil, sa justification dépend de l'usage qui en est fait. White Rabbit doit être vu comme une aide à la coopération des employés, et rien d'autre.

Une force de notre système réside dans son architecture générique. En effet, tout comme Arcadia et Yenta, White Rabbit pourrait, au coût d'efforts minimaux, être modifié pour être appliqué à une foule d'autres domaines. En fait, le domaine d'application de notre système dépend en grande partie de la base de connaissance qui lui est fournie. Or, cette base peut traiter de n'importe quel domaine, pour aussi peu qu'un expert soit en mesure d'inventorier les différents mots clé qui y sont reliés, puis de les réunir dans un graphe sémantique. L'utilisation d'un graphe sémantique constitue d'ailleurs un avantage marqué sur bon nombre de systèmes de jumelage. Aucun des systèmes étudiés dans ce mémoire n'en fait usage.

Par ailleurs, Yenta, le système présentant le plus de similarités avec le nôtre, et aussi peut-être le plus évolué de sa catégorie, est capable de découvrir des profils d'intérêts similaires, mais il est incapable de mettre en relief les profils complémentaires, c'est-à-dire ceux dont les capacités de l'un correspondent aux besoins de l'autre. Or,

notre représentation du profil de l'usager en deux facettes (capacités et intérêts) permet à notre système de le faire sans problème.

Plusieurs extensions à ce travail sont possibles pour en améliorer les résultats. La première de celles-ci serait d'augmenter les capacités d'analyse de la langue naturelle de notre système pour le rendre capable de mieux comprendre les subtilités de la conversation humaine. D'autres travaux pourraient être menés pour standardiser l'environnement de discussion afin que celui-ci rencontre mieux les critères de qualité de ses usagers, habitués au contact de ce genre d'interface. Et finalement, des tests en milieu industriel devraient être menés afin de déterminer si oui ou non, un tel système se révélerait un outil utile et atteindrait l'objectif d'améliorer la gestion des ressources d'une entreprise.

Bibliographie

[1] AÏMEUR, E. *METIS : un système et une méthode d'explicitations de Taxinomies destinées à l'identification de structures conceptuelles.* PhD thesis, Université Paris VI, Paris, France, 1994.

[2] BÉLANGER, S. Optimisation des interactions au sein d'un réseau de connaissances. Master's thesis, Université de Montréal, Montréal, Canada, 2000.

[3] BÉLANGER, S., THIBODEAU, M. A., FRASSON, C., AND AÏMEUR, E. Vai connection, expertise managing application for a company. In *IASTED, Artificial Intelligence and Applications* (Innsbruck, Austria, 2001). to appear.

[4] BERMUDEZ, J. L., PIRAS, A., AND RUBINSTEIN, M. Classification of lightning electromagnetic waveforms with a self-organizing kohonen map. In *Proceedings of the 13th International Wroclaw Symposium on Electromagnetic Compatibility* (Poland, june 1996), pp. 517–521.

[5] BIEBER, P., BROUCHOUD, H., CAMPS, V., CARPUAT, B., CROS, P., GLEIZES, M. P., GLIZE, P., LÉGER, A., MACHONIN, A., MALVILLE, E., PEZET, J., AND THOMAS, H. Arcadia : An architecture for cooperating information access agents. In *INET 98, Internet Society Symposium* (1998). http ://www.cert.fr/francais/deri/cros/Cros/Papers/INET98.ps.

[6] BIGUS, J. P., AND BIGUS, J. P. *Constructing Intelligent Agents Using Java.* John Wiley & Sons, Inc., 1998.

[7] BUCHANAN, B. G., AND SHORTLIFF, E. H. *Rule Based Expert Systems : The Mycin Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, 1984.

[8] DEBOECK, G., AND KOHONEN, T. *Visual Explorations in Finance.* Springer-Verlag, 1998.

[9] DYKE, N. W. V., LIEBERMAN, H., AND MAES, P. Butterfly : A conversation-finding agent for internet relay chat. In *Proceedings of the 1999 International Conference on Intelligent User Interfaces* (Redondo Beach, CA, USA, january 1999), pp. 39–41.

[10] website. http ://press.uoregon.edu/unixsamp/oracle/classif.htm.

[11] FELLBAUM, C. *WordNet : An Electronic Lexical Database.* MIT Press, 1998.

[12] FONER, L. N. Yenta : A multi-agent, refferal-based matchmaking system. In *Proceedings of the First International Conference on Autonomous Agents (Agents '97)* (Marina del Rey, CA, USA, february 1997), ACM Press, pp. 301–307.

[13] FONER, L. N. *Political Artifacts Personal Privacy : The Yenta Multi-Agent Distributed Matchmaking System.* PhD thesis, Massachusetts Institute of Technology, 1999.

[14] FONER, L. N., AND CRABTREE, I. B. Multi-agent matchmaking. In *Software Agents and Soft Computing. Toward enhancing machine intelligence. Concepts and applications* (Berlin, Germany, 1997), Springer-Verlag, pp. 100–115.

[15] Harvest web indexing. website. http ://www.tardis.ed.ac.uk/harvest/index.html.

[16] KUOKKA, D., AND HARADA, L. Matchmaking for information agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) '95* (Montréal, Qué, Canada, August 1995), Morgan Kauffman, pp. 672–678.

[17] LUGER, G. F., AND STUBBLEFIELD, W. A. *Artificial Intelligence : Structures and Strategies for Complex Problem Solving.* Benjamin/Cummings Publishing Company, Inc., 1997.

[18] MAES, P. Agents that reduce work and information overload. *Communications of the ACM 37*, 7 (1994), 30–40.

[19] MAES, P., LASHKARI, Y., AND METRAL, M. *Readings in Agents.* Morgan Kuffman Publishers, Inc., San Francisco, CA, USA, 1997, ch. Collaborative Interface Agents.

[20] MCKELVEY, B., MINTZBERG, H., PETZINGER, T., PRUSAK, L., SENGE, P., AND SHULTZ, R. The gurus speak : Complexity and organizations. *Emergence : A Journal of Complexity Issues in Organizations and Management 1*, 1 (1999), 73–91.

[21] MUSIL, M., AND PLESINGER, A. Discrimination between local microearthquakes and quarry blasts by multi-layer perceptrons and kohonen maps. *Bulletin of the Seismological Society of America 86*, 4 (1996), 151–160.

[22] Planetall. website. http ://www.planetall.com.

[23] QUINLAN, J. R. Induction of decision trees. In *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich, Eds. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1 :81–106, 1986.

[24] RYS, O. Software agents in the internet. In *Proceedings of the HUT Internetworking Seminar* (Helsinki, Finland, 1997), Helsinki University of Technology. http ://www.tml.hut.fi/Opinnot/Tik-110.551/1997/inwpohja.html.

[25] SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal 27* (july 1948), 379–423,623–656.

[26] SHARDANAND, U., AND MAES, P. Social information filtering : Algorithms for automating word of mouth. In *Proceedings of the CHI'95 Conference* (Denver, CO, USA, 1995), ACM Press, pp. 210–217.

[27] SILIPO, N., BORTOLAN, G., AND MARCHESI, C. Design of hybrid architectures based on neural classifier and rbf pre-processing for ecg analysis. *International Journal of Approximate Reasoning 21*, 2 (1999), 177–196.

[28] Sixdegrees. website. http ://www.sixdegrees.com.

[29] STANDISH, T. A. *Data Structures, Algorithms, and Software Principles.* Addison-Wesley Publishing Company, Reading, USA, 1994.

[30] SYCARA, K., LU, J., KLUSCH, M., AND WIDOFF, S. Dynamic service matchmaking among agents in open information environments. *Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems 28*, 1 (1999), 47–53.

[31] THIBODEAU, M. A., BÉLANGER, S., AND FRASSON, C. White rabbit - matchmaking of user profiles based on discussion analysis using intelligent agents. In *Intelligent Tutoring Systems* (Montréal, Qué, Canada, 2000), G. Gauthier, C. Frasson, and K. VanLehn, Eds., Springer, pp. 113–122.

[32] WOLRATH, A., AND WALDO, J. The java tutorial - rmi. website, october 2000. http ://java.sun.com/docs/books/tutorial/rmi/index.html.