Université de Montréal

HELP : Localisation et recommandation d'experts pour le développement d'un système d'aide collaborative

par

Anita Saleman

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 Maîtrise ès Sciences
en Informatique

décembre, 2005

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

HELP : localisation et recommandation d'experts pour le développement d'un système d'aide collaborative

Présenté par :

Anita Saleman

a été évalué par un jury composé des personnes suivantes :

Guy Lapalme, président-rapporteur

Esma Aïmeur, directrice de recherche

Douglas Eck, membre du jury

Mémoire accepté le 7 février 2006

Résumé

Pour résoudre un problème donné, il est nécessaire d'avoir recours à l'expertise appropriée. De nos jours, le développement de systèmes assistant les usagers à rechercher des experts dans un domaine précis fait l'objet de nombreux projets de recherche. Nous présentons ici HELP, un système qui permet de localiser et recommander les experts en vue du développement d'une aide collaborative. Il s'agit d'un environnement dans lequel chaque usager a accès à l'expertise d'autres usagers, en fonction de ses besoins. Nous considérons que chaque usager a des connaissances qui font de lui un expert dans un domaine, mais il peut également avoir peu ou pas de notions dans un autre domaine, ce qui fait alors de lui un apprenant. HELP permet à l'utilisateur de trouver rapidement une réponse à sa question dans un domaine précis. Pour cela, deux approches sont utilisées. La première consiste à rechercher s'il existe dans le système une requête similaire à la requête courante de l'utilisateur. Dans ce cas, nous utilisons le raisonnement à base de cas textuels. La deuxième approche vise à localiser les experts dans des domaines particuliers et de les mettre en contact avec des usagers qui sollicitent cette expertise. Dans cette optique, nous avons axé notre système sur la recommandation d'experts : lorsqu'un usager est à la recherche d'informations, HELP lui recommande des experts pertinents. Pour la recommandation d'experts, nous utilisons une technique hybride basée sur le filtrage collaboratif et le raisonnement à base de cas. Enfin, nous pensons que dans un contexte académique, l'aspect pédagogique joue un rôle important dans la recherche d'experts pertinents. Cet aspect est donc particulièrement étudié.

Mots-clés : recommandation d'experts, localisation d'experts, aspect pédagogique, filtrage collaboratif, raisonnement à base de cas, aide collaborative.

Abstract

To solve a given problem, we have to look for the appropriate expertise. Nowadays, the development of systems assisting users to locate experts who can help them has become a principal subject of research. In this thesis, we present a system called HELP, whose aim is to locate and recommend experts for the development of a collaborative aid. In this environment, each user can access the expertise of other users according to their needs. A user who has knowledge in one area is regarded as an expert, but he can also be regarded as a learner on a specific topic of another area if he has few or no concept in this topic. HELP allows the user to find quickly an answer to his question. For this we use two approaches. The first one consists in making the system retrieve one or more requests similar to the seeking-user's request. The method used in this case is the textual case-based reasoning technique. The second approach aims at locating the experts in specific areas in order to put them in contact with the user information-seeking. For this second approach we used an expert recommendation system: when a user needs some information, the system recommends a list of relevant experts to contact. The technique used for the recommendation system is a hybrid technique based on collaborative filtering and case-based reasoning. Lastly, we think that in an academic context, the pedagogical aspect plays a significant role in the search of relevant experts. This aspect is thus particularly studied in this thesis.

Keywords: expert recommendation, expert localization, pedagogical aspect, collaborative filtering, case-based reasoning, collaborative help.

Table des matières

Chapitre 1. Le Raisonnement à Base de Cas 8
1.1 Principes fondamentaux d'un système CBR 8
1.1.1. Conception du raisonnement à base de cas 8
1.1.2. Le cycle CBR 9
1.1.3. Fondement du raisonnement à base de cas 11
1.1.4. Qu'ést-ce qu'un cas ? 12
1.1.5. La Phase de Recherche 14
1.1.6. La phase d'adaptation 18
1.1.7. La phase de Révision 20
1.1.8. La phase de Maintenance 20
1.2. Les modèles CBR 20
1.2.1. Le modèle structurel 20
1.2.2. Le modèle conversationnel 21
1.2.3. Le modèle textuel 22
1.3. Le CBR textuel 23
1.3.1. Modèle en couches (Mario Lenz) 23
1.3.2. WordNet [Miller 95] 26
1.3.3. Conclusion 27
Chapitre 2 : La Recherche d'Informations et les systèmes de questions-réponses 29
2.1. La Recherche d'Informations 29
2.1.1. Le modèle booléen 29
2.1.2. Le modèle vectoriel 30
2.1.3. Le modèle probabiliste 34
2.1.4. Lemmatisation et algorithme de Porter [Porter 80] 35
2.2. Les systèmes de questions-réponses 37
2.2.1. FallQ [Lenz et Burkhard 97] 38
2.2.2. FAQ-Finder [Burke *et al.* 97] 40

2.3. Conclusion 42
Chapitre 3 : La localisation d'experts 43
3.1. Qu'est-ce qu'un expert ? 43
3.2. La localisation d'experts 44
3.2.1. Qu'est-ce qui provoque la recherche d'experts ? 44
3.2.2. Principes d'un système de localisation d'expertise 48
3.3. Les systèmes de localisation d'experts 52
3.3.1 Chicago Information Exchange [Kulyukin 98] 52
3.3.2 Expertise Finder [Crowder *et al.* 03] 57
3.3.3 Système Q&A [Budzik et Hammond 99] 60
3.3.4 « Support Routing » [Aberg et Shahmehri 01] 61
3.3.5 Expert Finder [Vivacqua et Lieberman 00] 62
3.3.6 Conclusion 63
Chapitre 4 : Les systèmes de recommandation 64
4.1. Introduction 64
4.2. Les techniques de recommandation 65
4.2.1. Le filtrage collaboratif 66
4.2.2. Le filtrage basé sur le contenu 71
4.2.3. Technique basée sur le raisonnement à base de cas 71
4.2.4. Le filtrage basé sur les données démographiques 72
4.2.5. Le filtrage basé sur l'utilité 72
4.2.6. Le filtrage basé sur « les connaissances » 72
4.3. Les systèmes hybrides 73
4.3.1. Pondération *(Weighted)* 73
4.3.2. Technique à commutation (*Switching*) 73
4.3.3. Technique mixte 73
4.3.4. Combinaison de caractéristiques 74
4.3.5. Cascade 74

Chapitre 5 : Conception et méthodologie 75
5.1. Notre approche 75
5.1.1. Objectifs 75
5.1.2. Fonctionnement global de HELP 76
5.1.3. Fonctions principales accessibles aux utilisateurs 77
5.1.4. Aspect pédagogique 78
5.1.5. Application de la méthodologie dans HELP 82
5.2. Architecture de *HELP* 83
5.2.1. Fonctionnalités du système 83
5.2.2. Environnement de l'utilisateur 86
5.3. Structuration du domaine d'application 88
5.4. Base de cas textuels 89
5.5. Module de traitement des questions 90
5.6. Module de gestion de la base de cas 95
5.7. Base des profils utilisateurs 96
5.7.1. Les données démographiques 97
5.7.2. Les données relatives à l'expertise 98
5.7.3. Les bases d'expertise et d'apprentissage 98
5.7.4. Préférences sur les critères de recherche 99
5.7.5. Liste des experts évalués par l'utilisateur 99
5.8. Recommandations d'experts 100
5.8.1. Recommandation basée sur le filtrage collaboratif 101
5.8.2. Recommandation basée sur le raisonnement à base de cas 105
5.9. Conclusion 107
Chapitre 6 : Implémentation 115
6.1. Implémentation de HELP 115
6.1.1. Les Servlets et les pages JSP (*Java Server Pages*) 115
6.2. Domaine d'application 117

6.3. Environnement d'un utilisateur 118
6.3.1. Enregistrement d'un utilisateur 118
6.3.2. Gestion de la messagerie 121
6.4. Validation du système 123
6.4.1. Rappels 123
6.4.2. Utilisation générale du système 124
6.4.3. Consultation des cas similaires 125
6.4.4. Analyse des choix des apprenants 128
6.4.5. Analyse des votes 131
6.4.6. Conclusion 134
Conclusion 136
Discussion 137
Travaux futurs 138

Liste des tableaux

Tableau 1 : Les différentes techniques de recommandation [Burke 02] 66
Tableau 2 : Nombre d'occurrences de chaque index dans les requêtes 1, 2 et 3 92
Tableau 3 : Vecteurs de termes pondérés des 3 requêtes 93
Tableau 4 : Vecteur de mots-clés pondérés de la nouvelle requête 95
Tableau 5 : Matrice des votes donnés par les utilisateurs aux experts 102
Tableau 6 : Votes donnés à Alice sur le concept Installing 106
Tableau 7 : Comparaison de systèmes 108
Tableau 8 : nombre de requêtes envoyées par catégorie d'utilisateurs 125
Tableau 9 : cas similaires trouvés et retenus 126
Tableau 10 : choix des apprenants par rapport aux recommandations du système 129
Tableau 11: analyse des votes 132

Liste des figures

Figure 1 : Concept du raisonnement à base de cas [Leake 96] 9
Figure 2 : Le cycle du processus CBR proposé par Aamodt et Plaza 10
Figure 3 : Partie du sous-réseau Nom de WordNet 27
Figure 4 : représentation schématique du *Model Space Vector* 32
Figure 5 : Exemple de l'organisation du domaine dans CIE 55
Figure 6 : Réception d'une requête et réponse de l'expert 56
Figure 7 : Aperçu de la problématique et concepts de Expertise Finder [Crowder *et al.* 03] 58
Figure 8 : Évaluation d'un expert 80
Figure 9 : Critères de recherche d'un expert 81
Figure 10 : Architecture de *HELP* 85
Figure 11 : Environnement d'un utilisateur de *HELP* 87
Figure 12 : Représentation hiérarchique du domaine d'application 89
Figure 13 : composantes du profil utilisateur 97
Figure 14 : Illustration d'une partie du domaine d'application 118
Figure 15 : Enregistrement de l'expertise d'un utilisateur 119
Figure 16 : Environnement de l'utilisateur 120
Figure 17 : Envoi d'une requête 121
Figure 18 : Recherche de cas similaires 122
Figure 19 : Recommandation d'experts 123
Figure 20 : Cas similaires trouvés pour chaque catégorie d'experts 126
Figure 21 : requêtes pour lesquelles un type d'expert est retenu pour la raison *i* 130
Figure 22 : Analyse de la validation des réponses 133

Remerciements

Je voudrais en premier lieu adresser mes sincères remerciements à ma directrice de recherche, Madame Esma Aïmeur, pour m'avoir dirigée et encadrée tout au long de ce mémoire. Je la remercie vivement de m'avoir prodigué de précieux conseils et de m'avoir toujours encouragée et soutenue.

Je tiens aussi à remercier Serge Mani Onana pour sa disponibilité, son aide et ses suggestions d'amélioration. Je remercie également Sébastien Gambs pour son aide à la relecture du mémoire.

Enfin, je remercie tous les autres membres du laboratoire Héron qui m'ont apporté leur soutien et leurs encouragements, dont Kamal Bakour, Narimel Bendakir, Soumaya Chaffar, Hicham Hage, Moez Mabrouk, Arnoldo Rodriguez, Kamal Yammine.

J'adresse également mes remerciements aux membres du jury qui ont acceptés d'être rapporteurs de mon mémoire.

La réalisation de cette recherche a fait partie du projet DIVA (Développement, Intégration et éValuation des technologies de formation d'Apprentissage) et a été rendue possible grâce à la participation financière de Valorisation-Recherche-Québec (VRQ).

Introduction

L'information. Ce concept vaste est devenu aujourd'hui l'un des principaux centres d'intérêts dans de nombreux domaines de recherche, l'objectif étant généralement toujours le même : comment obtenir l'information *utile*. De nos jours, obtenir dans de meilleurs délais l'information adéquate à une préoccupation donnée constitue un véritable défi. Pourtant, un tel défi doit être relevé constamment par de nombreuses entités à l'instar des entreprises, universités ou autres groupes d'individus. En effet, l'information est un enjeu important pour le développement de tout type d'entité. Dans notre cas, l'information est considérée comme étant simplement une connaissance.

Afin de mieux introduire notre problématique, nous allons d'abord parler de quelques concepts décrits dans les études de la gestion des connaissances (*Knowledge Management*). Nous présenterons ensuite nos objectifs et notre approche.

La gestion des connaissances (*Knowledge Management*)

De plus en plus d'organisations ont réalisé que la « connaissance » constitue une ressource stratégique [Na Ubon et Kimble 02]. Ces organisations, qu'il s'agisse de compagnies, de fournisseurs de service, d'universités, ont une masse importante d'informations et de connaissances circulant entre leurs membres. Les recherches en gestion des connaissances ont permis de développer des approches en vue de manipuler et d'organiser l'information sur ces connaissances, afin de la rendre accessibles par tous les membres d'une communauté [Dieng 00].

Les études portant sur la gestion des connaissances ont permis d'identifier deux types de connaissances [Skyrme 98] :

- Les connaissances **explicites**, qui peuvent être codifiées sous une forme tangible (rapports, documents techniques, etc.)
- Les connaissances dites **tacites** qui sont détenues par les individus. Celles-ci sont souvent des pratiques acquises par l'expérience ou lors d'un transfert

de connaissances issu d'une interaction face-à-face (selon le schéma classique du maître-apprenti).

Bien que les travaux dans le domaine de la gestion des connaissances se soient beaucoup focalisés sur le développement de systèmes permettant de gérer les connaissances explicites, une grande attention est désormais portée sur l'exploitation des connaissances tacites.

Dans tous les domaines, avec l'utilisation croissante d'Internet, les utilisateurs se retrouvent immergés dans une masse d'informations, bien connue sous le nom de « surcharge d'informations », et ne sont plus en mesure de trouver l'information appropriée dans un temps limité. Plusieurs systèmes, notamment les systèmes de recommandation [Burke 02], tentent de résoudre ce problème en allégeant la quantité d'informations proposée à l'utilisateur selon son profil : ses goûts, ses préférences, ses choix passés, etc.. On parle alors de personnalisation, ou encore d'adaptation à l'utilisateur.

Notre approche

Dans notre approche, nous nous intéressons uniquement aux connaissances *tacites*. L'objectif de notre système, HELP, est de répondre aux besoins d'aide et plus spécifiquement d'expertise d'une personne travaillant dans une organisation quelconque. En effet, il est fréquent pour les membres d'une organisation de se retrouver confrontés à des problèmes qu'ils n'arrivent pas à résoudre rapidement par manque de connaissances appropriées. Ces connaissances sont souvent non connues ou non accessibles bien qu'elles existent dans l'organisation [Skyrme 98]. Notre approche consiste donc à fournir un environnement d'aide collaborative, tant au niveau industriel qu'académique, où chaque apprenant peut avoir accès à l'expertise d'autres apprenants en fonction de ses besoins sur un domaine bien défini.

Imaginons une société informatique dont l'objectif est de vendre des services (Ex. : analyse des systèmes existants ou *consulting*) ou encore des produits (Ex. : applications informatiques) pour divers clients. La société est relativement grande et est divisée en plusieurs équipes.

Alice travaille depuis plusieurs années dans la société et occupe maintenant le poste de chef de projet, après avoir passé quelques temps dans le développement d'applications. Elle connaît divers langages de programmation : C, C++, Visual Basic, Java etc. Elle est actuellement en charge de la réalisation d'un projet important : il s'agit de simuler des jeux de données valides qui serviront par la suite de données d'entrée à divers systèmes satellitaires. Alice réunit une équipe dont les compétences techniques solides lui permettront de mener à bien son projet. Le langage de développement choisi est Java. Éric, Pascal et Claire doivent réaliser la conception et développement de la partie concernant le stockage des informations, l'accès et l'enregistrement des données utilisées par le simulateur. Ils choisissent XML, permettant de stocker des informations dans des fichiers au format bien spécifique. Ils se retrouvent bien vite confrontés à un problème : l'enregistrement des données sous fichiers XML n'est pas réalisé correctement. Les technologies mises en œuvre sont la sérialisation, l'utilisation d'XTREAM, des beans Java, la persistance des objets. Bien qu'ayant chacun des connaissances pointues en Java, ce lien entre les objets Java et leur sauvegarde sous fichiers XML leur paraît relativement complexe. Éric va donc chercher sur Internet ou dans d'autres sources d'informations la réponse à ce problème. Malgré les informations théoriques et pratiques trouvées, rien de s'applique spécifiquement à son problème. Chacun cherche alors autour de lui, parmi les membres de son équipe, si une personne a déjà connu ce type de problème afin de profiter de son expérience. Cette attitude est une façon rapide et généralement efficace pour résoudre un problème qui a déjà été résolu par une tierce personne. Cependant la recherche d'une tierce personne est souvent limitée au réseau social de la personne dans le besoin. Le réseau social d'Éric se compose des membres de son équipe et d'autres personnes qu'il connaît. Chacune des personnes interrogées a également son propre réseau de

connaissances mais il se pourrait qu'elles ne soient pas en mesure de l'aider. Ses recherches lui font d'une part perdre beaucoup de temps, et d'autre part, il peut se retrouver avec une solution qui ne remplit pas toutes les spécifications du projet. Pascal réussit cependant à trouver deux personnes qui lui parlent d'une certaine Christine, qui a déjà travaillé sur un projet ayant des technologies similaires, notamment sur la sérialisation des objets Java. Pascal recherche à contacter Christine mais elle ne travaille plus dans la société. Il tente alors de retrouver les membres de l'équipe de Christine d'alors, et réussit à trouver trois personnes, mais aucune ne se souvient précisément des solutions à ces mêmes problèmes qu'ils avaient rencontrés alors. Cela s'explique par le fait que seule Christine était alors en charge de la réalisation de l'enregistrement des données.

Que nous dit cette première partie d'exemple ? A un certain moment dans le temps, une personne au moins a rencontré le même problème que l'équipe actuelle, cependant le nom même de cette personne a été relativement long et difficile à trouver. De plus, cette personne ne travaille plus pour la société.

Reprenons ce même scénario. Cette fois-ci, chacun des membres de l'organisation a un accès à un logiciel interne qui est le système HELP. Éric se connecte au système HELP, écrit sa question en choisissant un certain nombre de mots-clés, suivis d'une description plus précise de son problème et l'envoie au système. HELP analyse sa question et recherche si des problèmes similaires déjà résolus sont présents dans sa banque de données. Si tel est le cas, Éric peut alors consulter les solutions qui ont été données. S'il n'est pas satisfait par les réponses reçues ou s'il n'existe aucune question similaire, HELP utilise simultanément le raisonnement à base de cas et le filtrage collaboratif (deux techniques de recommandation que nous détaillerons dans les chapitres 1 et 4) pour lui recommander des personnes à contacter et susceptibles de résoudre son problème. Dans ce cas, parmi toutes les propositions qui lui sont faites, il aura celle de Christine, mais également d'autres personnes que les membres de l'équipe n'ont pas pu « atteindre » grâce à leur réseau social. HELP leur permet ainsi de connaître les personnes susceptibles de les aider au sein même de l'organisme.

Cet exemple illustre l'un des problèmes posés par la recherche d'accès et de localisation de l'expertise des personnes d'une organisation. En effet, il n'est pas toujours facile de connaître les personnes qui sont en mesure de nous aider, ni où aller chercher l'information. Nous avons élaboré une approche permettant à un membre d'une organisation d'avoir accès, par l'intermédiaire du réseau Internet, à un groupe d'experts susceptibles de l'aider dans la résolution de son problème. Par ailleurs, un tel membre aura aussi à fournir son expertise lorsque requise par d'autres utilisateurs.

Notre objectif se heurte à certains problèmes, soulignés dans les études faites dans la gestion des connaissances, et qu'il est important de prendre en compte :

- Tout d'abord, un expert peut ne pas être intéressé de répondre aux questions, c'est ce que Budzik et Hammond [Budzik et Hammond 99] ont appelé le *conflit d'intérêt entre novices et experts* : le but du *novice* d'avoir de façon constante et permanente l'accès à l'information est en conflit direct avec le but de l'expert qui est d'utiliser son temps pour d'autres objectifs. Ce conflit peut être l'une des raisons du désintérêt de l'expert. Dans notre cas, cette raison nous parait essentielle : en effet, si un expert reçoit plusieurs fois la même question, il n'aura pas la motivation d'y répondre à chaque fois [Kulyukin 98].

- L'autre problème est relatif à la volatilité de l'expertise. En effet, un expert indisponible, ou qui quitte l'organisation, disparaît ainsi avec son expertise. Dans ce cas, comment faire pour que son expertise soit toujours disponible?

La prise en compte des deux problèmes évoqués ci-dessus a permis le développement de systèmes dont nous parlerons dans l'état de l'art (chapitres 2, 3 et 4). Ils ont également servi de base à l'élaboration de notre approche.

Pour apporter des éléments de réponse à ces problèmes, nous avons choisi d'utiliser la technique de raisonnement à base de cas. Cette technique permet de retrouver et réutiliser les solutions données à des problèmes similaires au problème courant [Aamodt et Plaza 94] [Kolodner et Leake 96] [Kolodner 93] [Leake 96]. Ainsi, les utilisateurs du système ayant des questions similaires à celles posées précédemment ont accès directement aux réponses qui ont déjà été données. L'expert est donc assuré de ne pas avoir à répondre plusieurs fois à des questions similaires et de ne pas perdre son temps. De plus, l'approche de questions-réponses est une manière de capturer les connaissances d'un expert de façon moins formelle, et donc moins contraignante. Si aucune question similaire n'a été trouvée, l'utilisateur a la possibilité de converser directement ou indirectement avec un expert choisi dans la liste d'experts recommandés par le système.

Concrètement, notre système permet de rechercher les personnes ayant de l'expertise dans un domaine particulier. Nous appellerons ces personnes des « experts ». Dans HELP, chaque utilisateur peut être vu à la fois comme un apprenant et un expert. Un apprenant, ou *demandeur*, est une personne recherchant une expertise ou une information spécifique. Un expert, ou *fournisseur*, donne quant à lui la solution à un problème formulé par l'apprenant. HELP permet de recommander des experts à des apprenants à la recherche de l'information ou de l'aide dans un domaine précis. Il permet donc non seulement le partage et la réutilisation des connaissances, mais aussi la mise en relation de personnes travaillant dans une même organisation. En d'autres termes, il s'agit de la création d'un environnement d'assistance collaborative. HELP peut donc être défini comme étant un système de recommandation d'experts et de gestion d'expertise implicite.

Ce mémoire est organisé de la façon suivante : dans le premier chapitre, nous présentons la technique de raisonnement à base de cas, et plus particulièrement le raisonnement à base de cas textuel, que nous utilisons dans le développement de notre application. Dans un deuxième chapitre nous ferons un état de l'art sur les systèmes basés

sur la recherche de questions similaires et la localisation d'experts. Nous décrirons ensuite l'architecture du système et les différents modules qui le composent. Enfin, nous terminerons par l'implémentation et la validation de notre système.

Chapitre 1. Le Raisonnement à Base de Cas

Le raisonnement à base de cas (*case-based reasoning* ou CBR [Aamodt et Plaza 94] [Kolodner et Leake 96] [Kolodner 93] [Watson 97]) est une technique de l'intelligence artificielle qui s'inspire du comportement humain lors de la résolution de certains problèmes dans notre vie courante. Nous essayons toujours de nous remémorer les expériences passées similaires au problème courant pour tenter de le résoudre. Dans ce chapitre, nous allons décrire les principales propriétés d'un système utilisant le raisonnement à base de cas, puis nous détaillerons l'un des modèles du CBR que nous utilisons dans notre système : le raisonnement à base de cas textuel.

1.1 Principes fondamentaux d'un système CBR

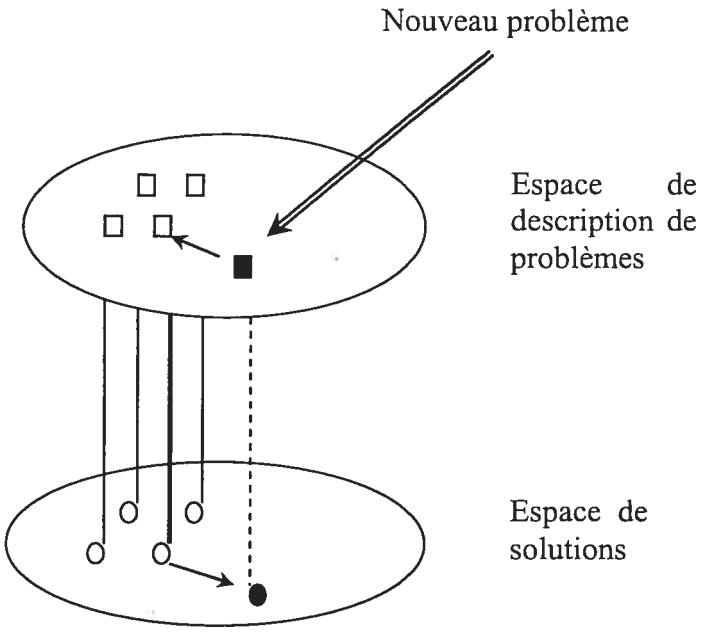
Le raisonnement à base de cas est une forme de raisonnement analogique. Cette section en présente les concepts associés et les types de connaissances à considérer lors du développement d'un système fondé sur les cas.

1.1.1. Conception du raisonnement à base de cas

A case-based reasoner solved new problems by adapting solutions that were used to solve old problems.

Riesbeck and Schank, 1989

La technique de raisonnement à base de cas peut être comprise d'un point de vue très simple : dans le but de trouver une solution à un problème courant, il suffit de chercher un problème similaire dans une base d'expériences, de prendre la solution du problème passé et de l'utiliser en tant que point de départ pour trouver la solution au problème actuel. La figure 1 décrit la démarche utilisée pour appliquer cette technique [Leake 96].



■ = Description d'un nouveau problème à résoudre
□ = Descriptions des problèmes précédemment résolus
O = Solutions précédentes
● = Nouvelles solutions résultant de l'adaptation des anciennes solutions

Figure 1 : Concept du raisonnement à base de cas [Leake 96]

1.1.2. Le cycle CBR

Aamodt et Plaza (1994) ont définit le CBR comme étant un processus cyclique composé des phases principales suivantes :

- La recherche (« Retrieve ») : cette phase permet de rechercher les cas de la base les plus similaires au nouveau problème à résoudre.

- L'adaptation (« Reuse ») : cette phase consiste à reprendre la solution du cas sélectionné dans la phase précédente, et de l'adapter si nécessaire au nouveau problème à résoudre. On obtient une solution possible pour résoudre le problème.

- La révision (« Revise ») : la solution proposée est testée dans l'environnement du problème initial et évaluée. Il s'agit de vérifier si la solution proposée peut résoudre le problème initial ou si cela nécessite de nouvelles modifications.

- La maintenance (« Retain ») : cette nouvelle expérience est ensuite mémorisée dans la base de cas en vue d'être utilisée dans de futures résolutions de problèmes.

La figure 2 ci-dessous représente les quatre phases que nous venons de décrire :

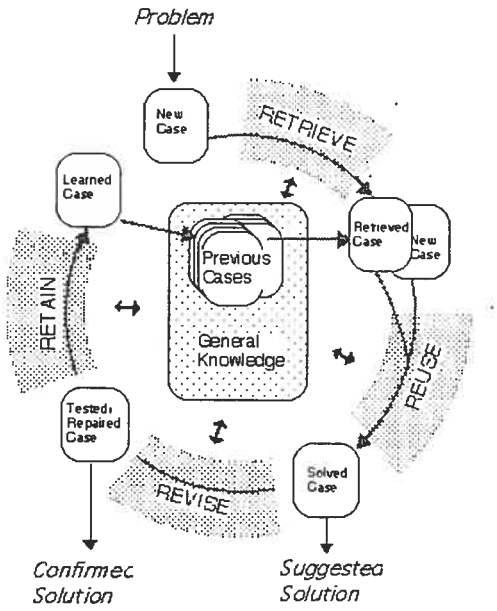


Figure 2 : Le cycle du processus CBR proposé par Aamodt et Plaza

La section suivante donne un bref historique du fondement du raisonnement à base de cas.

1.1.3. Fondement du raisonnement à base de cas

Les origines du raisonnement à base de cas remontent aux travaux de Schank et Abelson en 1977 [Schank et Abelson 77] [Riesbeck et Schank 89]. Leurs travaux (en particulier, leurs travaux sur la *compréhension du texte*) ont permis de fonder la *théorie de la mémoire dynamique* [Schank 82]. Cette théorie se base sur le phénomène de la compréhension et sur le phénomène de rappel. Schank définit le phénomène de rappel comme faire l'analogie d'une situation présente avec une situation déjà vécue et qui ont des caractéristiques ou des informations communes. Sa théorie de la mémoire dynamique s'appuie sur l'organisation de la mémoire humaine : nos processus cognitifs sont enregistrés dans nos mémoires sous une même structure, les *memory organizations packets* (MOPs) [Schank 82]. Les MOPs sont représentés sous forme de *scripts*, qui sont des structures permettant de décrire des évènements typiques que nous avons vécus, tels que « aller au restaurant ».

En 1983, Janet Kolodner a développé le premier système CBR appelé CYRUS à l'Université de Yale (groupe de Schank). CYRUS se basait sur la théorie de Schank et sur les MOPs pour la résolution de problèmes. CYRUS contenait les connaissances, représentées sous forme de cas, des voyages et des réunions de l'ancien Secrétaire de Etats-Unis, Cyrus Vance. Le système permettait aux utilisateurs de poser des questions sur ces événements. Le modèle de mémoire de cas utilisé dans ce système a servi ensuite dans plusieurs systèmes de raisonnement à base de cas [Aamodt et Plaza 94], comme MEDIATOR (1985), PERDUADER (1988), CHEF (1989), JULIA (1992), CASEY (1989).

A partir des années 1990, le CBR a largement été étudié et utilisé dans des applications commerciales principalement, pour la résolution de problèmes (diagnostic, planification, design), les systèmes d'aide à la décision, les « help desk » et la gestion des connaissances [Lamontagne et Lapalme 02].

Nous allons nous intéresser maintenant aux différentes connaissances à définir lorsque l'on souhaite développer un système CBR, puis nous décrirons plus en détails chaque phase du processus cyclique.

1.1.4. Qu'est-ce qu'un cas ?

Avant de détailler les différentes phases du cycle CBR vues en section 1.1.2, nous allons définir ce qu'est un cas.

Définition du concept « cas »

Dans la terminologie CBR, un cas peut être considéré comme étant une situation de problème, une expérience déjà passée qui a été résolue, et que l'on a gardé en vue de le réutiliser pour résoudre un nouveau problème. Il contient le problème passé et le contexte dans lequel ce problème s'est présenté [Leake 96].

Un cas se compose donc en deux parties qui sont la description du cas et la solution associée. Le cas est décrit par certaines caractéristiques appelées ***indices*** ou ***index***. Ce sont les index qui vont permettre de représenter et d'identifier un cas lors des différentes phases du cycle CBR.

Lorsqu'un nouveau problème se présente, il est transformé en *cas cible* dont la partie description du cas utilise le même formalisme de représentation que celui utilisé pour les *cas sources*, cas représentant les expériences passées et stockées dans une mémoire appelée *base de cas*. La partie solution du cas cible est construite en recherchant parmi les cas sources ceux dont la description du cas se rapproche le plus de la description du cas

cible. Cette similarité est calculée en fonction d'une heuristique, la plus courante est celle appelée la méthode des plus proches voisins. La solution du cas source jugé le plus similaire est adapté si nécessaire au contexte du nouveau problème et est ensuite évaluée. Si la solution convient, la base de cas est mise à jour par ajout de la nouvelle expérience sous forme de cas.

Le processus de construction d'une base de cas initiale nécessite le plus souvent une intervention manuelle car plusieurs points importants et difficilement automatisables sont à prendre en compte [Lamontagne et Lapalme 02] : connaissances du fonctionnement du domaine applicatif, structuration initiale de la base de cas, détermination des différentes structures de connaissances (*knowledge containers*).

Les structures de connaissances (*knowledge containers*)

En 1995, M. Richter introduit un modèle explicitant les connaissances nécessaires pour développer un système CBR [Richter 95]:

- *La base de cas* : constituée par l'ensemble des cas d'expériences.
- *Le vocabulaire d'indexation* : la définition d'un vocabulaire d'indexation est essentielle pour structurer les cas et faciliter la phase de recherche et de maintenance dans le cycle CBR. Le vocabulaire d'indexation contient l'ensemble des caractéristiques choisies pour identifier un cas. Ces caractéristiques sont appelées les *index* ou *indices*.
- *La mesure de similarité* : La construction d'une mesure de similarité adéquate est également importante pour la recherche des cas les plus pertinents. Cette mesure est définie en fonction du vocabulaire d'indexation et permet d'évaluer la similarité entre deux cas.

- *La technique d'adaptation* : des mesures d'adaptation permettant d'adapter la solution antécédente au nouveau problème.

Les structures de connaissances ou *Knowledge Containers* permettent de construire et de maintenir une base de cas, et ainsi préserver et exploiter les expériences passées.

1.1.5. La Phase de Recherche

La phase de recherche doit permettre d'extraire un ensemble de cas de la base de cas les plus pertinents pour la résolution du nouveau problème à résoudre. Il s'agit donc de pouvoir identifier un certain nombre de cas dont les indices sont jugés similaires aux indices du problème à résoudre. Les indices sont les caractéristiques qui décrivent les cas de la base ainsi que le nouveau problème. Un système de raisonnement à base de cas doit donc disposer d'une structure d'indexation dans le but de produire une recherche plus efficace.

On peut décomposer la phase de recherche en deux parties :

- Le calcul de la similarité entre le nouveau cas et les cas contenus dans la base de cas.

- La sélection des cas les plus similaires au problème à résoudre.

La fonction de similarité

Pour réaliser ces deux parties de la phase de recherche, il faut définir une fonction de similarité. Le but de la **fonction de similarité** est de déterminer la ressemblance entre deux cas de la base en se basant sur leurs attributs. On peut donc, à partir de la description d'un cas, obtenir une liste ordonnée (en fonction de la valeur donnée par la fonction de similarité) des cas similaires. La fonction de similarité est une fonction qui prend deux entités en paramètres et qui retourne une valeur reflétant la similarité entre ces deux entités

selon un certain but. Si l'intérêt de l'utilisateur est de partir en voyage pour une destination bien précise, il accordera plus d'importance donc plus de poids à l'attribut « destination » par rapport aux autres attributs du cas « voyage » qui peuvent être par exemple, type de forfait, activités, proximité plage, etc. Donc, pour comparer deux cas en se basant sur leurs attributs, la fonction de similarité globale sera une fonction pondérée par les différentes similarités des attributs. La métrique de similarité classiquement utilisée est la méthode des plus proches voisins (« k-nearest-neighbhors »).

La similarité entre deux cas q et c est définie par :

$$S(q,c) = \sum w_i S_i(q_i, c_i) \qquad (1)$$

q_i sont les attributs du cas q et c_i sont les attributs du cas c.

S_i sont les différentes similarités des attributs et w_i sont les poids de pondération.

Dans cette formule, on voit qu'il est important de définir les différentes similarités entre les attributs de deux cas.

Il existe d'autres métriques de similarité que nous allons présenter brièvement [Finnie G. et Sun Z. 02] [Núñez et *al.* 02].

La métrique dérivée de la distance de Minkowski

Cette distance est une généralisation de la distance euclidienne et de la distance de Manhattan.

La métrique est définie par : $d(C_i, C_j) = \left(\frac{\sum_{k=1}^{K} weight^r \left| d(A_{ki}, A_{kj} \right|^r}{\sum_{k=1}^{K} weight^r} \right)^{1/r}$

Équation 1

où k est le nombre d'attributs du cas. Lorsque $r = 1$ on obtient la distance de Manhattan et lorsque r=2 on obtient la distance euclidienne.

Mesures de similarité hétérogènes

Wilson et Martinez (1997) ont proposé deux mesures appelées : « Heterogeneous Value Difference Metric (HVDM) » et « Interpolated Value Difference Metric » (IVDM).

La métrique HVDM se définit par :

$$HVDM(i, j) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)}$$

Équation 2

où m représente le nombre d'attributs considérés. La fonction $d_a(x_a, y_a)$ mesure la distance entre deux valeurs x et y pour un attribut a.

La métrique IVDM est définit par :

$$IVDM(x, y) = \sum_{a=1}^{m} ivdm_a(x_a, y_a)^2$$

Équation 3

avec :

$$ivdm_a(x,y) = \begin{cases} vdm_a(x,y) = \sum_{c=1}^{C} \left| P_{a,x,c} - P_{a,y,c} \right|^2 & \text{si } a \text{ est une valeur discrète} \\ \sum_{c=1}^{C} \left| p_{a,c}(x) - p_{a,c}(y) \right|^2 & \text{sinon} \end{cases}$$

C est le nombre de classes dans la base de données. $P_{a,x,c}$ est la probabilité conditionnelle suivante : si l'attribut a est égal à la valeur x alors la classe « cible » est c et $P_{a,x,c} = \frac{N_{a,x,c}}{N_{a,x}}$ où $N_{a,x}$ est le nombre d'instances qui ont x comme valeur pour l'attribut a ; $N_{a,x,c}$ est le nombre d'instances qui ont x comme valeur pour l'attribut a et qui appartiennent à la classe c.

La sélection des cas

Une fois que les similarités entre le nouveau cas et les cas de la base sont calculés, un ensemble des cas les plus similaires est choisi. Il y a deux différentes méthodes pour les sélectionner :

- Sélectionner les cas dont la similarité dépasse un certain seuil. La difficulté est de définir la valeur du seuil, c'est-à-dire quelle est la valeur à partir de laquelle on peut considérer que les cas sont similaires.

- Sélectionner les n cas les plus similaires. La difficulté est également de choisir le nombre n des cas à sélectionner.

Dans les deux cas on est confronté au problème de savoir quelles sont les valeurs qui peuvent permettre de trouver les cas les plus pertinents lors de la recherche.

1.1.6. La phase d'adaptation

Une fois que les cas les plus pertinents sont sélectionnés, le système réutilise les solutions passées en les modifiant de manière à pouvoir les réutiliser pour le nouveau problème à résoudre. Il s'agit donc d'*adapter* la solution passée au nouveau problème à résoudre.

L'adaptation de cas est un des aspects du raisonnement à base de cas qui a été particulièrement étudié en raison de sa complexité. Cette étape est celle qui définit la qualité de la solution obtenue, et donc de l'efficacité du système auprès de l'utilisateur.

Les approches les plus utilisées pour l'adaptation de cas sont :

L'adaptation à base de règles

L'adaptation à base de règles contient plusieurs approches [Aïmeur et Boudina 99]:

- ✓ L'*adaptation par substitution* : consiste à substituer les valeurs attribuées aux caractéristiques de la nouvelle solution par les valeurs de l'ancienne solution. Janet Kolodner (1993) a distingué quatre méthodes de substitution :
  - *Le remplacement* : les caractéristiques de l'ancienne solution sont remplacées par des nouveaux paramètres.
  - *L'ajustement de paramètres* : les valeurs des paramètres d'une solution sont ajustés aux données du nouveau problème.

- *La recherche locale* : une partie de l'ancienne solution est remplacée par un nouvel objet.

- *L'interrogation de la mémoire de cas* : c'est la recherche d'une nouvelle structure pour trouver un élément correspondant à une situation précise.

✓ *L'adaptation transformationnelle ou structurelle* : consiste à ré-utiliser la solution passée et à appliquer des opérateurs de transformation en se basant sur les différences entre le cas retrouvé et le cas courant [Aamodt et Plaza 94]. Cette technique d'adaptation demande un modèle fortement dépendant du domaine en termes d'opérateurs d'adaptation.

✓ *L'approche générative ou dérivationnelle* : il s'agit de retracer les étapes qui ont conduit à la solution antécédente du cas retrouvé, et de les ré-appliquer au problème à résoudre de façon à générer une nouvelle solution [Aamodt et Plaza 94] [Lamontagne et Lapalme 02].

L'adaptation à base de cas

L'adaptation à base de règles [Leake *et al.* 95] demande un effort considérable dans le processus d'acquisition des connaissances, comparable au processus utilisé dans les systèmes à base de règles (versus les systèmes de raisonnement à base de cas). Leake *et al.* proposent alors d'appliquer le raisonnement à base de cas pour faire de l'adaptation de cas, en n'utilisant plus de règles d'adaptations mais des cas adaptés résultant des expériences d'adaptation passées.

1.1.7. La phase de Révision

Cette phase consiste à évaluer la solution générée lors de l'adaptation : si le résultat est un succès, alors le système apprend (*case retainment*) à partir de ce succès, sinon il est nécessaire de modifier de nouveau la solution.

Cette phase est généralement une phase qui se déroule en dehors du système CBR, car elle implique l'application de la solution suggérée au problème de départ, et généralement, c'est l'évaluation d'un expert du domaine qui indique s'il s'agit d'un succès ou d'un échec.

1.1.8. La phase de Maintenance

En général, cette phase est le processus permettant de rajouter dans la base des cas, le nouveau cas créé. Cela implique la sélection des informations concernant le problème à résoudre, son indexation selon les structures de connaissances choisies pour le développement du système, et l'intégration de ce nouveau cas dans la structure de la mémoire.

1.2. Les modèles CBR

Il existe plusieurs modèles de raisonnement à base de cas [Lamontagne Lamontagne et Lapalme 02]. Nous décrivons dans la section suivante les différents modèles, puis nous détaillerons le modèle utilisé dans notre système.

1.2.1. Le modèle structurel

Ce modèle est l'approche traditionnelle utilisée dans les systèmes CBR. Cette approche nécessite que le problème soit complètement décrit avant que ne débute la recherche dans la base de cas.

Dans ce modèle, toutes les caractéristiques importantes du problème sont déterminées à l'avance par le concepteur du système. Les cas sont structurés et se composent de paires ‹ attribut, valeur ›. Un attribut est une caractéristique importante du domaine d'application.

La similarité entre deux cas est mesurée en fonction de la distance entre les valeurs de mêmes attributs.

Toutefois, il est parfois difficile de déterminer à l'avance toutes les caractéristiques importantes nécessaires à la résolution d'un problème. D'autres problèmes permettent de contourner ces contraintes.

1.2.2. Le modèle conversationnel

Le modèle conversationnel définit le problème à résoudre en interrogeant l'usager du système (d'où le terme de « conversation »). Ainsi, cela permettra au système de sélectionner les solutions les plus appropriées.

Un cas du modèle conversationnel consiste en trois parties :

- Une brève description textuelle d'un problème. Cette description est utilisée pour faire une première recherche dans la base de cas.
- Des index exprimés sous forme de questions afin de définir de manière plus précise le problème à résoudre. Chaque question a un poids qui représente son importance par rapport au cas.
- Une description textuelle de la solution à mettre en œuvre pour ce problème.

Le schéma de résolution de problèmes pour le CBR conversationnel est basé sur le principe suivant : il s'agit de connaître progressivement les aspects et caractéristiques du problème à résoudre par interaction avec l'usager. Ainsi, les attributs permettant la

recherche sont trouvés en collaboration avec l'usager et permettent de trouver les solutions appropriées.

1.2.3. Le modèle textuel

Dans certains domaines, il est difficile de représenter les cas de manière structurée car les expériences à codifier ne sont disponibles que sous forme textuelle. On peut prendre comme exemple les fichiers FAQ (*Frequently-Asked Questions*), les documentations, les manuels des équipements techniques, les rapports divers, etc. La description des cas est entièrement textuelle, non structurée, et souvent, les structures de connaissances (indexation, mesures de similarité, techniques d'adaptation) sont définies selon des techniques de recherche d'informations ou de traitement de la langue naturelle. Si la description du cas textuel comporte un ensemble de caractéristiques dont un sous-ensemble est textuel, alors le cas est appelé *semi-structuré*.

Le modèle textuel est souvent utilisé dans des domaines où les situations ou descriptions de problèmes ne peuvent être codifiées que textuellement, c'est le cas par exemple des textes utilisés en jurisprudence [Lamontagne et Lapalme 02].

Selon Mario Lenz [Lenz 98a] [Lenz 98b] [Lenz 98c], il est important de pouvoir répondre aux trois questions suivantes pour construire un système CBR textuel :

- Quelles sont les structures de connaissances (*knowledge containers)* appropriées ?
- Comment la connaissance sera-t-elle représentée ?
- Quelles sont les sources de connaissances ?

Dans la section suivante, nous allons nous intéresser davantage au modèle textuel et aux différents travaux effectués dans ce domaine.

1.3. Le CBR textuel

Comme nous l'avons dit dans le chapitre précédent, l'idée fondamentale du CBR est de réutiliser les connaissances acquises lors de situations passées de résolutions de problème dans un contexte similaire. Cependant, en pratique, on observe certaines situations où les problèmes sont difficilement structurables. En effet, certaines expériences ne sont disponibles que sous forme textuelle [Lenz 98c], comme nous l'avons vu dans la section précédente.

Généralement, les connaissances (*knowledge containers*) utilisées dans les systèmes appliquant le CBR textuel sont les suivantes : l'indexation et une mesure de similarité appropriée [Lenz 98b]. En effet, la collection des cas est déjà prédéfinie (elle se compose souvent d'un ensemble de documents dans lesquels la recherche s'effectue à l'aide d'outils de recherche d'information) et la phase d'adaptation est très peu utilisée. De ce fait, nous en arrivons aux questions posées dans la section 1.2.3. La résolution de ces questions permettra de construire un système CBR textuel.

1.3.1. Modèle en couches (Mario Lenz)

Mario Lenz [Lenz 98c] a étudié plusieurs systèmes (FallQ [Lenz et Burkhard 97], ExperienceBook [Kunze et Hubner 98]) utilisant le CBR textuel et en a déduit qu'il est plus avantageux de séparer les connaissances en plusieurs couches, chacune ayant la responsabilité de traiter un type de connaissances bien spécifique.

- Couche « Mots-clés » : couche basique contenant un dictionnaire de mots-clés et un parseur de mots-clés utilisant ce dictionnaire pour reconnaître des expressions simples du document. Pour construire le dictionnaire de mots-clés, des techniques du domaine de la recherche d'information [Salton et McGill 83] peuvent être utilisées : statistiques sur la fréquence des mots, lemmatisation, etc.

- Couche « Phrases » : contient un dictionnaire de mots ou « expressions » liées au domaine d'application. Un parseur est également requis lors de la construction de cette couche pour reconnaître les expressions et ensemble de mots spécifiques au domaine. Le processus d'acquisition des connaissances doit inclure les sources de connaissances spécifiques (documents techniques, manuels d'application, etc.) pour l'extraction des phrases, et généralement, la validation par des experts du domaine est fortement recommandée. Une fois que le dictionnaire est construit, la difficulté réside dans la construction d'un parseur capable de reconnaître les phrases dans le document.

- Couche « Thesaurus » : cette couche contient les informations permettant d'établir les liens de similarité entre les différents mots-clés (un exemple connu de thesaurus est WordNet [Miller 95]).

- Couche « Glossaire » : cette couche est similaire à la couche « thésaurus », la différence est qu'elle établit des liens de similarités avec les termes spécifiques au domaine d'application. Ici encore des experts du domaine sont requis pour fournir les informations nécessaires valider les relations de similarité.

- Couche « Valeur des caractéristiques » : contient un ensemble de caractéristiques et leurs différentes valeurs possibles liées au domaine d'application. Les attributs s'obtiennent généralement par discussions avec les experts du domaine. Une fois que les valeurs pertinentes des différents attributs sont définies, il faut consulter les experts du domaine de nouveau afin d'établir la mesure de similarité qui permettra de déterminer les poids de chaque caractéristique ainsi que les similarités entre les différentes valeurs.

- Couche « Structure du domaine » : contient la description de la structure du domaine permettant de grouper les documents. Le fait de classifier les documents selon la structure du domaine permet d'améliorer la précision et la rapidité de la recherche.

- Couche « Extraction de l'information » : nous avons vu que la couche « Valeurs des caractéristiques » permet de structurer la connaissance sous forme de paires {attribut, valeur} à partir de la consultation d'experts; cependant certaines parties textuelles de documents ont aussi besoin d'être structurées sous ce même format, pour cela on utilisera des techniques d'extraction de l'information (*Information Extraction*). Un exemple simple donné dans [Lenz 98c] et qui nous intéresse est les questions posées par des utilisateurs : ces questions ne sont que des descriptions textuelles qui manquent de structuration.

Les couches « Mots-clés », « Phrases », et « Valeurs des caractéristiques » contiennent les connaissances permettant de représenter les cas textuels, et donc de construire la *base de cas* alors que les autres couches sont utilisées pour *la mesure de similarité* (ce sont deux des structures de connaissances décrites dans la section 1.3), donc pour la phase de recherche du cycle CBR (cf. chapitre 1).

Le modèle en couche a cet avantage de mettre en évidence les connaissances réutilisables dans d'autres domaines d'applications et celles qui nécessitent d'être codifiées pour chaque nouveau domaine. En particulier, les couches « phrases », « glossaire », « valeurs des caractéristiques » sont très spécifiques au domaine, et doivent être revues et adaptées pour toute nouvelle application. Un inconvénient majeur est la sollicitation des experts du domaine pour beaucoup de couches (surtout les couches spécifiques au domaine), ce qui peut demander un effort d'acquisition considérable.

1.3.2. WordNet [Miller 95]

WordNet [URL WordNet] est un thésaurus permettant de relier les mots entre eux selon des relations de synonymie. WordNet est très utilisé dans des systèmes utilisant le raisonnement à base de cas textuels car il permet d'établir divers types de relations entre plusieurs mots, qu'il s'agit de noms, verbes, adjectifs, etc. Ces relations permettent ensuite d'établir la similarité entre une paire de mots-clés.

WordNet [Miller 95] a été développé par le laboratoire des Sciences Cognitives à l'université de Princetown sous la direction du Professeur George A. Miller et est actuellement disponible en version anglaise uniquement.

L'unité basique de WordNet est un ensemble de synonymes, donc un ensemble de mots pouvant être interchangeables, appelé *synset*. WordNet consiste en quatre sous-graphes organisés selon la catégorie du mot : nom, verbe, adjectif, adverbe. Chaque sous-graphe a son propre ensemble de relations. Par exemple, les noms sont organisés en termes d'antonymie, hypéronymie – hyponymie, méronymie –holonymie.

L'antonymie est un rapport sémantique existant entre des mots dont les sens sont opposés (Source : Wikipedia en ligne [URL Wikipedia]). L'hypéronymie est une catégorie sémantique qui permet de définir une relation hiérarchique entre au moins deux mots. Par exemple, le terme « être vivant » est un hyperonyme du terme « animal ». L'hyponymie désigne des mots moins généraux qui accompagnent les hypéronymes. Par exemple, dans la phrase « Un oiseau est un ovipare qui vole et qui possède des plumes », les termes « ovipare », « qui vole » et « qui possède des plumes » sont des hyponymes du terme oiseau. La méronymie est une relation partitive hiérarchisée : un méronyme A d'un mot B dont le signifié désigne une sous-partie du signifié de B. La relation inverse est l'holonymie. Par exemple, le terme « bras » est un méronyme de « corps » et le terme « maison » est un holonyme de « toit ».

Le système Chicago Information Exchange, CIE [Kulyukin 98], utilise WordNet, et un exemple d'une partie du réseau est donné, que nous décrivons ici. Pour la catégorie des noms, seule les relations d'hypéronymie et d'hyponymie sont utilisées. Le terme dans WordNet est *isa* (is_a : est_un).

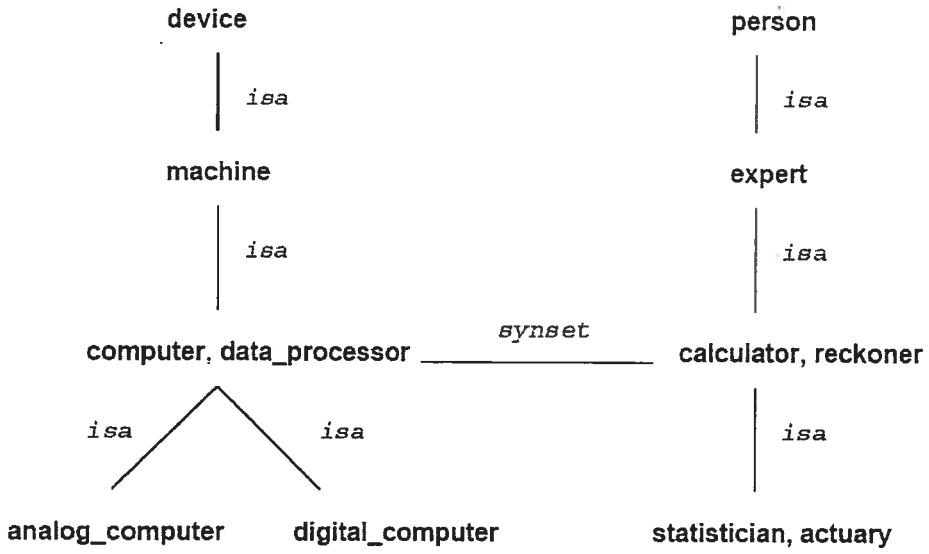


Figure 3 : Partie du sous-réseau Nom de WordNet

La figure 3 montre des hyperonymes et hyponymes du nom « computer ». La relation *synset* désigne une similarité de sens entre les deux mots reliés.

1.3.3. Conclusion

Nous désirons appliquer le CBR textuel pour répondre aux besoins exprimés dans l'introduction : permettre à un utilisateur de poser une question sur un domaine bien déterminé, afin que le système puisse chercher dans sa base de données s'il existe des

questions similaires ayant obtenues une solution. La solution peut alors être utilisée pour répondre à la nouvelle question. Dans le cas où aucune question similaire n'a été trouvée, la question est redirigée vers des experts, ou l'utilisateur est directement mis en contact avec un ou plusieurs experts. Cette fonction nécessite de faire de la localisation d'experts : déterminer quels sont les utilisateurs du système potentiellement considérés comme experts sur un sujet précis du domaine.

Dans la prochaine section, nous allons faire un état de l'art sur les systèmes basés sur la recherche de questions similaires et les systèmes basés sur la localisation d'experts.

Chapitre 2 : La Recherche d'Informations et les systèmes de questions-réponses

Dans ce chapitre nous allons présenter les principaux modèles en recherche d'informations (*information retrieval*) utilisés dans le CBR textuel pour l'indexation des cas et la recherche. Nous présenterons ensuite quelques systèmes de questions-réponses et nous terminerons par une comparaison des systèmes.

2.1. La Recherche d'Informations

Les techniques habituellement utilisées pour rechercher des documents textuels pertinents proviennent traditionnellement du domaine de la recherche d'informations [Salton et McGill 83]. Pour représenter les documents textuels, la plupart des modèles de la recherche d'informations utilisent les concepts de « termes » ou « d'index ».

2.1.1. Le modèle booléen

Dans le modèle booléen, les documents sont représentés par un ensemble de termes reliés entre eux par des opérateurs logiques, « ET », « OU », « NON » (respectivement, les opérateur « $\wedge$ », « $\vee$ », « $\neg$ »). Soit d un document, et t_i un terme représentant le document, alors le document sera représenté par :

$$d = t_1 \wedge t_2 \wedge \ldots \wedge t_i \wedge \ldots \wedge t_n$$

Il s'agit d'une indexation binaire : soit un terme apparaît dans le document, soit il n'apparaît pas.

Une requête Q est une expression booléenne dont les termes sont reliés par les opérateurs logiques :

$$q = t_1 \wedge t_3 \vee (t_4 \wedge \neg t_6)$$

La fonction de correspondance entre un document d et une requête q doit vérifier l'implication suivante :

$$d \Rightarrow q$$

Le résultat de cette fonction est binaire, ce qui est son principal inconvénient : en effet, si une requête et un document ne correspondent pas parfaitement, alors la requête q ne sera pas considérée comme pertinente. De plus, dans ce modèle il est impossible d'exprimer l'importance d'un terme par rapport à un autre, ce qui joue également sur la qualité de la pertinence des documents retrouvés.

2.1.2. Le modèle vectoriel

Le modèle vectoriel proposé par Salton a été utilisé la première fois dans le système SMART [Salton 71] et est actuellement l'un des modèles les plus utilisés dans le domaine de la recherche d'informations. Ce modèle permet de représenter les documents à travers les mots qu'ils contiennent. Chaque document et chaque requête peuvent être représentés comme un vecteur de mots ayant autant de dimensions qu'il existe de mots dans le vocabulaire d'indexation. Ces vecteurs sont construits après un certain traitement du contenu du document ou d'une partie du document. Le texte est découpé en mots, les mots considérés comme vides de sens (par exemple les articles : le, la, un...) sont éliminés, les mots qui sont sous forme fléchie sont ramenés à leurs racine. Les mots restants sont appelés les termes (dans la terminologie du modèle vectoriel, on peut également les appeler les *index* si l'on se place du point de vue du CBR).

Cependant, tous les termes n'ont pas la même importance : un mot est plus susceptible d'être pertinent dans un document *d* si sa fréquence d'occurrences est forte dans le document *d* et si elle est faible dans les autres documents. Chaque terme a donc un poids associé.

On note $\vec{d}_j$ le vecteur associé au document *j* et $\vec{q}$ le vecteur associé à la requête :

$$\vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{k,j}, ..., w_{n,j})$$

$$\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{k,q}, ..., w_{n,q})$$

avec $w_{k,j} \in [0,1]$ et $w_{k,q} \in [0,1]$

Dans ces représentations $w_{k,j}$ est le poids du terme t_k dans le document *j*, et $w_{k,q}$ est le poids du terme t_q dans la requête *q*.

2.1.2.1. Calcul de la similarité

Le modèle vectoriel permet de comparer les documents entre eux et de calculer leur degré de similarité selon les mots qu'ils contiennent. Le processus qui consiste à comparer une requête avec un document pour savoir s'ils sont relativement proches se définit comme étant un calcul de similarité entre deux vecteurs. Dans la figure ci-dessous, la requête *q* et le document d_j sont représentés par des vecteurs.

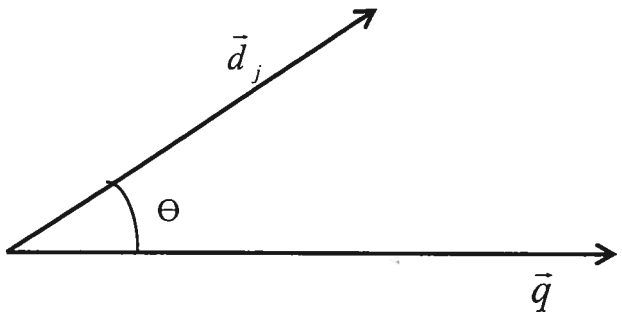


Figure 4 : représentation schématique du *Model Space Vector*

La similarité entre le document et la requête est donc calculée par le cosinus de l'angle Θ formé entre le vecteur représentant le document et celui représentant la requête. La formule du cosinus est donnée ci-dessous :

$$sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^{\ 2}} \times \sqrt{\sum_{i=1}^{n} w_{i,q}^{\ 2}}} \qquad \text{Équation 1}$$

La détermination du poids est importante dans le calcul de la similarité. En effet, le poids d'un terme mesure son importance par rapport à tout le document, ou par rapport à l'ensemble des documents (selon ce que l'on souhaite considérer). La méthode de calcul de poids que nous avons utilisé est connue sous le nom de TF*IDF (TF : *term frequency* et IDF : *Inverse document frequency*).

2.1.2.2. Mesure statistique TF*IDF

Le concept de la mesure *tfidf* est de permettre au système de mesurer la similarité globale entre la requête de l'utilisateur et chaque document de la base de connaissances, en prenant en compte la fréquence d'occurrences des termes dans un fichier [Burke *et al.* 97]. Cette mesure permet d'évaluer la relative rareté d'un terme dans un ensemble de documents, et de l'utiliser comme étant un poids mesurant la fréquence d'un terme dans un document particulier. Un terme apparaissant dans chaque document aura sans doute un *idf* égal ou proche de zéro. Un terme apparaissant dans un seul document aura sans doute la plus haute valeur possible de *idf*.

La **fréquence du terme** (tf : *term frequency*) mesure l'importance d'un terme pour un document. Plus le mot est utilisé, plus il est représentatif du document. Si $freq_{i,j}$ est le nombre d'occurrences du terme t_i dans le document d_j alors une expression du $tf_{i,j}$ est donnée par la formule :

$$tf_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}} \qquad \text{Équation 2}$$

dans laquelle $\max_l freq_{l,j}$ est le nombre d'occurrences du terme apparaissant le plus souvent dans le document d_j. Si le terme t_i n'apparaît pas dans le document d_j, alors $tf_{i,j} = 0$.

La **fréquence documentaire inverse** (idf : *inverted document frequency*) mesure si le terme est discriminant. Un terme est discriminant s'il apparaît dans un petit nombre de documents. Un terme qui apparaît dans tous les documents n'est pas discriminant. La mesure *idf* est utilisée pour donner moins d'importance aux termes peu discriminants. Soit

N le nombre de documents dans le corpus et n_i le nombre de documents dans lesquels le terme t_i apparaît, alors une expression de la valeur idf_i du terme t_i est la suivante :

$$idf_i = \log \frac{N}{n_i} \qquad \text{Équation 3}$$

Finalement, le poids mesurant l'importance d'un mot t_i dans le document d_j sera alors donné par la formule :

$$w_{i,j} = tf_{i,j} \times idf_i \qquad \text{Équation 4}$$

Dans HELP, nous utilisons le modèle vectoriel pour traiter les requêtes des utilisateurs et rechercher s'il existe des requêtes similaires à des cas précédemment résolus. Nous présentons au chapitre 5 un exemple qui met en application les formules citées ci-dessus.

2.1.3. Le modèle probabiliste

Ce modèle se base sur le calcul de la probabilité de la pertinence d'un document pour une requête donnée. On recherche donc quels sont les documents qui ont une forte probabilité d'être pertinents.

On note d_j un document et q la requête d'un utilisateur :

$$d_j = (w_{1,j}, w_{2,j}, ...w_{i,j}, ..., w_{n,j})$$

$$q = (w_{1,q}, w_{2,q}, ..., w_{i,q}, ..., w_{n,q})$$

où $w_{i,j} \in \{0,1\}$ et $w_{i,q} \in \{0,1\}$, la valeur de $w_{i,j}$ représente le fait que le terme t_i apparaît ou non dans le document d_j.

La fonction de correspondance permet de savoir si le document d_j est pertinent par rapport à la requête q.

$$sim(d_j, q) = \frac{P(rel/d_j)}{P(nrel/d_j)}$$

où $P(rel/d_j)$ représente la probabilité que le document d_j soit pertinent pour la requête q, et $P(nrel/d_j)$ la probabilité que le document d_j soit non pertinent pour la requête q.

2.1.4. Lemmatisation et algorithme de Porter [Porter 80]

Comme nous l'avons vu dans la section 2.1.2 lors de la description du modèle vectoriel, une description textuelle doit généralement faire l'objet d'une suppression de mots « inutiles », puis d'une lemmatisation sur les mots restants. Nous verrons dans l'état de l'art sur les systèmes de questions-réponses que c'est généralement la procédure suivie pour extraire les mots-clés d'un document. Ensuite, pour rechercher si une requête correspond à un document de la base, on compare les termes de la requête avec ceux du document. La comparaison consiste à vérifier tout simplement si deux mots sont identiques. Soit une requête contenant le mot *connect*, elle ne sera pas reconnue comme potentiellement similaire au document qui contient le mot *connects,* bien qu'il s'agisse d'un seul et même mot.

Un mot peut donc avoir plusieurs variantes; on comprend alors qu'il devient difficile d'extraire des mots-clés d'un document ou de comparer des mots entre eux si on doit considérer toutes les formes de chaque mot. Pour éviter cette opération, des algorithmes de lemmatisation (*stemming*) ont été créés. La lemmatisation consiste à faire l'analyse lexicale d'un contenu textuel en réduisant chaque mot à sa racine. Le mot ramené

à sa racine est appelé *lemme*. Le fait de ramener les termes appartenant à un même groupe à un seul terme (*lemme*) permet de réduire la taille et la complexité des données à traiter. La lemmatisation regroupe, pour chaque lemme, l'ensemble de ses différentes formes (noms, verbes, adjectifs, etc.). On peut donc appliquer cette opération pour réduire les mots à leur racine, et les mots ayant une racine commune ont généralement le même sens. Dans ce cas, l'analyse du contenu textuel portera sur le sens des mots en faisant abstraction de leurs formes.

[Fuller et Zobel 98] ont étudié plusieurs algorithmes de lemmatisation (analyse des mots en anglais uniquement) et les ont comparé selon l'ensemble des termes dérivés par lemmatisation et qui sont équivalents. Trois algorithmes de lemmatisation sont comparés avec une nouvelle méthode basée sur un dictionnaire, développé par l'équipe de [Fuller et Zobel 98]. Leur méthode de comparaison est basée sur l'approche suivante : la lemmatisation regroupe l'ensemble de tous les mots dérivés à partir d'une racine, la méthode consiste donc à comparer le nombre de mots dérivés corrects que chaque algorithme arrive à produire. Le résultat a donné une performance de 89% pour l'algorithme de Porter.

Nous présentons ici l'algorithme de Porter [Porter 80] que nous utilisons ensuite dans notre système. L'algorithme se base sur l'élimination des terminaisons des mots en anglais afin de garder uniquement la racine.

Une consonne est une lettre autre que A, E, I, O, U ou autre que Y précédé par une consonne. Si une lettre n'est pas une consonne il s'agit d'une voyelle.

Chaque mot peut être représenté par la formule : $[C] (VC)^m [V]$
V est une suite de voyelles (au moins une voyelle)
C est une suite de consonnes (au moins une consonne)
m est appelé la mesure d'un mot ou d'une partie d'un mot représentée sous la forme VC
Par exemple, $m = 0$ pour TR, TREE, BY

m = 1 pour TROUBLE, TREES

m = 2 pour TROUBLES, PRIVATE

L'algorithme de Porter contient une cinquantaine de règles sous la forme :

(condition) S1 → S2

Si la racine d'un mot se terminant par le suffixe S1 satisfait la condition alors S1 est remplacé par S2.

L'algorithme de Porter se déroule en 5 étapes, chaque étape contenant un ensemble de règles à appliquer. La première étape traite les mots au pluriel et les participes passés. Cette étape tient compte des cas particuliers : le mot *sized* deviendra *siz* après élimination du suffixe *ed*. Cependant l'application d'une autre règle (IZ → IZE) permettra d'obtenir le mot *size*. Les étapes suivantes permettent d'éliminer les dérivations.

Le paramètre *m* est utilisé dans les conditions, afin de ne pas effectuer d'élimination de suffixe sur les mots trop courts. Ceci est basé sur une observation et n'a pas de base linguistique.

Nous allons maintenant présenter quelques systèmes de questions-réponses.

2.2. Les systèmes de questions-réponses

De nombreux systèmes de questions-réponses ont été développés pour répondre à des besoins de service à la clientèle. Un grand nombre d'équipes travaille actuellement sur ce type de système, surtout depuis que les conférences TREC (Text REtrieval Conference) [URL Trec] ont considéré l'évaluation des systèmes de questions-réponses. Cependant ces systèmes se focalisent principalement sur la recherche de la réponse exacte à une question de l'utilisateur, en utilisant diverses techniques permettant d'extraire une chaîne de caractères (correspondant à la réponse recherchée) d'un gros corpus d'information (lors de la conférence TREC 9 en 2000, le corpus comportait 3 giga-octets d'informations sous forme de journaux généralistes et spécialisés).

Dans notre système, nous ne nous intéressons pas à la recherche de nouvelles réponses mais à la réutilisation des solutions données lors de questions précédemment posées et similaires à la question courante.

2.2.1. FallQ [Lenz et Burkhard 97]

FallQ, développé par [Lenz et Burkhard 97] a été développé pour le service à la clientèle d'une compagnie de téléphonie cellulaire. Lorsque des clients ont des questions ou des problèmes, ils les soumettent au personnel préposé à la ligne directe (*Hotline*), qui essaie de satisfaire au mieux les clients.

Les connaissances de la compagnie sont stockées sous forme de documents (descriptions des imperfections et des problèmes durant le développement des modules, spécifications des modules, les documents *Frequently-Asked-Questions*, autres documentations) et sous forme de *expériences*. En effet, [Lenz et Burkhard 97] mettent l'accent sur les connaissances détenues par des membres du personnel expérimenté qui développent des logiciels depuis des années, en comparant le manque d'expériences des nouveaux employés. L'objectif de FallQ est donc un support technique pour les membres de la Hotline, qui recherche les problèmes similaires passés et les solutions ayant été suggérées, tout en laissant le personnel **décider** de la réponse à donner au client. Cet objectif a pour principale motivation la volonté d'améliorer la rapidité de la réaction des membres du personnel du support technique, afin de satisfaire au mieux le client.

Méthodologie

Pour répondre aux objectifs fixés, [Lenz et Burkhard 97] ont développé un système d'informations utilisant le raisonnement à base de cas textuels. Nous décrivons ci-dessous les étapes de construction et de codage des différentes structures de connaissances (*knowledge containers*).

1ière phase : Définition de la structure des cas

Selon la définition donnée en section 2.3, le système développé utilise des cas textuels semi-structurés : la partie textuelle de la description du cas correspond à la question de l'utilisateur, la partie structurée correspond à un ensemble de caractéristiques communes à tous les cas (par exemple, l'identification du client).

2ième phase : Structure et construction de la base de cas

Toutes les informations utiles au support technique (documentations sur les différents types de téléphone fournis par la compagnie, problèmes survenus dans le passé, description des défauts des appareils, requêtes des clients, etc.) ont permis de servir de base à la construction de la base de cas [Lenz 98c].

Ce système se base sur un modèle appelé *Case Retrieval Nets* [Lenz *et al,* 98b]. Ce modèle contient un graphe dont les nœuds représentent soit un descripteur d'un cas, soit une Entité d'Information (*Information Entitie* : IE). Les liens entre deux nœuds IE évaluent leur similarité. Les liens allant d'un descripteur de cas vers tous les IE spécifient ce cas (la mesure de poids associée au lien permet de définir l'importance de l'IE vis-à-vis du cas). La recherche de cas pertinents consiste à activer un mécanisme de propagation : les entités d'informations extraites de la requête sont activées dans le réseau. L'activation est propagée selon les liens de similarité jusqu'à arriver aux nœuds de descripteur de cas les plus pertinents.

3ième phase : Étapes de la phase de recherche

La phase de recherche se base sur le modèle *Case Retrieval Nets* et se décompose en quatre étapes :

- ✓ Analyse de la requête afin d'en extraire les mots-clés et identifier les IEs.
- ✓ Activation des IEs identifiées dans la requête

- ✓ Propagation de la similarité parmi les IEs afin de rechercher les mots-clés similaires.

- ✓ Propagation des IEs vers les nœuds représentant les cas afin d'accéder aux cas où l'on retrouve les entités d'informations extraites de la requête ou des entités d'informations similaires. Un ordre de préférence est établi en se basant sur les différentes activations.

2.2.2. FAQ-Finder [Burke *et al.* 97]

FAQFinder [Burke *el al.* 97] est également un système de questions-réponses qui utilise comme base de connaissances des fichiers FAQ. Le système reçoit les questions des utilisateurs en langage naturel et identifie les FAQ de USENET qui sont les plus similaires à cette question.

Méthodologie

La méthodologie employée est le raisonnement à base de cas textuel. La base de connaissances est constituée de fichiers FAQ et chaque fichier FAQ possède un ensemble de questions et de réponses associées.

La recherche d'une réponse pertinente à une question posée par l'utilisateur se déroule en deux étapes :

1ière étape : recherche des fichiers FAQ pertinents

La première étape consiste à diriger la recherche vers un ensemble de fichiers Frequently-Asked-Questions susceptibles de contenir la réponse à la question de l'utilisateur. L'utilisateur peut ou non confirmer ce choix.

Pour procéder à cette première étape, le système utilise une technologie standard de la recherche d'information : le système public SMART [Salton 71]. La question de

l'utilisateur est traitée comme étant une requête soumise à un traitement de comparaison avec les fichiers FAQ. Le système SMART traite chaque mot afin d'en extraire la racine et supprime les mots appartenant à une liste de mots les plus fréquents et non utiles. Ensuite, le système construit un vecteur de termes à partir de la requête ainsi traitée, et les compare de nouveau avec les vecteurs similaires déjà créés pour les fichiers *Frequently Asked Questions* dans une étape d'indexage off-line. Les fichiers ayant le plus haut score sont retournés à l'utilisateur pour une sélection.

2ième étape : recherche de la réponse appropriée

La deuxième phase consiste en l'appariement de la question avec chaque question du fichier FAQ sélectionné. Puis un calcul de similarité est effectué afin de déterminer les questions les plus pertinentes pour répondre à la question de l'utilisateur. Plusieurs mesures de similarités sont utilisées et sont décrites dans la section suivante.

Les mesures de similarités

Mesure de similarité statistique

Une paire question/Réponse est représentée par un vecteur de termes qui associe à chaque terme un poids. Le poids est celui qui est connu sous le nom de *tfidf* (cf. section 2.1.2).

Une fois que le vecteur de termes est construit, il est ensuite comparé aux autres vecteurs en utilisant une métrique de la recherche d'information : calcul du cosinus de l'angle entre le vecteur représentant la question de l'utilisateur et le vecteur représentant la paire Question/Réponse.

Mesure de similarité sémantique

FAQFinder base principalement sa recherche sur l'utilisation d'un réseau sémantique (WordNet [Miller 95]) permettant d'établir les similarités sémantiques entre les mots d'une requête. Ceci permet d'améliorer la précision de recherche d'une réponse.

Similarité globale

La similarité globale entre deux questions est mesurée par une somme pondérée de plusieurs métriques : métrique statistique, métrique sémantique (utilisation du thésaurus WordNet), et métrique de recouvrement.

2.3. Conclusion

Les systèmes de questions-réponses que nous avons décrits ont tous un point commun : ils recherchent une question similaire à la requête courante, et réutilise les solutions sans adaptation afin de les présenter à l'utilisateur. La différence est dans le calcul de la similarité, c'est-à-dire les connaissances utilisées dans la phase de recherche du cycle de raisonnement à base de cas. Ces systèmes mettent donc en évidence l'importance des mesures de similarité et des techniques d'extraction d'informations afin d'obtenir les requêtes les plus similaires, et par conséquent les réponses les plus adéquates.

Chapitre 3 : La localisation d'experts

Les systèmes assistant les utilisateurs à trouver d'autres utilisateurs ayant les capacités suffisantes pour résoudre leurs problèmes intéressent de plus en plus les communautés de recherche scientifique ainsi que les organisations [McDonald et Ackerman 98], principales bénéficiaires de ces systèmes. Nous allons d'abord nous intéresser au terme « expert », afin de le comparer à notre propre définition. Puis nous expliquerons pourquoi la recherche d'experts fait l'objet de plus en plus de recherches et d'applications. Enfin nous terminerons par décrire et comparer quelques systèmes de localisation d'experts.

3.1. Qu'est-ce qu'un expert ?

Selon Yiman-Seid et Kobsa [Yimam-Seid et Kobsa 03] un expert peut être une *source d'informations*, comparable aux sources d'informations explicites telles que les documentations et les bases de données; ou il peut également être *une personne qui réalise une fonction sociale ou organisationnelle précise*, c'est-à-dire une personne qui utilise son savoir spécifique pour exécuter une tâche. Ce sont les besoins de l'utilisateur qui permettront d'affiner la recherche d'un expert : dans la première catégorie, il s'agit de se poser la question : « Qui a des connaissances sur le sujet x », alors que dans la deuxième catégorie, les questions sont plus précises : « quel est le degré de connaissances de l'expert sur le sujet x », « quel est l'expert qui a les connaissances les plus appropriées pour réaliser efficacement telle tâche ».

Crowder *et al.* (2003) définissent un expert comme étant une personne possédant une connaissance spécifique qui permet de faire de cette personne une autorité dans un domaine; cette personne est reconnue par tous comme étant compétente dans ce domaine. Typiquement, il s'agirait d'une personne qui aurait écrit le plus grand nombre de rapports liés à un certain domaine, travaillé avec un nombre important de personnes, et ayant une certaine expérience dans l'organisation (en termes « d'ancienneté »).

Dans notre approche, nous avons considéré la définition donnée par Yiman-Seid et Kobsa : un expert est une source potentielle d'informations, capable de résoudre le problème d'un individu à un moment donné, ou tout simplement de lui fournir un apport de connaissances.

3.2. **La localisation d'experts**

La technologie doit permettre non seulement d'accéder aux connaissances explicites que l'on peut retrouver dans les documents de l'entreprise mais également aux connaissances tacites détenues par les individus [Yimam-Seid et Kobsa 03]. De plus, la capacité à trouver rapidement des informations sur l'expertise des personnes de l'entreprise peut jouer un rôle critique pour entretenir la formation des entreprises/organisations virtuelles, réseaux d'expertise, etc. La reconnaissance du besoin d'entretenir le partage de l'expertise a récemment fait l'objet de la recherche par, entres autres, les communautés de KM (*Knowledge Management*) et de CSCW (*Computer Supported Collaborative Work*). Des concepts tels que capitalisation d'expertise, extraction de connaissances, management des compétences, management du capital intellectuel, réseaux d'expertise et systèmes de partage des connaissances sont les sujets discutés.

3.2.1. Qu'est-ce qui provoque la recherche d'experts ?

Cette section présente les motivations évoquées dans la littérature sur la recherche d'experts dans une organisation.

3.2.1.1. Le réseau social

Le réseau social d'une personne est l'ensemble des contacts connus de la personne, et qu'elle utilise généralement lorsqu'elle cherche à résoudre un problème. Le réseau social d'un individu est souvent composé de personnes avec qui il partage des intérêts similaires

[Kautz *et al.* 97]. Il est naturel pour toute personne ayant un problème à résoudre de demander autour d'elle une personne susceptible de l'aider ou une personne pouvant lui recommander quelqu'un d'autre. Ainsi un ensemble de connexions se crée au fur et à mesure, et élargit le réseau social de la personne. Selon McDonald et Ackerman (1998), beaucoup de personnes, et spécialement les personnes qui ont plusieurs années d'ancienneté dans une organisation, ont du mal à localiser qui a telle connaissance concernant tel domaine. Ces personnes « seniors » mentionnent souvent l'*expérience* pour identifier les personnes possédant une certaine expertise. Ces réseaux sociaux sont formés dans le temps, au fur et à mesure des travaux faits, et surtout des interactions sociales générées par le travail.

Cependant, toute personne est amenée à changer de poste ou d'entreprise, avec pour conséquence des modifications plus ou moins importantes dans les réseaux sociaux formés [Crowder *et al.* 03]. De plus, toute nouvelle personne dans l'entreprise n'a pas cette compréhension implicite de *qui sait quoi* [McDonald et Ackerman 98]. Ce modèle n'est donc pas fiable, et de plus dans une grande organisation, les réseaux sociaux sont souvent limités, c'est-à-dire qu'une personne ne pourra sans doute jamais connaître tous les membres (et leur expertise) d'une organisation.

3.2.1.2. Pourquoi a t-on besoin d'un expert ?

Les deux principales motivations pour rechercher un expert dans un domaine spécifique sont [Yimam-Seid et Kobsa 03] : un besoin d'informations ou un besoin d'expertise.

Un besoin d'informations

En effet, toutes les informations dans une organisation ne sont pas nécessairement explicitées dans des documents. Certaines informations ne peuvent être transférées qu'à travers l'apprentissage, l'expérience, les conversations informelles.

De plus, on peut se retrouver dans des situations où l'information dont les utilisateurs ont besoin soit mal spécifiée et requiert un dialogue avec un expert pour l'identifier.

Les utilisateurs veulent souvent minimiser leurs efforts et leur temps dans la recherche d'une information spécifique (en particulier pour ce qui représente « beaucoup de travail » pour l'utilisateur correspond à « peu de travail » pour l'expert). Ceci inclut d'utiliser les experts comme des filtres d'information efficace et de confiance dans la sélection de l'information utile parmi la masse d'informations disponible.

Ackerman et McDonald (1996) parlent d'un certain type d'aide, dite *aide spécialisée* : un utilisateur à la recherche d'expertise spécifique pouvant résoudre son problème spécifique est en fait à la recherche d'une aide ou assistance spécialisée. Cette aide peut bien sur être fournie par de la documentation, des archives, ou des tentatives de résoudre le problème par lui-même, mais bien vite, cet utilisateur se tournera vers ses collègues.

Un besoin d'expertise

Le besoin d'une personne possédant un certain type d'expertise est demandé lorsque cette personne doit jouer un rôle dans un projet particulier, ou lorsqu'un partenariat permanent entre l'expert et la personne est requis. Becks *et al.* (2003) définissent ce besoin d'expertise comme étant une *fonction de groupe* : le système est ici utilisé pour catégoriser

les profils de tous les utilisateurs dans le but de présenter un ensemble d'expertise dans un but d'analyse et d'exploration. Considérons un environnement d'entreprise où un directeur de projet essaie d'identifier l'expertise des membres de son personnel qui peut potentiellement s'occuper d'une certaine sous tâche.

Dans un mode de fonction de filtre [Becks *et al.* 03], l'utilisateur utilise le système pour rechercher des utilisateurs qui sont similaires à son profil ou à des parties de son profil. Dans un contexte de plateforme d'apprentissage, un apprenant peut vouloir rechercher d'autres apprenants ayant les mêmes antécédents, les mêmes intérêts, les mêmes connaissances dans le but par exemple de construire un groupe d'apprentissage. Dans le contexte de scénarios dans le domaine de gestion des connaissances (*Knowledge Management*), ce type de système peut également avoir une fonction de filtre : considérons par exemple un environnement d'entreprise où un expert dans un certain domaine veut former un réseau d'experts ayant des antécédents de projet similaires dans le but de partager leurs expériences ou de développer un nouveau projet. De manière similaire au processus de filtrage de profils selon son propre profil, un utilisateur peut soumettre une requête au système de recherche d'expert dans le but de trouver des experts qui correspondent à ses besoins explicites.

3.2.1.3. Conclusion

Finalement, le besoin d'informations ou aide spécialisée est utilisé lorsque une personne a besoin d'une connaissance bien spécifique, et uniquement dans le but de résoudre son problème ou de se renseigner sur un point précis. Il est clair que l'utilisateur et la personne fournissant l'information n'ont pas les mêmes buts. Dans le cas du besoin d'expertise, la recherche aboutira à un ensemble de personnes expertes ayant un but commun, celui de résoudre une tâche bien spécifique. Notre système fait également la distinction entre ces deux concepts, comme nous le verrons dans le chapitre 5.

Trois principaux facteurs sont à considérer pour comprendre l'intérêt de tels systèmes dans les organisations [Yimam-Seid et Kobsa 03] [Crowder *et al.* 03] : la taille de l'organisation, sa distribution géographique, et l'homogénéité de la composition de ses membres. Plus l'organisation est importante en taille et distribuée dans l'espace, et plus ses membres sont hétérogènes (connaissances différentes, parcours de carrière/parcours scolaires différents, division stricte en plusieurs départements), moins les contacts informels et les échanges se réalisent. De plus, les personnes ne restent pas forcément au même poste dans l'entreprise, la recherche de personnes ayant déjà effectué une tâche devient alors plus complexe.

Les systèmes de localisation d'expertise ont donc pour but de pallier le mieux possible aux différents problèmes évoqués ci-dessus.

3.2.2. Principes d'un système de localisation d'expertise

Les systèmes de profiling d'expertise permettent de rendre les connaissances explicites et implicites détenues par des individus visibles et accessibles aux autres [Becks *et al.* 03].

> *Expert finding means matching a prototype set of personal data (i.e. profile of a certain user in the application environment or a query profile) against a collection of others' personal profiles in order to determine a ranking of fitting actors*
>
> *Becks, et al., (2003).*

Nous avons vu dans la section précédente l'intérêt d'une recherche d'expertise (« pourquoi »), nous allons voir maintenant les données à prendre en compte pour construire des systèmes de localisation d'expertise (« comment »).

McDonald et Ackerman (1998) ont fait une étude sur l'identification et la sélection des sources potentielles d'information pour résoudre un problème spécifique, dans une compagnie fournissant des logiciels. Cette étude leur a permis ensuite de connaître les données utiles et les mécanismes d'utilisation de ces données, et donc les principes importants à considérer lors de la construction d'un système de localisation d'expertise.

Deux étapes sont nécessaires à la localisation d'expertise :

- ✓ **Identification de l'expertise :** il s'agit de savoir quelles informations spécifiques et quelles connaissances détient chaque individu.
- ✓ **Sélection de l'expertise :** choisir parmi les personnes ayant l'expertise demandée, c'est-à-dire fournir de l'expertise.

Nous allons voir en détails quels sont les mécanismes de l'identification et de la sélection de l'expertise.

3.2.2.1. Identification de l'expertise

Nous allons tout d'abord décrire rapidement l'étude effectuée par McDonald et Ackerman (1998) dans la société MSC (*Medical Software Corporation*).

Les mécanismes d'identification d'expertise dans la société MSC

[McDonald et Ackerman 98] ont identifié deux mécanismes d'identification d'expertise, utilisés dans une entreprise : MSC, *Medical Software Corporation*, dans laquelle ils ont étudié les informations, les données, les personnes impliquées dans la localisation d'expertise. MSC est une compagnie qui développe, vend et fait du support technique de logiciels de gestion médicale et dentaire. Ces deux mécanismes sont :

l'utilisation d'*historique* et l'utilisation de *« concierges d'expertise »*. Nous allons voir ci-dessous les définitions et implications de ces deux mécanismes.

Un des mécanismes utilise l'historique ou les archives maintenues par l'organisation. Les programmeurs et le personnel du support technique de MSC utilisent les modifications faites sur les fichiers sources pour identifier les experts potentiels. La règle est la suivante : chaque développeur qui modifie un fichier doit obligatoirement inscrire son identifiant et la date à laquelle les modifications ont été effectuées. Ainsi, le dernier développeur qui a modifié le code est supposé être celui qui a une bonne mémoire du code.

> *When a programmer makes a change in a program he is supposed to add his mnemonic to the line and update the date. This is how we know who last changed the program. Whoever made the last change in the program is the default expert in that program...It's close enough. The logic is that the person who spent time on it last has it freshest in memory and so they are the best person to ask a question.*

Cependant, cette méthode a plusieurs désavantages, dont celui, si l'on considère toujours l'exemple donné, de l'importance des modifications apportées. Le développeur qui a fait un petit changement dans le code sera la dernière personne ayant travaillé sur le fichier, et donc sera identifié comme expert potentiel.

Les mécanismes utilisant les archives ou historiques peuvent être une bonne source d'identification à condition qu'elles soient maintenues régulièrement et qu'elles soient associées à des informations supplémentaires.

Les « concierges d'expertise » sont des personnes ayant un rôle important dans une entreprise : ce sont les personnes qui savent *qui sait quoi*. Ce sont elles qui vont diriger les personnes ayant besoin d'aide vers les personnes appropriées.

De même, ces personnes peuvent être de bonnes sources d'informations, cependant elles sont soumises aux mêmes règles que les autres employés : leur indisponibilité, leur départ de la compagnie, peuvent rendre « inutilisable » cette source d'identification d'expertise.

Nous avons vu à travers un exemple les sources potentielles d'informations d'une compagnie. Pour réaliser cette étape d'identification d'expertise, un système de localisation d'expertise nécessite de trouver un moyen de collecter les connaissances, et de créer et maintenir les profils d'experts. Pour cela deux grandes approches existent : l'approche *traditionnelle* (manuelle) et l'approche *automatique*.

La collecte d'informations : l'approche traditionnelle et l'approche automatique

L'approche *traditionnelle* requiert tout simplement que les utilisateurs fournissent les données qui décrivent leur expertise et leur domaine d'intérêts. Cette approche a bien sûr plusieurs inconvénients [Becks *et al.* 03] : il est nécessaire de définir une compréhension commune des différents attributs possibles pouvant définir les profils personnels. En effet, certains types de vocabulaire peuvent donner lieu à plusieurs interprétations. Afin d'obtenir une concordance automatique des profils personnels, un vocabulaire doit être défini englobant tous les profils possibles. De plus, il est nécessaire d'entretenir une certaine motivation auprès des différents acteurs, pour d'une part créer leurs profils, et d'autre part, les mettre à jour. C'est pourquoi il est important d'avoir des mécanismes de collectes d'informations permettant de mettre à jour automatiquement ces données.

L'approche *automatisée* consiste à analyser les comportements spécifiques, des documents produits par les experts, etc. et modifier en conséquence leurs profils utilisateurs. Le système Yenta [Foner 98] analyse les échanges d'e-mail pour établir des

profils utilisateurs. Nous verrons d'autres exemples d'utilisation de l'approche automatisée dans la section.

3.2.2.2. Sélection de l'expertise

Selon Allen, cité par McDonald et Ackerman, les personnes recherchant de l'information prennent en compte plusieurs facteurs pour choisir le ou les experts à contacter. Ces facteurs se basent sur des critères personnels tels que l'état psychologique engendré par le fait de *demander* de l'information (et donc de faire état de ses propres ignorances avec la peur d'une possible perte de statut). De plus d'autres facteurs permettent de différencier les experts à contacter : la réciprocité attendue (la probabilité d'aider l'expert à son tour), et l'équité sociale (jusqu'à quel point connaissent-ils la personne socialement).

3.3. Les systèmes de localisation d'experts

Dans cette section, nous présentons quelques systèmes qui ont pour but de localiser les experts pour les recommander aux utilisateurs selon leurs besoins.

3.3.1 Chicago Information Exchange [Kulyukin 98]

De plus en plus d'organisations ont à répondre aux questions de leurs clients. Cependant, les problèmes observés lors des études sur le *Knowledge Management* se posent également ici : les clients perdent du temps à rechercher l'information et les experts de l'organisation ne sont pas motivés car ils ont à répondre aux mêmes questions de manière répétitive, ou bien ils reçoivent des questions qui sont non pertinentes à leur domaine d'expertise. Afin de résoudre ce problème, Kulyukin (1998) a développé des systèmes d'Echange d'Information, accessibles sur le Web, et permettant l'organisation et l'accès à l'expertise textuelle en ligne.

L'approche utilisée dans ce type de systèmes consiste à voir l'expertise comme étant un ensemble de concepts, chaque concept contenant une collection dynamique de paires de questions-réponses. Un concept est considéré comme un domaine de l'expertise de l'organisation. Une paire de Q&A est une question qui a été précédemment répondue par un expert. Les clients et les experts ont à leurs dispositions plusieurs interfaces et outils leur permettant de gérer l'expertise et la recherche d'expertise.

Ces idées ont été implémentées dans un système appelé *Chicago Information Exchange* (CIE), une application permet de gérer l'expertise en ligne de l'Université de Chicago, Département Informatique. Les clients sont les étudiants et les parents. Les experts sont la faculté, les étudiants gradués et les secrétaires.

L'exemple suivant montre la problématique et permet de comprendre l'objectif de CIE :

> *X is an undergraduate enrolled in CS115, which uses the Scheme programming language. X finds the textbook for the class terse and wants to know if there is another book on Scheme that he could read, too. How does the Department make sure that X gets answers quickly? How can the Department see to it that once the answers to such questions are available they can be reused?*

Il s'agit donc de rechercher si un expert peut répondre aux questions de l'étudiant, et ensuite mettre les solutions données à la disposition de tout le monde.

Brève description du système

Les concepts et sujets associés sont structurés de manière hiérarchique. Lors de l'enregistrement d'un expert dans le système, son compte d'informations est automatiquement associé à chaque concept ou sujet du domaine. Outre ses informations personnelles, le compte d'informations d'un expert contient l'ensemble des questions

auxquelles il a répondu, ainsi qu'une liste de contacts (souvent des experts auxquels il peut envoyer des questions non pertinentes à son domaine d'expertise). La figure 5 est un exemple de l'organisation du domaine dans CIE : les concepts sont organisés selon une certaine hiérarchie (telle qu'elle est définie dans le Département Informatique de l'Université de Chicago), et à chaque concept, comme nous pouvons le voir, est associé un ou plusieurs comptes d'informations. Le compte se compose des informations personnelles (IP sur la figure 5), de la description de son expertise (DE) et de la base de questions auxquelles il a répondu (Q&A).

Lorsqu'un utilisateur pose une question, le système recherche d'abord les concepts les plus pertinents à la question posée. Une fois que le concept est trouvé, CIE vérifie s'il existe des comptes d'informations associés à ce concept, et si c'est le cas, pour chaque compte d'information, le système calcule la similarité entre la question posée et les questions posées dans la collection des questions non pertinentes. Dans ce cas, le compte d'information n'est plus considéré. Dans les autres cas, le système compare la question posée à chaque paire de questions-réponses (paires Q&A) provenant de la base de Q&A associée au compte. La Q&A la plus similaire est envoyée au client, comme on peut le voir sur la figure 5.

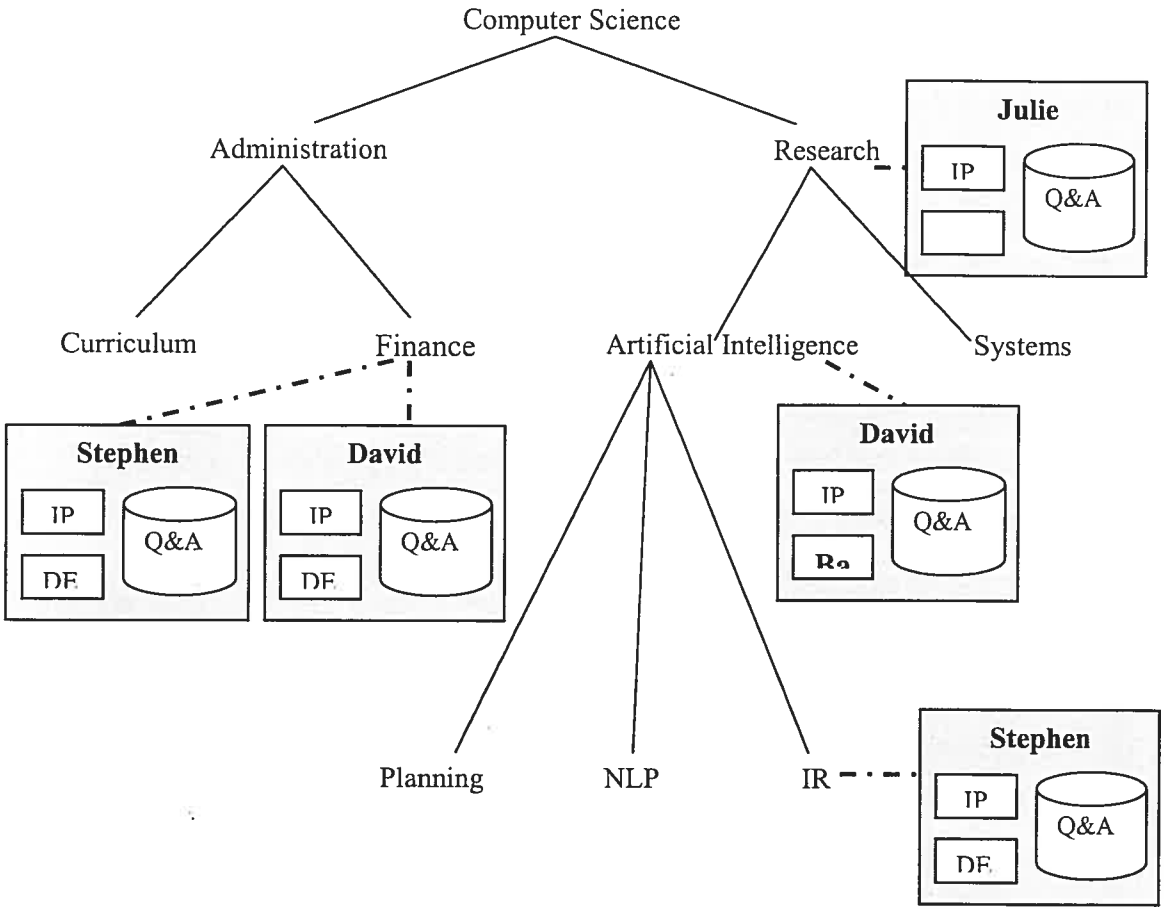


Figure 5 : Exemple de l'organisation du domaine dans CIE

La figure 6 montre clairement les choix donnés à l'utilisateur concernant la réponse de l'expert.

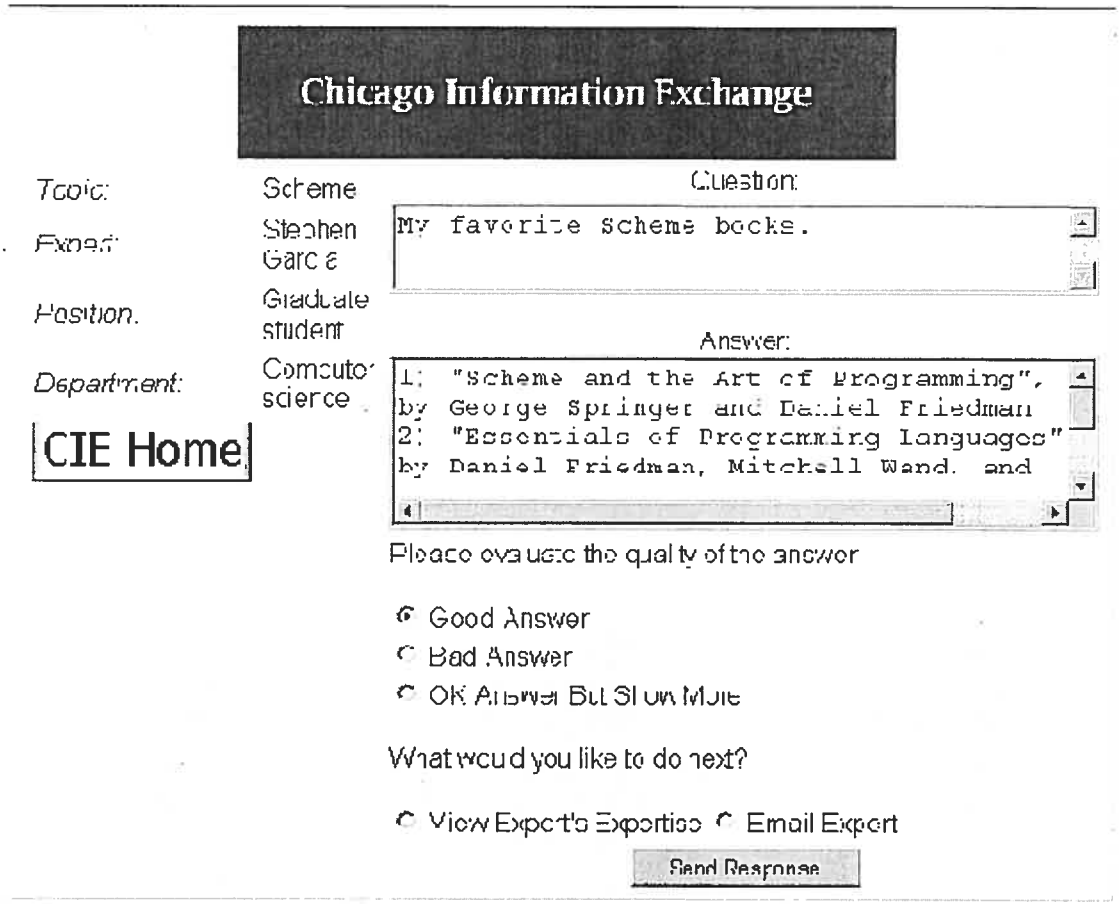


Figure 6 : Réception d'une requête et réponse de l'expert

Méthodologie

CIE utilise le modèle vectoriel de Salton, et combine trois métriques afin de calculer la pondération des termes du vecteur. Deux métriques proviennent du modèle vectoriel : la métrique *tf*idf* (décrite au chapitre 2), et une métrique nommée *condensation clustering*. La troisième métrique est basée sur la notion d'activation propagée (*spreading activation*) dans WordNet. L'activation propagée a pour but de réduire la dépendance lexicale dans les termes des questions pour chaque paire de Questions&Réponses, c'est-à-dire autoriser un

petit degré de variation lexicale dans la question du client. Par exemple, dans la figure 3 (chapitre 1), si la question du client contient le mot « machine » et la question de l'expert contient le mot « computer », les deux termes correspondent à cause de leur relation sémantique.

Conclusion

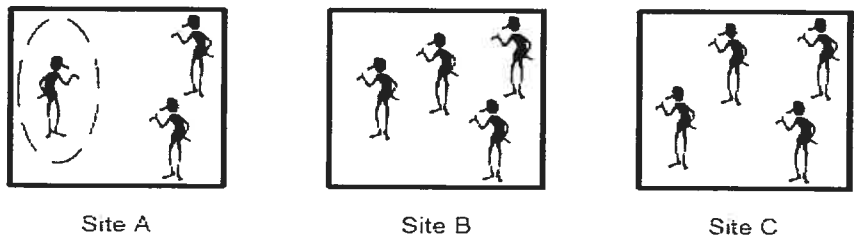
Nous avons vu que l'utilisateur avait la possibilité d'évaluer la qualité de la réponse (« Good Answer » ou « Bad Answer », « Ok Answer but show more »), cependant le résultat de cette réponse ne sert qu'à relancer la recherche dans le cas où l'utilisateur n'est pas satisfait. L'une de nos principales motivations est de savoir **pourquoi** l'utilisateur n'est pas satisfait.

3.3.2 Expertise Finder [Crowder *et al.* 03]

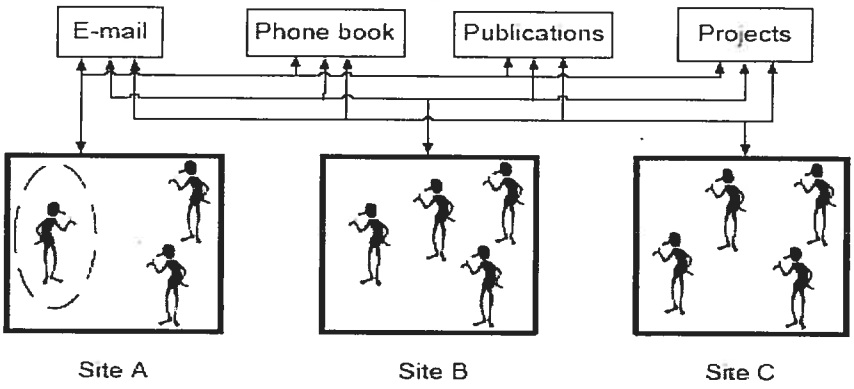
Expertise Finder a été conçu pour localiser la meilleure expertise dans une organisation de taille importante pour résoudre des problèmes spécifiques. Le système a pour objectif d'aider à résoudre les problèmes qui peuvent intervenir dans les réseaux sociaux des personnes (cf. section 3.2.1.), cependant ce système n'a pas la prétention de remplacer le réseau social mais d'accélérer les connexions entre les personnes de l'organisation. Le résultat est la recommandation de contacts et de documents appropriés aux besoins de l'utilisateur.

La problématique est décrite en figure 7, et peut se résumer par :

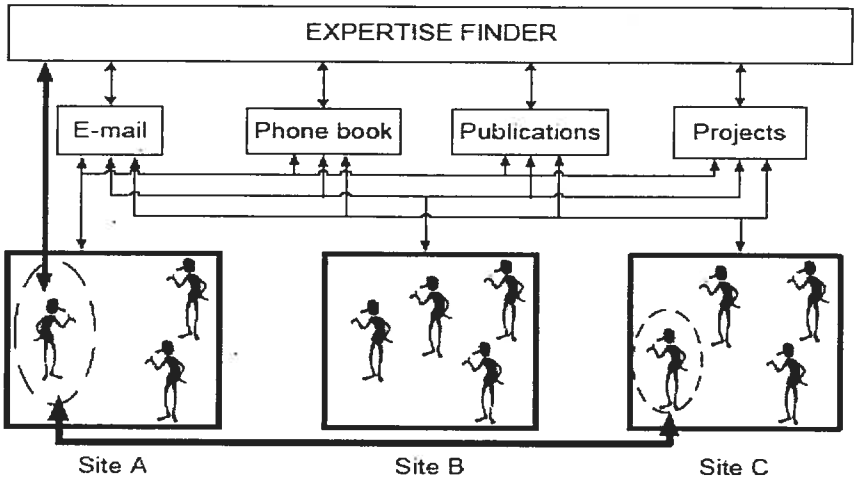
How does a person located in Site A, locate the best expertise to solve a specific problem? [Crowder *et al.* 03]



(a) Local social network fails to find an expert.



(b) Individuals are known to the organisation electronic resources



(c) Expertise Finder identifies an expert regardless of location

Figure 7 : Aperçu de la problématique et concepts de Expertise Finder [Crowder *et al.* 03]

Les sources de connaissances utilisées pour la localisation d'experts sont les informations internes de la compagnie: e-mail, contacts téléphoniques, bases de données des ressources humaines, rapports et publications. Ces deux dernières informations sont particulièrement utilisées pour retrouver la personne adéquate. En effet, le prototype a été développé pour recommander des personnes et de la documentation en utilisant les publications scientifiques des personnes. Suivant la définition donnée à l'expert par Crowder *et al.* les experts recommandés seront ceux qui auront eu, entre autres, le plus de publications.

Expertise Finder est un système multi-agents composés d'un agent principal, l'agent *Expertise Finder*, et d'un ensemble d'agents appelés *Source Agents*. Les agents de sources sont responsables de collecter les informations nécessaires pour déterminer une liste de personnes et de documents à recommander. L'agent *Expertise Finder* est responsable des interactions avec les agents de sources de manière à trouver un résultat final pour l'utilisateur. Des agents Web s'occupent des interactions entre l'utilisateur et l'agent *Expertise Finder*.

Lorsqu'un utilisateur pose une question sur un certain sujet, la requête est transmise à l'agent *Expertise Finder* par les agents Web. L'agent Web demande alors aux Agents de Publications (qui sont des entités Agents de Source) de trouver les publications correspondants à la requête. La requête est transformée en requête SQL de façon à pouvoir interroger la base de données des publications. La base de données des publications liste les auteurs et leurs adresses e-mail, et l'Agent de Publications utilise les services d'un autre agent, nommé *Directory Service Agent,* pour identifier les auteurs. L'Agent EF enregistre toutes les données envoyées par l'agent de Publications et compte le nombre de fois où chaque auteur apparaît dans les publications. L'agent EF maintient également une liste des personnes non identifiées. Tous ces résultats (auteurs, nombre d'occurrences et leur statut dans le département) sont envoyés à l'utilisateur, ainsi que les auteurs non identifiés (qui sont souvent des collaborateurs externes, ou des personnes ayant quitté le département).

Ce système a mis en évidence l'importance des ressources utilisées pour la localisation d'experts selon le domaine d'application choisi. Cependant, ce système ne considère pas suffisamment la requête de l'utilisateur, ni son profil (critique faite par les auteurs eux-mêmes d'Expertise Finder).

De plus, de notre point de vue, il ne s'agit pas vraiment de recommandations mais plutôt d'une recherche d'un ensemble de personnes susceptibles d'aider l'utilisateur. En effet, le principe de la recommandation consiste à proposer à l'utilisateur des objets (de manière générale) qui puissent l'intéresser, et qui prennent donc en compte son profil [Burke 02]. Dans ce système, l'utilisateur risque de se retrouver submergé par un ensemble de personnes qu'il pourrait contacter, et de plus nous ne pensons pas qu'une personne ayant fait un maximum de publications soient la personne adéquate pour répondre aux besoins de l'utilisateur. Bien sur, Crowder *et al.* prônent l'efficacité du système en termes de connexions (extension du réseau social), cependant, même à ce niveau, l'utilisateur aura peut-être à contacter plusieurs personnes « expertes » avant de trouver celle qui correspondra le mieux à sa requête.

3.3.3 Système Q&A [Budzik et Hammond 99]

Budzik et Hammond (1999) ont développé un système appelé Q&A. Q&A est un système de questions-réponses accessible sur le Web qui permet de gérer les interactions entre un expert et un utilisateur. Lorsqu'un utilisateur pose une question, le système recherche s'il existe une question similaire ayant déjà été répondue. S'il n'existe pas de questions similaires, Q&A propose une liste d'experts à l'utilisateur qui sont le plus susceptibles de pouvoir répondre à la question. Lorsqu'un expert répond à l'une des questions, la paire question-réponse est capturée et indexée selon un sujet choisi par l'expert pour une utilisation ultérieure.

Q&A organise l'expertise de manière hiérarchique : les experts sont en haut de la hiérarchie, puis les sujets (domaines d'expertise) reliés à chaque expert, et enfin, pour chaque domaine d'expertise l'ensemble des questions-réponses correspondant. Les experts sont représentés en tant que vecteurs de termes dérivés des vecteurs calculés lors de l'analyse de leurs questions et réponses. Ainsi la recherche des experts s'effectue en utilisant la formule du cosinus sur l'angle formé par le vecteur de la requête et le vecteur de l'expert.

3.3.4 « Support Routing » [Aberg et Shahmehri 01]

Aberg et Shahmehri (2001) ont développé un système basé sur une approche qu'ils ont appelé le « support routing ». Le rôle du « support routing » est de décider quelle est la meilleure façon de répondre à la question de l'utilisateur : le guider vers un assistant humain ou utiliser un système de questions-réponses utilisant les mêmes techniques de recherche d'information vues précédemment (modèle vectoriel). Le choix se fera en fonction du domaine d'application du système qui déterminera les facteurs les plus importants à prendre en compte (comme par exemple la confiance et préférences des utilisateurs, le coût des ressources, l'efficacité du support).

Les connaissances extraites des assistants sont représentées sous forme de composant Question-Réponse (question avec réponse associée). Le système utilise la technique du modèle vectoriel afin de sélectionner les questions qui correspondent le plus à la question de l'utilisateur.

Tel que nous l'avons vu dans les systèmes de questions-réponses au chapitre 2, le système se compose principalement de deux parties : un indexeur et un manageur de questions. L'indexeur est responsable de la création d'une représentation interne des composants questions-réponses en associant à la question un vecteur de termes pondérés (selon les étapes suivies par le système FAQ-Finder). Le manager de questions est

responsable de la représentation interne de la requête utilisateur. Le vecteur produit sera utilisé pour la phase de recherche.

Dans ce système, les experts sont regroupés par catégorie représentant leur domaine d'expertise. Lorsqu'un utilisateur est dirigé vers un assistant humain, le système recherche uniquement les experts appartenant à la catégorie demandée par l'utilisateur.

3.3.5 Expert Finder [Vivacqua et Lieberman 00]

De nombreux systèmes de localisation d'experts se base sur l'analyse des documents produits par les utilisateurs du système pour déterminer leurs domaines et leur degré d'expertise. Expert Finder [Vivacqua et Lieberman 00] est un agent qui permet de recommander des experts en Java à des débutants. Le système se base uniquement sur la localisation d'experts en analysant les programmes sources en Java pour établir les niveaux d'expertise en Java de chaque développeur.

Chaque fichier source produit par un utilisateur est analysé afin de connaître les classes et méthodes qu'il utilise. Puis, le modèle vectoriel de Salton est appliqué afin de produire un vecteur de mots-clés pondérés. La pondération permet d'établir le niveau d'expertise de l'utilisateur pour chaque mot-clé. Par exemple, pour un mot-clé donné, une pondération proche de 1 permettra au système de classer l'utilisateur comme expert **pour ce mot-clé**.

Lorsqu'un utilisateur pose une question (il a la possibilité soit d'entrer des mots-clés soit de sélectionner des classes appartenant au langage Java), l'agent calcule alors un vecteur de mots-clés et fait la comparaison entre le vecteur représentant la requête avec chaque vecteur représentant l'expertise de chaque utilisateur de la base. Les n similarités les plus hautes déterminent les experts à recommander à l'utilisateur.

Le modèle d'Expert Finder a cet inconvénient de ne se baser que sur les fichiers sources produits par les utilisateurs pour déterminer leur niveau d'expertise et faire par la

suite des recommandations. Or, l'expertise d'une personne n'est pas seulement déterminée par ce qu'elle produit. Cela représente peut-être qu'une partie de son expertise.

3.3.6 Conclusion

Tous ces systèmes ont une motivation commune : permettre à des clients de trouver l'expert capable de répondre à leurs questions le plus rapidement possible. Cependant, les systèmes décrits précédemment ne prennent en compte dans le profil des utilisateurs que les domaines d'expertise (extraits automatiquement ou fournis par l'expert lui-même) pour établir leur liste de recommandation. D'une requête à une autre, l'apprenant pourrait donc avoir les mêmes propositions d'experts même si ceux-ci ne leur conviennent pas.

Pour améliorer la pertinence des recommandations, nous pensons qu'il est nécessaire d'apporter des informations supplémentaires concernant l'expert lui-même, afin de pouvoir mieux connaître quelles sont les attentes de l'utilisateur vis-à-vis de l'expert ou de ses connaissances. Nous verrons dans le chapitre 5 l'approche adoptée.

Chapitre 4 : Les systèmes de recommandation

De nos jours, la quantité d'informations disponibles devient de plus en plus importante, notamment depuis l'avènement de l'Internet. Une personne recherchant une information spécifique se sent souvent submergée par la masse de connaissances qu'elle a à sa disposition, et parmi lesquelles elle doit trouver elle-même l'information *utile*. Les systèmes de recommandation ont alors émergé depuis plusieurs années pour faciliter et aider les usagers à faire face à cette surcharge d'informations : en effet, les systèmes de recommandation ont pour but de recommander des items ou objets qui intéressent l'utilisateur, donc adaptés à son profil.

Nous allons parler dans ce chapitre des techniques de recommandations les plus utilisées.

4.1. Introduction

Comme nous l'avons dit, les systèmes de recommandation ont pour but de permettre à l'usager d'avoir un service adapté à son profil : tout système produisant des recommandations personnalisées ou ayant pour fonction de guider l'utilisateur à choisir des produits ou services qui lui potentiellement utiles ou intéressants sont appelés des systèmes de recommandation [Burke 02]. De nombreux sites commerciaux proposent des services personnalisés basés sur les techniques de recommandations, tels que Amazon.com ou CDNow.com. Les recommandations sur ces sites ont pour but de suggérer à l'utilisateur des items particuliers susceptibles de leur plaire. Les systèmes de recommandations sont également utilisés dans d'autres domaines, dont l'un nous intéresse particulièrement : l'apprentissage en ligne ou e-learning. Dans ce domaine, nous pouvons citer par exemple un système qui recommande des articles à des étudiants [Tang et McCalla 03]. Une des particularités de ce système est de prendre en compte la dimension pédagogique des articles et de les recommander selon les compétences et préférences de l'étudiant : un apprenant

peut vouloir connaître en détails un sujet particulier, le système lui recommandera donc des articles très techniques. Citons également DIA [Yammine *et al* 04], un système permettant de recommander des livres à des étudiants en fonction de la similarité de leur profil (comprenant leur style d'apprentissage) avec d'autres usagers.

Dans les sections suivantes nous décrirons les différentes techniques de recommandation.

4.2. Les techniques de recommandation

Les techniques de recommandations peuvent être classées en fonction de plusieurs critères [Burke 02] : les données sur lesquelles les recommandations sont basées, et l'utilisation qui en est faite. Généralement, la technique de recommandation peut être vue comme une fonction qui prend comme informations de départ les données du système (il peut s'agir de produits si on considère les systèmes à but commercial) et les interactions de l'utilisateur avec le système. La fonction combine alors ces deux types d'informations pour produire en sortie des recommandations ou des suggestions.

Burke [Burke 02] a distingué 5 techniques de recommandations regroupées dans le tableau 1. **I** est l'ensemble des « items » ou objets sur lesquels des recommandations peuvent être faites, **U** est l'ensemble des utilisateurs du système de recommandation. L'utilisateur **u** est l'utilisateur cible du système pour lequel les recommandations doivent être calculées et **i** est un des objets de l'ensemble **I**.

Tableau 1 : Les différentes techniques de recommandation [Burke 02]

Technique	Background	Informations nécessaires	Fonctionnement
Collaborative	Ratings from **U** of items in **I.**	Ratings from **u** of items in **I.**	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i.**
Content-based	Features of items in **I.**	**u**'s ratings of items in **I.**	Generate a classifier that fits **u**'s rating behaviour and use it on **i.**
Demographic	Demographic information about **U** and their ratings of items in **I.**	Demographic information about **u.**	Identify users that are demographically similar to **u**, and extrapolate from their ratings of **i.**
Utility-based	Features of items in **I.**	A utility function over items in **I** that describes **u**'s preferences.	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I.** Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

Pour chacune des techniques décrites ci-dessous, nous verrons quels en sont les avantages et les limites.

4.2.1. Le filtrage collaboratif

La recommandation basée sur le filtrage collaboratif est la plus connue des techniques de recommandations. Le principe du filtrage collaboratif est d'utiliser un certain type de données permettant d'identifier une corrélation entre plusieurs utilisateurs. Ce type

de données représente le degré d'appréciation d'un utilisateur sur des objets du système. Généralement ce degré d'appréciation est représenté sous forme de votes donnés sur les objets proposés par le système [Herlocker *et al.* 00] [O'Sullivan *et al.* 02].

Les étapes principales du filtrage collaboratif sont les suivantes :

- ✓ Chaque utilisateur donne un vote sur les différents objets proposés par le système.
- ✓ Le système identifie les utilisateurs les plus similaires en basant la *fonction de similarité* sur les votes. Si deux utilisateurs ont donné les mêmes votes ou des votes proches sur les mêmes objets, alors ces deux utilisateurs sont considérés comme « proches ».
- ✓ De nouvelles recommandations sont alors générées en fonction des utilisateurs les plus similaires. Deux types de recommandations peuvent êtres produits [Sarwar *et al.* 01] :
  - Une prédiction du vote que l'usager actif accorderait à un objet *i*.
  - Une liste de recommandations comprenant des objets potentiellement intéressants (et non encore évalués par l'usager) pour l'utilisateur actif. Ce type de résultat est désigné par l'expression d'une recommandation *Top-N*.

Breese et *al.* [Breese *et al.*, 1998] identifient deux classes principales d'algorithmes de filtrage collaboratif : les algorithmes fondés sur la mémoire et les algorithmes basés sur les modèles.

4.2.1.1 Algorithmes fondés sur la mémoire (*memory-based*)

Les algorithmes basés sur la mémoire utilisent tous les profils utilisateurs afin de générer de nouvelles prédictions.

En général, les algorithmes fondés sur la mémoire ont deux étapes :

- ✓ Rechercher quels sont les utilisateurs ayant voté de façon similaire les mêmes éléments que l'utilisateur courant (pour qui on souhaite faire des prédictions sur ses votes).
- ✓ Calculer ensuite, en fonction des votes des utilisateurs similaires, la ou les prédictions sur les objets que l'utilisateur n'a pas encore évalués.

Plusieurs techniques permettent de calculer la similarité entre deux usagers. Breese *et al.* [Breese *et al.*, 1998] citent deux techniques : *la similarité vectorielle* et la *corrélation de Pearson*.

Nous avons vu que la méthode du modèle vectoriel est une technique de recherche d'informations qui permet de mesurer la similarité entre deux documents : les documents sont représentés par des vecteurs de mots-clés et la formule du cosinus permet de calculer la similarité entre les deux vecteurs (cf. section 2.1.2.). Ce formalisme est utilisé dans le cas du filtrage collaboratif, en considérant qu'un utilisateur prend le rôle d'un « document », les noms des objets seront les mots-clés et les votes seront les poids donnés par les utilisateurs aux objets. Ainsi, la *similarité vectorielle* entre deux usagers peut être calculée par la formule du cosinus [Salton et McGill 83].

La *corrélation de Pearson* est une technique qui a été utilisée dans de nombreux systèmes de recommandation, et pour la première fois (dans le contexte des systèmes de recommandation) dans le projet GroupLens en 1994 [Breese, *et al.*, 1998] [Resnick *et al.* 94]. La corrélation entre deux personnes est définie par la formule suivante :

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v_a})(v_{i,j} - \overline{v_i})}{\sqrt{\sum_j (v_{a,j} - \overline{v_a})^2 \sum_j (v_{i,j} - \overline{v_i})^2}} \qquad \text{Équation 5}$$

où j représente chacun des objets ayant été à la fois voté par l'utilisateur a et l'utilisateur i.

La prédiction qu'un utilisateur a vote sur un objet j est calculée ensuite par la formule suivante :

$$p_{a,j} = \overline{v_a} + k\sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v_i}) \qquad \text{Equation 6}$$

où n est le nombre d'utilisateurs ayant un poids non égal à zéro, et k est un facteur de normalisation calculé de telle sorte que la somme des valeurs absolues des poids est égale à une unité c'est-à-dire 1.

4.2.1.2 Algorithmes à base de modèles (*model-based*)

Cette deuxième technique est basé sur « les profils utilisateurs pour estimer ou apprendre un modèle, qui sera ensuite utilisé pour faire des prédictions » (traduction de la définition donnée par [Breese *et al.*, 1998]) pour un utilisateur donné. Dans cette approche, un modèle est donc calculé, souvent de manière hors-ligne, permettant par exemple de regrouper des utilisateurs ayant des caractéristiques communes ; ce modèle est ensuite utilisé en ligne lors du calcul des prédications. Ainsi, contrairement aux algorithmes fondés sur les utilisateurs, tous les profils utilisateurs de la base de données ne sont pas utilisés mais simplement une partie représentative de cette base.

Plusieurs techniques existent pour construire des modèles, dont le *modèle des clusters* et le *modèle basé sur les réseaux bayesiens*.

4.2.1.3 Les limites du filtrage collaboratif

Les principales limites du filtrage collaboratif sont liées au calcul de corrélation entre les utilisateurs. En effet, ce calcul est basé sur des votes donnés à des objets en commun, et peut donc soulever deux problèmes, celui du « nouvel utilisateur » et celui du « nouvel item » [Burke 02]:

- ✓ Le problème du « nouvel utilisateur » : un nouvel utilisateur a généralement peu ou pas de votes, ce qui peut faire de lui un utilisateur non comparable avec aucun autre utilisateur du système. Et dans ce cas, cet utilisateur a peu de chances d'avoir des recommandations pertinentes.

- ✓ Le problème du « nouvel item » : un nouvel item qui n'a pas reçu beaucoup de votes est également un problème car ce nouvel item ne sera pas facilement recommandé (même si un utilisateur vote pour cet item, il a peu de chances d'avoir une corrélation avec d'autres utilisateurs).

Il existe également un autre inconvénient appelé *sparsity problem* ou problème d'espacement [Sarwar *et al.* 01] : si peu d'utilisateurs ont voté pour les mêmes objets, alors peu de corrélations peuvent être faites entre les utilisateurs. Ce problème survient dans des systèmes possédant beaucoup d'items et une base de données d'utilisateurs relativement petite.

Enfin, le problème de performance, notamment pour les algorithmes basés sur les utilisateurs, doit également être souligné : la complexité du temps de calcul augmente avec le nombre d'utilisateurs et le nombre d'objets [Sarwar *et al.* 01].

4.2.2. Le filtrage basé sur le contenu

Dans un système utilisant le filtrage basé sur le contenu, les recommandations sont faites à partir des objets que l'utilisateur a aimé dans le passé : le système lui suggère des objets ayant un contenu similaire.

Cette approche requiert une représentation des objets sous une forme qui puisse être utilisée pour la comparaison. Les objets doivent donc être décrits avec des caractéristiques communes.

L'inconvénient ici réside surtout dans la représentation des objets. En effet, la comparaison se base sur les caractéristiques des objets, il faut donc pouvoir comparer « correctement » les objets avec ceux enregistrés dans le profil utilisateur. Burke (2002) donne l'exemple de personnes aimant écouter aussi bien du jazz que de la musique classique. Un système basé sur le filtrage par contenu ne recommandera jamais des musiques classiques à des passionnés de jazz, étant donné que ces deux types de musique ne partagent pas les mêmes caractéristiques. Une manière de contourner ce problème serait d'enregistrer et d'utiliser comme critère de comparaison les préférences des utilisateurs.

4.2.3. Technique basée sur le raisonnement à base de cas

> *The basic philosophy in case-based recommendation is to recommend items that are similar to those items that the user has liked in the past.*
>
> Smyth et Cotter (1999)

Cette technique est très proche du filtrage par contenu. La différence réside dans la représentation des objets à comparer : dans la recommandation basée sur le raisonnement à base de cas, les objets sont représentés de manière structurée, alors que le filtrage collaboratif utilise des objets dont la représentation est difficilement structurable et est principalement textuelle [Aguzzoli *et al.* 02] [Bradley *et al.* 00].

4.2.4. Le filtrage basé sur les données démographiques

Ce type de système utilise les données personnelles de l'utilisateur afin de le catégoriser selon des classes démographiques, et lui faire ainsi des recommandations. L'avantage de cette méthode est qu'elle ne demande pas de garder un historique des votes des utilisateurs, et n'a donc pas les inconvénients liés à ce système de votes (problème du « nouvel utilisateur » et problème du « nouvel item »). Cependant, collecter les informations démographiques utiles peut poser des problèmes, qu'il s'agisse du problème de temps mis par l'utilisateur à fournir ces données, ou du problème des données privées ou confidentielles (spécialement dans le domaine du e-commerce).

4.2.5. Le filtrage basé sur l'utilité

Dans cette approche, chaque utilisateur a une fonction d'utilité qui lui est attribué. Le système se base alors sur le calcul de l'utilité de chaque objet pour faire des recommandations à un utilisateur. La difficulté est de trouver la fonction d'utilité. L'avantage est que la fonction peut ne pas dépendre du produit, c'est-à-dire qu'elle intègre des données autres que celles des objets recommandés.

4.2.6. Le filtrage basé sur « les connaissances »

Cette approche considère les besoins et préférences des utilisateurs pour en déduire des recommandations, avec comme particularité d'avoir des « connaissances fonctionnelles », c'est-à-dire que ces systèmes ont des connaissances sur comment un objet particulier peut correspondre aux besoins d'un utilisateur. Par exemple, le système « Entrée » [Burke 02] utilise ses connaissances sur la cuisine pour déduire des similarités entre des restaurants.

4.3. Les systèmes hybrides

Les systèmes de recommandations hybrides consiste à combiner au moins deux techniques de recommandations dans le but d'améliorer la performance du système en essayant de contourner les problèmes posés par une seule technique [Burke 02].

4.3.1. Pondération *(Weighted)*

Les résultats produits (votes ou scores) par différentes techniques de recommandation sont combinées de manière à produire une seule recommandation.

4.3.2. Technique à commutation (*Switching*)

Cette technique permet de choisir une technique de recommandation entre plusieurs selon certains critères. La détermination de la technique à utiliser dépend de la situation, le système doit donc définir les critères de commutation (dans quels cas utiliser une autre technique). Cependant, cela permet au système de connaître les forces et faiblesses des techniques de recommandations qui le constituent.

4.3.3. Technique mixte

La technique mixte permet de donner à l'utilisateur des recommandations provenant de plusieurs techniques simultanément [Smyth et Cotter 99]. Un exemple de technique mixte est l'utilisation des techniques de filtrage collaboratif et de filtrage par contenu. Cela permet d'éviter un des problèmes du filtrage collaboratif : le « démarrage à froid » du nouvel item. En effet, la technique basée sur le contenu permet d'obtenir des recommandations des nouveaux objets en se basant sur leurs descriptions.

4.3.4. Combinaison de caractéristiques

Dans cette technique, les caractéristiques des informations fournies par les différentes méthodes de recommandation sont combinées pour permettre l'utilisation d'une seule technique sur l'ensemble de ces données. Par exemple, le filtrage collaboratif utilise principalement les votes des utilisateurs comme sources de données, elles peuvent être ajoutées aux informations utilisées par le filtrage par contenu pour produire des recommandations.

4.3.5. Cascade

Cette technique hybride se déroule en deux étapes : une première technique est utilisée pour produire un ensemble de candidats potentiels, et la deuxième technique sert à raffiner les recommandations faites.

L'avantage de cette méthode hybride est que la deuxième technique ne servira que si les recommandations produites par la première nécessitent une discrimination supplémentaire (dans le cas où les recommandations sont trop nombreuses par exemple). Par contre, si la première technique donne trop peu de recommandations, ou si celles-ci sont déjà ordonnées pour permettre une sélection rapide, alors la deuxième technique ne sera pas utilisée.

Chapitre 5 : Conception et méthodologie

Après avoir passé en revue les différents systèmes de questions-réponses et de localisation d'experts, nous allons décrire l'approche adoptée pour le développement du système *HELP*. Dans ce chapitre, nous présentons l'architecture globale du système, le domaine d'application et sa représentation, ainsi que la description précise des différents modules.

5.1. Notre approche

Dans cette partie, nous donnons une description de HELP en termes d'objectifs, de fonctionnement, d'aspect pédagogique et de méthodologie utilisée.

5.1.1. Objectifs

Notre système a pour objectif principal la localisation des personnes ayant de l'expertise dans un domaine afin de les recommander aux utilisateurs qui en ont besoin. En particulier, nous prenons en compte les problèmes soulignés dans les études de gestion des connaissances (cf. introduction), notamment celui de l'expert sollicité en permanence pour les mêmes questions, ou encore celui de l'expertise volatile.

Par ailleurs, les systèmes décrits dans le chapitre 3 mettent en évidence l'importance de la localisation d'experts selon leurs domaines de compétences techniques. Mais ces systèmes ne prennent pas en compte la pédagogie employée par l'expert lorsqu'il répond à l'apprenant. L'aspect pédagogique est par contre très important pour nous. En effet, toute personne a sa propre méthode pour répondre à une question ou expliquer un concept. Ceci peut se faire en donnant des exemples ou des explications détaillées, en insistant sur les points importants, etc. Notre but est donc de recommander à l'apprenant des personnes dont la méthode pédagogique lui convienne, c'est-à-dire, qui lui permette de résoudre efficacement son problème.

Nous allons maintenant décrire le fonctionnement global du système HELP.

5.1.2. Fonctionnement global de HELP

Nous avons vu en introduction que chaque utilisateur peut être vu à la fois comme apprenant et comme expert : un utilisateur peut connaître certains aspects d'un domaine particulier, dans ce cas il est vu comme un expert sur ce domaine. En même temps sur les autres aspects de ce même domaine qu'il ne connaît pas ou peu, ou encore sur un autre domaine, il sera considéré comme un apprenant.

Dans notre système, un expert peut avoir 4 catégories de compétence : débutant, intermédiaire, avancé, et très avancé. Dans la suite de ce mémoire, le terme général « expert » désigne une personne indépendamment de sa catégorie.

Lors de sa première connexion au système, l'utilisateur précise, en plus de ses informations personnelles, ses domaines d'expertise. Lorsqu'il se trouve confronté à un problème, il a la possibilité d'envoyer au système une question décrivant son problème. Il est possible qu'un autre utilisateur ait déjà rencontré ce même problème et que celui-ci ait été résolu antérieurement. Dans ce cas, HELP va rechercher s'il existe des questions similaires dans la base de cas des experts potentiels. En effet, chaque utilisateur possède une base de cas des problèmes qu'il a résolus en tant qu'expert. Si une ou plusieurs questions similaires sont trouvées, celles-ci sont envoyées à l'utilisateur avec les solutions associées ainsi que des informations concernant les auteurs des dites solutions. Ceci permet d'éviter aux experts de recevoir toujours les mêmes questions.

Si aucune question similaire n'est trouvée, HELP envoie alors la requête à plusieurs experts qu'il détermine selon des techniques de recommandation. Lors de la recherche d'experts potentiels, HELP prend en compte aussi bien leurs *compétences techniques* que leurs *compétences pédagogiques*. Nous verrons dans la section suivante ce que nous entendons par compétence pédagogique. Lorsque l'apprenant reçoit une ou plusieurs

réponses provenant des experts, il a la possibilité de voter pour chaque expert selon des critères d'évaluation (cf. **Figure 1**). Une fois que l'apprenant a voté, la requête et sa solution sont enregistrées dans deux bases de cas : l'une est la *base de cas d'apprentissage* de l'apprenant, l'autre est la *base de cas d'expertise* de l'expert. Cette stratégie de questions-réponses permet d'une part à l'apprenant d'avoir de nouvelles connaissances qui seront stockées dans sa base de cas d'apprentissage, et d'autre part, elle permet de capturer une partie des connaissances de l'expert. Enfin cette stratégie associée au vote permet la localisation et donc la recommandation d'experts.

HELP permet une autre alternative à l'utilisateur : au lieu d'envoyer sa question, il peut demander au système de lui recommander une liste d'experts en fonction de leurs domaines d'expertise. Pour chaque expert recommandé, HELP associe une explication concernant la recommandation, ainsi que des informations telles que le nombre de questions auxquelles l'expert a répondu. Ainsi, l'utilisateur peut mieux cibler l'expert à contacter.

5.1.3. Fonctions principales accessibles aux utilisateurs

A partir de la description faite précédemment, nous pouvons dégager les fonctions principales du système. Chaque utilisateur a accès :

- à une messagerie lui permettant de gérer l'envoi et réception de requêtes,
- à une fonction d'élaboration d'une liste de contacts
- à un compte personnel regroupant ses informations d'identification et les données acquises (réponses aux requêtes, cas enregistrés, votes reçus, etc.) lors de l'utilisation du système.

5.1.4. Aspect pédagogique

Lorsqu'un utilisateur reçoit une réponse à une question qu'il a posée, il y a deux possibilités : soit il accepte la réponse parce qu'elle résout bien son problème, soit alors il juge que la réponse ne lui convient pas et par conséquent, la rejette. Nous avons voulu savoir pourquoi une réponse peut ne pas être acceptée par un utilisateur. Dans un contexte d'apprentissage, nous pensons qu'un apprenant qui a besoin d'une réponse précise à l'une de ses questions a besoin de comprendre rapidement la réponse et pour cela, il lui faut une réponse correspondant à sa façon d'apprendre. Un apprenant peut mieux comprendre à *l'aide d'exemples*, un autre a besoin de connaître parfaitement le sujet et donc a besoin d'une liste de références pour avoir une vue globale sur le sujet, un autre encore a simplement besoin d'une réponse courte et précise, etc. Il semble donc important de prendre en compte non seulement les compétences techniques de l'expert, mais également ses compétences pédagogiques, c'est-à-dire sa manière de répondre. Ce sont ces deux types de compétences (techniques et pédagogiques) qui influent sur le fait d'accepter ou non une réponse reçue.

La question qui se pose alors est : comment déterminer les compétences pédagogiques d'un expert ? Il est difficile de demander à l'expert de répondre lui-même à cette question car, seules ses interactions avec des apprenants nous paraissent adéquates pour déterminer les caractéristiques de sa pédagogie. L'approche que nous avons suivie est celle d'un système de votes : chaque apprenant qui reçoit une réponse d'un expert doit l'évaluer sur la base des critères de vote présentés dans la Figure 8. L'aspect pédagogique dans notre système est donc déterminé par la communauté des utilisateurs par l'intermédiaire des votes.

La question « Recommanderiez-vous cet expert ? » est un **vote global** que l'apprenant donne à l'expert. L'ensemble des votes globaux pour un expert représente son

degré de reconnaissance par tous les utilisateurs qui l'ont contacté lors de l'envoi de requêtes.

Ensuite, l'utilisateur explique la raison de son évaluation globale en donnant pour chacun des autres critères d'évaluation une note entre 1 et 5 (5 étant la meilleure note). Ces autres critères sont définis dans le cadre d'une « **évaluation pédagogique** ». Par exemple, si l'utilisateur donne une note égale à 5 à l'expert sur le critère « réponse rapide », cela signifie que l'utilisateur *estime* que l'expert a donné une réponse dans un temps jugé relativement court. De même, si l'expert reçoit une note égale à 1 ou 2 sur le critère « fournit des exemples », cela signifie que cet expert n'a en général pas l'habitude de donner des exemples, il ne sera alors pas recommandé aux personnes dont les préférences d'apprentissage se basent entre autres sur les exemples. Le critère « utilisation d'un vocabulaire adéquat » fait référence au langage qu'utilise l'expert pour expliquer sa réponse : en effet, si l'expert utilise un vocabulaire trop technique, cela peut rendre la compréhension de la réponse plus difficile, surtout si l'utilisateur n'a pas les mêmes prérequis que l'expert en termes de compétences techniques.

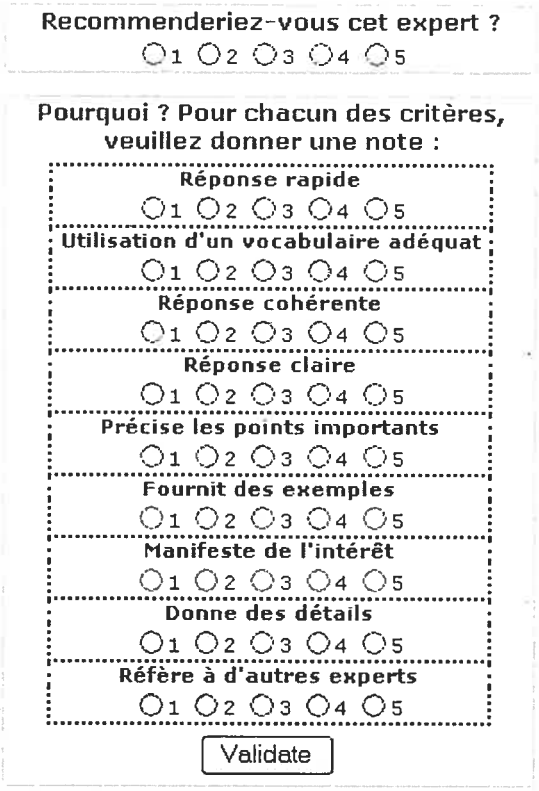


Figure 8 : Évaluation d'un expert

Nous voyons dans la Figure 8 que les votes définissent les caractéristiques de l'expert *du point de vue* de l'apprenant. Ainsi l'apprenant va pouvoir donner au système des informations concernant sa perception de la pédagogie de l'expert recommandé. Lors de prochaines recommandations, le système cherchera quels sont les experts qui ont été évalués de la même manière, soit par l'apprenant lui-même, soit par d'autres utilisateurs (qui ont reçu les mêmes votes), et pourra les recommander à l'apprenant. Ces critères ont été définis sur la base d'un questionnaire d'évaluation utilisé dans le cadre des cours donnés au Départ Informatique et de Recherche Opérationnelle de l'Université de Montréal.

Nous avons dit également que notre système va permettre à l'utilisateur de rechercher des experts selon des critères de recherche. La Figure 9 donne un aperçu des critères de recherche d'un expert. Ceux-ci dépendent d'une part de statistiques déduites des interactions entre les utilisateurs (nombre de questions posées, nombre de réponses envoyées ou reçues, etc.) mais également des votes donnés par un apprenant à un expert (« réponse rapide », « clarté de la réponse », etc.).

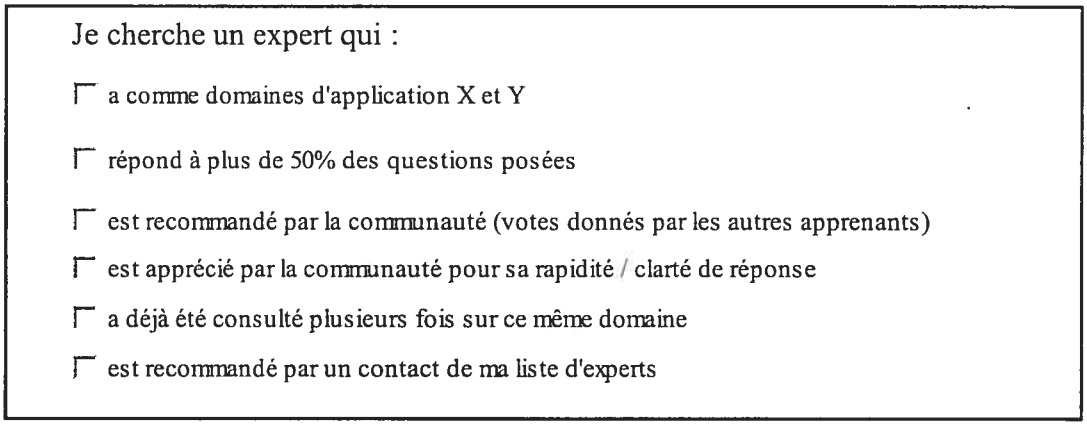


Figure 9 : Critères de recherche d'un expert

Les critères de recherche d'un expert permettent à l'utilisateur de spécifier ce qui lui parait important pour déterminer la valeur d'un expert. Par exemple, un utilisateur *u* considère qu'un expert qui répond à plus de 50% des questions qu'il reçoit, et qui de plus, de l'avis général, donne des réponses dans un temps relativement court, est un « bon » expert. C'est-à-dire que *u* va considérer que ces deux critères (« répond à plus de 50% des questions posées » et « est apprécié pour sa rapidité de réponse ») doivent faire partie des caractéristiques des experts qui lui seront proposés.

Cette liste non exhaustive propose des critères de sélection afin d'affiner si nécessaire les recherches des utilisateurs. Lorsque beaucoup d'experts sont recommandés, l'apprenant peut choisir ses préférences et cibler plus rapidement les experts qui sauront l'aider de manière efficace.

5.1.5. Application de la méthodologie dans HELP

L'un des principaux processus du système consiste à rechercher les questions similaires à celle posée par l'utilisateur. Nous avons utilisé l'approche de *raisonnement à base de cas textuels* en prenant comme cas d'expériences les paires de questions-réponses. Les phases principales du cycle de raisonnement à base de cas dans notre système sont les phases de *recherche* et de *maintenance*. En effet, nous n'utilisons pas de phase *révision* de la solution, celle-ci est présentée directement à l'utilisateur. Ce dernier a ensuite la possibilité d'accepter la solution, de demander plus de détails à l'auteur de la réponse s'il est disponible, etc. Si aucune solution trouvée dans la phase *recherche* ne convient à l'utilisateur, la question est ensuite envoyée aux experts les plus susceptibles de donner une réponse.

Dans un tout autre processus, nous utilisons une technique hybride mixte (cf. chapitre 4 : les systèmes de recommandation, section 4.3). La technique hybride mixte consiste à donner les recommandations produites par les deux techniques : *filtrage collaboratif* et *raisonnement à base de cas*. Comme nous l'avons dit, nous pensons que, dans un contexte académique, l'aspect pédagogique joue un rôle important dans la recherche d'experts appropriés. Nous prenons donc en compte plusieurs critères de vote qui serviront par la suite à l'utilisateur à rechercher quels sont les experts qui lui conviennent le mieux, c'est-à-dire quels sont les critères qui lui permettront de mieux comprendre les réponses de l'expert. Un apprenant appréciera toujours de se voir recommander des experts ayant les mêmes caractéristiques que ceux qu'il a aimés dans ses précédentes interactions ; c'est pour cette raison que nous utilisons la technique de raisonnement à base de cas. Mais se baser uniquement sur l'expérience d'un utilisateur ne nous paraît pas suffisant, nous pensons qu'il faut aussi exploiter les opinions des autres utilisateurs sur l'expert afin d'obtenir une meilleure appréciation de ses compétences. Nous avons par conséquent également choisi la méthode du filtrage collaboratif.

Dans les sections suivantes, nous donnons des détails sur la conception de HELP à travers son architecture et la description fonctionnelle de ses différents composants.

5.2. Architecture de *HELP*

L'architecture (cf. Figure 10) décrit toutes les fonctionnalités qu'un utilisateur peut obtenir de HELP.

5.2.1. Fonctionnalités du système

Un apprenant peut :

- *Envoyer une requête* sur un domaine précis : HELP recherche s'il existe des cas similaires, et dans ce cas, affiche les solutions correspondantes à l'utilisateur. S'il n'existe pas de réponses similaires, ou si l'utilisateur n'est pas satisfait, le système recherche des experts susceptibles de répondre à sa question.

- *Rechercher directement des experts*. L'apprenant peut préciser ses critères de recherche et obtenir des recommandations d'experts en fonction de ces critères et en fonction également des performances *techniques* et *pédagogiques* de l'expert.

- *Rechercher des apprenants similaires à lui-même*. De la même manière, l'apprenant peut préciser certains critères de recherche, et former ensuite une communauté d'apprenants en fonction des recommandations qui lui seront faites.

Un expert peut :

- *Gérer les requêtes* qu'il reçoit : il peut y répondre directement, les transférer à d'autres experts de sa connaissance qui n'ont pas encore été localisés par le système, indiquer au système les requêtes non pertinentes à son domaine d'expertise.

- *Rechercher d'autres experts* selon des critères de recherche et former ainsi une communauté d'experts.

Les communautés d'apprenants et d'experts ont pour but d'aider à étendre le réseau social dans l'organisation. Rappelons que le réseau social d'une personne (cf. chapitre 3) consiste en la formation d'un réseau de personnes se connaissant. Ce réseau peut s'étendre au fur et à mesure dans le temps mais peut avoir certaines limites, telles expliquées dans le chapitre 3. HELP permet donc de mettre en relation des personnes selon des affinités, c'est ce que nous avons défini comme étant les communautés d'experts ou d'apprenants. Un utilisateur pourra plus facilement, par exemple, sélectionner des personnes pour former un groupe de travail sur un projet.

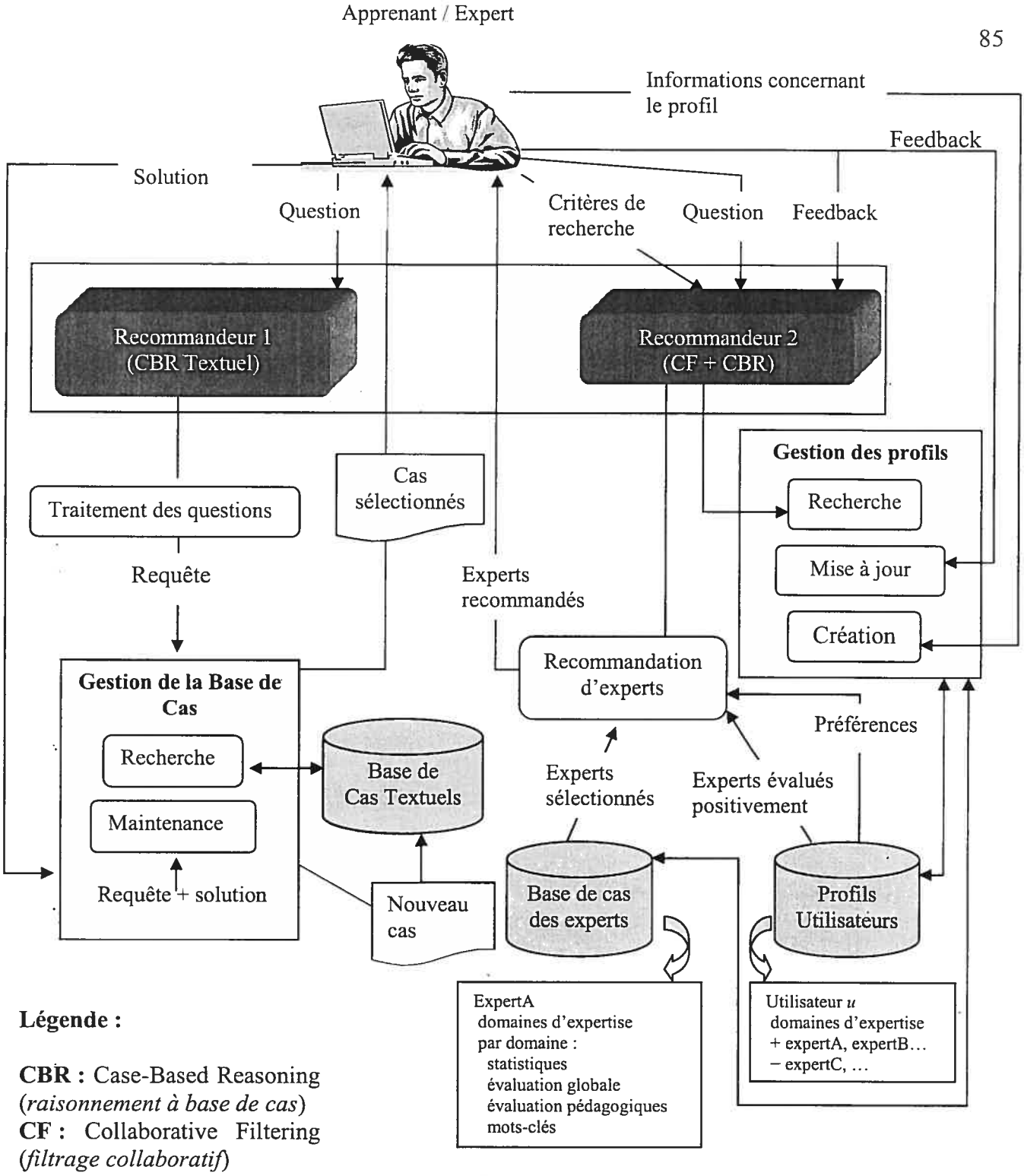


Figure 10 : Architecture de *HELP*

L'architecture présente les différents composants du système HELP. Elle se compose de deux types de recommandeurs : le premier (*Recommandeur 1*) permet de rechercher des requêtes similaires à la question de l'utilisateur. La question posée est alors envoyée au module *traitement des questions* qui la transforme en requête (cf. section 5.5). Les sous-modules *recherche* et *maintenance* du module *Gestion de la Base de Cas* permettent respectivement de rechercher des cas similaires à la requête et de rajouter de nouveaux cas textuels dans la base de cas lorsqu'un expert répond à une question. La méthode utilisée est le CBR Textuel (*case-based reasoning* ou raisonnement à base de cas, modèle textuel (cf. chapitre 1)).

Le deuxième recommandeur (*Recommandeur 2*) permet de recommander des experts en fonction de la question de l'utilisateur ou de critères de recherche (cf. section 5.1.4). Les techniques utilisées sont le filtrage collaboratif (CF. : *Collaborative Filtering*) et le raisonnement à base de cas (CBR : *Case-Based Reasoning*). Ces techniques nécessitent de prendre en compte les experts qui ont été sélectionnés et évalués positivement (vote significatif à partir d'un certain seuil que nous avons pris égal à 3 dans notre cas, puisque les votes sont compris entre 1 et 5), et d'aller rechercher d'autres experts similaires dans la base de cas des experts. L'architecture montre deux bases de données : la base de cas des experts, qui contient l'ensemble des experts susceptibles d'être recommandés, et la base des utilisateurs qui contient l'ensemble des profils utilisateurs. Physiquement, la base des utilisateurs et la base des experts constituent une seule et même base puisque chaque utilisateur est un expert potentiel. Cependant, nous séparons logiquement la base des utilisateurs en deux bases afin de marquer la différence entre les utilisateurs du système et ceux considérés comme experts en fonction de la requête courante.

5.2.2. Environnement de l'utilisateur

Chaque utilisateur de HELP doit s'enregistrer avant de pouvoir accéder aux différentes fonctionnalités du système.

Nous expliquerons plus en détails ce que contiennent les bases d'apprentissage et d'expertise de l'utilisateur dans la section 5.7. La liste de contacts contient d'une part les apprenants ou experts faisant partie de sa communauté ; d'autre part, les experts ayant répondu à ses questions sont aussi gardés dans sa liste de contacts (liste d'experts).

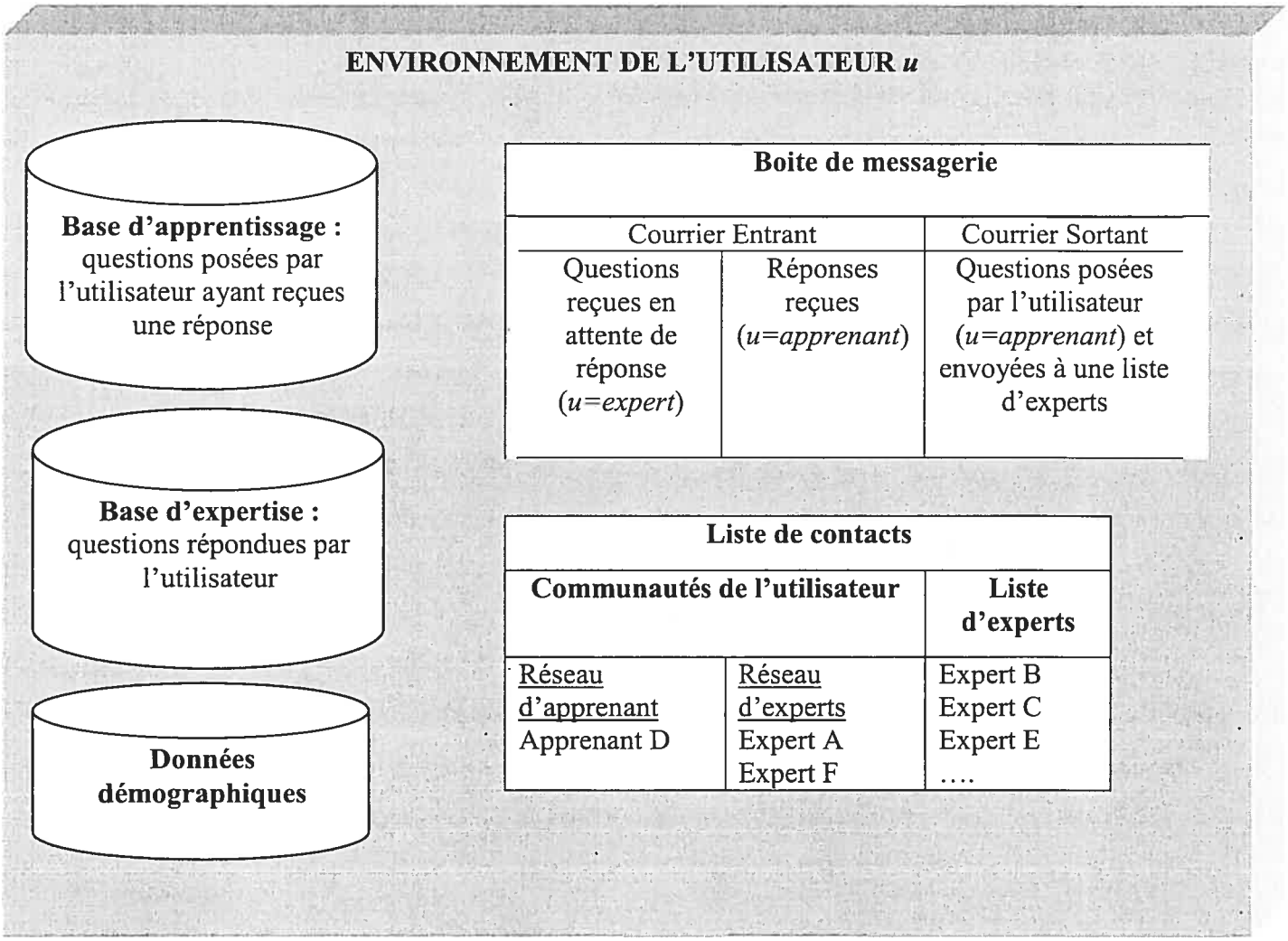


Figure 11 : Environnement d'un utilisateur de *HELP*

La boite de messagerie permet à l'utilisateur de gérer ses interactions avec les autres utilisateurs du système. Il doit gérer :

- ✓ les questions qu'il reçoit en tant qu'expert : y répondre, les transférer à une personne de sa connaissance qu'il pense être capable de répondre à la question, ou tout simplement notifier au système et à l'auteur de la requête que celle-ci ne relève pas de son domaine.

- ✓ les questions qu'il envoie en tant qu'apprenant.

- ✓ les réponses qu'il reçoit en tant qu'apprenant. Lorsqu'un utilisateur envoie une requête, le système sélectionne automatiquement les experts pertinents en fonction du domaine concerné par la question, et en fonction des critères pédagogiques enregistrés dans les préférences de l'utilisateur. Ensuite les experts ont la possibilité de répondre, l'utilisateur consulte les réponses et vote pour chaque auteur des réponses reçues.

- ✓ les réponses qu'il donne en tant qu'expert. Dès qu'un utilisateur reçoit une requête, nous avons vu qu'il doit y répondre s'il le peut. L'expert sera ensuite évalué par l'apprenant (cf. Figure 9).

Dans les sections suivantes, nous allons parler du domaine d'application, ainsi que des différents modules qui composent l'architecture.

5.3. Structuration du domaine d'application

Nous avons choisi de structurer le domaine d'application de manière hiérarchique. En Figure 12 est présentée la hiérarchisation du domaine d'application. Il est important de mentionner que le système peut s'appliquer à plusieurs domaines.

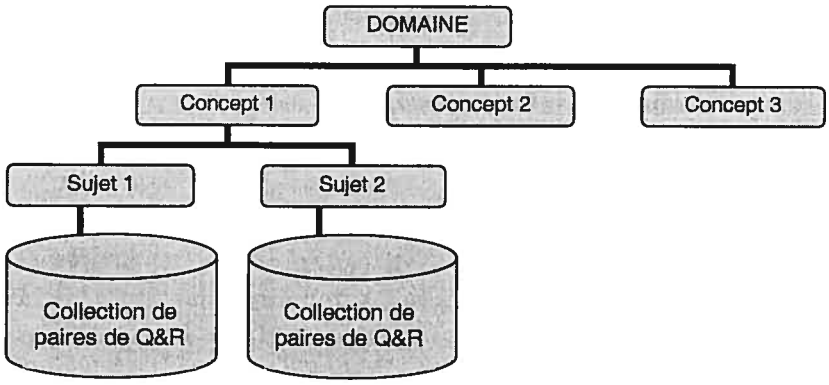


Figure 12 : Représentation hiérarchique du domaine d'application

A chaque concept du domaine d'application sont associés un ou plusieurs sujets, et à chaque sujet est associée une collection de paires de Questions et Réponses (Q&R) correspondantes. Il peut exister des concepts sans sujets associés et dans ce cas, la collection de paires de questions et réponses se rattache directement au concept.

Les paires questions et réponses sont les questions qui ont été posées par un utilisateur et répondues par un expert. Nous verrons dans les sections suivantes que ces bases correspondent en fait à une seule base de cas textuels.

5.4. Base de cas textuels

La base de cas textuels se compose de cas textuels semi-structurés construits à partir de la requête de l'utilisateur et de la solution associée. Un cas est constitué d'une partie « description de problème » et « description de la solution » (cf. chapitre 1). Généralement, la partie « description de problème » est structurée en paires {attribut, valeur}. Cependant, certains cas ne peuvent être complètement structurés car une partie de leur description n'est disponible que sous forme textuelle : ce sont les cas textuels semi-structurés.

La partie description de cas comporte les éléments suivants :

- Auteur : auteur de la requête
- Date : date à laquelle la requête a été envoyée au système
- Concept et éventuellement sujet relatif à la requête
- Libellé de la requête
- Description du problème
- Vecteurs de termes associés. Ce vecteur de termes est dérivé du traitement effectué sur le libellé de la requête : le vecteur contient des mots-clés extraits du libellé de la requête et sont associés à un poids mesurant leur importance. Ce vecteur permettra ensuite de calculer la similarité entre la requête et les cas de la base, ce que nous verrons plus en détails dans la section suivante.

La partie solution de cas comporte les éléments suivants :

- Auteur : auteur de la solution
- Date : date à laquelle la solution a été envoyée à l'utilisateur
- Description de la solution

5.5. Module de traitement des questions

Ce module est responsable de l'indexation des questions. L'indexation est effectuée par l'extraction des mots-clés, la question ayant été préalablement filtrée par rapport à une liste de mots usuels et ayant fait l'objet de lemmatisation (*stemming*).

Nous avons décidé d'utiliser la technique du modèle vectoriel, décrite dans la section 2.1.2, dont les expériences sur divers systèmes ont démontré l'efficacité. L'indexation de la requête comporte plusieurs étapes :

- *Tokenisation* : ce traitement consiste à découper la question en mots (ou *tokens*).

- *Filtrage des mots dits usuels* : certains mots sont considérés comme non significatifs car ils n'ont pas d'utilité en tant qu'index. Les mots tels que « the », « like » etc. n'ont pas d'utilité pour la compréhension de la phrase. On utilise donc une « stoplist » contenant les mots à ne pas retenir : tous les mots appartenant à la « stop-list » sont simplement supprimés de la requête de l'utilisateur.

- *Lemmatisation* : elle consiste à transformer les mots conjugués afin de les ramener à une même racine. Des études ont montré l'efficacité de l'algorithme de Porter [Porter 80]. C'est donc cet algorithme que nous avons utilisé.

- Les mots restants peuvent être considérés comme des index et reçoivent une pondération. Nous utilisons la mesure de pondération TF×IDF (*Term Frequency* × *Inverse Document Frequency*), décrite dans la section 2.1.2.

Nous avons décrit en section 2.1.2 le modèle vectoriel de Salton, qui permet de construire un vecteur de termes pondérés. Voici un exemple de l'application de ce modèle, que nous utilisons dans notre système.

Soient C1, C2, C3 trois cas textuels contenus dans la base de cas, donc la partie « description de problème » ne contient que deux attributs : la question de l'utilisateur posée en langage naturel et le vecteur de mots-clés associé à cette requête.

Question 1: *how to compile servlets and run servlets?*

Question 2: *I have to send an e-mail to an admin[...]. I'm having problems with sending the e-mail trough a servlet.*

Question 3: *I use an HTML form. I need to use either servlets or JSP to generate an XML file. This XML file should contain the choices made by the user and for every user selection made on the HTML form, this XML file should automatically change.*

Les mots soulignés correspondent aux mots appartenant au vocabulaire d'indexation. Nous allons considérer que ces trois questions ont été préalablement traitées : suppression des mots « inutiles » (tous les mots non soulignés), et lemmatisation des mots restants (par exemple les mots *send* et *sending* sont ramenés à leur racine *send*). Les vecteurs sont de dimension $n = 8$ puisqu'il y a 8 index (les mots soulignés) différents.

Le Tableau 2 indique le nombre d'occurrences de chaque index dans les trois cas, c'est-à-dire dans les trois questions de la base. A partir de ce tableau, on calcule le poids de chaque index (ou terme) pour chaque question en appliquant l'équation 4 (chapitre 2, section 2.1.2).

Tableau 2 : Nombre d'occurrences de chaque index dans les requêtes 1, 2 et 3

	Question 1	Question 2	Question 3
Compile	1	0	0
Run	1	0	0
Servlet	2	1	1
Send	0	2	0
E-mail	0	2	0
HTML form	0	0	2
JSP	0	0	1
XML File	0	0	3

Par exemple, le calcul du poids $w_{compile,Question1}$ du terme *compile* dans la question 1 donne :

$$w_{compile,Question1} = tf_{compile,Question1} \times idf_{Question1} = \frac{1}{2} \times \log 3 = 0.24$$

Le Tableau 3 montre le résultat des calculs des poids de chaque terme.

Tableau 3 : Vecteurs de termes pondérés des 3 requêtes

	Question 1	Question 2	Question 3
Compile	0.24	0	0
Run	0.24	0	0
Servlet	0	0	0
Send	0	0.48	0
E-mail	0	0.48	0
HTML form	0	0	0.32
JSP	0	0	0.16
XML File	0	0	0.48

Si nous comparons les lignes 3 issues du Tableau 2 et du Tableau 3, nous remarquons que le mot *servlet* apparaît au moins une fois dans toutes les questions (Tableau 2), alors que son poids est égal à 0 (Tableau 3). Ceci s'explique par le calcul de la fréquence documentaire inverse (section 2.1.2) : en effet, selon sa définition, plus un mot est rare dans l'ensemble des documents, plus il est discriminant pour le document auquel il appartient. Nous pouvons alors nous permettre de dire que le mot qualifie le document. Ici

le mot *servlet* apparaît dans l'ensemble des questions de la base, il ne peut donc être un terme discriminant. Les trois colonnes qui correspondent respectivement aux questions 1, 2 et 3 sont les vecteurs de termes pondérés des trois requêtes initiales. Nous obtenons alors les trois cas C1, C2, C3 dans la base de cas (on suppose que chaque requête a une solution).

Nous allons maintenant voir un exemple d'application de la formule du cosinus (chapitre 2, section 2.1.2). Prenons l'exemple d'une nouvelle requête Q :

> Q : *I want to send data from an HTML form into a XML file. How to generate an XML file through JSP or servlet?*

Nous cherchons à savoir s'il existe une requête dans la base de cas similaire à la requête Q. Il faut donc traiter la requête (suppression des mots « inutiles » et lemmatisation des mots restants). Nous procédons de la même manière en calculant le nombre d'occurrences des mots-clés de la requête Q et ensuite on applique l'équation 4 (chapitre 2, section 2.1.2). Le résultat est donné dans le Tableau 4.

Nous appliquons maintenant la formule du cosinus (équation 1, chapitre 2, section 2.1.2) pour connaître la similarité entre la requête de l'utilisateur et chaque cas de la base : nous calculons la similarité entre le vecteur de mots-clés de la question et chaque vecteur de mots-clés de tous les cas de la base.

Similarité (Q, Question 1) = 0

Similarité (Q, Question 2) = 0.20

Similarité (Q, Question 3) = **0.98**

Tableau 4 : Vecteur de mots-clés pondérés de la nouvelle requête

	Question Q
Compile	0
Run	0
Servlet	1
Send	1
E-mail	0
HTML form	1
JSP	1
XML File	2

Question Q
0
0
0
0.24
0
0.24
0.24
0.48

La question la plus similaire est de toute évidence la question 3. Ci-dessous le détail des calculs de la similarité entre la requête Q et la question 3 (application de l'équation 1).

$$\text{Similarité (Q, Question 3)} = \frac{0.24\times0.32 + 0.16\times0.24 + 0.48\times0.48}{\sqrt{0.32^2 + 0.16^2 + 0.48^2} \times \sqrt{0.24^2 + 0.24^2 + 0.48^2}}$$

Cet exemple montre comment rechercher des cas similaires à la requête courante. Les cas les plus similaires seront retournés à l'utilisateur, avec la solution associée.

5.6. Module de gestion de la base de cas

Le module qui gère la base de cas contient deux phases du raisonnement à base de cas (cf. chapitre 1).

La phase de *recherche* permet de rechercher les cas les plus similaires à la requête de l'utilisateur. Comme nous l'avons dit plus haut, chaque cas contient un vecteur de termes pondérés le représentant. La similarité entre chaque cas de la base et le *cas cible* (la

question de l'utilisateur analysée par le module de traitement des questions) est calculée en utilisant la *formule du cosinus* (cf. section 2.1.2), qui mesure le degré de similarité entre deux vecteurs de mots-clés. Cette phase est cruciale pour le *Recommandeur 1* car c'est celle qui permettra de recommander des cas similaires à la requête courante et donc de donner des solutions qui pourront aider à résoudre le problème de l'utilisateur. Dans le cas où la phase de recherche ne donne aucun résultat ou que les cas similaires trouvés ne sont pas retenus par l'apprenant, la requête est alors envoyée au module *Recommandeur 2*.

La phase de *maintenance* permet de rajouter un nouveau cas dans la base de cas textuels. Étant donné qu'un cas se compose d'une requête et d'une solution, c'est seulement lorsqu'un expert a répondu à la question et a été évalué par l'utilisateur, auteur de la question, que le cas est rajouté dans la base de cas. Nous avons vu en section 5.1.4 ce que signifie *évaluer un expert* : il s'agit de lui attribuer des notes entre 1 et 5 à chacun des critères énoncés sur la Figure 8.

5.7. Base des profils utilisateurs

Dans cette section, nous allons présenter quelle est la composition d'un profil utilisateur (cf. Figure 13), et l'utilité de ses différentes données.

La Figure 13 est une reprise de l'environnement de l'utilisateur (Figure 11) mais ici on ne s'intéresse qu'au profil de l'utilisateur et à ses composantes.

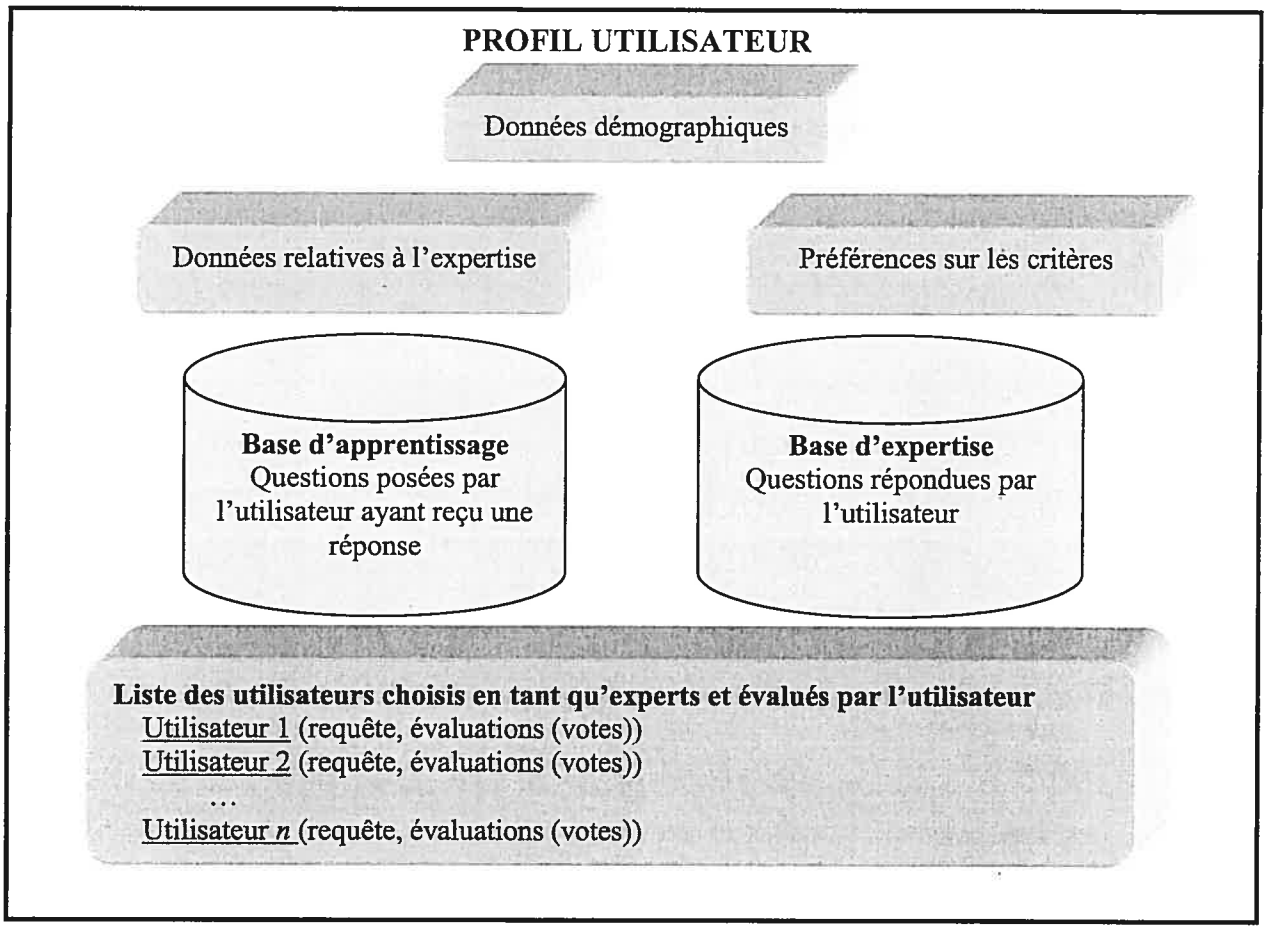


Figure 13 : composantes du profil utilisateur

5.7.1. Les données démographiques

Les données démographiques correspondent aux données personnelles de l'utilisateur : nom, prénom, adresse e-mail, pseudonyme et mot de passe. C'est par le pseudonyme que l'utilisateur est identifié par le système et donc par les autres personnes utilisant le système. Le pseudonyme doit donc être unique.

5.7.2. Les données relatives à l'expertise

Dans la phase d'enregistrement, l'utilisateur fournit ses domaines d'expertise au système. Puis, au fur et à mesure de ses interactions avec les autres utilisateurs, HELP met son profil à jour à travers les éléments suivants :

- Des statistiques, comme par exemple, le rapport entre le nombre de questions répondues sur le nombre de questions posées, ou encore le pourcentage de réponses évaluées positivement, etc.
- Les votes reçus des autres utilisateurs en fonction du domaine.
- Les mots-clés déduits des questions auxquelles il a répondu.

5.7.3. Les bases d'expertise et d'apprentissage

Lorsque l'utilisateur, en tant qu'apprenant, reçoit une réponse à sa question, il évalue l'expert qui lui a répondu, et l'ensemble {question, solution} est transformé en cas et est stocké dans la base de cas textuels. De plus, le cas est enregistré dans la **base d'apprentissage** de l'apprenant. Cette base d'apprentissage contient l'ensemble des cas dans lesquels sont contenus les requêtes résolues de l'apprenant. A tout moment, l'apprenant peut consulter cette base, pour se remémorer la solution, pour adapter lui-même la solution à un autre problème qui lui semble similaire, pour contacter de nouveau l'expert, etc.

De même, l'utilisateur en tant qu'expert possède une **base d'expertise** dans laquelle sont stockés tous les cas qu'il a solutionnés. Lorsqu'une question est envoyée à un expert, le système recherche d'abord s'il existe une requête similaire dans sa base d'expertise et si tel est le cas, la solution est envoyée directement à la personne qui est à l'origine de la question. A chaque question, sont associés non seulement la solution, mais aussi les votes qui ont été donnés à l'expert au moment de la consultation de la réponse par l'apprenant.

Ces votes reçus sont très importants car ils vont permettre de « définir » les caractéristiques pédagogiques de l'expert.

5.7.4. Préférences sur les critères de recherche

Nous avons vu en section 5.1.3 ce que signifie l'aspect pédagogique dans notre système. Lorsque l'utilisateur recherche des experts, nous avons vu qu'il peut choisir plusieurs critères de recherche (cf. Figure 9). Si par exemple, l'utilisateur recherche des experts donnant une « réponse rapide » et répondant « à plus de 50 % des questions posées », ces paramètres seront enregistrés dans son profil avec un poids associé. Le poids sera calculé selon la formule :

$$w(critère_i) = \frac{nb_i}{\sum_k nb_k}$$

où nb_i (resp. nb_k) est le nombre de fois que le critère i (resp. k) a été sélectionné par l'utilisateur. On calcule donc le nombre de fois qu'un critère a été choisi sur le nombre total de critères sélectionnés. Cela permettra de savoir les critères que l'utilisateur considère plus importants pour la recherche d'experts.

Ces préférences sont ensuite utilisées pour affiner les recommandations d'experts, comme nous le verrons dans la section 5.8.

5.7.5. Liste des experts évalués par l'utilisateur

Dans le profil utilisateur, HELP mémorise toutes les interactions de l'utilisateur avec les experts qu'il a évalués, c'est-à-dire ceux dont il a considéré la réponse. Outre le pseudonyme de l'expert, le système enregistre la requête et les évaluations données. Ce

sont principalement ces données et le contenu de la base d'expertise qui vont servir dans les processus de recommandation.

5.8. Recommandations d'experts

Lorsqu'un apprenant envoie une requête au système, il peut ne pas y avoir de requêtes similaires ayant déjà été résolues. Dans ce cas, il faut rechercher les experts appropriés pour répondre aux questions. Comme nous l'avons dit dans les sections précédentes, nous souhaitons que la recommandation d'experts se base sur certains *critères pédagogiques*. Un apprenant doit être en contact avec une personne qui a non seulement les compétences techniques pour lui répondre, mais également qui est capable de faire comprendre sa réponse. De plus, avoir l'avis de l'apprenant sur l'expert nous paraît également important afin que le système améliore les futures recommandations. Pour cela, à chaque réponse qu'envoie un expert, l'apprenant doit l'évaluer sur les questions illustrées en Figure 8.

L'expert est ainsi évalué par plusieurs personnes et par la suite, ces votes peuvent servir pour la recommandation. Nous utilisons une technique hybride de recommandation, basée sur le filtrage collaboratif et le raisonnement à base de cas.

Le filtrage collaboratif se base sur les votes donnés par les utilisateurs du système : l'utilisateur cible (pour lequel on souhaite faire des recommandations) est comparé à tous les utilisateurs ou un ensemble d'utilisateurs du système en se basant sur les votes fournis par chacun. On recherche donc des corrélations entre les utilisateurs par rapport aux évaluations données sur les objets enregistrés dans chaque profil utilisateur (*corrélation de Pearson)*. Pour cela, on ne considère que le vote donné à la question : « Recommanderiez-vous cet expert ? ». L'utilisateur cible a donc un ensemble de « voisins proches » et les objets qui se trouvent dans leurs profils et absents dans le profil de l'utilisateur cible peuvent lui être recommandés.

Dans le cas de la recommandation basée sur le raisonnement à base de cas, sachant qu'un utilisateur a aimé un ou plusieurs experts dans le passé, le système recherche des experts similaires à ceux qu'il a aimés selon certains critères et les lui recommande. Quels sont ces critères ? En fait, il s'agit de toutes les données qui caractérisent un utilisateur en tant qu'expert : d'une part les votes qu'il a reçus, et d'autre part toutes les autres données : nombre de questions reçues auxquelles il a répondu, nombre de questions qu'il a transférées à d'autres expert, etc.

De plus, le système permet à l'utilisateur de rechercher des experts selon plusieurs critères de sélection illustrés en Figure 9. A partir de ces critères, nous appliquerons le filtrage collaboratif sur les votes correspondant aux critères.

5.8.1. Recommandation basée sur le filtrage collaboratif

Comme nous l'avons vu, le filtrage collaboratif se base sur les votes donnés par les apprenants. Nous allons donner un exemple simple de recommandations basées sur le filtrage collaboratif. Normalement (cf. chapitre 4 : les systèmes de recommandation), les votes sont regroupés dans une matrice « utilisateurs × objets ». Or, ici les objets sont les « experts » eux-mêmes. En effet, notre but est de savoir si un expert peut être recommandé à un utilisateur u en considérant les similarités de u avec les autres utilisateurs (dont l'expert lui-même). Le Tableau 5 montre ce que nous venons d'énoncer.

Tableau 5 : Matrice des votes donnés par les utilisateurs aux experts

		←---------- Objets ----------→				
		Alice	Marc	David	Carole	Sarah
Utilisateurs	Alice		5	4	1	2
	Marc	3		2	5	5
	David	4	3		5	4
	Carole			3		1
	Julie	4	3	?	5	?
	Peter	4	3	2	5	1
	Claire	4	3		5	4
	Sarah	5	5		2	

Les votes donnés dans ce tableau sont ceux qui correspondent à la question « Recommanderiez-vous cet expert ? » Nous voyons qu'il y a les mêmes noms du coté « utilisateurs » et du coté « objets », ce qui est normal car Alice par exemple, peut avoir voté pour Marc en tant qu'expert. Par contre, dans la matrice, une case vide correspond au fait qu'il n'y a eu aucun vote donné : soit c'est un expert qui n'a pas encore reçu de vote de la personne (c'est le cas de Carole qui n'a pas voté pour Alice et Marc), soit il s'agit des utilisateurs qui ne doivent pas voter pour eux-mêmes (Alice ne votera jamais pour l'expert « Alice »).

Nous considérons dans cet exemple que les experts : Alice, Marc, David, Carole et Sarah ont le même domaine d'expertise : l'intelligence artificielle.

A partir de ces votes, nous allons déterminer la prédiction de Julie sur l'expert « Sarah », c'est-à-dire quel serait le vote que donnerait Julie à l'expert Sarah. Cela nous

permettra de savoir si l'expert Sarah peut être recommandé ou non à Julie. Dans le tableau 5, les points d'interrogation « ? » indiquent les prédictions que nous cherchons à calculer.

Selon les formules décrites dans la section 4.2.1., nous allons d'abord calculer la moyenne des votes de chaque utilisateur :

$$\overline{v_{Alice}} = \frac{12}{4} = 3 \qquad \overline{v_{Marc}} = 3.75 \qquad \overline{v_{David}} = 4 \qquad \overline{v_{Carole}} = 2$$

$$\overline{v_{Julie}} = 4 \qquad \overline{v_{Peter}} = 3.75 \qquad \overline{v_{Claire}} = 4 \qquad \overline{v_{Sarah}} = 4$$

Nous calculons ensuite la similarité selon la formule de la corrélation de Pearson (cf. section 4.2.1, Équation 4) entre l'utilisateur « Julie » et les autres utilisateurs. Les résultats sont donnés ci-dessous :

w(*Julie, Alice*) = -0.8944

w(*Julie,Marc*) = 0.3402

w(*Julie, David*) = 1.000

w(*Julie, Peter*) = 0.3950

w(*Julie, Claire*) = 1.000

w(*Julie, Sarah*) = - 0.8658

La corrélation entre Julie et Carole n'a pas été calculée tout simplement parce que Julie et Carole n'ont aucun vote en commun, elles n'ont pas voté pour les mêmes experts. Elles ne sont donc pas corrélées. De plus nous voyons que la corrélation entre Julie et David est égale à 1, de même que la corrélation entre Julie et Claire. Cela signifie que ces deux personnes (David et Claire) sont vraiment très proches, du point de vue des votes, de Julie. Si nous observons le Tableau 5, nous voyons en effet que David et Claire ont donné

les mêmes votes ou sensiblement les mêmes que ceux donnés par Julie, ce qui n'est pas le cas d'Alice par exemple.

Nous pouvons maintenant calculer la prédiction de Julie sur l'expert « Sarah » : quel vote donnerait Julie à Sarah. Nous appliquons l'équation 5 (chapitre 4, section 4.2.1), et le résultat est

$$P_{Julie,"Sarah"} = 4 + \frac{1}{3.6296}[-0.8944*(2-3) + 0.3402*(5-3.75) + 1.000*(4-4) + 0.3950*(1-3.75) + 1.000*(4-4)]$$

$$P_{Julie,"Sarah"} = 4.0642$$

On en déduit que Julie aurait de grandes chances de voter 4 sur l'expert « Sarah ». Le système recommandera donc Sarah à Julie en donnant l'explication associée, à savoir « Nous vous recommandons Sarah, car elle a été apprécié de manière générale (vote global de 4) par des personnes qui ont donné les mêmes votes que vous aux mêmes experts ».

Cet exemple montre le cas d'une recommandation basée uniquement sur le critère de sélection du domaine, c'est-à-dire que l'utilisateur a seulement voulu rechercher un expert dans un domaine particulier. Mais nous avons vu que l'utilisateur peut choisir d'autres critères de sélection.

Prenons comme exemple que Julie ait choisit le domaine X et le critère « réponse rapide ». Dans ce cas, au lieu d'appliquer le filtrage collaboratif sur les votes donnés à la question « Recommanderiez-vous cet expert », nous allons considérer les votes donnés pour « réponse rapide ».

5.8.2. Recommandation basée sur le raisonnement à base de cas

L'idée générale ici est de réutiliser les précédentes interactions de l'utilisateur avec le système afin de lui recommander une liste d'experts susceptibles de lui apporter l'aide ou les conseils dont il a besoin.

Dans notre cas, la recommandation basée sur le raisonnement à base de cas consiste à recommander des experts dont les caractéristiques sont similaires à ceux que l'utilisateur a aimés dans le passé, c'est-à-dire à ceux qui ont reçu un vote global de 4 ou 5 (experts évalués positivement) en fonction du domaine. Le profil utilisateur contient de nombreuses informations concernant ses précédentes interactions avec HELP. La base de cas des experts dans la Figure 10 contient l'ensemble des utilisateurs en tant qu'experts. Si l'on prend l'exemple de l'expert A (Figure 10), celui-ci est défini par ses domaines d'expertise et en fonction de chaque domaine : ses statistiques (nombre de questions posées, répondues, etc.), son évaluation globale, son évaluation pédagogique, et ses termes déduits des questions qu'il a reçues et résolues. Cet ensemble de caractéristiques va permettre de rechercher les experts ayant des caractéristiques communes dans la base de cas des experts.

Les étapes de la recommandation sont celles-ci :

- Le système établit les caractéristiques des experts enregistrés dans le profil de l'utilisateur.
- Le système établit également les caractéristiques des experts de la base des profils utilisateurs selon le domaine choisi.
- Une similarité est calculée entre les experts préférés de l'utilisateur et ceux qu'il n'a pas encore choisis. La méthode utilisée est la méthode des plus proches voisins.

Prenons un exemple pour illustrer les étapes ci-dessus : Julie est en difficulté dans le concept *Installing* du langage de programmation Java. Les mots-clés de sa question sont : *install*, *server*, *tomcat*, *servlet*, *work*. Les experts qu'elle a appréciés dans le passé sont : Alice (vote global de 4), David (vote global de 4), Carole (vote global de 5).

1ère étape : établir les caractéristiques des experts enregistrés dans le profil de l'utilisateur et ayant reçu un vote positif. Dans notre exemple, il s'agit d'Alice, David et Carole. Supposons par exemple qu'Alice présente les caractéristiques (statistiques) suivantes : nombre de questions qu'elle a reçues au total : 6 ; nombre de questions auxquelles elle a répondu : 5. De plus, à propos du concept *Installing*, elle a répondu à 3 questions sur les 3 reçues et a reçu les votes consignés dans le Tableau 6.

Tableau 6 : Votes donnés à Alice sur le concept Installing

	Réponse 1	Réponse 2	Réponse 3
Vote général	4	4	5
Réponse rapide	3	4	2
Vocabulaire adéquat	5	4	4
Réponse cohérente	5	5	5
Réponse claire	5	4	4
Points importants	1	5	2
Fournit des exemples	1	1	1
Manifeste de l'intérêt	4	3	3
Donne des détails	4	5	4
Réfère à d'autres experts	1	1	1

Nous ne voyons ici qu'une partie du profil d'Alice, qui représente l'essentiel des informations nécessaires (avec les mots-clés et le domaine concerné par la question) au processus de recommandation. Seuls les experts faisant partie du domaine concerné sont considérés, ce qui est le cas de David et Carole.

2ième étape : établir les caractéristiques des autres experts de la base des profils utilisateurs. Il s'agit de parcourir la liste des autres experts dans le même domaine (ici, le concept *Installing*). Le système ne trouve que les experts Marc et Tom. Comme dans le cas d'Alice, leurs caractéristiques sont établies.

3ième étape : recommandation des experts. A partir des caractéristiques établies ci-dessus, il s'agit de calculer le degré de similarité entre les experts Alice et Marc, puis Alice et Tom. Si, par exemple la similarité entre Alice et Marc est de 0.3 et la similarité entre Alice et Tom est de 0.8, alors Tom sera recommandé à Julie. De même, les similarités sont calculées entre David et Marc, et David et Tom, et enfin entre Carole et Marc et Carole et Tom.

Ainsi, les experts les plus similaires à ceux qui ont été appréciés dans le passé par Julie lui seront recommandés.

5.9. Conclusion

Le système que nous avons conçu permet à des apprenants de définir eux-mêmes leurs préférences en matière de pédagogie. Il ne s'agit pas seulement de rechercher des experts selon leurs compétences techniques, mais également en fonction d'autres compétences qui permettront à l'utilisateur de mieux comprendre la réponse. Outre le fait de mettre en relation des personnes ayant des affinités pédagogiques (la façon d'expliquer de l'expert se combine parfaitement avec la manière de comprendre de l'apprenant), cela permet également de réduire le temps de recherche de l'apprenant.

Comparaison avec d'autres systèmes

Dans cette section, nous comparons notre système avec les système de localisation et recommandation d'expertise décrits dans le chapitre 3.

Les deux premiers critères sont généraux et les suivants sont plus spécifiques au processus de recommandation. Le tableau ci-dessous permet la comparaison sur les différents critères que nous expliquons en page suivante.

Tableau 7 : Comparaison de systèmes

	Chicago Information Exchange (CIE)	Expertise Finder	Système Q&A	Expert Finder	Système « Support routing »	HELP.
Utilisation d'un système de questions-réponses	Oui	Non	Oui	Non	Oui	Oui
Simule le réseau social	Oui	Oui	Non	Non	Non	Oui
Source de données concernant l'expertise	Expertise fournie par l'utilisateur lors de la 1ère connexion	Informations internes : e-mails, contacts téléphoniques et rapports de publications	Expertise fournie par l'utilisateur lors de la 1ère connexion	Fichiers sources en Java produits par les développeurs	Expertise fournie par l'utilisateur lors de la 1ère connexion	Expertise fournie par l'utilisateur lors de la 1ère connexion
Données statiques ou dynamiques ?	Statiques	Dynamiques	Statiques	Dynamiques	Statiques	Dynamiques
Prises en compte de plusieurs niveaux d'expertise	Non	Non	Non	Oui	Non	Oui
Validation de la réponse donnée par l'expert	Oui Trois choix possibles : « ok », « non Ok », « pas de mon domaine »	Non	Non	Non	Non	Oui Evaluation générale et évaluation pédagogique
Base de recommandation	Expertise et requête de l'utilisateur	Expertise uniquement Pas de prise en compte de la requête de l'utilisateur	Expertise et requête de l'utilisateur	Expertise et requête de l'utilisateur	Expertise	Expertise + requête de l'utilisateur + profil pédagogique de l'expert

Utilisation d'un système de question réponse : ce critère permet de savoir s'il existe une fonctionnalité de recherche de cas similaires, qui permettra, comme nous l'avons dit en introduction, d'une part d'accélérer la production d'un résultat pour l'utilisateur dans le cas où une question similaire à la sienne a déjà été résolue ; et d'autre part cela permettra aux experts de ne pas avoir à répondre toujours aux mêmes questions.

Réseau social : nous avons vu que le réseau social d'une personne se compose de l'ensemble de ses contacts (section 3.2.1).

Source de données concernant l'expertise : manuelle, c'est-à-dire fournie par les utilisateurs, ou automatisée, c'est-à-dire après analyse de fichiers produits par les utilisateurs.

Données statiques ou dynamiques ? Le système permet t-il de mettre à jour les données, surtout celles concernant l'expertise des personnes, ou sont-elles statiques (ce sont les informations de départ qui sont prises en compte).

Validation de la réponse donnée ? Un utilisateur a t-il la possibilité d'évaluer la réponse fournie, donc d'évaluer l'expertise de la personne qui lui a répondu. Cela peut être utile pour les futures recommandations.

Base de la recommandation : la recommandation est-elle basée sur l'expertise seule de l'expert, c'est-à-dire sur ses connaissances techniques ou d'autres données sont prises en compte ? La requête de l'utilisateur est-elle entièrement prise en compte ? Ce critère donne un aperçu des sources de données utilisées pour la recommandation.

Conclusions sur la comparaison

CIE est le seul système (hors le système HELP) à avoir une validation des réponses données, validation donnée en sélectionnant l'un des trois choix suivants lors de la réception d'une réponse : « réponse OK », réponse « non OK », « Pas de mon domaine ».

Cette validation ne sert dans ce cas qu'à relancer la recherche dans le cas où l'utilisateur n'est pas satisfait. Dans notre système, la validation a beaucoup plus d'importance : elle permet d'évaluer la réponse fournie, et servira pour les futures recommandations.

De plus, HELP et CIE simulent tous les deux le réseau social : chacun des systèmes permet à l'utilisateur d'avoir accès à une liste de contacts choisis par l'utilisateur lui-même. Cependant les contacts ajoutés dans le système CIE n'ont pas toujours une relation avec les contacts établis lors de la phase « questions-réponses ». Alors que dans le cas du système HELP, les contacts disponibles lors du processus de questions-réponses et / ou du processus de recommandation peuvent être rajoutés dans la liste de contact avec un rappel de la raison pour laquelle ce contact a été ajouté.

Le principe général des systèmes Q&A et du système utilisant le « support routing » est le même que celui de HELP : des cas similaires à la requête courante sont d'abord recherchés, et dans le cas où aucun cas n'est trouvé, le processus de recherche d'experts est lancé. Cependant, plusieurs différences importantes sont à noter : les systèmes Q&A et « support routing » ne se basent que sur des données statiques pour la recherche des experts, c'est-à-dire que seules les données fournies lors de la première connexion au système sont prises en compte. De plus, aucune validation n'est faite sur la réponse donnée, et celle-ci est enregistrée automatiquement sous forme de cas.

Expertise Finder est un bon système de localisation d'expertise puisque les données se basent sur de nombreuses ressources tangibles et produites par les experts eux-mêmes. Cependant, le système ne se base pas suffisamment sur la requête de l'utilisateur lors de la recherche d'experts. Seul le profil « technique » de l'expert est considéré : les recommandations sont donc toujours les mêmes, et ne changent que lorsque de nouvelles données sur l'expert sont produites. De même, Expert Finder se base également sur des fichiers produits par les experts pour établir leur profil technique et faire les recommandations. Contrairement à Expertise Finder, la requête de l'utilisateur est bien

prise en compte pour le calcul de la recommandation. Cependant, il n'y a pas de systèmes de questions-réponses et aucune validation de la réponse n'est faite.

Imaginons trois étudiants en doctorat, Alice, Eric et Julie, chacun travaillant dans des domaines de recherche différents. Les cinq systèmes vus précédemment, ainsi que le système HELP, permettent à un utilisateur quelconque de trouver l'aide ou l'expertise dont il a besoin à un moment donné ; cependant les résultats obtenus sont différents pour chacun de ces systèmes. Nous allons voir à travers des exemples simples les résultats obtenus selon les données d'entrée pour chaque système en les comparant systématiquement avec HELP.

Alice envoie une requête au système CIE ; Le système recherche s'il existe une question similaire dans sa base ; il en existe effectivement une et celle-ci est retournée à l'étudiante Alice. Celle-ci a ensuite la possibilité d'accepter ou non la réponse, avec également la possibilité de demander à avoir une liste d'experts susceptibles de l'aider dans son domaine. Elle reçoit alors une liste de 17 experts, recommandés par similarité de leur profil (chaque expert est représenté par un vecteur de mots-clés déduit de toutes ses interactions questions-réponses) avec la requête. Ces experts ont sensiblement le même profil, Alice en choisit deux « au hasard » dans un premier temps afin de les contacter directement. Si le même exemple est appliqué au système HELP, nous verrons que CIE permettra de trouver « plus » de questions similaires ou avec une meilleure précision, car CIE utilise des techniques plus évoluées prenant en compte la similarité sémantique de la requête de Alice avec celles contenues dans sa base. Cependant, la recommandation d'experts dans CIE se base sur les mots-clés de la requête donc sur les questions auxquelles les experts ont répondu (avec application du modèle vectoriel). HELP prend en compte l'expertise des utilisateurs de manière plus générale, mais surtout prend en compte le profil des experts ainsi que celui des utilisateurs (vus en tant qu'apprenant) afin de proposer des « ressources » plus adaptées à l'utilisateur.

Le système Q&A a sensiblement le même principe que le système CIE, mis à part le fait qu'aucune relation sémantique n'est utilisée dans le calcul de la similarité, et qu'une validation de la réponse fournie par l'expert n'est disponible. Les questions et réponses sont indexées automatiquement sous le compte de l'expert par catégorie d'expertise (cf. section 3.3.3).

Expertise Finder est un système multi-agents dont les agents « sources » ont pour tâche de récupérer des informations concernant des experts. La recommandation se base donc sur des données produites par les experts eux-mêmes, et Expertise Finder recommande les *n* premiers auteurs ayant le plus grand nombre d'apparition dans les données analysées par les agents sources. Soient Eric, Alice et Julie qui envoient des requêtes totalement différentes, mais faisant référence au même sujet, au système Expertise Finder. Tous trois reçoivent les mêmes recommandations d'experts : exp_1, exp_2, exp_3, exp_4, exp_5, exp_6, exp_7, exp_8. En effet, Expertise Finder fait évoluer ses recommandations uniquement en fonction de l'évolution des données sur les experts. Aucune prise en compte n'est faite de la requête ni du profil de l'apprenant. Parallèlement, avec HELP, Alice obtiendra exp_1 et exp_3 (avec la même requête) car ceux-ci ont un profil pédagogique adapté à celui d'Alice. Eric obtiendra la liste exp_3, exp_9 et exp_{10} (sa requête est différente, et concerne donc une expertise différente, et parmi ces experts, un filtrage est effectué prenant en compte les précédentes interactions (avec votes) d'Eric). Enfin Julie obtiendra deux requêtes similaires trouvées par le système dans sa base de cas.

Le système « support routing » utilise également un système de questions-réponses selon le même principe que HELP. Un étudiant envoyant une requête obtiendra la même réponse dans ce système que dans HELP. Alice envoie une requête $q_{Alice,1}$ au système, puis quelque temps plus tard une question $q_{Alice,2}$ sur le même domaine *d*. Dans les deux cas, elle obtiendra la même liste d'experts : la recherche d'experts se base sur l'expertise donnée en entrée par l'expert et ces données ne sont pas mises à jour. Imaginons ce même scénario dans HELP : après avoir envoyé la question $q_{Alice,1}$, Alice reçoit comme recommandation

les experts exp_1, exp_2, exp_3, exp_4, et exp_5. L'apprenant Alice sélectionne les réponses qu'elle souhaite garder, vote pour des experts dont elle a apprécié la réponse, ou au contraire n'a pas apprécié la réponse. Puis quelque temps plus tard, elle envoie sa question $q_{Alice,2}$ dans le même domaine. Elle aura alors comme recommandation exp_1, exp_3, exp_6 : exp_1 et exp_3 seront sélectionnés par le système comme ayant un profil pédagogique cohérent par rapport aux précédentes interactions d'Alice avec le système (Alice a voté positif sur exp_1 et exp_3 lors de ses précédentes requêtes) et selon ses votes, le système prédit que Alice votera positif pour exp_6. De plus, pour chaque recommandation, HELP associe la raison de la recommandation, ce qui ne peut que renforcer la confiance d'Alice dans HELP (elle sait pourquoi on lui recommande ces experts).

Enfin, Expert Finder analyse également les documents produits par les experts pour connaître leur expertise et leur niveau. De même que les autres systèmes basés sur les sources produits, cela peut ne pas être suffisamment complet. L'étudiant E7 / exp_7 est nouveau à l'Université, cependant il est très bon dans le domaine *d*. Dans HELP il renseigne son expertise et ainsi pourra être recommandé d'abord une première fois car le système va le conseiller en tant qu'expert du domaine *d*, puis, il sera recommandé parce qu' ayant reçu des avis favorables sur son expertise et sa pédagogie. Par contre, dans Expert Finder, l'étudiant E7 ne sera jamais recommandé s'il ne travaille pas (et donc ne produit pas de documents) dans ce domaine *d*.

En conclusion, nous ajoutons également que le système HELP est le seul système à prendre en compte les réponses fournies par les experts et à les faire évaluer par un système de votes. Ceci permet en outre d'améliorer la pertinence des recommandations en fonction du profil de l'apprenant et de l'expert. De plus, HELP permet un filtrage sur

l'enregistrement des cas puisque cet enregistrement se base sur cette même validation (enrichissement de la base de cas).

Nous pouvons donc conclure que HELP est le seul système, parmi ceux auxquels il a été comparé, qui cherche à savoir pourquoi un utilisateur pourrait ne pas être satisfait et qui utilise ces raisons pour améliorer les recommandations.

Nous allons voir dans le chapitre suivant le détail de l'implémentation, les technologies utilisées et les captures d'écran de HELP.

Chapitre 6 : Implémentation

Dans ce chapitre, nous parlerons de l'environnement de développement de notre système, puis nous donnerons des scénarios d'utilisation du système, autant du point de vue *apprenant* que du point de vue *expert*.

6.1. Implémentation de HELP

HELP a été développé sous le système d'exploitation Windows XP et le serveur web Apache Tomcat, version 5.0. Nous avons utilisé le langage de programmation JAVA (J2SE version 1.4.2) sous l'environnement de développement Eclipse, version 3.1. Les données sont stockées sous formes de fichiers XML.

Nous avons donc, plus précisément, utilisé les technologies suivantes :

- Java Servlet et JSP (Java Server Pages) version 2.3 sous le serveur Tomcat version 5.0
- La technologie DOM pour parser les données des fichiers XML
- HTML et Javascript pour les interfaces du programme

Nous décrivons brièvement ci-dessous les technologies utilisées.

6.1.1. Les Servlets et les pages JSP (*Java Server Pages*)

Depuis quelques années, de plus en plus de sites Web se sont développés. Plusieurs technologies et langages de programmation liés au Web sont apparus, permettant d'améliorer la gestion et le design des sites web : HTML, ASP, XML, PHP, ASP.NET, etc.

Nous allons introduire le fonctionnement des Servlets et des pages JSP, liés au langage de programmation Java, et permettant de développer des applications client-serveur dynamiques.

Architecture client-serveur

Un site WEB doit permettre l'échange d'informations entre un client (le navigateur) et le serveur Web. Pour cela, il existe plusieurs protocoles, dont le protocole HTTP (HyperText Markup Language). Lorsque le client saisit une URL (Uniform Resource Locator) dans son navigateur, une requête http est envoyée au serveur Web, c'est-à-dire que le client interroge le serveur Web. Le serveur Web renvoie ensuite une réponse basée également sur le protocole http.

Les Servlets

Une servlet Java est une classe qui a pour but de générer une réponse à une sollicitation d'un service. La classe de base d'une servlet est *GenericServlet*. Il existe plusieurs types de servlets selon le protocole de communication entre le client et le serveur Web. Nous parlerons des servlets que nous avons utilisé : les servlets *HttpServlet*.

La méthode principale est la méthode *service*, définie dans la classe javax.servlet.GenericServlet. Un objet de servlet est instancié en un seul exemplaire, par contre, plusieurs utilisateurs peuvent demander ce service (c'est-à-dire cette servlet). Dans ce cas, un nouveau *thread* est démarré et il exécute la méthode *service*. Cette méthode prend deux paramètres de types *ServletRequest* et *ServletResponse*. La méthode de type *ServletRequest* permet de manipuler la requête du client et la méthode de type *ServletResponse* permet de manipuler la réponse envoyée par le serveur au client.

La technologie JSP (Java Server Pages) est liée aux servlets. Elle permet de créer des interfaces Web dynamiques, en permettant l'intégration de code HTML statique et du

code JAVA rendant la page dynamique. Les pages JSP sont automatiquement transformées en Servlets par le serveur Web.

Technologie XML

XML (eXtensible Markup Language) fournit une notation pour la structuration de documents et nous l'utilisons dans HELP pour stocker les informations de l'application : profil des utilisateurs, ensemble des questions et réponses, etc.

XML est un ensemble de règles de syntaxe qui permet de représenter; grâce à des balises, des données dans des documents de type texte. Cette représentation précise la structure et les données.

6.2. Domaine d'application

Le domaine d'application que nous avons choisi est le langage de programmation Java. HELP permet donc à des développeurs de programmation Java, ou tout simplement à toute personne recherchant de l'information sur Java ou sur la programmation orientée objet, tel que Java, de poser sa question en langage naturel (en anglais).

Nous avons défini la hiérarchisation du domaine d'application d'après le site java.sun.com. En Figure 14, nous pouvons voir une partie de la hiérarchisation du domaine. Un utilisateur peut donc préciser lors de son enregistrement les concepts et sujets associés dans lesquels il pense être expert.

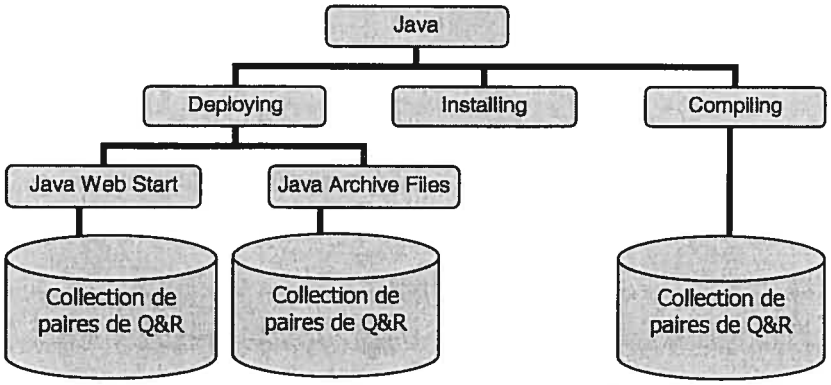


Figure 14 : Illustration d'une partie du domaine d'application

Nous allons maintenant voir les différentes étapes par lesquelles un utilisateur doit passer pour accéder aux différentes fonctionnalités du système, puis nous montrerons des captures d'écran de ces fonctionnalités.

6.3. Environnement d'un utilisateur

Dans cette section, nous verrons quels sont les composants de l'environnement de l'utilisateur lorsque celui-ci se connecte au système.

6.3.1. Enregistrement d'un utilisateur

Un utilisateur doit s'enregistrer afin d'accéder au système. S'il se considère comme un expert, il doit alors préciser ses domaines d'expertise.

L'utilisateur entre préalablement son nom, son e-mail, son nom d'utilisateur et son mot de passe. C'est son nom d'utilisateur qui permettra de l'identifier dans tout le système, il doit donc être unique.

En cliquant sur **Deploying**, l'utilisateur voit apparaître les sujets associés au concept, et en cliquant sur chacun des sujets, une description est donnée. Cela permet à l'utilisateur de savoir si ce sujet ou concept constitue réellement son expertise

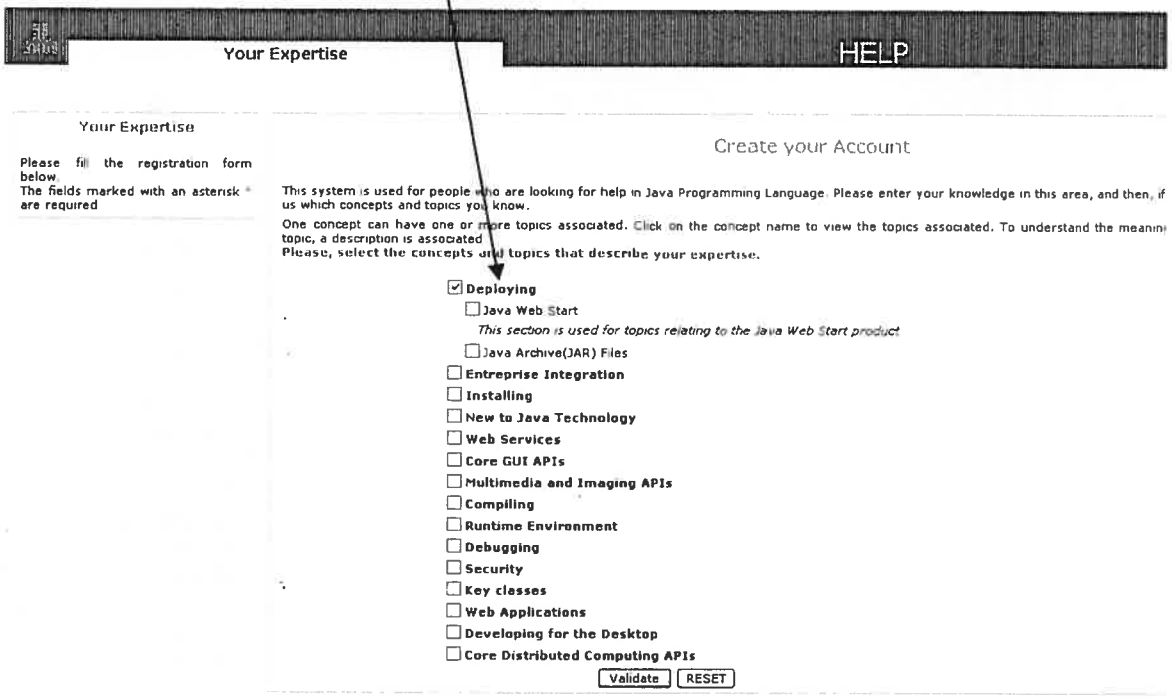


Figure 15 : Enregistrement de l'expertise d'un utilisateur

Une fois l'utilisateur enregistré, il a la possibilité d'accéder à son environnement, c'est-à-dire à toutes les fonctionnalités offertes par le système (cf. Figure 16) : *My Log* lui permet de consulter ses données personnelles et les données reliées à son expertise; *Mailbox* lui permet de gérer sa boite de messagerie; *Expert Research* lui permet de rechercher des experts selon certains critères de sélection.

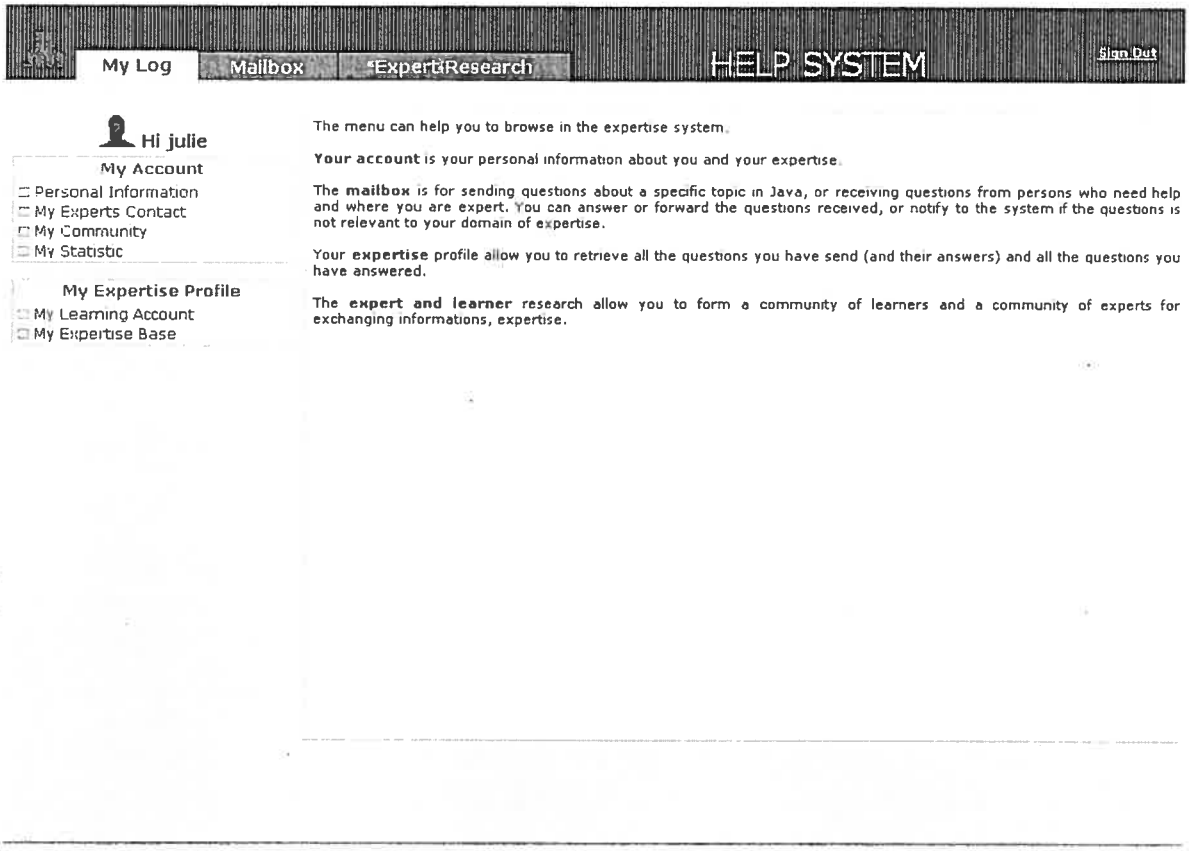


Figure 16 : Environnement de l'utilisateur

Le menu MY LOG est constitué de deux sous-menus :

- ✓ *My Account* : c'est le compte personnel de l'utilisateur, il peut consulter ses informations personnelles (nom, prénom, e-mail, etc.), la liste de ses contacts, autant les experts que les apprenants, et enfin, connaître les informations relatives à son expertise, comme les votes qu'il a reçu, le nombres de questions auxquelles il a répondu, le nombre de questions qu'il a envoyé par domaine, etc.

- ✓ *My Expertise Profile* : ce sous-menu contient sa base d'apprentissage et sa base d'expertise.

6.3.2. Gestion de la messagerie

La boite de messagerie permet à l'utilisateur de gérer ses requêtes en tant qu'expert et en tant qu'apprenant.

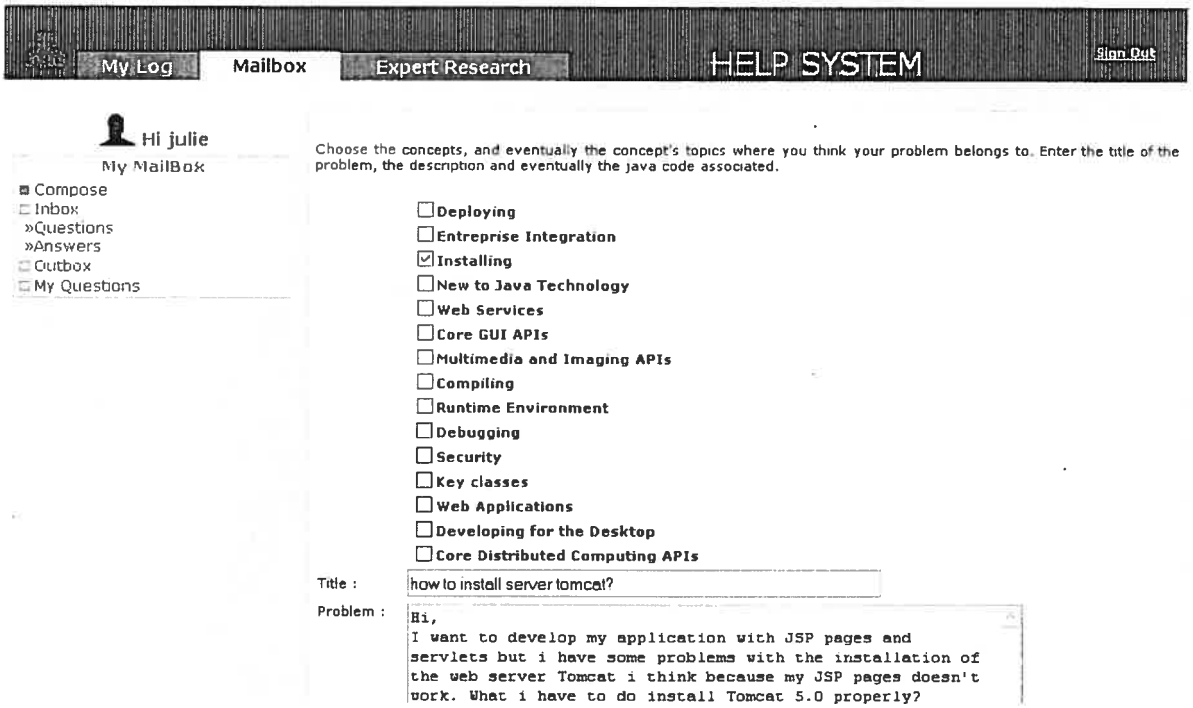


Figure 17 : Envoi d'une requête

Sur la Figure 17, Julie envoie une requête au système par l'intermédiaire de sa boite de messagerie. Elle choisit le domaine qu'elle considère être celui correspondant à sa requête (elle peut en choisir plusieurs). Si Julie ne choisit aucun domaine, tous les domaines seront alors considérés pour la recherche.

Une fois la requête envoyée, le système recherche s'il existe des requêtes similaires. Si c'est le cas, ces requêtes sont envoyées à l'utilisateur. Sinon, le système sélectionne les

experts qui seront les plus susceptibles de répondre à la requête et la leur envoie. Julie doit alors attendre sa réponse.

Elle peut consulter les autres catégories de sa boite de messagerie que nous allons détailler ici :

- ✓ *Inbox* contient deux sous-catégories, qui correspondent respectivement aux deux types de messages qu'elle peut recevoir : *Questions* sont les questions qu'elle reçoit en tant qu'experte. Elle peut soit y répondre, soit les transférer à une autre personne. *Answers* sont les réponses qu'elle reçoit des experts.
- ✓ *Outbox* sont les réponses qu'elle envoie en tant qu'experte.
- ✓ *My Questions* sont les questions qu'elle a envoyées.

Marc se connecte à son tour, consulte sa boite de messagerie et voit le message de Julie. Il peut alors le consulter et y répondre.

Dans le cas ou Julie pose une question possédant des requêtes similaires, le système les lui propose et Julie a la possibilité de les consulter et de les conserver dans sa base d'apprentissage.

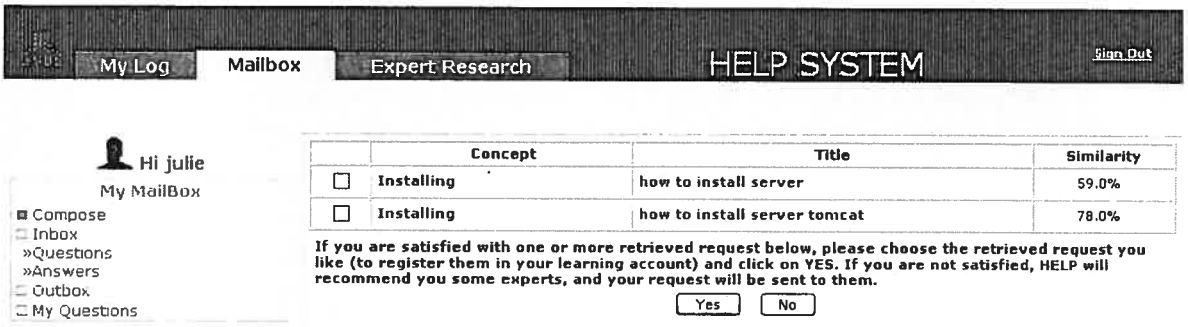


Figure 18 : Recherche de cas similaires

Si Julie n'est pas satisfaite, le processus de recommandation est déclenché et sa requête est envoyée aux experts cités dans la figure ci-dessous :

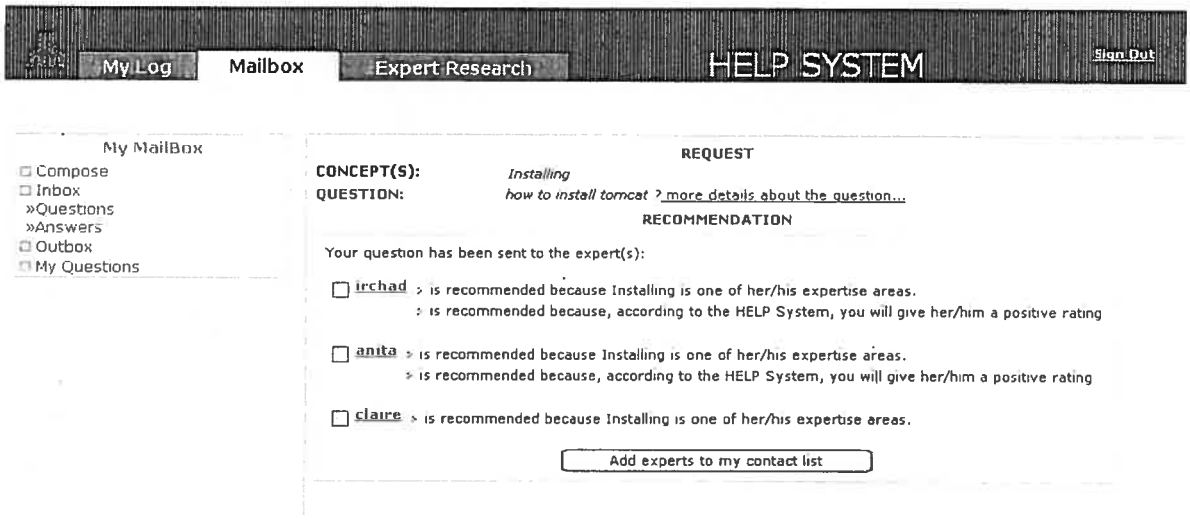


Figure 19 : Recommandation d'experts

Julie a la possibilité d'enregistrer les experts recommandés dans sa liste de contacts. Le système indique toujours la ou les raisons pour lesquelles il recommande l'expert. Cela permet à l'apprenant de pouvoir choisir les experts à qui envoyer la requête.

6.4. Validation du système

Nous présentons dans cette section la validation de notre système et les différents obtenus.

6.4.1. Rappels

Il existe 4 catégories d'experts : débutant, intermédiaire, avancé et très avancé.

Le terme « expert » est utilisé ici dans son sens général et désigne une personne qui est susceptible de répondre à une question, toute catégorie confondue.

Le terme « apprenant » sera utilisé pour toute personne utilisant le système pour lancer une requête.

Notre base de données initiale contient 54 cas issus du site : forum.sun.com. Rappelons que le domaine d'application générale du système est le langage de programmation Java. Nous avons décomposé le domaine d'application en 15 domaines, eux-mêmes pouvant être composés de sous domaines (entre 0 et 4 par domaine). Les 54 cas couvrent l'ensemble de ces 15 domaines.

6.4.2. Utilisation générale du système

Le domaine d'application de notre système est le langage de programmation Java; nous avons donc choisi des personnes ayant travaillé et travaillant toujours dans le développement d'applications (« programmeurs »). La plupart ont eu une formation scolaire assez standard en Java, et ont eu ensuite plus ou moins l'occasion de l'utiliser dans le cadre de leur travail. On obtient donc des utilisateurs à des niveaux différents.

Les personnes doivent s'enregistrer et indiquer au système leur catégorie : débutant, intermédiaire, avancé, très avancé. Ensuite ils choisissent les domaines du langage JAVA dans lesquels ils pensent être experts (par exemple, le domaine « Security », qui permet de regrouper toutes les méthodes relatives à l'aspect sécurité des données et des traitements d'une application; la Figure 17 permet de voir les domaines de Java considérés dans le système).

Le système a été utilisé par 63 utilisateurs, toute catégorie confondue.

Les tests ont été effectués sur une période de 15 jours environ.

L'analyse des résultats a été faite à partir des données récupérées dans les fichiers XML enregistrés au niveau des utilisateurs.

Notre première observation porte sur le nombre d'utilisateurs par catégorie d'expertise.

Tableau 8 : nombre de requêtes envoyées par catégorie d'utilisateurs

	Débutant	Intermédiaire	Avancé	Très avancé	***Total***
Nombre d'utilisateurs	3	12	32	16	**63**
Nombre total de requêtes envoyées	14	35	46	12	**107**

6.4.3. Consultation des cas similaires

Lorsqu'une requête est envoyée par un utilisateur, le système recherche tout d'abord s'il existe des cas similaires. Rappelons que le calcul de la similarité est basé sur le ou les **domaines choisis** par l'utilisateur et sur le **titre** de la requête.

Nous avons d'abord analysé le nombre de **cas similaires trouvés** par le système en différenciant les requêtes envoyées par catégorie d'utilisateurs, ainsi que les **cas retenus**. Le **Tableau 9** montre le nombre de requêtes pour lesquelles x cas similaires ont été trouvés par le système (lignes grisées) et les y cas retenus par les utilisateurs (et qu'ils ont donc trouvés pertinents).

Par exemple, parmi les 14 requêtes totales envoyées par les utilisateurs « débutants », le système n'a trouvé aucun cas similaire pour 11 d'entres elles, et 1 cas similaire pour les 3 autres. Parmi ces 3 requêtes dont 1 cas similaire a été trouvé, aucun utilisateur n'a retenu ce cas.

Tableau 9 : cas similaires trouvés et retenus

	0 cas	1 cas	2 cas	3 cas	≥ 4 cas
Débutant	11 requêtes	3 requêtes	0	0	0
		0			
Intermédiaire	19 requêtes	11 requêtes	3 requêtes	1 requête	1 requête
		8 requêtes	2 requêtes	1 requête	1 requête
Avancé	22 requêtes	16 requêtes	6 requêtes	2 requêtes	0
		7 requêtes	4 requêtes	1 requête	
Très avancé	6 requêtes	5 requêtes	1 requête	0	0
		1 requête	1 requête		

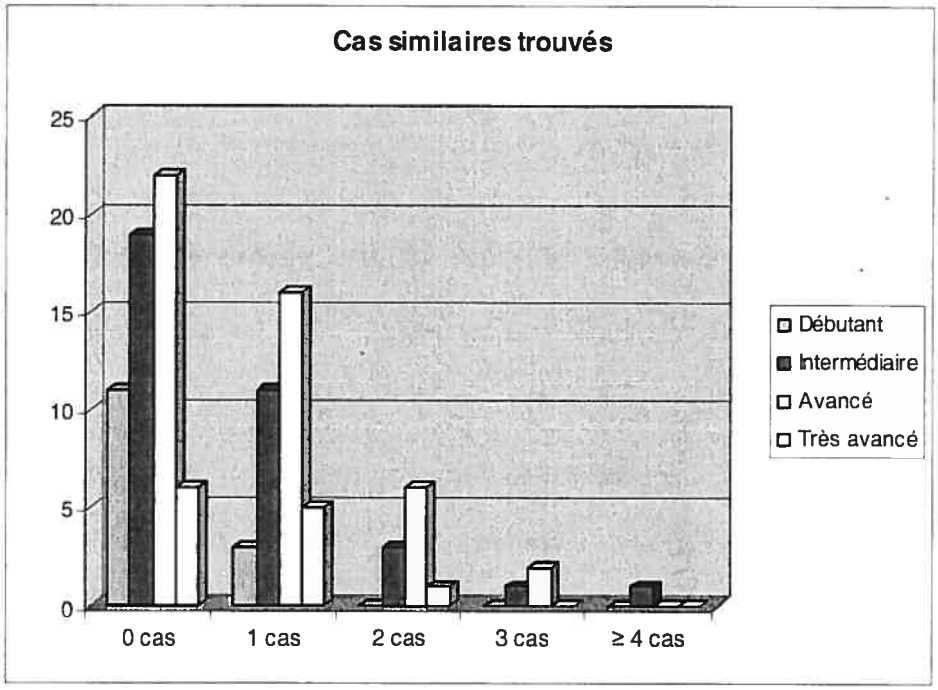


Figure 20 : Cas similaires trouvés pour chaque catégorie d'experts

La Figure 20 donne une représentation graphique du nombre de cas similaires trouvés par catégorie d'experts.

Interprétation des résultats :

Aucun débutant n'a retenu de cas. Un débutant n'a généralement pas le vocabulaire adéquat pour poser sa question. Nous avons vu que le titre de la requête est important car c'est à partir de ce titre que sont extraits les mots-clés. Si les mots choisis par l'utilisateur débutant ne décrivent pas correctement son problème, il peut obtenir des cas dont le titre est similaire au sien, mais dont le contenu ne correspond pas à ce qu'il recherche. Il va donc rejeter le cas trouvé.

Pour les catégorie « intermédiaire » et « avancé », les utilisateurs sont partagés : on observe environ la moitié de cas retenus. En effet, par rapport à un débutant, un membre de ces catégories posera mieux sa question et retrouvera donc un contenu plus adéquat à ce qu'il cherche. De plus, il va préférer avoir une réponse rapide plutôt que d'attendre la réponse d'un expert : étant donné qu'il a déjà des connaissances en Java, des éléments de réponses peuvent lui suffire pour trouver la réponse complète lui-même.

Pour la catégorie « très avancé », peu de cas ont été retenus. Ceci peut s'expliquer d'une part par le caractère très technique et très pointu de la question de l'expert. D'autre part, un expert préfèrera souvent dialoguer avec un autre expert et donc rejettera de manière « presque systématique » les cas trouvés pour lancer la recherche d'experts.

Les raisons qui conduisent un apprenant à retenir au moins un cas parmi les cas similaires trouvés sont nombreuses et très subjectives. Chaque utilisateur peut avoir ses propres raisons et celles-ci peuvent varier selon la requête et l'urgence de son besoin. Il est alors difficile d'en tirer une tendance générale sur le choix de retenir ou non un cas.

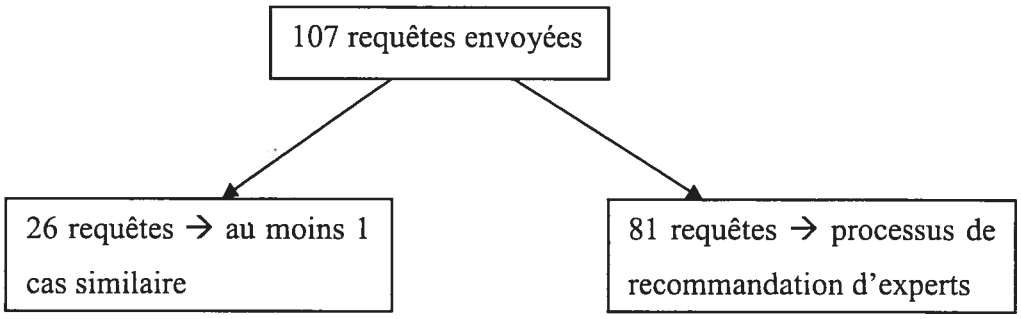
Nous remarquons également que l'analyse uniquement du titre est plutôt limitée pour la recherche de cas similaires. En effet, plusieurs cas peuvent avoir des titres similaires mais des contenus très différents.

6.4.4. Analyse des choix des apprenants

Nous allons maintenant analyser les choix des apprenants par rapport aux recommandations données par le système.

Il y a 107 requêtes envoyées au total, toute catégorie confondue. Parmi ces 107 requêtes, 26 ont retenu au moins un cas similaire correspondant, et 81 ont lancé le processus de recommandation.

Lorsqu'au moins un cas similaire a été retenu, le système considère que la requête est traitée et donc le processus de recommandation ne s'appliquera pas sur cette requête.



Un expert peut être recommandé par le système pour 4 types de raisons, que nous appellerons arbitrairement raison 1, raison 2, raison 3, raison 4. Pour chaque expert recommandé, la raison de ce choix est affichée pour permettre à l'apprenant de faire son tri (il peut choisir autant d'experts qu'il veut).

Tableau 10 : choix des apprenants par rapport aux recommandations du système

		Utilisateurs qui ont envoyé les 81 requêtes			
		Débutant 14 requêtes	Intermédiaire 23 requêtes	Avancé 34 requêtes	Très avancé 10 requêtes
Raison 1	Débutant	0	0	0	0
		0	0	0	0
	Intermédiaire	30	45	22	1
		30	62	35	6
	Avancé	82	90	32	5
		82	103	47	11
	Très avancé	43	92	16	7
		43	92	25	14
Raison 2	Débutant	0	0	0	0
		0	0	0	0
	Intermédiaire	3	2	0	0
		3	2	0	0
	Avancé	2	1	2	1
		2	1	3	1
	Très avancé	2	0	1	2
		2	0	1	2
Raison 3	Débutant	0	0	0	0
		0	0	0	0
	Intermédiaire	5	4	1	0
		5	4	1	0
	Avancé	3	1	3	2
		3	1	3	2
	Très avancé	2	0	1	2
		2	0	1	2
Raison 4	Débutant	0	0	0	0
		0	0	0	0
	Intermédiaire	6	1	1	0
		6	1	1	0
	Avancé	1	3	12	3
		1	3	12	3
	Très avancé	0	0	2	5
		0	0	2	5

Les raisons sont :

Raison 1 : le domaine de la requête correspond au domaine de l'expert recommandé (toute catégorie confondue).

Raison 2 : prédiction d'un vote positif (supérieur ou égal à 3).

Raison 3 : l'expert recommandé est similaire à l'apprenant (ils ont voté de la même façon par rapport à d'autres experts).

Raison 4 : l'expert a déjà été recommandé et choisi par l'apprenant.

Le Tableau 10 donne le résultat sur le choix des apprenants par rapport aux experts recommandés, en différenciant les catégories de ces experts.

Les lignes grisées représentent le nombre de requêtes pour lesquelles un type d'expert est recommandé par le système pour la raison *i*. Les lignes non grisées représentent le nombre de requêtes pour lesquelles l'expert recommandé (pour la raison *i*) a été retenu.

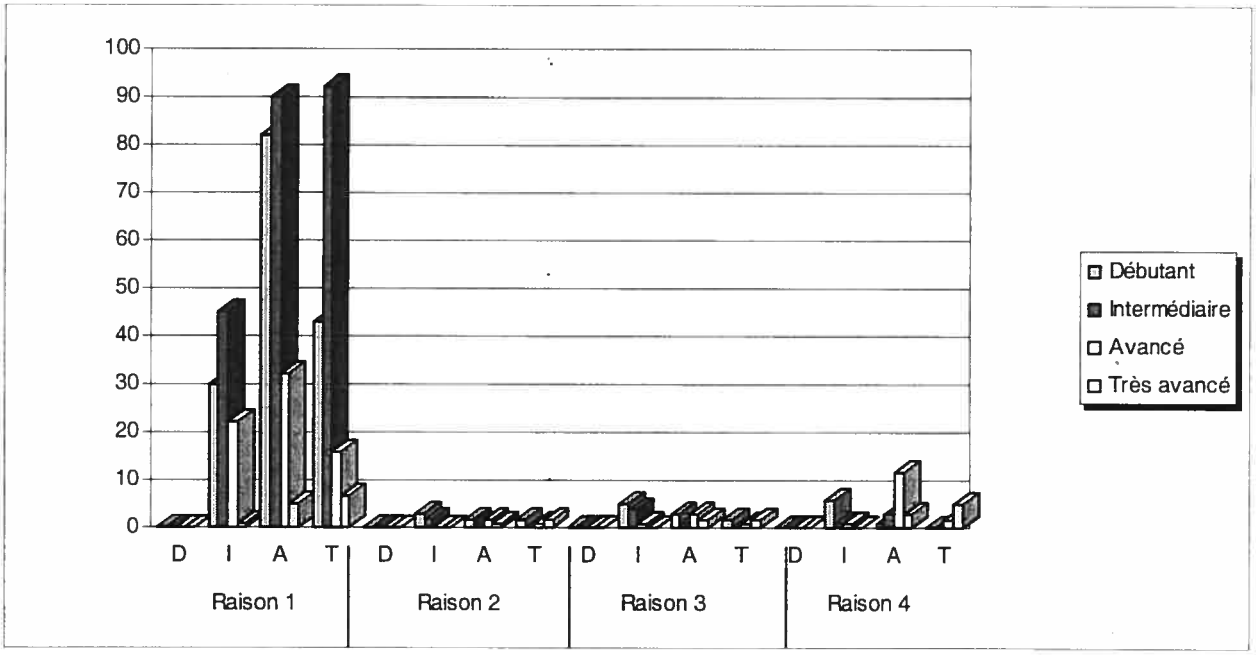


Figure 21 : requêtes pour lesquelles un type d'expert est retenu pour la raison *i*

Interprétation des résultats

Etant donné qu'il y a peu de débutants (3), ils ont chacun, d'après les résultats, choisis tous les experts qui leur sont recommandés.

Nous pensons que ces résultats sont peu significatifs dans la mesure où il y a très peu de débutants et que l'on peut penser que chaque utilisateur « débutant » a le même comportement d'une requête à l'autre. Le débutant n'a aucun critère de sélection, l'important pour lui est surtout d'avoir au moins une réponse.

Pour la catégorie intermédiaire, les résultats montrent que, à chaque fois que l'apprenant reçoit des recommandations d'experts pour d'autres raisons que la raison 1, il va choisir les experts qui ont des chances de lui répondre et qui ont déjà été recommandés par ailleurs donc il choisira moins d'experts pour la raison 1.

Pour les catégories avancés et très avancés, on remarque qu'en général ils choisissent les autres raisons que la raison 1 car leurs requêtes sont très précises, et ils préfèrent cibler leurs interlocuteurs.

On observe que peu de recommandations sont faites pour la catégorie « très avancé » pour la raison 1. Ceci peut s'expliquer par le fait que les requêtes des experts sont beaucoup plus ciblées au niveau du domaine, donc ils ont moins de chances d'obtenir des recommandations larges.

6.4.5. Analyse des votes

Dans cette section, nous allons analyser d'abord part le nombre de réponses envoyées par rapport aux requêtes reçues ; puis le nombre de votes reçus suite aux réponses envoyées ; enfin, nous allons observer le nombre de votes positifs donnés parmi l'ensemble des votes donnés.

Tableau 11: analyse des votes

	Débutant				Intermédiaire				Avancé				Très avancé			
	0 requête reçue				18 requêtes reçues				42 requêtes reçues				47 requêtes reçues			
	D	I	A	E	D	I	A	E	D	I	A	E	D	I	A	E
Nombre de réponses envoyées	0	0	0	0	13	2	1	0	2	6	18	7	0	2	13	8
Nombre de votes reçus	0	0	0	0	11	2	1	0	2	5	14	4	0	2	9	6
Nombre de votes positifs reçus	0	0	0	0	6	1	1	0	1	3	12	3	0	0	2	5

Le **Tableau 11** regroupe l'ensemble des résultats relatifs à l'analyse de votes. Pour une meilleure compréhension de la lecture de ce tableau, nous allons expliquer ce que représente la colonne grisée. 42 requêtes (toute catégorie confondue) ont été reçues par l'ensemble des membres de la catégorie « avancé ». 6 est le nombre total de réponses envoyées par les avancés à la catégorie intermédiaire, représenté par la lettre I dans le tableau (on sous-entend que les requêtes correspondantes ont été envoyées par ces mêmes « intermédiaires »).

Suite aux réponses envoyées, et reçues par ces intermédiaires, ceux-ci ont ensuite la possibilité de voter. Et donc le nombre de votes reçus par les avancés est de 5.

Enfin, parmi ces 5 votes, 3 sont des votes positifs (vote supérieur ou égal à 3).

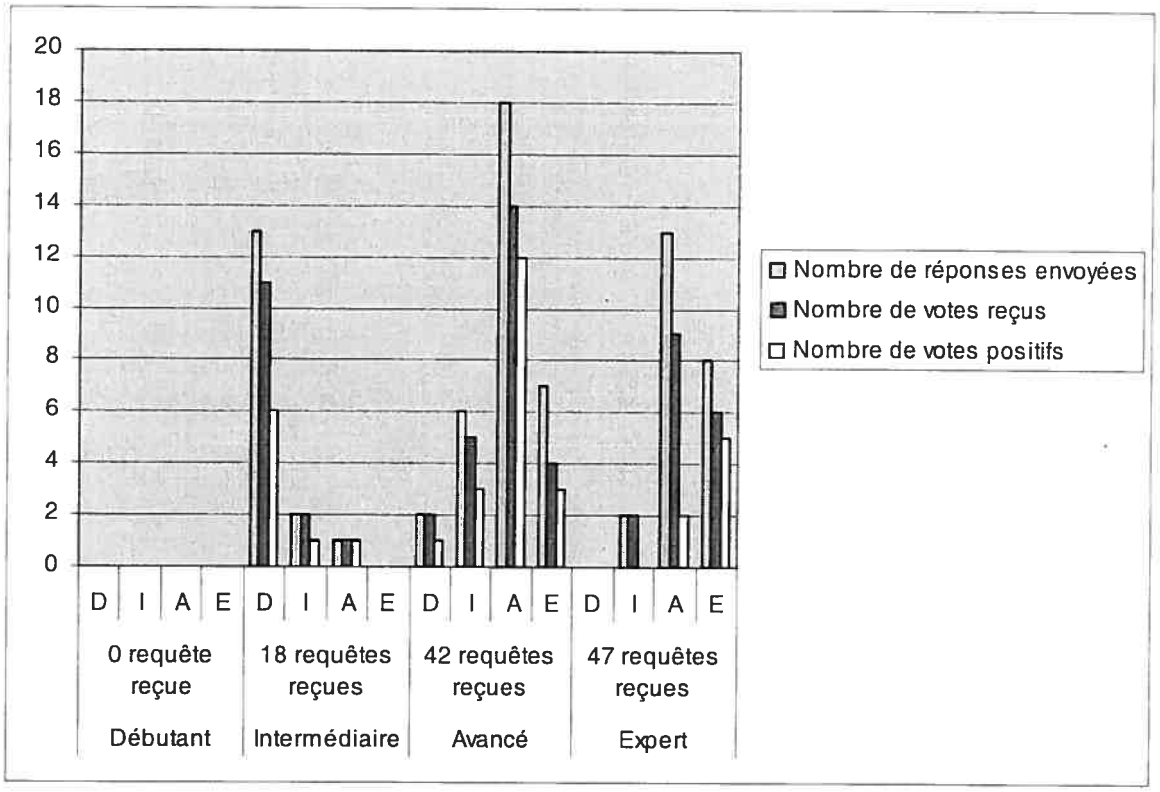


Figure 22 : Analyse de la validation des réponses

Interprétation des résultats :

Les débutants ne reçoivent aucune requête car ils ne se sont enregistrés dans aucun domaine d'expertise, donc ils ne sont jamais recommandés et ne reçoivent donc pas de votes. Par conséquent, la technique du filtrage collaboratif (basée sur les votes) ne fonctionne pas pour de tels débutants.

A l'inverse les catégories « très avancés » et « avancés » sont susceptibles de s'enregistrer dans plusieurs domaines et donc d'être recommandés plus souvent, et par conséquent d'avoir plus de votes.

Les débutants ont tendance à retenir tous les experts car ils se doutent qu'ils auront peu de réponses et de plus, ils pensent avoir des réponses plus ou moins complexes par rapport à leur compréhension.

La réponse d'un expert est-elle validée ? (La validation d'une réponse se traduit par le vote de l'utilisateur sur la réponse de l'expert) Donc la question revient à se demander si l'utilisateur a voté ? Nous avons également voulu différencier les votes positifs (≥ 3 car les votes sont compris entre 1 et 5) des votes négatifs, et donc les réponses restant sans votes.

De manière générale on observe que les votes sont donnés (peu de personnes ne votent pas). Ceux qui votent le moins sont les personnes de la catégorie « très avancé », ce qui peut s'expliquer par le fait que qu'ils ne votent que pour les experts qu'ils ont appréciés (tous leurs votes ne sont pratiquement que des votes positifs).

6.4.6. Conclusion

Ces différentes analyses nous permettent de répondre à un certain nombre d'interrogations sur l'intérêt d'un tel système.

Tout d'abord, la phase de recherche de cas similaires est-elle utile ou non ? D'après les résultats, cette recherche de cas similaires intéresse particulièrement les utilisateurs ayant une bonne compétence. On peut vraisemblablement penser que cela leur permet un gain de temps appréciable dans la recherche de réponses à leurs requêtes.

Ensuite, le système de votes est-il réellement utilisé ? Il fonctionne bien puisque les utilisateurs votent en général dès qu'ils ont une réponse, ce qui montre que les utilisateurs de ce système souhaitent partager leurs avis et leurs points de vue par rapport aux réponses fournies à leurs requêtes.

Enfin, nous avons remarqué que le fait de mettre à la disposition des apprenants les informations sur les experts recommandés permet de faciliter leur choix parmi les recommandations proposées.

Conclusion

Obtenir l'expertise appropriée lorsqu'on en a besoin, tel était notre objectif principal. Localiser un expert afin de le proposer à une personne désireuse de ses connaissances fait partie des travaux dans le domaine de la gestion des connaissances. Nous nous sommes intéressés au terme « appropriée » associée à l'expertise. Comment obtenir une expertise « appropriée » à nos besoins ? Ou en d'autres termes, quel est l'expert qui pourra nous aider ? Notre approche se basait sur le principe généralement associé aux systèmes tutoriels intelligents : avoir une réponse adaptée à l'apprenant, c'est-à-dire une réponse adaptée à sa façon d'assimiler les connaissances. Généralement, un système tutoriel intelligent repose sur l'utilisation de plusieurs stratégies pédagogiques utilisées en fonction du profil de l'apprenant. Dans notre système, nous avons également voulu nous focaliser sur la pédagogie. Chaque utilisateur, en tant qu'expert, a sa propre pédagogie pour faire partager ses connaissances. HELP a pour but de connaître la stratégie pédagogique appréciée par l'apprenant, celle utilisée par l'expert, et enfin recommander l'expert dont la « stratégie pédagogique » est la plus susceptible de répondre aux besoins de l'apprenant. Nous avons évoqué à travers ce chapitre notre approche pour atteindre ces objectifs.

HELP permet aux apprenants de définir eux-mêmes leurs préférences en matière de pédagogie. Il ne s'agit pas seulement de rechercher des experts selon leurs compétences techniques ; mais bien plus, il faut tenir compte d'autres compétences qui permettront à l'utilisateur de mieux comprendre la réponse. Outre le fait de mettre en relation des personnes ayant des affinités pédagogiques (la façon d'expliquer de l'expert se combine parfaitement avec la manière de comprendre de l'apprenant), cela permet aussi de réduire le temps de recherche de l'apprenant. Enfin, les connaissances capturées à travers le système de questions-réponses permet de favoriser l'apprentissage personnel de l'apprenant (base d'apprentissage).

De nombreuses applications de ce type de système restent à exploiter : l'intégration dans une organisation de ce système pour faciliter les prises en charge de projet et la distribution de travail ou la délégation de tâches. Tout d'abord, le système peut recommander en fonction des domaines d'expertise et si, par exemple, un chef de projet souhaite qu'une personne soit chargée des contacts avec les clients, il peut exploiter ce système d'évaluation pédagogique et rechercher des personnes selon les critères qui lui paraissent adéquats pour ce type de tâche.

D'autre part, il serait également intéressant d'étudier les conséquences des évaluations pédagogiques aussi bien dans un contexte industriel qu'académique, car chaque apprenant/expert est évalué sur ses compétences et certaines qualités personnelles relatives à son travail. On pourrait ainsi, à partir de ces résultats, construire des outils capables de donner des statistiques concernant par exemple les sujets les moins compris par les apprenants, ou connaître les lacunes qui ont besoin d'êtres comblées par une formation.

Discussion

Dans HELP, la recherche de requêtes similaires associe à chaque terme de la question un poids calculé selon une technique bien connue de la recherche d'informations. Cependant, cette technique ne permet de mettre en évidence certains mots-clés, comme par exemple les mots du domaine pour lesquels il serait intéressant d'augmenter le poids afin d'améliorer la précision des résultats de la recherche. Pour cela, une perspective intéressante serait de développer un réseau sémantique de mots-clés selon le domaine d'application.

Notre système de recommandation se base essentiellement sur des données subjectives (les votes). Il est nécessaire d'avoir également un avis objectif sur les experts, et pour cela recueillir des données et les prendre en compte dans le calcul d'un score de crédibilité de l'expert.

Tout système de recommandation basé sur le filtrage collaboratif peut faire l'objet d'attaques pouvant provoquer le biais des résultats donnés par le système [Massa et Avesani 04] [Massa et Bhattacharjee 04]. Dans HELP, il suffirait qu'un utilisateur « malveillant », désireux de faire en sorte qu'un certain expert soit de plus en plus recommandé et donc de plus en plus consulté vienne créer plusieurs comptes avec plusieurs pseudonymes différents (alors qu'il s'agit d'une même personne). Par la suite, il donnerait des votes excellents à un expert particulier. Cela forcerait le système à recommander cet expert, c'est-à-dire à lui accorder plus de crédit. Nous n'avons pas pris en compte ce type de problème dans le système actuel. Cependant, nous en tiendrons compte pour les futurs développements. Il faudrait par exemple demander un feedback sur les recommandations données afin d'établir la crédibilité de la recommandation : la recommandation donnée est-elle bonne ? Si non, pourquoi ? Si l'expert recommandé reçoit plusieurs avis défavorables, il conviendrait de s'interroger sur son expertise et faire en sorte que sa crédibilité soit revue.

Travaux futurs

Dans notre système, nous avons simulé un système peer-to-peer où chaque apprenant possède ses propres bases de connaissances. Dans nos travaux futurs, nous souhaiterions développer un système multi-agents où chaque apprenant aurait un agent personnel responsable de gérer son environnement, et responsable des interactions avec les autres agents.

Une autre approche intéressante serait d'étendre le système en permettant aux utilisateurs d'avoir accès à des ressources explicites, des documents, des liens, etc. qui seraient proposés ou recommandés par des experts.

Bibliographie

[Aamodt et Plaza 94] Aamodt A. et Plaza E., "Case-based reasoning: foundational issues, methodological variations, and system approaches", *AICom – Artificial Intelligence Communications*, 7 (1), pp. 39-59, 1994.

[Aberg et Shahmehri 01] Aberg A. et Shahmehri N., "Collection and exploitation of expert knowledge in web assistant system", *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34)*, Maui, Hawaii, Janvier 2001.

[Ackerman et McDonald 96] Ackerman M. S. et McDonald D. W., "Answer Garden 2: merging organizational memory with collaborative help", *Proceedings of the ACM conference on Computer-Supported Cooperative Work (CSCW'96)*, pp. 97-105, Boston, Massachusetts, Etats-Unis, Novembre 1996.

[Aïmeur et Boudina 99] Aïmeur E. et Boudina K., "Financial analysis by case base reasoning", *Proceedings of the 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-99)*, pp. 388-397, Caire, Egypte, Mai / Juin 1999.

[Aguzzoli *et al.* 02] Aguzzoli S., Avesani P. et Massa P., "Collaborative case-based recommendation systems, *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning (ECCBR-02)*, Aberdeen, Ecosse, Royaume-Uni, septembre 2002.

[Becks *et al.* 03] Becks A., Reichling T. et Wulf V., "Supporting collaborative learning by matching actors", *Proceedings of the 36th Annual Hawaii Conference on System Science (HICSS'03)*, Big Island, Hawaii, Janvier 2003.

[Bradley *et al.* 00] Bradley K., Rafter R. et Smyth B., "Case-based user profiling for content personalisation", *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH-2000)*, pp. 133-143, Trento, Italie, 2000.

[Budzik et Hammond 99] Budzik J. et Hammond K. J., "Q&A: a system for the capture, organization and reuse of expertise", *Proceedings of the 62nd Annual Conference of*

the American Society for Information Science (ASIS), Medford, Octobre / Novembre 1999.

[Burke 02] Burke R., "Hybrid recommender systems: surveys and experiments", *User Modeling and User-Adapted Interaction*, **12** (4), Novembre 2002.

[Burke *et al.* 97] Burke R., Hammond K., Kulyukin V., Lytinen S. et Tomuro N., "Question answering from frequently-asked question files: experiences with the FAQ Finder system", *Artificial Intelligence Magazine*, **18** (2), pp. 57-66, 1997.

[Crowder *et al.* 03] Crowder R., Hughes G. et Wendy H., "An agent based approach to finding expertise in the engineering design environment", *Proceedings of the 14th International Conference on Engineering Design (ICED 03)*, pp. 179-188, Stockholm, août 2003.

[Dieng 00] Dieng R., "Knowledge management and the internet", *IEEE Intelligent Systems*, **15** (3), pp. 14-17, mai/juin 2000.

[Finnie G. et Sun Z. 02] Finnie G. et Sun Z., "Similarity and Metrics in Case-based reasoning", International Journal of Intelligent Systems, **17**(3), pp. 273-287, Mars 2002.

[Foner 98] Foner L. N., "Yenta: a multi-agent, referral-based matchmaking system", *Proceedings of the 1st International Conference on Autonomous Agents (Agent'97)*, pp. 301-307, Marina Del Rey, Californie, février 1997.

[Fuller et Zobel 98] Fuller M. et Zobel J., "Conflation-based comparison of stemming algorithm", *Proceedings of the 3rd Australian Document Computing Symposium*, pp. 8-13, Sydney, Australie, Août 1998.

[Herlocker *et al.* 00] Herlocker J. L., Konstan J. A. et Riedi J., "Explaining collaborative filtering recommendations", *Proceedings of the 2000 Conference on Computer Supported Cooperative Work (CSCW'00)*, pp. 241-250, Philadelphie, Pennsylvanie, Etats-Unis, 2000.

[Kautz *et al.* 97] Kautz H. A., Selman B. et Shah M., "Referral Web: "Combining Social Networks and Collaborative Filtering", *Communications of the ACM*, **40** (3), pp. 63-65, 1997.

[Kulyukin 98] Kulyukin V., "An interactive and collaborative approach to answering questions for an organization", *Proceedings of the American Society for Information Science (ASIS-98) Midyear Conference*, Orlando, Floride, mai 1998.

[Kunze et Hubner 98] Kunze M. et Hubner A., "CBR on semi-structured documents: the experience book and the FallQ project", *Proceedings of the 6th German Workshop on Case-Based Reasoning*, Berlin, Allemagne, Mars 1998.

[Kolodner 93] Kolodner J. L., *Case-Based Learning*, Kluwer Academic Publishers (Editeur), 1993.

[Kolodner et Leake 96] Kolodner J. L., Leake D. B., Chapitre "A Tutorial Introduction »,. dans *Case-Based Reasoning: Experiences, Lessons & Future Directions"*, Leake, D. (Editeur), 1996.

[Lamontagne et Lapalme 02] Lamontagne L. et Lapalme G., "Raisonnement à base de cas textuel - état de l'art et perspectives futures", *Revue d'intelligence artificielle*, **16** (3), pp. 339–366, 2002.

[Leake 96] Leake D. B., Chapitre "CBR in context: the present and future", dans *Case-Based Reasoning: Experiences, Lessons, and Futures Directions*, Leake, D. (Editeur), 1996.

[Leake *et al.* 95] Leake D.B., Kinley A. et Wilson D., "Learning to improve case adaptation by introspective reasoning and CBR", *Proceedings of the 1st International Conference on Case-Based Reasoning*, Sesimbra, Portugal, Octobre 1995.

[Lenz 98a] Lenz M., "Textual CBR and Information Retrieval – A comparison", *Proceedings of the 6th German Workshop on CBR,* Berlin, Allemagne, Mars 1998.

[Lenz 98b] Lenz M., "Defining knowledge layers for textual case-based reasoning", *Proceedings of the 4th European Workshop on Advances in Case-Based Reasoning (EWCBR-98),* Dublin, Ireland, Septembre 1998.

[Lenz 98c] Lenz M., "Knowledge sources for textual CBR applications", *Proceedings of the 15th American Association for Artificial Intelligence (AAAI-98,) Workshop on Textual Case-Based Reasoning*, pp. 24-29, Madison, Wisconsin, Etats-Unis, Juillet 1998.

[Lenz et Burkhard 97] Lenz M. et Burkhard H-D, "CBR for document retrieval - the FallQ project", *Proceedings of the 2nd International Conference on Case-Based Reasning (ICCBR-97)*, pp. 84-93, Rhode Island, Etats-Unis, Juillet 1997.

[Lenz *et al.* 98a] Lenz M., Hbner A. et Kunze M., "Question answering with textual CBR", *In Proceedings of the International Conference on Flexible Query Answering Systems*, pp. 236-247, Denmark, 1998.

[Lenz *et al.* 98b] Lenz M., Bartsch-Sporl B. et Burkhard H.-D., *Case-based reasoning technology – from foundations to applications*, Stephan Wess (Editeur.), Lecture Notes in Artificial Intelligence, Springer Verlag, 1998.

[O'Sullivan *et al.* 02] O'Sullivan D., Wilson D. et Smyth B., "Improving case-based recommendation, a collaborative filtering approach", *Proceedings of the 6th European Conference on Advances in Case-Base Reasoning (ECCBR'02)*, pp. 278-291, Londres, Royaume-Uni, 2002.

[Massa et Avesani 04] Massa P. et Avesani P., "Trust-aware collaborative filtering for recommender systems", *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pp. 492-508, Agia Napa, Cyprus, October 2004.

[Massa et Bhattacharjee 04] Massa P. et Bhattacharjee B., "Using Trust in Recommender Systems: an Experimental Analysis", *Proceedings of International Conference iTrust2004*, 2004.

[McDonald et Ackerman 98].McDonald D. W. et Ackerman M. S., "Just talk to me: a field study of expertise location", *Proceedings of the 1998 ACM Conference on Computer-Supported Cooperative Work (CSCW '98)*, pp. 315-324, Seattle, Washington, Novembre 1998.

[Miller 95] Miller G. A., "WordNet: a lexical database for english", *Communications of the ACM*, **38** (11), pp.39-41, Novembre 1995.

[Na Ubon et Kimble 02] Na Ubon A. et Kimble C., "Knowledge management in online distance education", *Proceedings of the 3rd International Conference Networked Learning*, pp. 465-473, Sheffield, Royaume-Uni, Mars 2002.

[Núñez et *al.* 02] Núñez H., Sànchez-Marrè M., Cortès U., Comas Q., Martínez M. et Poch M., "Classifying Environmental System Situations by means of Case-Based Reasoning: a Comparative Study", *Proceedings of the First Biennial Meeting of the international Environmental Modelling and Software Society (iEMSs)*, pp. 450-455, Lugano, Suisses, Juin 2002.

[Porter 80] Porter M. F., "An algorithm for suffix stripping", *Program,* **14** (3), pp. 130-137, 1980.

[Resnick *et al.* 94] Resnick, P., Iacovou N., Suchak M., Bergstrom P. et Riedl J., "GroupLens: an open architecture for collaborative filtering of netnews", *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work (CSCW'94)*, pp. 175-186, Chapel Hill, Caroline du Nord, Etats-Unis, Octobre 1994.

[Richter 95] Richter M., "Introduction", *Case-based reasoning technology: from foundations to applications*, Stephan Wess (Editeur.), Springer Verlag, 1995.

[Riesbeck et Schank 89] Riesbeck C. K. et Schank R. C., "*Inside case-based reasoning*", Lawrence Erlbaum (Editeur.), 1989.

[Salton 71] Salton G., *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall (Editeur), 1971.

[Salton et McGill 83] Salton G. et McGill M. J., *Introduction to Modern Information Retrieval*, McGraw Hill (Editeur), 1983.

[Sarwar *et al.* 01] Sarwar B., Karypis G., Konstan J. et Reidl J., "Item-based collaborative filtering recommendation algorithms", *Proceedings of the 10th International Conference on World Wide Web*, pp. 285-295, Hong Kong, 2001.

[Schank 82] Schank R., *Dynamic memory*, Cambridge University Press, 1982.

[Schank et Abelson 77] Schank R. et Abelson R., *Scripts, Plans, Goals and Understanding: An Inquiry into Human knowledge Structures*, Lawrence Erlbaum Associates (Editeur), 1977.

[Skyrme 98] Skyrme D., "Knowledge Management Solutions: The Role of Technology", *ACM SIGGROUP Bulletin, Special Issue on Knowledge Management at Work*, Mars 1998.

[Smyth et Cotter 99] Smyth B. et Cotter P., "Surfing the digital wave: generating personalised television guides using collaborative, case-based recommendation", *Proceedings of the 3rd International Conference on Case-based Reasoning*, pp. 561-571, Munich, Allemagne, Juillet 1999.

[Tang et McCalla 03] Tang T. Y. et McCalla G., "Towards pedagogy-oriented paper recommendation and adaptive annotations for a web-based learning system", *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Workshop on Knowledge Representation and Automated Reasoninf for e-learning system"*, pp. 72-80, Acapulco, Mexique, Août 2003.

[Vivacqua et Lieberman 00] Vivacqua A. et Lieberman H., "Agents to Assist in Finding Help", *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI-2000)*, pp. 65-72, Le Hague, Hollande, Avril 2000.

[Watson 97] Watson, *Applying Case-Based Reasoning: Techniques for Enterprise Systems*, Morgan Kaufmann Publishers (Editeur), 1997.

[Yammine *et al* 04] Yammine K., Abdel Razek M., Aïmeur E. et Frasson C., "Discovering intelligent agent: a tool for helping students searching a library", *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS'2004)*, pp. 720-729, Maceio, Alagoas, Brésil, Août /Septembre 2004.

[Yimam-Seid et Kobsa 03] Yimam-Seid D. et Kobsa A., "Expert Finding Systems for Organizations: Problem and Domain Analysis and the DEMOIR Approach", *Journal of Organizational Computing and Electronic Commerce*, **13** (1), pp. 1-24, 2003.

[URL Wordnet] http://wordnet.princeton.edu/

[URL Wikipedia] http://fr.wikipedia.org/wiki/