Université de Montréal

RARE : Un système de recommandation de cours basé sur les règles d'association

par

Narimel Bendakir

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

Août, 2006

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

RARE : Un système de recommandation de cours basé sur les règles d'association

présenté par :
Narimel Bendakir

a été évalué par un jury composé des personnes suivantes :

Claude Frasson, président-rapporteur
Esma Aïmeur, directrice de recherche
Philippe Langlais, membre du jury

Mémoire accepté le 13 octobre 2006

Résumé

Les étudiants inscrits à des programmes d'études supérieures sont souvent confrontés à deux défis: une myriade de cours parmi lesquels choisir, et un manque de connaissances au sujet des cours à suivre. C'est généralement grâce aux recommandations de leurs superviseurs, leurs collègues et leurs amis que la plupart des étudiants choisissent leurs cours et s'enregistrent. Les systèmes de recommandation de cours ont été proposés dans la littérature comme un outil permettant d'aider les étudiants à choisir les cours les plus appropriés à leurs champs d'intérêt et à leurs besoins. Bien qu'un grand nombre de techniques aient été appliquées à ces systèmes, la combinaison du forage de données avec les évaluations des utilisateurs sous la forme d'une méthode hybride n'a pas été utilisée auparavant. Ce mémoire présente *RARE* (a course **R**ecommender system based on **A**ssociation **R**ul**E**s) [Bendakir *et al.*, 2006], un système de recommandation de cours basé sur les règles d'association. En partant d'un historique de données réelles provenant du Département d'Informatique et de Recherche Opérationnelle (*DIRO*) de l'Université de Montréal, *RARE* effectue dans un premier temps un processus de forage de données. Il analyse les différents comportements des anciens étudiants par rapport aux choix de cours qu'ils ont effectués. Le but étant de découvrir des règles significatives qui associent les cours suivis conjointement. Ces règles sont employées, par la suite, pour l'inférence des recommandations pour de nouveaux étudiants, ce qui permet à *RARE* de résoudre le problème de démarrage à froid qui représente une question centrale dans les systèmes de recommandation. Étant donné que le processus de forage de données s'effectue hors ligne, la question de gérer des données à grande échelle (*scalability*) peut également être surmontée, et plus le volume de données est important, meilleure est la qualité des connaissances découvertes. Pour tenir compte du changement des appréciations des étudiants actuels au sujet des cours et afin de bénéficier de leurs avis, *RARE* offre à ses utilisateurs la possibilité d'évaluer ses recommandations. Ainsi, il peut mettre à jour les règles et les enrichir. En d'autres termes, *RARE* combine les avantages de l'expérience des anciens étudiants avec le feedback des étudiants actuels afin de personnaliser les recommandations offertes et de permettre à ses utilisateurs de faire des choix de cours

judicieux à partir d'un grand nombre d'alternatives même si leur expérience personnelle sur ce sujet est limitée.

Mots-clés : Forage de données, systèmes de recommandation, règles de classification, règles d'association, recommandation de cours.

Abstract

Students pursuing higher education degrees are faced with two challenges: a myriad of courses from which to choose, and a lack of knowledge about which courses to follow and in what sequence. It is according to their supervisors, friends and colleagues' recommendations that the majority of them choose their courses and register. Course recommender systems have been suggested in the literature as a tool to help students make informed course selections. Although a variety of techniques have been proposed in these course recommender systems, combining data mining with user ratings in order to improve the recommendation has never been done before. This paper presents *RARE*, a course **R**ecommender system based on **A**ssociation **R**ul**E**s [Bendakir *et al.*, 2006], which was used on real data coming from the Department of Computer Science and Operations Research (*DIRO*) at the Université de Montréal. *RARE* incorporates a data mining process together with user ratings for recommendation. Starting from the history of real data, *RARE* analyses the past course selection behaviour of the former students. The aim is to discover significant rules that associate academic courses that are followed together. These rules are later used to predict recommendations for new students and, consequently, to circumvent the cold start problem, which is a central question in recommender systems. Moreover, the off line data mining process helps us overcome the problem of scalability, and the more the volume of data increases the better is the quality of extracted knowledge. Since student course appreciation changes over time, *RARE* offers to users the possibility to rate the recommendations and benefits from their opinions, thus leading to an improvement of the rules. Therefore, *RARE* combines the benefits of both former students' experience and current students' ratings in order to personalize its recommendations and to help its users to make the most appropriate choices among all the possible courses even if their personal experience on this mater is limited.

Keywords: Data mining, recommender systems, classification rules, association rules, course recommendation.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Je voudrais en premier lieu adresser tout particulièrement mes remerciements à ma directrice de recherche, professeure Esma Aïmeur, qui m'a toujours orientée dans mon travail et qui a été toujours disponible pour moi, avec ses conseils et ses encouragements.

Je tiens aussi à remercier les membres de l'honorable jury qui ont bien voulu évaluer mon travail, les professeurs Claude Frasson et Philippe Langlais.

Un remerciement spécial à mon collègue Sébastien Gambs, qui s'est avéré d'une aide précieuse tout au long de ce projet.

Je remercie tous mes collègues du laboratoire Héron qui m'ont toujours soutenue.

Je remercie également toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

Dédicaces

À mes parents, à mon mari et mon fils, à mes frères et sœurs, à mes amies et à toutes les personnes que j'estime.

Chapitre 1 : Introduction

Le volume des données mémorisées concernant les étudiants inscrits dans les programmes des études supérieures ne cesse d'augmenter et de s'enrichir jour après jour. Compte tenu du volume important de cette information qui est centrée sur les cours suivis, les notes obtenues, les spécialités, etc., il serait préjudiciable que toutes ces connaissances qui peuvent être décisives restent cachées et donc inutilisables. De ce fait, il est nécessaire de développer des outils d'analyse et d'interprétation de données qui permettront de construire les différents profils d'étudiants et ceci afin de mieux les décrire et comprendre qui ils sont, comment ils se comportent et à quel point ils sont semblables à d'autres étudiants. Ces profils permettent d'adapter les programmes d'étude, les cours offerts et les styles d'apprentissage aux capacités des étudiants, à leurs niveaux cognitifs et à leurs préférences. Vu la croissance du volume de ces données et la complexité des objectifs à atteindre, la technologie de forage de données [Fayyad *et al.*, 1996], [Han *et al.*, 2001], [Larose, 2005] devient un outil indispensable. En définition, cette technologie constitue un processus d'extraction automatique de connaissances à partir de gros volumes de données. En utilisant une variété d'outils d'analyse, les connaissances découvertes peuvent prendre plusieurs formes à savoir des relations, des modèles ou des règles qui auparavant étaient implicites et cachés dans les données. Ces différentes formes de connaissances peuvent constituer des profils d'étudiants qui sont précis et complets. Après l'établissement de ces profils, il pourrait être très utile d'intégrer des systèmes de recommandation [Resnick *et al.*, 1997], [Adomavicius *et al.*, 2005] dans l'expérience d'études. L'objectif de ces systèmes est de permettre aux étudiants de repérer, choisir, ou trier des cours, des ressources, ou des références appropriées et, par conséquent, leur fournir des conseils qui tiennent compte du profil de chacun d'eux. Un système de recommandation basé sur le forage de données [Adomavicius *et al.*, 2001], [Perugini *et al.*, 2004], [Schafer, 2005] et qui a établi des profils d'étudiants peut donc personnaliser les informations fournies et aider ses utilisateurs à faire de bons choix à partir d'un grand nombre d'alternatives même sans avoir une expérience personnelle suffisante.

1.1 Motivation et objectifs

Les étudiants inscrits aux programmes d'études supérieures sont souvent confrontés à deux défis : une myriade de cours parmi lesquels il faut choisir et un manque de connaissance au sujet des cours à suivre. Ainsi, trouver le bon cours à prendre au bon moment peut devenir une tâche ardue. C'est généralement grâce aux recommandations de leurs superviseurs, leurs collègues et leurs amis que la plupart des étudiants choisissent leurs cours et s'enregistrent. Or, il pourrait être très utile de réaliser des systèmes de recommandation dont l'objectif est de chercher les cours appropriés aux étudiants, en tenant compte de leurs connaissances et de leurs préférences. Ainsi, ces systèmes ont la capacité de réduire la quantité des cours présentés aux étudiants, leur rendant ainsi le choix beaucoup plus aisé.

Dans ce mémoire, nous proposons ***RARE*** (a course **R**ecommender system based on **A**ssociation **R**ul**E**s) [Bendakir *et al.*, 2006], un système de recommandation de cours. *RARE* se base sur le principe de faire profiter ses utilisateurs d'une expérience collaborative d'étudiants. Vu que le volume des données mémorisées des étudiants qui ont été enregistrés et qui ont achevé leurs études est important, nous pensons que l'application des techniques de forage de données sur cet ensemble de données nous permettrait de capturer le comportement des étudiants dans leurs choix de cours et, par conséquent, révéler des relations cachées entre les cours suivis. Ainsi, une première phase de développement du système consiste en un processus de forage de données et sera effectuée hors ligne. Pour disposer d'un ensemble de données réelles, nous avons collecté des données qui décrivent les profils de 230 étudiants ayant suivi le programme de maîtrise en informatique entre l'année 1990 et 1999, au sein du Département d'Informatique et de Recherche Opérationnelle de l'Université de Montréal. L'objectif du processus de forage de données est d'établir des règles initiales à partir des données collectées. Étant donné que les règles de classification et les règles d'association sont deux types de modèles fréquemment recherchés dans les processus de forage de données, et vu qu'elles constituent des modèles compréhensibles et faciles à exploiter [Larose, 2005], nous avons opté pour l'extraction de ces deux types de règles. Selon les résultats obtenus, les règles les mieux adaptées à nos besoins seront utilisées. Dans une deuxième

phase de développement de *RARE*, nous mettons en œuvre un processus de recommandation qui se base, dans sa prédiction des cours à recommander, sur les règles établies. Pour ce faire, une comparaison des règles avec l'historique des cours suivis par l'utilisateur est effectuée afin de permettre au système de personnaliser les recommandations. Cependant, nous devons souligner la nécessité de garantir la pertinence des recommandations offertes, et ceci par le biais d'une mise à jour des règles. Le but est d'enrichir ces dernières ainsi que de les adapter, au fil du temps, aux appréciations des étudiants au sujet des cours. De ce fait, *RARE* sollicite la collaboration de ses utilisateurs en leur permettant d'évaluer tous les cours qu'ils suivent. Au fur et à mesure que les étudiants procurent leur feedback, la recommandation de *RARE* s'améliore.

En résumé, nous avons deux objectifs à atteindre:

- Effectuer d'abord un processus de forage de données afin d'apprendre, à partir d'un historique de données, des règles qui décrivent les profils comportementaux des anciens étudiants dans les choix des cours qu'ils ont accomplis.
- Ensuite, combiner l'approche du forage de données avec l'approche de collaboration des étudiants actuels à travers leur feedback.

Un problème fondamental que connaissent la plupart des systèmes de recommandation au stade initial de leur fonctionnement est *le démarrage à froid* [Pazzani, 1999], [Perugini *et al.*, 2004], [Rafaeli *et al.*, 2005]. Pour produire des recommandations précises, ces systèmes requièrent une période d'apprentissage afin de détecter les préférences de leurs utilisateurs. Contrairement à ces systèmes, *RARE* se base sur les règles établies lors du forage de données. Ceci lui permet de recommander les cours dès son démarrage. Ainsi, *RARE* propose, d'une part, une solution au problème de démarrage à froid. Étant donné que le processus de forage de données s'effectue hors ligne, ceci aide *RARE* à dépasser le problème de données à grande échelle (*scalability*) [Li *et al.*, 2004], et plus le volume de données est important, meilleure est la qualité des connaissances découvertes.

D'autre part, les évaluations des utilisateurs constituent une dynamique qui permet l'enrichissement des règles et garantit ainsi une qualité progressive et une utilité permanente des recommandations futures. Avec une telle hybridation, nous estimons tirer profit à la fois des expériences passées et présentes des étudiants. Nous pensons également être capables d'offrir aux utilisateurs de *RARE* des recommandations utiles et appropriées à leurs besoins et attentes tout en gagnant leur confiance et leur fidélité.

À la fin de chacune des deux phases d'implémentation (la phase hors ligne et la phase en ligne) de *RARE*, nous avons procédé à une évaluation expérimentale.
Dans la phase en ligne et après avoir extrait les deux types de règles (règles de classification et d'association) par le processus de forage de données, nous avons opté pour les règles d'association vu qu'elles répondaient au besoin de notre application. Nous avons ensuite expérimenté ces règles afin de tester leur efficacité dans la recommandation. L'expérimentation portait sur la mesure du recouvrement et de l'exactitude des recommandations produites par les règles. Nous avons alors collecté un ensemble de données qui décrivent les cours suivis par 40 étudiants qui ont été enregistrés dans le programme de maîtrise en informatique après l'année 2000 et qui ont achevé leurs cours. Ainsi, parmi l'ensemble des règles extraites, l'expérimentation nous a permis de choisir 253 règles qui offraient le recouvrement et l'exactitude les plus élevés.
L'évaluation de la phase en ligne, quant à elle, portait sur l'expérimentation de *RARE*. Cette fois-ci, 52 étudiants ont pris part à l'évaluation. Les résultats obtenus ont montré que les étudiants ont été fortement satisfaits par le système. Ils l'ont jugé d'une grande importance et ont trouvé ses recommandations appropriées. À leur avis, *RARE* constitue une solution efficace pour le partage d'informations et d'expériences entre les étudiants passés et actuels.

1.2 Organisation du mémoire

Ce mémoire est constitué des parties suivantes : l'état de l'art, la conception et l'implémentation du système et, enfin, la conclusion.

L'état de l'art est divisé en deux chapitres. Dans le chapitre 2, nous présentons en détail le forage de données. Nous commençons par définir quelques concepts de base comme les entrepôts de données et les technologies de l'intelligence d'affaires, dont le forage de données fait partie. Nous citons les étapes à suivre dans un processus de forage de données et nous décrivons ses différentes tâches. Vu que les objectifs de notre travail consistent à découvrir les règles de classification et les règles d'association, nous expliquons plus en détail chacune de ces deux tâches. Nous concluons par une brève description du rôle du forage de données ainsi que de son application dans les systèmes de recommandation.

Dans le chapitre 3, nous nous intéressons aux systèmes de recommandation. Nous définissons dans un premier temps la recherche et le filtrage d'information. Nous décrivons par la suite la notion de profil d'utilisateur, nous expliquons la recommandation et nous évoquons les différentes techniques que cette dernière utilise. Les facteurs qui influencent la qualité de ces systèmes sont également mentionnés. Nous abordons ensuite la recommandation de cours et nous présentons une revue de quelques systèmes qui ont été développés dans ce domaine.

Le chapitre 4, quant à lui, présente l'approche suivie dans la conception de *RARE* et schématise son architecture. Il décrit ensuite plus en détails les différentes composantes de l'architecture proposée ainsi que les bases de données utilisées par le système.

Dans le chapitre 5, nous présentons l'implémentation de toutes les étapes expliquées dans la conception de *RARE*. Nous éclaircissons ensuite les fonctionnalités offertes par ce système à travers un scénario de recommandation. Finalement, l'évaluation de *RARE* est également présentée dans ce chapitre.

Pour conclure, nous discutons de *RARE* dans le chapitre 6. Nous exposons ses forces et ses faiblesses à travers sa comparaison avec les autres systèmes de recommandation de cours. Nous suggérons des travaux futurs pour remédier aux faiblesses mentionnées et pour améliorer l'efficacité du système.

Notons que dans la suite de ce mémoire, l'utilisation du masculin sera utilisée pour référer aussi bien au masculin qu'au féminin.

Chapitre 2 : Forage de données

Jour après jour, de plus en plus de données sont collectées et accumulées par les entreprises. En même temps, l'automatisation est de plus en plus présente suite à l'augmentation des guichets automatiques, des télévirements, des cartes de crédit et débit, et de l'étiquetage électronique. Tous ces facteurs mènent à la production et à la collection des données qui alimentent des bases de données de taille sans précédent. Cependant, les données ne sont jamais stockées dans un seul endroit et ne sont typiquement pas dans le format adéquat pour soutenir les processus de décision dans les affaires. Ainsi, d'importants moyens sont mobilisés par les entreprises pour collecter et conserver les données dans des aires communes connues par le nom d'*entrepôts de données* (*Data Warehouses*). Leur mission est de contenir des données qui sont intègres, consistantes, nettoyées et prêtes pour une utilisation stratégique. Par ailleurs, il est nécessaire d'avoir la capacité de comprendre aisément ces grandes quantités de données et d'employer efficacement les informations qu'elles contiennent. Pour avoir cette capacité, les entreprises se sont trouvées obligées de faire appel à *l'intelligence d'affaires* (*Business intelligence*) qui est leur seul moyen de passer des données à l'information, puis à la connaissance et finalement à l'action. Une technologie de pointe de l'intelligence d'affaires est *le forage de données* (*Data Mining*). Cette technologie consiste à analyser des grandes quantités de données pour en faire sortir les informations significatives et précédemment inconnues afin de les utiliser pour la prise des décisions économiques importantes.

Dans ce chapitre, nous présentons un état de l'art sur le forage de données. Nous commençons par définir quelques concepts de base comme les entrepôts de données et les technologies de l'intelligence d'affaires, dont le forage de données fait partie. Nous ne nous arrêtons pas seulement aux définitions, mais nous présentons aussi une comparaison des trois technologies afin de mettre en évidence les points forts du forage de données. Nous décrivons ensuite la modélisation de données ainsi que les étapes à suivre dans un processus de forage de données. Afin de comprendre les tâches pouvant être accomplies par le forage de données, nous décrivons d'abord chacune d'elles. Vu que les objectifs de

notre application consistent à classifier les données ou à découvrir des règles d'association, nous expliquons plus en détail chacune de ces deux tâches. Nous récapitulons ensuite les caractéristiques de toutes les tâches sans oublier d'expliciter quelques techniques et algorithmes utilisés dans leurs implémentations. Finalement, la conclusion décrit brièvement le rôle du forage de données dans la description et la prédiction, et introduit son application dans les systèmes de recommandation.

2.1 Entrepôt de données

L'entrepôt de données est un processus qui permet d'organiser le stockage de grands ensembles de données d'une manière facilitant la recherche et la récupération d'informations pour des fins d'analyse de données [URL1]. Son objectif principal est de fournir des données intègres, conformes, propres et utilisables. La composante la plus fondamentale dans un entrepôt de données est la base de données relationnelle qui représente l'endroit où les données sont stockées [Thearling, 2000]. Ainsi, de la perspective d'utilisateur, l'entrepôt constitue une source de données. L'accès à celle-ci est fourni par la requête de l'utilisateur, les outils de création de rapport (*reporting*), l'analyse, ou les outils de forage de données. Ainsi, l'entrepôt de données est considéré comme un serveur d'informations et un outil d'aide à la décision [Berson *et al.*, 2000], [Kantardzic, 2002].

Dans la section qui suit, nous présentons les différentes technologies de l'intelligence d'affaires et nous décrivons comment chacune d'elles interroge les entrepôts de données.

2.2 Les technologies de l'intelligence d'affaires

Le terme *intelligence d'affaires* est employé pour décrire tous les processus, les techniques et les outils qui soutiennent la prise de décision d'affaires [Cabena *et al.*, 1998]. Dans cette section, nous tenons compte des trois technologies d'intelligence d'affaires à savoir : *les requêtes* (*Ad Hoc Query*), *le traitement analytique en ligne* (*on-line analytical processing : OLAP*) et *le forage de données* (*Data Mining*).

Grâce aux outils traditionnels d'analyse tels que les requêtes et le traitement analytique en ligne (*OLAP*), l'analyste génère une série de questions basées sur sa connaissance du domaine. Guidé par une idée ou une hypothèse qu'il veut tester, il explore les données à la recherche des réponses à ses questions. Ces réponses sont utilisées pour la déduction d'un modèle de données et la vérification de son hypothèse. Il est donc de la responsabilité de l'analyste d'établir un modèle de données satisfaisant. Comme la structure des bases de données se développe et la complexité des questions et des objectifs augmente, il devient pratiquement impossible pour qu'une personne connaisse les données suffisamment pour dire, avec confiance, quelles sont les variables qui affectent un certain comportement. Le forage de données est une technologie qui utilise diverses techniques pour extraire, d'une grande quantité de données, l'information cachée et utile sous forme de modèles explicites et compréhensibles. Pour mieux comprendre les capacités du forage de données, il est utile de le comparer aux deux autres technologies d'intelligence d'affaires que sont les requêtes et le traitement analytique en ligne. Les trois sections suivantes décrivent les technologies suscitées. Une comparaison entre elles est présentée dans la section 2.2.4.

2.2.1 Requêtes

Dans une application qui utilise les requêtes (*Ad Hoc Query*), l'utilisateur a la capacité d'accéder à l'information sur demande. Ainsi, il n'obtient que ce qu'il a demandé, ni plus, ni moins. Cette technologie est utile lorsqu'on sait exactement ce qu'on cherche et qu'on peut le décrire formellement [Kantardzic, 2002]. Par exemple, si l'utilisateur pose la question : « *quel est le revenu produit par chaque client pendant cette année* ? ». Les réponses à cette question contiennent les noms des clients et les revenus pour l'année choisie, ce qui est exactement l'information demandée [Wu, 2000].

2.2.2 Traitement analytique en ligne

Le traitement analytique en ligne (*OLAP*) est basé sur des concepts de bases de données multidimensionnelles qui permettent aux utilisateurs d'analyser les données en utilisant une grande variété de vues raffinées, multidimensionnelles et complexes [URL1]. Il

procure aux utilisateurs la capacité d'explorer et d'analyser manuellement l'information récapitulative et détaillée [Wu, 2000], et les assiste dans la construction et la production des requêtes très complexes. La définition du traitement analytique en ligne a été aussi récapitulée sous la forme de cinq mots clés : *analyse rapide d'information multidimensionnelle partagée* (*Fast Analysis of Shared Multidimensional Information* : *FASMI*) [URL4]. Par exemple, un directeur de vente peut vouloir comprendre pourquoi les ventes ont chuté dans une zone particulière. Les outils de traitement analytique en ligne lui permettent de poser des questions à travers des dimensions multiples, telles que les ventes par magasin, les ventes par produits et les ventes par périodes de temps. En regardant les données sous différents angles, le directeur peut analyser quels facteurs parmi : magasin, produit, ou temps ont influencé les ventes [Zaima, 2003]. En revanche, cette technologie d'intelligence d'affaires ne peut pas révéler les relations cachées ou implicites dans les données.

2.2.3 Forage de données

Malgré que la plupart des techniques utilisées dans le forage de données, existaient depuis des années ou des décennies dans le milieu académique, c'est seulement lors de la dernière décennie que le forage de données, dans le domaine du commerce, a adopté ces techniques et a commencé à les utiliser de façon intensive [Berry *et al.*, 2004]. Depuis ce moment là, plusieurs définitions lui ont été données dont nous citons quelques exemples :

- Le terme *KDD* (*Knowledge Discovery in Databases*) est employé pour désigner le processus global de découverte de la connaissance utile dans les données. Le forage de données représente une étape particulière dans ce processus. Il consiste à appliquer des algorithmes spécifiques pour extraire des modèles à partir des données. Les étapes additionnelles du processus de *KDD* à savoir : le choix et la préparation des données, la suppression des erreurs, l'incorporation de la connaissance et l'interprétation appropriée des résultats, assurent que la connaissance utile soit dérivée des données [Fayyad *et al.*, 1996].

- Le forage de données est un processus qui consiste à trouver des tendances et des modèles dans les données. Son objectif est de fouiller dans de grandes quantités de données pour découvrir des nouvelles informations. L'idée est de transformer cette nouvelle connaissance en résultats fonctionnels, tels qu'augmenter la probabilité d'acheter un produit par les clients, ou diminuer le nombre de réclamations de fraudes [Groth, 1999].

- Le forage de données emploie une variété d'outils d'analyse de données pour découvrir des modèles et des rapports qui peuvent être employées pour faire des prédictions raisonnablement précises. Cependant, le forage de données n'est pas une technique ou un algorithme particulier, il represente plutôt un processus dont l'objectif ultime est la prédiction qui constitue la généralisation d'un modèle à d'autres données. Explorer et décrire la base de données est simplement le point de départ [Edelstein, 2003].

- Le forage de données représente une des applications majeures des entrepôts de données dont l'avantage par rapport aux autres outils et applications est sa capacité d'extraire l'information cachée et non triviale [Kantardzic, 2002]. C'est un champ interdisciplinaire rassemblant des techniques issues de l'apprentissage machine, de l'identification de modèle, des statistiques, des bases de données, et de la visualisation pour attaquer la question de l'extraction d'information à partir des grandes bases de données [Larose, 2005].

Prenons un exemple d'application de forage de données où l'utilisateur pose la question suivante: « *quelle est la couleur de siège de voiture qu'un parent préférerait acheter pour son enfant* ? » C'est une question qui est plus difficile à répondre et exige de prendre en compte de nombreuses variables. Celles-ci pourraient être l'âge de l'enfant, le niveau de revenu des parents, la couleur de la voiture, le sexe de l'enfant, etc. Même la couleur des cheveux de l'enfant pourrait avoir un impact sur la couleur du siège qui va être acheté. Il pourrait ainsi y avoir tout genre d'informations qui pourraient aider à comprendre quelle préférence de couleur un parent pourrait avoir.

2.2.4 Comparaison entre les technologies d'intelligence d'affaires

Grâce à l'utilisation de l'intelligence artificielle et de techniques statistiques et mathématiques, les capacités du forage de données sont au delà de celles des autres technologies d'intelligence d'affaires. Une comparaison générale entre celles-ci et le forage de données est présentée dans le tableau 2.1.

Tableau 2.1 : Forage de données comparé aux outils de requête et de traitement analytique en ligne

Outils de requête et de traitement analytique en ligne	***Forage de données***
L'interrogation de données s'effectue par l'utilisateur.	L'interrogation de données s'effectue par les algorithmes de forage de données [Mena, 1998].
Les outils de requête posent des questions spécifiques.	Les outils de forage de données recherchent l'information liée à un but défini [Berson *et al.*, 2000].
Les outils de traitement analytique en ligne peuvent aider l'utilisateur à répondre à des questions multidimensionnelles très complexes.	Les résultats de forage de données peuvent prendre la forme de modèles, de tendances, ou de règles qui sont implicites dans les données [Wu, 2000].
Les outils de requête et les outils de traitement analytique en ligne permettent l'exploitation et l'analyse de l'historique de données. Ces deux technologies sont rétrospectives.	Le forage de données permet d'établir des modèles prédictifs qui prévoient le futur en se basant sur l'analyse du passé. Le forage de données est donc prospectif [Noonan, 2000], [Zaima, 2003].
Les outils de requête et les outils de traitement analytique en ligne ne peuvent pas faire face aux volumes très grands de données.	Le forage de données devient de plus en plus utile à mesure que la quantité de données stockées augmente [Groth, 1999].

En fait, le but du forage de données peut se résumer à faire sortir, d'une masse importante de données, des connaissances implicites et utiles. Ces connaissances sont représentées sous une forme qu'on appelle « *modèle* ». Dans la section suivante, nous donnons une définition précise à ce terme, et nous classons ensuite les modèles dans deux types principaux que sont : les *modèles descriptifs* et *prédictifs*.

2.3 Le modèle, le profilage et la prédiction

La modélisation consiste à établir un modèle qui représente une description de la base des données actuelles, et qui peut être utilisée pour faire des prédictions sur des nouvelles données [Thearling, 2005a]. D'un certain point de vue, un modèle est un algorithme ou un ensemble de règles qui relie une collection des données d'entrée, en général sous forme de champs dans une base de données, à une sortie ou un résultat cible particulier. La *régression*, les *règles d'association*, les *arbres de décision* et beaucoup d'autres techniques sont utilisées pour produire des modèles. Une fois établi, le modèle aide dans la prise de décision. Il doit donc être interprétable, utile et intuitivement simple. Cependant, le but principal du forage de données est la stabilité de ces modèles. Une bonne manière de décider si un modèle ou une règle est stable est de comparer sa performance sur plusieurs échantillons choisis aléatoirement à partir du même ensemble de données [Berry *et al.*, 2004]. On peut distinguer entre deux types de modèles produits par les techniques de forage de données :

- ***Modèle descriptif (modèle pour le profilage)***

  Du point de vue de la description, on peut par exemple construire un modèle à partir d'une base de données des clients qui ont répondu à une offre particulière ou qui ont acheté un produit particulier. Ce modèle peut servir à identifier les profils des clients potentiels [URL3]. Pour produire ces modèles, les applications utilisent divers types de données des clients. Les deux types principaux sont : *effectif* et *transactionnel* [Adomavicius *et al.*, 2001]. Le type effectif se veut répondre à la question « *qui est le client* ? », tandis que le type transactionnel s'occupe de « *que fait le client* ? ».

- ***Modèle prédictif***

  La prédiction utilise les données issues du passé pour prédire ce qui est susceptible de se produire à l'avenir. Cela est une utilisation plus puissante des données que simplement le modèle descriptif [Berry *et al.*, 2004]. La prédiction fournie par un modèle s'appelle habituellement un score. Ce dernier, qui est typiquement une valeur numérique, est assigné à chaque enregistrement de la base de données. Il indique la probabilité que l'enregistrement (le client) dont le score a été calculé exhibe un comportement particulier [Zaima *et al.*, 2003], [Thearling, 2005b].

En réalité, le forage de données n'est pas juste l'application d'une technique pour l'extraction de modèles. Il est plutôt un processus constitué de plusieurs étapes qui doivent être réalisées dans un ordre particulier. Nous présentons en détails le processus de forage de données dans la section qui suit.

2.4 Le processus de forage de données

La génération des modèles est seulement une étape du long processus de transformation de données en connaissance. Le plus grand défi de l'utilisation de forage de données est de savoir extraire, intégrer, nettoyer et préparer les données pour résoudre un problème particulier. Selon [Cabena *et al.*, 1998], un processus global de forage de données se compose de cinq étapes : la *détermination de l'objectif*, la *préparation de données*, le *forage de données*, l'*analyse de résultats* et l'*assimilation de la connaissance*. Cependant, le modèle *CRISP-DM* (*CRoss Industry Standard Process for Data Mining*) [URL5], [Shearer, 2000] a ajouté aux cinq étapes précédentes, l'étape de la compréhension de données, et a organisé le processus de forage de données en six étapes : la *compréhension d'affaires*, la *compréhension de données*, la *préparation de données*, la *modélisation*, l'*évaluation* et le *déploiement*. Dans la section qui vient, nous présentons en détails ces six étapes.

2.4.1 Détermination de l'objectif (compréhension d'affaires)

La détermination des objectifs ou la compréhension d'affaires est un ingrédient essentiel de n'importe quel projet de forage de données. Elle consiste à définir correctement et clairement les objectifs de l'application. Elle consiste à identifier le problème d'abord, pour ensuite le présenter sous la forme de l'une des six tâches de forage de données. Ceci a pour but de faciliter le développement d'un plan préliminaire conçu pour atteindre les objectifs [URL6], [Clifton *et al.*, 2001]. Un rapport efficace inclut la manière de mesurer les résultats du projet et même une justification de coût. Par ailleurs, le développement d'une hiérarchie de choix et de décisions qui sont nécessaires avant de commencer le forage de données est conseillé. Cette hiérarchie est constituée des points suivants : l'objectif d'affaires, le type de la prédiction, le type du modèle, l'algorithme et le produit à utiliser [URL3]. On peut aussi formuler plusieurs hypothèses pour un seul problème. L'étape de détermination de l'objectif nécessite donc une expertise combinée d'un expert en forage de données et un expert de l'application. Cette coopération continuera tout le long du processus de forage de données [Kantardzic, 2002].

2.4.2 Compréhension de données

La phase de compréhension de données commence par une première collecte des données nécessaires, car il est possible que certaines données utiles n'aient été jamais rassemblées auparavant. Il est alors nécessaire d'identifier toutes les sources internes ou externes d'information et choisir les sous-ensembles de données qui sont pertinents et nécessaires [URL2]. Par la suite, il faudra augmenter la familiarité avec les données, identifier des problèmes de qualité de données, découvrir des connaissances initiales dans les données et détecter les sous-ensembles pouvant être intéressants pour former des hypothèses sur l'information cachée [URL5].

2.4.3 Préparation de données

La préparation de données couvre toutes les activités pour préparer, à partir des données initiales, l'ensemble de données final qui sera utilisé pour la construction de modèles.

Elle est constituée de cinq étapes : la *sélection*, le *nettoyage*, la *construction*, l'*intégration* et le *formatage* des données.

2.4.3.1 Sélection de données

Les sources de données qui sont utiles et disponibles varient d'un problème à un autre. Il est donc important d'éliminer les données inutiles ou non pertinentes. Les critères pouvant aider à décider quelles sont les données importantes peuvent inclure la pertinence des données par rapport aux objectifs, leur qualité et même les contraintes techniques telles que des limites sur le volume ou les types de données [Shearer, 2000], [URL6].

2.4.3.2 Nettoyage de données

L'étape de nettoyage de données est habituellement nécessaire. Elle consiste à détecter et supprimer les erreurs et les contradictions figurant dans les données. Cette étape prend tout son sens lorsque plusieurs sources de données doivent être combinées du fait qu'il y a souvent des redondantes dans différentes représentations [Chen *et al.*, 2004]. Le nettoyage tient compte aussi d'autres types de problèmes de qualité comme : les valeurs de champs incorrectes (par exemple le numéro d'identification de sécurité sociale se trouvant à la place du revenu d'une personne), des valeurs qui semblent être correctes (par exemple les mâles enceintes), les valeurs absentes, un même nom employé pour différentes entités ou différents noms employés pour la même entité, etc. [URL3].

2.4.3.3 Construction de données

La construction de données consiste à développer des enregistrements entièrement nouveaux ou produire des attributs dérivés des attributs existants. Ainsi, certaines variables qui ont peu d'effet lorsque prises seules, peuvent être combinées avec d'autres variables en utilisant l'arithmétique ou les opérations algébriques, telles que l'utilisation du ratio « *dette* / *revenu* » au lieu de simplement la *dette* ou du *revenu*. Quelques variables qui sont définies sur un grand intervalle peuvent aussi être modifiées en créant

une meilleure variable telle que l'utilisation du *log* du revenu au lieu du revenu [URL3]. Un autre type d'attribut dérivé est la transformation *simple-attribut* comme la transformation des champs symboliques en des valeurs numériques. Le but de cette transformation est de se conformer au format requis par les outils de construction de modèle [Shearer, 2000], [Zaima *et al.*, 2003].

2.4.3.4 Intégration de données

L'intégration de données combine les données de différentes sources en une seule base de données. Ceci consiste en la combinaison de l'information provenant des différentes tables ou enregistrements. L'intégration de données couvre également les *agrégations* qui consistent à calculer des nouvelles valeurs en récapitulant l'information de plusieurs enregistrements et/ou plusieurs tables [URL6]. Par exemple, une agrégation pourrait inclure la conversion d'une table des achats de client, où il y a un enregistrement pour chaque achat, en une nouvelle table où il y a un enregistrement pour chaque client [Shearer, 2000].

2.4.3.5 Formatage de données

Le formatage de données est souvent nécessaire afin de convertir les données en un format requis par la technique de forage de données qui sera employée. Il est commun pour certains types d'information, d'être représentées dans différents formats si provenant de différentes sources de données. La *date* est probablement le cas le plus commun. Le champ *date* extrait à partir d'une source de données européenne dans le format « *DD-MM-YYYY* », pourrait être transformé dans le format « *MM-DD-YYYY* » pour faciliter l'assimilation des résultats de forage de données [Mendonca *et al.*, 1999]. Il est nécessaire, donc, de s'assurer que ce type d'information est représenté dans un format cohérent et connu par l'entrepôt de forage de données.

2.4.4 Modélisation

Différentes techniques de forage de données peuvent être utilisées pour résoudre un problème particulier. La phase de modélisation consiste à sélectionner une technique ou une combinaison de techniques appropriée comme, par exemple, les arbres de décision et les réseaux de neurones afin d'établir, examiner et choisir les modèles. Par conséquent, il est nécessaire d'avoir une importante variété d'outils et de technologies afin de trouver le meilleur modèle [URL3]. Indépendamment du choix des techniques, la phase de modélisation est rapide et automatisée [Cabena *et al.*, 1998]. Une chose importante à se rappeler est que la modélisation est un processus itératif. On doit donc explorer les différents modèles établis pour trouver le plus utile pour la résolution du problème. Lors de la modélisation, quelques techniques ont des conditions spécifiques sur la forme de données. Par conséquent, un retour à la phase de préparation de données est souvent nécessaire [Edelstein, 2000].

2.4.5 Évaluation

Avant de passer à la phase de déploiement final du modèle construit, il est important de l'évaluer et de revoir les étapes exécutées lors de sa construction afin d'être certain qu'il répond aux objectifs fixés. Un point clé est de déterminer s'il y a une question importante qui n'a pas été suffisamment adressée. À la fin de cette phase, la décision concernant l'utilisation des résultats de forage de données doit être prise [URL5].

2.4.6 Déploiement

La création du modèle ne constitue pas la fin du projet. Même si le but du modèle est d'acquérir une nouvelle connaissance, celle-ci doit être organisée et présentée de telle manière à ce le client puisse l'employer [Shearer, 2000]. Souvent les modèles trouvés font partie d'un processus d'affaires tel que l'analyse de risque, l'autorisation de crédit, ou la détection de fraude. Le modèle ou l'intelligence produite peut être déployée de différentes manières. On peut, par exemple, l'intégrer dans la gestion de rapport de client (*CRM* : *Customer Relationship Management*). Un autre exemple consiste à intégrer un

modèle prédictif dans une application de prêt hypothécaire pour aider un agent de prêt à évaluer le demandeur. Indépendamment de la façon de déployer le modèle, le forage de données ajoute de l'intelligence aux systèmes d'information des organisations sous forme de scores, de prédictions, de descriptions ou de profils [Zaima *et al.*, 2003].

Les étapes du processus de forage de données ne sont pas égales en termes de temps et d'effort dépensé. La figure 2.1 représente selon [Cabena *et al.*, 1998], les étapes du processus de forage de données ainsi que l'effort relatif à chacune d'elles. Ainsi, on peut voir que 60% du temps est conservé à la préparation de données. L'étape de l'application des techniques de forage de données afin d'établir les modèles constitue typiquement 10% de l'effort global.

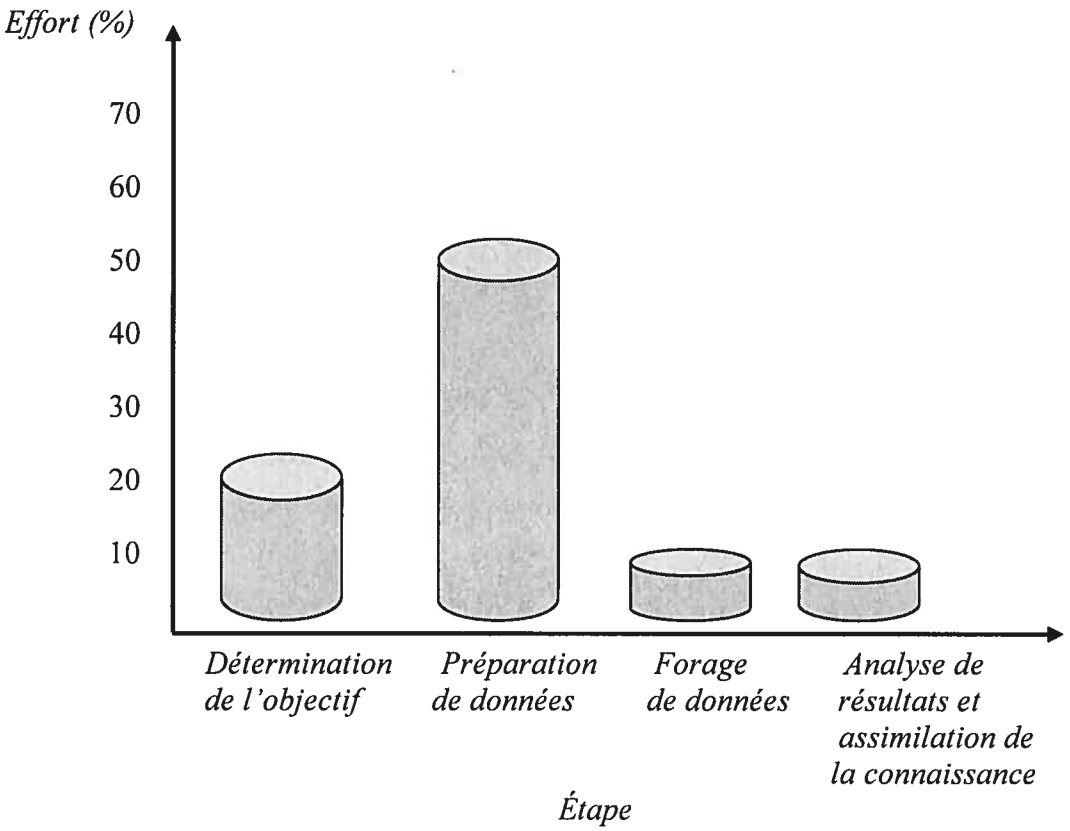


Figure 2.1 : Effort relatif à chaque étape du processus de forage de données [Cabena *et al.*, 1998]

Afin de résoudre un problème quelconque, celui-ci doit être présenté sous la forme de l'une des tâches pouvant être accomplie par le forage de données. La section 2.6 explique les différentes tâches et essaye de récapituler leurs caractéristiques sans oublier de citer des exemples de techniques utilisées par chacune d'elles.

2.5 Les tâches de forage de données

Les tâches du forage de données spécifient le genre de modèle qu'on souhaite trouver. En général, elles peuvent être regroupées en deux catégories de forage : *le forage de données descriptif* et *le forage de données prédictif* [Han *et al.*, 2001]. Le rôle de la description est de caractériser les propriétés générales des données. Elle cherche donc à décrire des relations et des motifs figurant dans les données. Les modèles dérivés doivent être interprétables par l'humain et offrent une vue globale des données analysées. De même, la prédiction utilise quelques variables ou champs de l'ensemble de données pour prévoir des valeurs futures ou inconnues d'autres variables. Son objectif est de produire un modèle qui peut être utilisé pour faire de la classification, de la régression, ou d'autres tâches similaires. Le compromis entre l'importance de la prédiction d'une part et de la description d'autre part, suivant les applications, varie considérablement [Kantardzic, 2002].

Dans la littérature de l'apprentissage machine, on trouve les deux termes suivants: *forage de données supervisé* et *forage de données non supervisé* pour décrire les deux catégories de forage de données. Dans le forage de données supervisé, on a toujours une variable cible pré-spécifiée. Cette variable représente l'élément qu'on veut classer, estimer, ou prédire [Berry *et al.*, 2004]. L'algorithme de forage de données reçoit en entrées plusieurs exemples dont les valeurs de la variable cible sont déjà connues. Il apprend à partir de ces exemples quelles valeurs de la variable cible sont associées à quelles valeurs des variables prédicatrices [Larose, 2005]. La modélisation dans ce cas, consiste à trouver des règles qui expliquent les valeurs connues de la variable cible. Dans le forage de données non supervisé, aucune variable cible n'est identifiée. Le but est de générer les modèles globaux qui prennent en considération toutes les variables. La forme la plus commune de forage de données non supervisé est *la détection automatique des*

segments (*clustring*) qui trouve les groupes des enregistrements similaires. Dans ce cas, toutes les variables appropriées sont des entrées de l'algorithme de forage de données sans aucune spécification d'une variable cible. La plupart des techniques utilisées pour effectuer les tâches de forage de données sont supervisées. Cependant, la technique qui peut être supervisée ou non supervisée est *la recherche des règles d'association* [Larose, 2005]. La section suivante explique les six tâches de forage de données.

2.5.1 Classification

Supposons qu'on dispose d'un ensemble d'objets caractérisés par certains attributs et qui appartiennent à différentes classes. Chaque objet possède des variables appelées les variables prédicatrices. Celles-ci constituent des informations sur la variable cible ou la classe [Larose, 2005]. Un algorithme de classification examine l'ensemble des objets afin d'extraire un modèle de classification qui décrit et permet de distinguer les différentes classes de données. Ce modèle est ensuite utilisé pour la prédiction des classes d'objets inconnues [Han *et al.*, 2001]. Par exemple, un problème d'évaluation de risque de crédit d'un client peut être transformé en un problème de classification. On peut ici distinguer entre deux classes de client qui sont: *bon client* et *mauvais client*. Dans un premier temps, un modèle de classification est produit à partir des données existantes sur les clients et leurs comportements de crédit. Ce modèle est ensuite employé pour classer un nouveau client dans l'une des deux classes mentionnées précédemment, ce qui permet d'accepter ou de rejeter son crédit [Shearer, 2000]. Les modèles de classification peuvent être représentés sous différentes formes, allant des arbres de décision aux règles de classification (*si-alors*), ou encore aux réseaux de neurones. Afin d'apporter plus d'éclaircissement sur la classification, nous présentons dans la section suivante le modèle d'*arbres de décision*. Nous décrivons également l'algorithme *C4.5* qui fait partie des algorithmes de génération d'arbres de décision les plus cités.

2.5.1.1 Arbres de décision

Un arbre de décision est une manière de structurer les données sous la forme d'un arbre. Ce dernier part d'un nœud racine qui représente l'ensemble des cas. Chaque nœud interne

constitue ensuite un test sur un attribut dont le résultat va conditionner quelle branche il faut suivre dans l'arbre. Les nœuds feuilles, aussi appelées extrémités, constituent les classes. Pour classer un nouveau cas, les valeurs de ses attributs sont testées en traversant l'arbre de décision. Ainsi, un chemin parcouru part de la racine de l'arbre et descend d'un nœud à un autre jusqu'à la rencontre d'une feuille. Au niveau de chaque nœud intermédiaire, le résultat du test indique comment se déplacer dans l'arbre. Chaque nœud intermédiaire peut être considéré comme étant le nœud racine d'un sous arbre. La fin du processus de classement se produit lorsqu'on atteint une feuille qui représente la valeur prédite de la classe de ce nouveau cas [Berson *et a*l., 2000], [Berry *et al.*, 2004].

Plusieurs algorithmes peuvent être utilisés dans la construction d'arbres de décision parmi lesquels on peut citer : *CHAID* (*Chi-squared Automatic Interaction Detection*) [Hartigan, 1975], *CART* (*Classification And Regression Trees*) [Breiman *et al.*, 1984], *C4.5* [Quinlan, 1993]. On appelle *arbres de classification*, les arbres de décision utilisés pour la prédiction des classes. Les arbres de décision utilisés pour la prédiction de variables continues sont désignés par le nom d'*arbres de régression* [URL3]. À titre d'exemple, l'algorithme *CART* peut être utilisé à la fois pour la construction d'arbres de classification et d'arbres de régression.

2.5.1.2 L'algorithme *C4.5*

Un des premiers algorithmes de construction d'arbres de décision est l'algorithme *ID3* qui fut développé par [Quinlan, 1986]. Il base la construction d'un arbre de décision sur le principe « *diviser pour régner* ». Son fonctionnement est expliqué ci-dessous. Plus tard, *ID3* fut amélioré par la version *C4.5* [Quinlan, 1993]. Celle-ci fonctionne avec les variables prenant des valeurs continues et tient compte des variables comportant des valeurs manquantes. La dérivation des règles a été aussi introduite pour cette version [Berson *et al.*, 2000]. Dans la suite de cette section, qui se base sur les références suivantes : [Quinlan, 1986], [Quinlan, 1993], [Han *et al.*, 2001] et [URL7], nous présentons le fonctionnement de l'algorithme *C4.5*.

- **Construction d'arbre de décision**

L'arbre commence par un nœud représentant l'ensemble des cas d'entraînement T.

- Si l'ensemble T contient un cas ou plus, tous appartenant à une seule classe, alors l'arbre de décision est une feuille définie par cette classe.
- Si l'ensemble T ne contient aucun cas, alors l'arbre de décision est une feuille, mais la classe associée à la feuille doit être déterminée à partir d'information autre que l'ensemble. L'algorithme C4.5 utilise la classe la plus fréquente.
- Si l'ensemble T contient des cas appartenant à un mélange de classes, alors un test est choisi en se basant sur un attribut particulier qui a un ou plusieurs résultats mutuellement exclusifs $\{O_1, O_2, \ldots, O_n\}$. L'ensemble T est partitionné en des sous-ensembles $T_1, T_2, \ldots, T_n$, dont T_i contient tous les cas dans T qui ont la valeur O_i du test choisi. L'arbre de décision de T est constitué d'un nœud de décision qui identifie le test, et une branche pour chaque résultat possible. La même construction d'arbre est appliquée récursivement à chaque sous-ensemble d'entraînement, de telle manière que la branche i mène à l'arbre de décision construit à partir du sous- ensemble T_i des cas d'entraînement.

La partition successive de l'ensemble des cas d'entraînement s'effectue jusqu'à ce que les sous-ensembles soient constitués des cas appartenant à une seule classe. L'algorithme *C4.5* se base sur une notion de la *théorie de l'information* appelée *entropie* [Shannon, 1948]. Une mesure appelée le *gain d'information* est utilisée pour la sélection de l'attribut de test au niveau de chaque nœud de l'arbre. Plus un attribut apporte de l'information au niveau d'un nœud, plus il est intéressant de le considérer pour être un attribut de test pour la partition de données au niveau de ce nœud. Ainsi, cet attribut minimise l'information nécessaire pour la classification des cas dans les partitions résultantes.

Soit S un ensemble de données d'entraînement. Supposons qu'on a m classes distinctes $C_1, C_2, \ldots, C_m$. Soit s_i le nombre des cas de S dans la classe C_i. L'information prévue nécessaire dans la classification d'un cas donné est donnée par l'équation (1).

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad (1)$$

p_i étant la probabilité qu'un cas arbitraire appartient à la classe C_i et est estimé par la fréquence $|s_i| / |s|$. Soit A un attribut ayant v valeurs distinctes (a_1, a_2, …, a_v). L'attribut A peut être utilisé pour partitionner S en v sous-ensembles $\{S_1, S_2, \ldots, S_v\}$, dont S_j contient les cas de S qui ont la valeur a_j de A. Si A est considéré comme étant le meilleur attribut de test, alors les sous-ensembles correspondent aux branches dérivées du nœud qui contient l'ensemble S.

Soit s_{ij} le nombre des cas de la classe C_i dans un sous ensemble S_j. L'entropie ou l'information prévue basée sur la partition à des sous-ensembles par l'attribut A est calculée par l'équation (2) :

$$E(A) = \sum_{j=1}^{v} \frac{s_{ij} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj}). \quad (2)$$

Pour un sous-ensemble donné S_j,

$$I(s_{1j}, s_{2j}, \ldots, s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij})$$

Où $p_{ij} = \frac{s_{ij}}{|S_j|}$ est la probabilité qu'un cas dans S_j appartienne à la classe C_i.

L'information apportée par l'attribut A est calculée par l'équation suivante:

$$Gain(A) = I(s_1, s_2 \ldots, s_m) - E(A) \quad (3)$$

Gain (A) est la réduction prévue dans l'entropie causée par la connaissance de la valeur de l'attribut A. L'algorithme calcule le gain d'information pour chaque attribut. Ensuite, l'attribut avec le gain d'information le plus élevé est choisi comme attribut de

test pour l'ensemble donné S. Un nœud est crée pour chaque valeur de l'attribut, et les cas sont partitionnés en conséquence. Prenons comme exemple l'ensemble de données illustré dans le tableau 2.2.

Tableau 2.2 : Données d'entraînement

ID	*Age*	*Revenu*	*Étudiant*	*Estimation_crédit*	*Classe : Acheter_ordinateur*
1	≤30	haut	non	acceptable	non
2	≤30	haut	non	excellent	non
3	31...40	haut	non	acceptable	oui
4	>40	moyen	non	acceptable	oui
5	>40	bas	oui	acceptable	oui
6	>40	bas	oui	excellent	non
7	31...40	bas	oui	excellent	oui
8	≤30	moyen	non	acceptable	non
9	≤30	bas	oui	acceptable	oui
10	>40	moyen	oui	acceptable	oui
11	≤30	moyen	oui	excellent	oui
12	31...40	moyen	non	excellent	oui
13	31...40	haut	oui	acceptable	oui
14	>40	moyen	non	excellent	non

L'attribut classe « *acheter_ordinateur* » peut avoir l'une des deux valeurs : *oui* ou *non*. On a donc deux classes distinctes et m = 2. Soit C_1 la classe qui correspond à la valeur *oui* et C_2 la classe qui correspond à la valeur *non*. Neuf cas appartiennent à la classe *oui* et cinq cas appartiennent à la classe *non*. Pour calculer le gain d'information apporté par chaque attribut, on utilise d'abord l'équation (1) afin de calculer l'information nécessaire dans le classement d'un cas donné :

$$I(s_1, s_2) = I(9,5) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

Après, on calcule l'entropie de chaque attribut. Commençons par l'attribut age:

Pour age = "$\leq$ 30'': $s_{11}= 2$, $s_{31} = 3$, $I(s_{11}, s_{21}) = 0.971$

Pour age = "31...40'': $s_{12}= 4$, $s_{22} = 0$, $I(s_{12}, s_{22}) = 0$

Pour age = "> 40'': $s_{13}= 3$, $s_{23} = 2$, $I(s_{13}, s_{23}) = 0.971$

En utilisant l'équation (2), l'information prévue nécessaire pour le classement d'un cas dans le cas de la partition selon l'age est :

$$E(age) = \frac{5}{14} I(s_{11}, s_{21}) + \frac{4}{14} I(s_{12}, s_{22}) + \frac{5}{14} I(s_{13}, s_{23}) = 0.694$$

Donc, le gain d'information apporté par l'attribut age est :

$$Gain(age) = I(s_1, s_2) - E(age) = 0.246$$

De la même façon, on obtient :

Gain (revenu) = 0.029;
Gain (étudiant) = 0.151;
Gain (estimation_crédit) = 0.048;

L'attribut « *age* » apporte donc le plus haut gain d'information par rapport à tous les autres attributs. Il sera choisi comme attribut de test. Un nœud est crée et est marqué par « age », et les branches sont crées pour chaque valeur de cet attribut. L'ensemble de cas est partitionné en conséquence. Les cas appartenant à la tranche d'age « 31..40 » appartiennent à la même classe. Une feuille doit donc être crée à la fin de cette branche et marquée avec la valeur « oui ». L'arbre de décision final est illustré dans la figure 2.2.

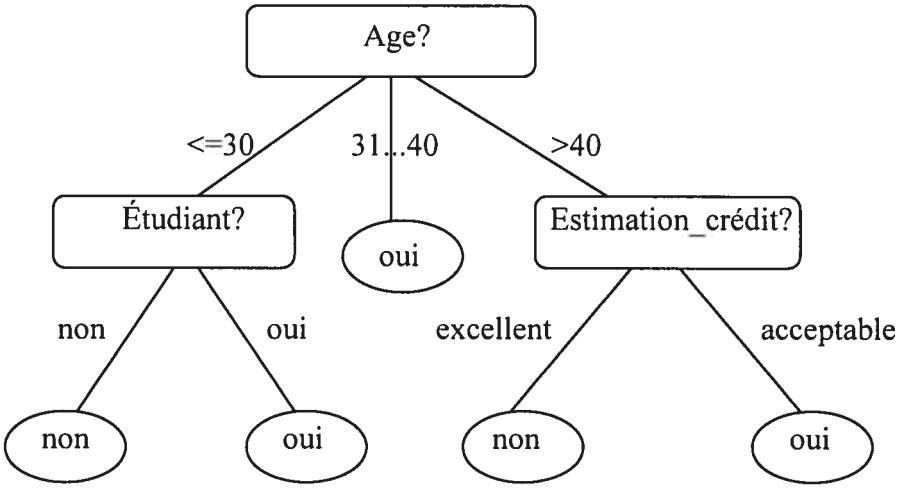


Figure 2.2 : Arbre de décision final

- **Extraction des règles de classification**

Les arbres de décision peuvent facilement être convertis en des règles de classification. Pour ce faire, la connaissance représentée par les arbres de décision peut être extraite et représentée sous forme de règles de classification « *SI-ALORS* ». Chaque règle est crée à partir d'un chemin qui part de la racine à une feuille. Chaque paire attribut-valeur au long d'un chemin donné forme une conjonction dans l'antécédent de la règle (la partie *SI*). La feuille représente la prédiction de la classe, elle forme la conséquence de la règle (la partie *ALORS*).

Les règles extraites de l'arbre de décision dans l'exemple précédent sont les suivantes :

SI age ≤ 30 ET étudiant = non ALORS acheter_ordinateur = non;

SI age ≤ 30 ET étudiant = oui ALORS acheter_ordinateur = oui;

SI age = 31…40 ALORS acheter_ordinateur = oui;

SI age >40 ET estimation_crédit = excellent ALORS acheter_ordinateur =non;

SI age> 40 ET estimation_crédit =juste ALORS acheter_ordinateur =oui.

C4.5 utilise les cas d'entraînement pour estimer l'exactitude de chaque règle. Puisque ceci constituerait une évaluation biaisée de l'exactitude de chaque règle, C4.5 utilise un ensemble de cas de test indépendant de l'ensemble d'entraînement pour estimer cette exactitude. Une règle peut aussi être élaguée en enlevant n'importe quelle condition dans son antécédent qui n'améliore pas l'exactitude estimée de la règle. Ainsi, il est possible qu'un cas d'essai donné ne satisfasse aucun antécédent de règle, une règle de défaut assignant la classe de majorité étant par conséquent ajoutée à l'ensemble résultant de règles.

2.5.2 Association

À l'origine, le problème d'extraction de règles d'association a été développé pour étudier les bases de donnés de transactions de ventes dans le cas de l'analyse du panier de la ménagère. Ce concept a été introduit pour la première fois par [Agrawal *et al.*, 1993]. Dans un supermarché comportant une grande collection d'articles, les décisions économiques typiques que la gestion du supermarché doit prendre, incluent quoi mettre en vente, comment concevoir les bons (coupons), comment placer les marchandises sur les étagères afin de maximiser le bénéfice, etc. L'analyse des données de transactions est une approche généralement utilisée afin d'améliorer la qualité de telles décisions. À partir d'une base de données de transactions, il est possible d'extraire des règles d'association permettant de déterminer les produits qui sont le plus fréquemment achetés ensemble. Un exemple de règle d'association est : « *90% des clients qui achètent du pain et du beurre achètent aussi du lait* ». L'antécédent de la règle est composé du pain et du beurre, tandis que la conséquence est composée du lait. Aujourd'hui, cette technique est appliquée à tout domaine cherchant à explorer les rapports évidents entre les objets d'un ensemble pour transformer les données en association qui regroupe ces objets entre eux.

La section qui suit introduit d'abord quelques définitions de base. Nous expliquons ensuite le fonctionnement de l'algorithme *Apriori*. Cet algorithme a été présenté pour la première fois en 1994 par [Agrawal *et al.*, 1994] et est fréquemment utilisé dans la génération de règles d'association. Cette section repose sur les références [Agrawal *et a*l., 1994], [Han *et al.*, 2001] et [URL8].

2.5.2.1 Définitions

Les règles d'association sont de type X => Y. L'antécédent X se compose d'une condition ou de conditions multiples qui doivent toutes être vraies pour que le conséquent Y soit vrai. On a alors : « $A_1 \wedge A_2 \wedge \ldots \wedge A_m \Rightarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n$ », où Ai (pour $i \in \{1, \ldots, m\}$) et B_j (pour $j \in \{1, \ldots, n\}$) sont des paires de valeurs des attributs. La règle d'association X=>Y peut être interprétée ainsi : « les tuples dans la base de données qui satisfont les conditions dans X satisfont aussi les conditions de Y ».

Le support et la confiance

Le support et la confiance sont deux mesures de l'importance d'une règle d'association qui représentent respectivement l'utilité et la certitude des règles découvertes. Un support de 2% d'une règle par exemple, signifie que 2% des transactions montre que les deux produits X et Y se vendent ensemble. Une confiance de 60% indique que 6 clients sur 10 qui achètent le produit X achètent aussi le produit Y.

Soit $I = \{i_1, i_2, ..., i_m\}$ un ensemble d'items. Soit D l'ensemble de transactions de la base de données, chaque transaction T étant un ensemble d'items dont $T \subseteq I$. Chaque transaction est associée à un identificateur appelé TID. Soit A un ensemble d'items. Une transaction T contient A si et seulement si $A \subseteq T$.

Une règle d'association est une implication de type : A=>B, où $A \subset I$, $B \subset I$ et $A \cap B = \Phi$. La règle A=>B apparaît dans l'ensemble de transaction D avec un support s, où s est le pourcentage des transactions de D qui contient $A \cup B$ (l'union de A et b). Il est considéré comme la probabilité $P(A \cup B)$.

La règle A=>B a une confiance c dans l'ensemble de transactions D, si c est le pourcentage des transactions de D contenant A, qui contiennent aussi B. La confiance c peut être définie par la probabilité conditionnelle $P(B \mid A)$.

$$Support(A \Rightarrow B) = P(A \cup B)$$

$$Confiance(A \Rightarrow B) = P(B \mid A)$$

Le support et la confiance varient entre 0% et 100%. Les règles qui vérifient le seuil minimum du support (*min_sup*) et le seuil minimum de la confiance (*min_conf*), sont considérées comme étant fortes (ou représentatives). Les deux seuils sont généralement fixés par l'utilisateur ou l'expert du domaine.

Un ensemble d'items est appelé un « *itemset* ». Un *itemset* qui contient k items est un k-*itemset*. Un *itemset* qui satisfait le support minimum est un *itemset* fréquent. L'ensemble des k-*itemsets* fréquents est noté par L_k.

2.5.2.2 L'algorithme *Apriori*

La génération de règles d'association est un processus composé de deux étapes:

- ✓ *Trouver tous les itemsets fréquents* : chacun de ces *itemsets* doit apparaître un nombre de fois au moins égal au support minimum prédéfini.
- ✓ *Générer les règles d'association fortes à partir des itemsets fréquents* : ces règles doivent satisfaire le support et la confiance minimums.

- **Recherche des *itemsets* fréquents**

Apriori-gen est la fonction utilisée par *Apriori* pour la génération des candidats. Elle prend comme argument L_{k-1}, l'ensemble de tous les (k-1)-*itemsets* fréquents, et retourne l'ensemble de tous les k-*itemsets*.

Apriori commence donc par chercher l'ensemble des 1-*itemsets* fréquents. Cet ensemble est nommé L_1. L_1 est utilisé pour la recherche de L_2, l'ensemble de 2-*itemsets* fréquents. L_2 va aussi être utilisé dans la recherche de L_3, et ainsi de suite, jusqu'à ce qu'aucun autres k-*itemsets* fréquents ne puissent être trouvés. La recherche de chaque L_k requiert un balayage (un scan) complet de la base de données.

Ce processus se fait en deux étapes qui sont *la jointure* et *l'élagage*. La jointure $L_{k-1} \times L_{k-1}$ permet de former un ensemble C_k, tandis que l'élagage permet d'obtenir l'ensemble L_k à partir de C_k selon la propriété suivante:

Propriété de l'algorithme Apriori :

Tous les sous-ensembles non vides d'un *itemset* fréquent doivent aussi être fréquents.

Pour réaliser cela, on élimine tous les sous-ensembles de C_k qui ne sont pas membres de L_{k-1} (sous-ensembles non fréquents) afin de générer L_k. Cette fonction permet de générer un nombre minimal de candidats. Pour bien comprendre le mécanisme de recherche des *itemsets* fréquents, considérons un exemple simple. On a une base de données de transactions D, illustrée dans le tableau 2.3. D contient 9 transactions, donc $|D| = 9$. I_i représente l'item i.

Tableau 2.3 : Données de transactions

TID	*Liste des items*
T100	I_1 ,I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

Les étapes (1 à 8) suivantes ainsi que la figure 2.3, illustrent l'algorithme *Apriori* dans sa recherche des *itemsets* fréquents de D.

1. Lors de la première itération de l'algorithme, chaque item est un membre de l'ensemble des candidats 1-*itemsets*, C_1. L'algorithme scanne ensuite toutes les transactions pour compter le nombre des occurrences de chaque item.

2. Supposons que le support minimum des transactions est égal à 2 (min_sup = 2/9= 22%). L'ensemble des 1-*itemsets* fréquents, L_1, est constitué des candidats 1-*itemsets* qui satisfont le support minimum requis.
3. Pour obtenir l'ensemble des 2-*itemsets* fréquents, L_2, l'algorithme utilise la jointure $L_1 \times L_1$, pour générer un ensemble candidat de 2-*itemsets*, C_2. Celui-ci est constitué de $\begin{pmatrix} | L_1 | \\ 2 \end{pmatrix}$ 2-*itemsets*.
4. Les transactions dans D sont scannées et le compte du support de chaque *itemset* candidat dans C_2 est comptabilisé.
5. L'ensemble des 2-*itemsets* fréquents, L_2, est alors déterminé, constituant des candidats 2-*itemsets* dans C_2 ayant le support minimum.
6. La génération de l'ensemble des candidats 3-*itemsets*, C_3, se fait comme suit :
   - *Jointure*:

     En premier, $C_3 = L_2$ X $L_2 = \{\{I_1, I_2, I_3\}, \{I_1, I_2, I_5\}, \{I_1, I_3, I_5\}, \{I_2, I_3, I_4\}, \{I_2, I_3, I_5\}, \{I_2, I_4, I_5\}\}$.
   - *Élagage* :

     En se basant sur la propriété de *Apriori* définie précédemment, les 2-items sous-ensembles de $\{I_1, I_2, I_3\}$ qui sont $\{I_1, I_2\}$, $\{I_1, I_3\}$, $\{I_2, I_3\}$ sont membres de L_2, donc on garde $\{I_1, I_2, I_3\}$ dans C_3.

     On fait la même vérification avec tous les sous-ensembles des autres candidats pour garder les candidats dont les sous-ensembles sont fréquents.

     Les quatre derniers candidats ne sont pas fréquents. Par conséquent, on les enlève de C_3, et on économise ainsi l'effort de calculer leurs comptes pendant le scan suivant de D qui sert à déterminer L_3. Il faut noter qu'étant donné un candidat k-*itemset*, on a seulement besoin de vérifier si ses (k-1)-sous-ensembles sont fréquents selon la stratégie de recherche d'*Apriori*.

     $C_3 = \{\{I_1, I_2, I_3\}, \{I_1, I_2, I_5\}\}$ après l'élagage.
7. Les transactions dans D sont scannées pour déterminer L_3. L_3 est constitué des candidats 3-*itemsets* dans C_3 ayant le support minimum.
8. L'algorithme utilise $L_3 \times L_3$ pour générer un ensemble candidat de 4-*itemsets*, C_4. Bien que le résultat de la jointure soit $\{\{I_1, I_2, I_3, I_5\}\}$, cet *itemset* est élagué puisque

son sous-ensemble $\{\{I_2, I_3, I_5\}\}$ n'est pas fréquent. Ainsi, $C_4 = \Phi$ et l'algorithme termine.

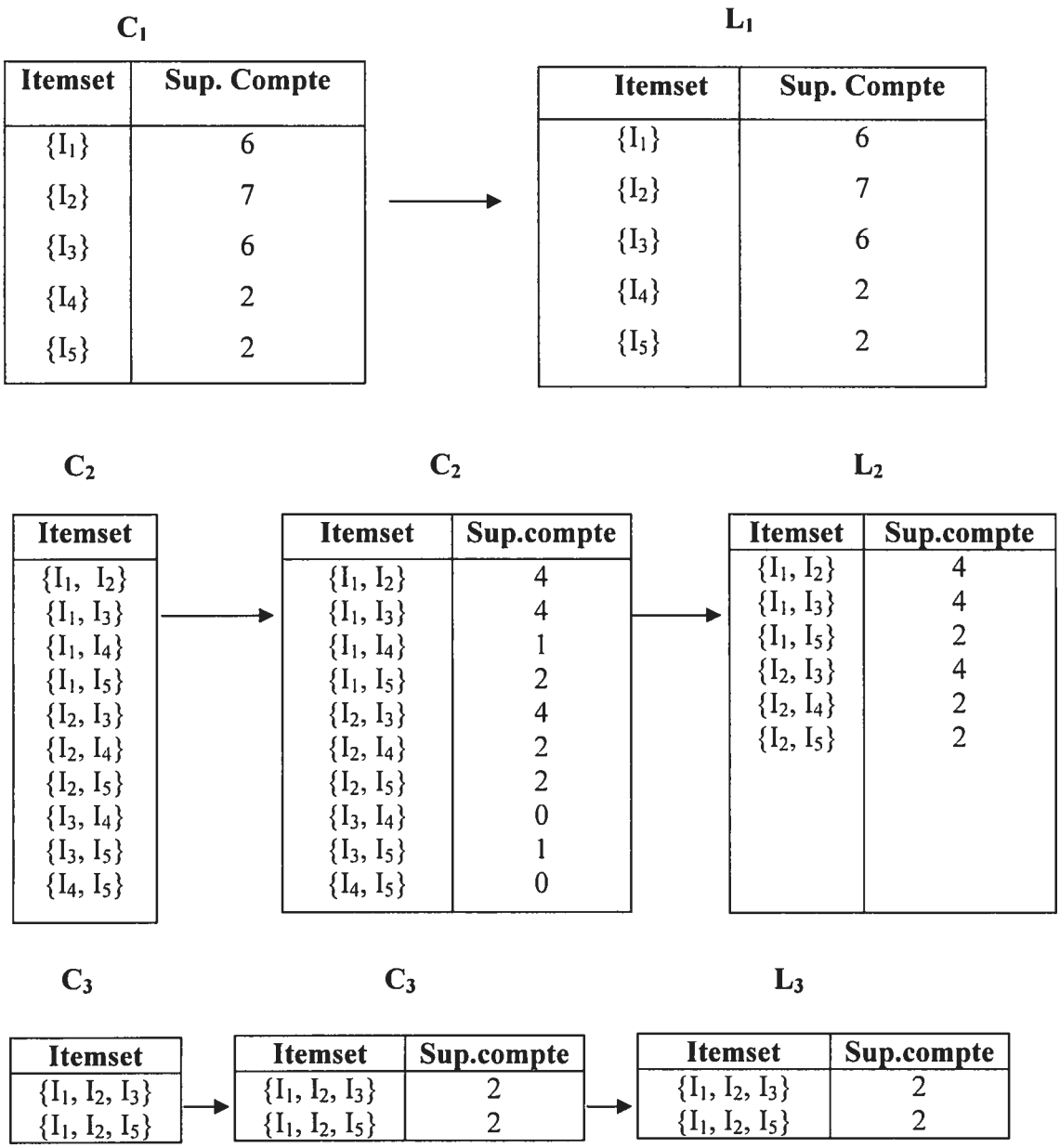


Figure 2.3 : *Apriori* lors de sa recherche des *itemsets* fréquents de D

- **Génération des règles d'association à partir des *itemsets* fréquents**

La génération des règles d'association se fait à partir de l'équation suivante :

$$Confiance(A \Rightarrow B) = P(B \mid A) = \frac{Support\_compt(A \cup B)}{Support\_comp(A)}$$

Pour chaque *itemset* fréquent *l*, générer tous les sous-ensembles non vides de *l*.
Pour chaque sous-ensemble non vide *s* de l'*itemset l*, générer la règle « *s* => (*l* - *s*) » si :

$$\frac{Support\_compt(l)}{Support\_compt(s)} \geq \min\_conf$$

La variable min_*conf* est le seuil minimum de la confiance.
Puisque les règles sont générées à partir des *itemsets* fréquents, chacune satisfera automatiquement le support minimum.
Suivant l'exemple suscité, essayons de générer les règles à partir de l'*itemset* fréquent *l* = $\{I_1, I_2, I_5\}$. Les sous-ensembles non vides de *l* sont $\{I_1, I_2\}$, $\{I_1, I_5\}$, $\{I_2, I_5\}$, $\{I_1\}$, $\{I_2\}$ et $\{I_5\}$. Les règles d'association générées sont ainsi les suivantes:

$I_1 \wedge I_2 \Rightarrow I_5$, confiance = 2/4 = 50%

$I_1 \wedge I_5 \Rightarrow I_2$, confiance = 2/2 = 100%

$I_2 \wedge I_5 \Rightarrow I_1$, confiance = 2/2 = 100%

$I_1 \Rightarrow I_2 \wedge I_5$, confiance = 2/6 = 33%

$I_2 \Rightarrow I_1 \wedge I_5$, confiance = 2/7 = 29%

$I_5 \Rightarrow I_1 \wedge I_2$, confiance = 2/2 = 100%

Supposons que la confiance minimale soit égale à 70%, la deuxième, la troisième et la dernière règles sont acceptées parce qu'elles sont considérées fortes.

2.5.3 Segmentation

La segmentation est le groupement des enregistrements, des observations ou des cas dans des segments ou dans des sous-groupes d'objets qui partagent des caractéristiques communes [URL3]. Le but est aussi de séparer les objets dissemblables. Il existe des techniques de « *détection automatiques des segments* (*clustering*) » qui sont capables de découvrir, dans les données, des structures cachées et non définies précédemment,

permettant la segmentation [Shearer, 2000]. Ces techniques consistent en une division des données dans des groupes d'objets. Chaque groupe, appelé segment (cluster), se compose des objets qui sont semblables entre eux-mêmes et différents des objets des autres groupes [Berkhin, 2002]. En d'autres termes, étant donné un ensemble d'enregistrements, les techniques de détection automatique des segments consistent à analyser puis segmenter l'ensemble à des sous groupes descriptifs dont la similarité des enregistrements dans un groupe est maximisée et la similarité avec des enregistrements en dehors du groupe est minimisée [Larose, 2005]. Un exemple simple de la segmentation serait le groupement que les personnes exécutent quand elles font la blanchisserie, groupant la pression permanente, le nettoyage à sec, les blancs, et les vêtements brillamment colorés. Cela est important vu les caractéristiques semblables des attributs au sujet de la manière dont ils se comportent dans le lavage [Berson *et al.*, 2000]. Cependant, il ne faut pas confondre la classification avec la détection automatique des segments. À la différence de la classification qui consiste en une manière de segmenter les données en les assignant aux classes prédéfinies, la détection automatique des segments se rapporte au problème de segmenter les données dans des groupes qui ne sont pas connus précédemment [Edelstein, 2000], [Candillier, 2004]. En conséquence, la détection des segments produit une description de niveau élevé de l'ensemble d'éléments basée seulement sur des mesures de distance entre ses éléments et aucun modèle de classification n'est explicitement construit [Mendonca *et al.*, 1999]. En d'autres termes, la détection automatique des segments n'a pas de variable cible et n'essaye pas alors de classer, estimer ou prédire la valeur d'une variable cible.

2.5.4 Description

Parfois les chercheurs et les analystes sont simplement intéressés à trouver une description utile des données. La description consiste à décrire de manière explicite des relations ou des modèles implicites contenus dans les données. Son objectif est de fournir une représentation des propriétés générales des données permettant ainsi aux utilisateurs d'avoir une vue d'ensemble [Han *et al.*, 2001]. Ainsi, on peut utiliser plusieurs techniques de forage de données pour créer des modèles de données. Toutefois, ces modèles doivent être clairs, transparents et favorables à l'interprétation et à l'explication intuitive

[Kantardzic, 2002]. De ce fait, quelques modèles sont plus favorables que d'autres à l'interprétation transparente. Les arbres de décision, par exemple, sont des modèles qui fournissent une explication intuitive des résultats pouvant être facilement compréhensibles par l'humain. D'autre part, les réseaux de neurones représentent un modèle opaque vu leur complexité et la nature de leur structure. Les techniques de forage de données qui donnent une bonne description de données sont les arbres de décision et les règles d'association [Larose, 2005]. La description de données représente un sous objectif dans presque tous les projets de forage de données. Aux premières étapes de ce processus, une description de données peut être utile. Son objectif est d'analyser et explorer les données dans l'optique de comprendre leur nature et de trouver des hypothèses potentielles sur l'information cachée [Shearer, 2000].

2.5.5 Estimation

L'estimation consiste à examiner les attributs d'un ensemble d'entités (produits, processus, ressources, etc.) et, en se basant sur les valeurs de ces attributs, assigner des valeurs à un attribut dont la valeur est inconnue et qu'on veut mesurer. Un exemple typique d'une tâche d'estimation est d'employer les attributs qui caractérisent un projet pour estimer (prédire) ses coûts [Mendonca *et al.*, 1999]. L'estimation est similaire à la classification sauf que la variable cible est numérique et continue au lieu d'être catégorielle. Ainsi, les modèles qui sont établis par la classification prévoient des catégories discrètes tandis que l'estimation fonctionne avec les résultats qui ont des valeurs continues comme par exemple les nombres réels entre 1 et 1 000 ou un pourcentage entre 0 et 100% [Groth, 1999]. Dans le domaine de l'analyse statistique plusieurs méthodes d'estimation existent comme par exemple l'estimation d'un point, la régression linéaire simple et la régression multiple. Les réseaux de neurone peuvent aussi être utilisés pour faire de l'estimation [Larose, 2005].

2.5.6 Prédiction

Une fois que les modèles sont établis, ils peuvent être employés pour prévoir le résultat des événements futurs. Malgré que les données historiques ne peuvent pas prévoir le

futur, les motifs présents dans les données tendent à se répéter de telle sorte que si un modèle représentatif d'un ensemble de données est établi, des prédictions pourront être faites à partir de ce modèle [Groth, 1999]. Toutes les méthodes et les techniques utilisées pour la classification (les arbres de décision, les règles de classification, les réseaux de neurone, etc.) et l'estimation (l'estimation d'un point, la régression linéaire simple, etc.) sont aussi utilisées pour la prédiction [Berry *et al.*, 2004]. Bien que la prédiction peut se rapporter à la valeur d'une donnée ou d'une classe, elle constitue souvent la prédiction des valeurs de données [Han *et al.*, 2001]. Ainsi, dans les statistiques, la prédiction est synonyme de régression. Par exemple, on veut développer un modèle pour prédire le salaire des diplômés d'université avec 10 ans d'expérience professionnelle.

2.6 Récapitulatif des tâches de forage de données

Les techniques utilisées dans la création de modèles de forage de données ne sont pas restreintes à une seule tâche. Elles peuvent être employées de différentes manières, chacune satisfaisant un objectif spécifique. Les réseaux de neurones par exemple peuvent être utilisés pour résoudre des problèmes de classification (où la variable cible est catégorielle) ou dans la régression (où la variable cible est continue). On peut également noter que plusieurs tâches avec des buts différents peuvent être appliquées successivement pour atteindre un résultat désiré. Par exemple, pour déterminer quels sont les clients susceptibles d'acheter un nouveau produit, un analyste d'affaires pourrait employer d'abord la détection automatique des segments pour segmenter la base de données des clients, puis appliquer la régression pour prévoir le comportement d'achat de chaque groupe [Goebel *et al.*, 1999]. Le choix d'une combinaison de tâches à appliquer dans une situation quelconque dépend de la nature de la tâche de forage de données, la nature des données disponibles et des qualifications et préférences des explorateurs de données [Berry *et al.*, 2004]. Le tableau 2.4 donne un récapitulatif des tâches de forage de données en présentant leurs caractéristiques générales et en montrant quelques exemples de techniques utilisées pour effectuer chacune d'elles.

Tableau 2.4 : Récapitulatif des tâches de forage de données

Tâche	Type		Caractéristiques générales	Objectif		Techniques et algorithmes utilisés
	Supervisé	Non supervisé		Description	Prédiction	
Classification	✓		Les modèles de classification peuvent être employés pour comprendre les données existantes et pour prévoir comment les nouveaux exemples se comporteront [URL3]. L'objectif ultime n'est pas d'explorer les données pour découvrir des classes intéressantes, mais de décider plutôt comment des nouveaux cas devraient être classés [Chen *et al.*, 2004].		✓	Arbres de décision (C4.5, CART, CHAID), réseaux de neurones, k-plus proches voisins (*k*-nearest neighbor)
Association	✓	✓	Bien que l'association puisse être employée pour la prédiction, elle est la plupart du temps employée pour la compréhension (la description) de données [Shearer, 2000].	✓		Règles d'association : Apriori, GRI, FP-growth.
Segmentation		✓	Après avoir trouvé les segments (qui segmentent raisonnablement la base de données), on peut les employer pour classer des nouveaux cas [Edelstein, 2000]. La connaissance du comportement typique d'un segment particulier peut être une information importante si on veut prévoir le comportement d'un membre inconnu de ce segment [Zaima *et al.*, 2003].	✓		Techniques de détection automatique des segments : (Hierarchical and *k*-means clustring, Kohonen networks).

Tableau 2.4 : Récapitulatif des tâches de forage de données (suite)

Tâche	Type		Caractéristiques	Objectif		Techniques et algorithmes utilisés
	Supervisé	*Non supervisé*		*Description*	*Prédiction*	
Description	✓	✓	Une bonne description d'un comportement suggère souvent une explication pour ce comportement [Berry *et al.*, 2004].	✓		Arbres de décision, règles d'association, détection automatique de segments.
Estimation	✓		Dans la pratique, l'estimation est souvent employée pour accomplir une tâche de classification où la variable cible est numérique [Larose, 2005].		✓	Techniques statistiques (Estimation de point, régression linéaire, régression multiple, ...), réseaux de neurones, k-plus proches voisins (*k*-nearest neighbor), CART.
Prédiction	✓		Les modèles établis par une tâche de classification ou d'estimation peuvent être utilisées pour la prédiction [Groth, 1999].		✓	Toutes les techniques de classification et d'estimation.

2.7 Conclusion

La clé de la réussite de la résolution de problèmes par le forage de données est de s'assurer d'abord de la bonne qualité des données disponibles. Cependant, le processus de collecter, nettoyer et transformer les données provenant de plusieurs sources peut être laborieux. Malgré cela, le forage de données dépasse les outils de requête et de traitement analytique en ligne. Il remédie à la faiblesse que ces outils ne représentent que l'approche descriptive de données. En fait, la description est une tâche différente de la prédiction, car même si un modèle descriptif est établi, ce n'est pas sûr que ce modèle puisse être appliqué au delà de la base de données utilisée pour son extraction. Toutefois, le forage de données vérifie que les modèles extraits puissent être employés pour la prédiction. Pour ce faire, il utilise différentes méthodes de test comme la partition de la base de données en deux. La première partie sert à développer le modèle tandis que la deuxième sert à le tester. Et même dans sa recherche des modèles dans la résolution d'un même problème, le forage de données a l'avantage d'avoir une multitude d'algorithmes, ce qui permet de comparer les différents résultats et d'opter pour le meilleur.

L'objectif de notre travail consiste à développer et implémenter un système de recommandation basé sur le forage de données. Dans le chapitre suivant, nous décrivons les systèmes de recommandation et nous expliquons entre autres l'utilisation du forage de données par ces systèmes.

Chapitre 3 : Systèmes de recommandation

Le Web est devenu un dépôt universel de connaissances permettant le partage de l'information d'une manière très efficace. Vu la demande croissante en matière d'accès à l'information, plusieurs outils ont été développés dans l'optique d'aider l'utilisateur à retrouver, sélectionner et exploiter l'information. Parmi ces outils, on peut citer les moteurs de recherche et les systèmes de filtrage d'information. Cependant, nous devons souligner la nécessité de garantir la pertinence de l'information offerte à l'utilisateur par le biais de ces outils afin de répondre à ses besoins et de satisfaire ses espérances. Ainsi, l'intégration des systèmes de recommandation dans la vie de tous les jours est devenue inévitable. L'objectif est d'apprendre d'abord les préférences des utilisateurs pour les aider, ensuite, à faire les meilleurs choix.

Dans ce chapitre, nous présentons dans un premier temps les concepts de recherche et de filtrage d'information, deux techniques d'accès à l'information. Nous définissons par la suite la notion du profil d'utilisateur et nous expliquons la recommandation ainsi que les différentes techniques qu'elle utilise. Ces techniques connaissent quelques faiblesses que nous présentons aussi dans ce chapitre. Vu que ces faiblesses ne sont pas les seuls facteurs d'influence sur la qualité des systèmes de recommandation, nous étudions celle-ci selon différents points de vue. Nous abordons ensuite la recommandation des cours, un domaine d'application peu étudié, et nous présentons une revue de quelques systèmes qui s'attaquent à ce domaine.

3.1 Recherche d'information

Le rôle d'un système de recherche d'information est de satisfaire le besoin d'information de l'utilisateur sur un sujet ou une matière donnée. Pour être efficace dans sa tentative, le système de recherche d'information se base sur l'adéquation entre le contenu effectif des documents et l'information recherchée par l'utilisateur. Il doit, d'une façon ou d'autre, interpréter le contenu des termes présents dans les documents et les ranger selon un degré de pertinence vis à vis de la requête [Baeza-Yates *et al.*, 1999]. Les termes des documents

sont également modélisés dans des matrices qui permettent de mesurer la similarité entre les vecteurs dans un espace multidimensionnel. Par conséquent, le résultat de la recherche satisfait les buts de l'utilisateur à cours terme [Perugini *et al.*, 2004]. Dans sa sélection des documents, un processus de recherche d'information passe par quatre phases principales : l'indexation, la formulation de la requête, la comparaison et le feedback [Lewis, 1991].

- **L'indexation**

Dans cette phase, les documents doivent être convertis, par des méthodes de représentation de texte, en une forme parfois appelée le représentant de document. Ce dernier est utilisé par le système de recherche d'information. Les principales formes de recherche d'information pour des documents textuels sont le modèle booléen, le modèle vectoriel et le modèle probabiliste [Baeza-Yates *et al.*, 1999], [Tollari, 2003].

- **La formulation de la requête**

La description complète du besoin d'information d'un utilisateur ne peut pas être utilisée directement pour chercher l'information. L'utilisateur doit donc, via les interfaces courantes des moteurs de recherche, traduire son besoin d'information en une requête qui peut être traitée par le système [Baeza-Yates *et al.*, 1999]. Celle-ci peut être en langage naturel, sous forme de mots clés ou de document exemple. Le système sélectionne les mots importants contenus dans la requête pour les utiliser dans sa recherche.

- **La comparaison**

Le système, dans cette étape, compare la requête de l'utilisateur aux documents stockés et décide quels sont les documents recherchés et en quel ordre de pertinence il faut les trier.

- **Le feedback**

Étant donné que le contenu d'un document et les besoins d'un utilisateur ne sont pas toujours parfaitement représentés, une recherche initiale donne rarement les résultats

attendus. Par conséquent, plusieurs modifications de la requête initiale pourraient être nécessaires pour arriver aux résultats souhaités par l'utilisateur. Pour ce faire, ce dernier peut modifier explicitement sa requête, ou communiquer au système une évaluation de chaque document recherché. Par la suite, le système met à jour la requête. Cette dernière opération est appelée « *retour de pertinence* » [Gallardo-lopez, 2002], [Bottraud *et al.*, 2003], [Tchienehom, 2004], et permet au moteur de recherche de raffiner la présentation de la requête de l'utilisateur [Burke, 2002].

Dans la section suivante, nous commençons par définir la technique du filtrage d'information. Ensuite, nous décrivons le profil d'utilisateur et nous présentons une comparaison entre le filtrage et la recherche d'information.

3.2 Filtrage d'information

L'augmentation rapide du volume d'information disponible sur le Web a vu l'émergence de la problématique de la surcharge de données. Cette dernière a engendré une nécessité grandissante de trouver des mécanismes pour éliminer les données non pertinentes et de cibler ainsi les informations satisfaisant au mieux le besoin de l'utilisateur [Perugini *et al.*, 2004]. Le filtrage d'information est donc, un terme utilisé pour désigner tout processus impliquant la livraison de l'information requise aux utilisateurs qui en ont besoin. Alors que la recherche d'information est une tâche interactive, qui concerne un seul usage du système par un utilisateur disposant à la fois d'une requête et d'un but immédiat, le filtrage d'information constitue une tâche passive. Il permet des utilisations répétitives du système, par un utilisateur ou plusieurs utilisateurs, et avec des intérêts à long terme [Belkin *et al.*, 1992]. Cela signifie qu'en utilisant le filtrage d'information, l'utilisateur ne formule pas explicitement une requête, comme en recherche d'information. Le filtrage se base plutôt sur une représentation de l'utilisateur qui tient compte de ses caractéristiques et qu'on appelle « *profil d'utilisateur* ». Pour mieux personnaliser les résultats renvoyés, ce profil sert à mesurer la similarité entre les informations disponibles et l'utilisateur [Bottraud *et al.*, 2003].

3.2.1 Profil d'utilisateur

Le profil d'utilisateur est un ensemble d'informations qui décrivent et caractérisent un utilisateur [Adomavicius *et al.*, 1999a], [Adomavicius *et al.*, 2001]. Il permet à un processus de recherche ou de filtrage d'information de développer une représentation des préférences et des intérêts de cet utilisateur afin d'adapter l'information à ses besoins et à ses attentes [Belkin *et al.*, 1992]. Le profil d'utilisateur peut être construit selon l'une des deux méthodes suivantes : en sollicitant explicitement le feedback de l'utilisateur à travers l'évaluation d'un ensemble de produits ou de services, ou en le surveillant pour déduire ses intérêts à travers des déclarations implicites [Perugini *et al.*, 2004]. Vu qu'il est difficile aux utilisateurs de spécifier leurs intérêts, la construction des profils est également difficile. Les intérêts varient d'un utilisateur à un autre et changent constamment. Ils peuvent être reliés aux buts et aux types de l'information et même aux caractéristiques de celle-ci comme la qualité et la complexité [Stadnyk *et al.*, 1992]. Cependant, nous soulignons la nécessité de réunir le maximum de critères descriptifs des informations manipulées par les processus de recherche et de filtrage. Mieux les informations sont décrites, plus il est aisé de satisfaire les utilisateurs en s'adaptant aux caractéristiques particulières de chacun d'eux [Tchienehom, 2004].

3.2.2 Recherche d'information versus filtrage d'information

La recherche et le filtrage d'information partagent le même objectif qui est d'aider l'utilisateur à obtenir des informations pertinentes. Le tableau 3.1 représente une comparaison entre les deux techniques [Belkin *et al.*, 1992], [Berrut *et al.*, 2003].

Tableau 3.1 : La recherche d'information versus le filtrage d'information

La recherche d'information	*Le filtrage d'information*
Rechercher les données sur un flux entrant	Éliminer les données sur un flux entrant
L'utilisateur obtient les données extraites	L'utilisateur obtient les données qui demeurent après le filtrage
Des besoins à cours terme	Des intérêts à long terme
Résultat non personnalisé	Résultat personnalisé (selon le profil de l'utilisateur)
Utilise souvent des mots clés	Différents dispositifs utilisés
Non adaptatif	S'adapte au changement du profil de l'utilisateur
Non dynamique	Filtre dynamiquement l'information entrante

Le filtrage a ensuite donné naissance à la *recommandation*; une notion que nous allons à présent définir. Nous évoquons ensuite les différentes techniques qu'elle utilise.

3.3 La recommandation

Quotidiennement, nous sommes parfois obligés de faire des choix parmi plusieurs possibilités sans avoir assez d'expérience personnelle. Ainsi, nous demandons l'aide d'autres personnes, nous comptons sur les lettres de recommandation, les journaux, etc. Les systèmes de recommandation [Resnick *et al.*, 1997], [Adomavicius *et al.*, 2005] se fixent comme objectif la simulation et l'automatisation de ce processus social [Perugini *et al.*, 2004]. Or, ils sont devenus un domaine de recherche dès l'apparition du premier papier sur le filtrage collaboratif en 1992 par *Goldberg* et *al* [Goldberg *et al.*, 1992]. Ce papier parle de *Tapestry*, un système de filtrage collaboratif développé au centre de recherche *Xerox*, dont l'intention était la gestion du courrier électronique. Cependant, le terme *système de recommandation* fut introduit pour la première fois par *Resnick* et *Varian* en 1997 [Resnick *et al.*, 1997]. Depuis, les systèmes de recommandation se sont étendus à plusieurs domaines pouvant intéresser les utilisateurs. Leur principe est de

collecter, d'abord, des informations sur les préférences des utilisateurs, puis de cibler au mieux ce qui peut être susceptible de les intéresser. Ainsi, le rôle de ces systèmes est de permettre le filtrage d'un ensemble important d'items comme les livres, les pages Web, les films, les CDs, etc. L'objectif étant de fournir aux utilisateurs un conseil de qualité sur les décisions à prendre, en tenant compte du profil de chacun d'eux [Karypis, 2001], [Schwaighofer *et al.*, 2003]. L'évolution des systèmes d'information à partir de la recherche d'information jusqu'aux systèmes de recommandation est illustrée dans le tableau 3.2 [Perugini *et al.*, 2004].

Tableau 3.2 : L'évolution des systèmes de recommandation à partir de la recherche d'information [Perugini *et al.*, 2004]

Le concept	***Les modèles de matrice***	***L'année***
Recherche d'information	Termes × Documents	Années 70 - 80
Filtrage d'information	Caractéristiques × Documents	Années 80 - 90
Filtrage basé sur le contenu	Caractéristiques × Artefacts	Années 90
Filtrage collaboratif	Utilisateurs × Documents	Années 90
Système de recommandation	Utilisateurs × Artefacts	À partir de 97

3.4 Les techniques de recommandation

Les systèmes de recommandation utilisent différentes techniques de filtrage afin de prédire l'utilité des items pour les utilisateurs. Le processus de recommandation le plus communément utilisé par ces systèmes repose sur les évaluations des items en utilisant les trois principales techniques de filtrage : *le filtrage basé sur le contenu*, *le filtrage collaboratif* et *le filtrage hybride* [Adomavicius *et al.*, 2005]. Une autre approche utilisée par ces systèmes est *la recommandation basée sur le forage de données* [Adomavicius *et al.*, 2001], [Perugini *et al.*, 2004], [Schafer, 2005]. Au lieu d'employer les évaluations des

utilisateurs, cette approche apprend, à partir d'un historique de données, des règles ou des modèles qui serviront, par la suite, pour la production des recommandations. Toutefois, les différentes approches de recommandation partagent le même objectif qui consiste à assister l'utilisateur dans sa recherche des items adéquats. Les sections suivantes sont dédiées à l'explication de chacune des approches susmentionnées.

3.4.1 Le filtrage basé sur le contenu

On entend dire par la recommandation basée sur le contenu le fait de proposer à un utilisateur des items similaires à ceux qu'il a préférés auparavant [Perugini *et al.*, 2004]. Les systèmes de recommandation basés sur le contenu sont conçus la plupart du temps pour recommander des objets basés sur le texte, tels que les articles, les pages Web, les courriels, etc. Le contenu dans ces systèmes est habituellement décrit par des mots-clés. Cependant, l'importance de ces mots dans un objet, c.à.d. l'information qu'ils apportent, est déterminée avec une mesure qui peut être définie de différentes manières [Adomavicius *et al.*, 2005]. Un exemple simple de cette mesure peut être la fréquence des mots dans un document. Or, la recommandation se fait suite à une comparaison entre le contenu du document et le profil de l'utilisateur. Ce dernier est parallèlement représenté sous forme de termes contenus dans le document que l'utilisateur a jugé intéressant, ainsi que des pondérations de ces termes. Plusieurs algorithmes ont été proposés pour l'analyse du texte des documents et la recherche des régularités qui servent de base dans la recommandation [Pazzani, 1999], tels que les arbres de décision et les réseaux de neurones. Par ailleurs, la recommandation basée sur le contenu peut être vue comme une continuation de la recherche d'information [Belkin *et al.*, 1992]. En d'autres termes, un moteur de recherche d'information capable de raffiner les requêtes de ses utilisateurs, à travers un processus de retour de pertinence, représente une forme simple de recommandation basée sur le contenu [Burke, 2002]. Ainsi, les deux approches ont des techniques en commun.

3.4.1.1 Les limites du filtrage basé sur le contenu

Les systèmes basés sur ce type de filtrage sont limités à l'analyse d'un certain type de contenu. Ils sont conçus, la plupart du temps, pour recommander des éléments basés sur le texte. De ce fait, ils se basent sur la comparaison des termes présents ce qui engendre :

- La difficulté de traiter d'autres critères de pertinence [Adomavicius *et al.*, 1999b] ainsi que d'indexer des documents multimédia.
- La limitation aux caractéristiques associées explicitement aux éléments à recommander. Dans le cas de la recommandation d'un film, par exemple, le titre et les noms des acteurs font parties des caractéristiques qu'on lui associe [Burke, 2002].
- La difficulté de traiter des nouveaux termes. C'est à dire que si de nouveaux termes apparaissent, vu un nouvel axe de recherche par exemple, l'utilisateur n'aura pas la chance d'avoir ce document. Ce dernier sera filtré vu qu'il contient des termes ne figurant pas dans le profil de l'utilisateur malgré qu'il s'agisse du même domaine d'intérêt de cet utilisateur [Berrut *et al.*, 2003].
- La difficulté de distinguer entre un bon document et un mauvais si tous les deux contiennent les mêmes termes [Balabanovic *et al.*, 1997].
- La difficulté de recommander des objets qui sont trop différents de ceux que l'utilisateur a appréciés précédemment ou, au contraire, des objets qui sont trop semblables, comme dans le cas des articles qui traitent le même événement malgré qu'ils le font différemment.
- Un utilisateur doit évaluer un nombre suffisant d'items avant qu'un système basé sur le contenu puisse comprendre les préférences de l'utilisateur et lui présenter des recommandations fiables. Par conséquent, un nouvel utilisateur, ayant fourni très peu d'évaluations, ne pourrait pas obtenir des recommandations précises [Adomavicius *et al.*, 2005].

3.4.2 Le filtrage collaboratif

La recommandation collaborative consiste à trouver des utilisateurs avec des profils similaires au profil de l'utilisateur cible, puis recommander à ce dernier les items qu'ils ont appréciés. Le calcul de la similarité se fait donc entre les utilisateurs au lieu d'être effectué entre les items [Balabanovic *et al.*, 1997]. Pour ce faire, le système maintient dans une base de données les préférences de ses utilisateurs. Il cherche les utilisateurs dont les préférences peuvent être proches au profil de l'utilisateur cible, et recommande à ce dernier les items qu'ils ont jugés pertinents. Cette approche suppose que les préférences d'un utilisateur sont généralement pareilles à celles d'un ou de plusieurs autres utilisateurs, et qu'un nombre suffisant d'évaluations d'items par cet utilisateur est disponible. Les items qui n'ont pas été évalués par un nombre suffisant d'utilisateurs ne peuvent pas être efficacement recommandés [Mooney *et al.*, 2000]. *GroupLens* est un exemple de système basé sur le filtrage collaboratif d'articles ou de messages de « *Usenet news* ». Il consiste en un système de forums permettant des échanges entre les personnes du monde entier. Dans cet espace de discussion, les forums sont classés par centre d'intérêt, dont chacun est appelé « *groupe de discussion* (*newsgroup*) ». Par son processus de filtrage collaboratif, le système *GroupLens* permet à un lecteur la sélection du contenu qui l'intéresse le plus [Herlocker *et al.*, 1997].

3.4.2.1 Les limites du filtrage collaboratif

Les systèmes de recommandation basés sur le filtrage collaboratif connaissent quelques faiblesses parmi lesquelles nous citons :

- Les évaluations des items par les utilisateurs sont les seuls facteurs qui influencent la performance du système. Ainsi, si les évaluations se réduisent la performance diminue également [Balabanovic *et al.*, 1997].
- Ces systèmes dépendent des données qui doivent se chevaucher. Or, ils trouvent des difficultés quant le nombre des utilisateurs ou d'évaluations est petit relativement au nombre d'items, ou dans le cas où peu d'utilisateurs partagent des évaluations en commun [Schwaighofer *et al.*, 2003]. Cela engendre un écart

(*sparcity problem*) entre les évaluations et, par conséquent, une réduction dans le nombre et la précision des recommandations à fournir. De très grands ensembles de données (*scalability problem*) peuvent également mener à une attente inacceptable pour fournir les recommandations [Li *et al.*, 2004].

- Un utilisateur unique dans ses goûts ou ses préférences risque de ne pas recevoir des recommandations [Balabanovic *et al.*, 1997].

3.4.3 Le filtrage basé sur le contenu versus le filtrage collaboratif

Contrairement au filtrage basé sur le contenu, le problème d'indexation dans le filtrage collaboratif ne se pose pas, ainsi :

- ✓ La possibilité de traiter n'importe quel genre de contenu en se servant exclusivement des évaluations des utilisateurs [Schwaighofer *et al.*, 2003], [Burke, 2002]. Or, un item est identifié par les évaluations qu'on lui a attribuées.
- ✓ Une conséquence de cette habilité est que contrairement au filtrage basé sur le contenu, le filtrage collaboratif arrive à recommander des items sans tenir compte de leurs caractéristiques.
- ✓ L'utilisateur peut recevoir n'importe quel item du fait que les utilisateurs similaires l'ont jugé intéressant. Pour un article, par exemple, le jugement des utilisateurs peut découler de plusieurs facteurs de pertinence tels que la clarté de l'article, l'importance du sujet, la force de son argumentation, ou le style de son écriture, et non pas pour la seule raison de la présence des termes recherchés dans cet article [Berrut *et al.*, 2003], [Adomavicius *et al.*, 2005].

Toutefois, les forces du filtrage collaboratif citées ci-dessus ne contredisent pas le fait que les systèmes qui utilisent la méthode du filtrage basé sur le contenu aient des avantages:

- ✓ Ils sont capables de recommander des items qui n'ont pas été évalués auparavant.
- ✓ Ils peuvent, également, recommander des items aux utilisateurs ayant des intérêts uniques.

- ✓ Ces systèmes caractérisent chaque profil sans aucune comparaison avec les intérêts ou les préférences des autres profils, ce qui facilite la production des explications des caractéristiques des items recommandés [Mooney *et al.*, 2000].

Par ailleurs, un problème commun aux deux techniques de filtrage est « *le problème de démarrage à froid* » qui arrive au premier stade de la recommandation. Pour produire des recommandations précises, le filtrage basé sur le contenu dépend d'une période d'apprentissage afin de détecter les préférences des utilisateurs. Le filtrage collaboratif, quant à lui, dépend de la disponibilité d'une masse critique d'utilisateurs pour qu'il arrive à filtrer efficacement. Or, le premier utilisateur ne peut pas être connecté à d'autres vu qu'il est unique [Pazzani, 1999]. De même, il est difficile pour le système de recommander un nouvel item qui vient d'être évalué ou qui possède peu d'évaluations [Mooney *et al.*, 2000], [Perugini *et al.*, 2004].

3.4.4 Le filtrage hybride

Le filtrage hybride consiste en une combinaison des deux approches de filtrage, collaboratif et basé sur le contenu, et cela dans un seul système de recommandation. La combinaison peut se faire selon différentes techniques permettant de dépasser les faiblesses des deux approches en tirant profit de leurs avantages afin d'améliorer la performance du système [Burke, 2002] [Balabanovic *et al.*, 1997]. La technique mixte est un exemple d'approche utilisée dans les systèmes hybrides. Elle consiste à implémenter plusieurs méthodes de filtrage de façon séparée puis de combiner leurs prédictions [Adomavicius *et al.*, 2005]. *Fab* est un exemple de système hybride pour la recommandation de pages Web [Balabanovic *et al.*, 1997]. Ce système, dans son apprentissage et son maintien des profils des utilisateurs, se sert des techniques basées sur le contenu. Il représente le contenu de page Web avec les 100 mots les plus importants. Il compare par la suite les profils pour trouver les similarités entre eux et fournit des recommandations collaboratives.

3.4.5 La recommandation basée sur le forage de données

Les systèmes de recommandation basés sur le forage de données [Adomavicius *et al.*, 1999b], [Adomavicius *et al.*, 2001], [Perugini *et al.*, 2004], [Schafer, 2005] consistent à analyser un historique de données décrivant les attributs et les actions des utilisateurs pour établir des profils comportementaux. Les connaissances découvertes peuvent prendre plusieurs formes à savoir des segments, des règles de classification ou des règles d'association. Une fois construits, ces profils permettent une compréhension complète des utilisateurs et aident à savoir qui sont-ils, quels sont leurs préférences, la façon dont ils se comportent, et combien sont-ils semblables à d'autres utilisateurs. Un exemple de comportement est le suivant : « *dans 95% des cas, quand Anne achète les céréales, elle les achète chez Costco* ». En se basant sur les modèles ou les règles déjà établies, les systèmes de recommandation peuvent inférer leurs recommandations et, par conséquent, personnaliser les services ou les produits qu'ils fournissent à leurs utilisateurs.

Par définition, la personnalisation est un processus itératif constitué des étapes schématisées sur la figure 3.1 [Adomavicius *et al.*, 2001]. La première étape consiste à collecter les données, les nettoyer, les préparer puis les stocker. En se basant sur les données préparées, la deuxième étape construit les profils d'utilisateurs qui doivent être compréhensibles et exactes. Dans la troisième étape, le système de recommandation doit rapprocher les produits ou les services aux différents profils d'utilisateurs. L'étape suivante qui constitue la livraison, peut présenter les produits selon différentes manières telle qu'une liste ordonnée selon la pertinence des recommandations. La dernière étape qui consiste à mesurer la réponse (*le feedback*) de l'utilisateur sert à évaluer l'efficacité de la recommandation. Son objectif est d'améliorer, s'il ya lieu, les autres étapes du processus de personnalisation. Ainsi, cette étape procure un feedback à toutes les autres étapes.

Les données des utilisateurs peuvent être modélisées par différents types de règles, comme les règles de classification et les règles d'association (voir chapitre 2 : forage de données). Pour ce faire, plusieurs algorithmes de forage de données peuvent être utilisés, tels que l'algorithme *Apriori* pour l'extraction des règles d'association et *CART* pour la génération des règles de classification. Comme expliqué dans le chapitre 2

(forage de données), nous pouvons extraire, à partir d'une base de données des transactions de vente, des règles d'association pour déterminer les produits qui sont le plus fréquemment achetés ensemble. Cela permet de recommander aux consommateurs des produits en se basant sur d'autres produits qu'ils ont achetés. Dans le domaine du forage du Web, le problème de trouver les pages visitées ensemble est semblable à trouver des associations dans les bases de données des transactions de vente [Géry *et al.*, 2003]. Chaque page Web représente un item et l'ensemble des pages visitées par les utilisateurs constitue les transactions de la base de données. Ainsi, le système procède à l'extraction des règles d'association. Étant donné les pages visitées par un utilisateur, le système cherche les règles dont l'antécédent est équivalent à ces pages. Une fois trouvées, les conséquents de ces règles prédisent les pages à suggérer à l'utilisateur.

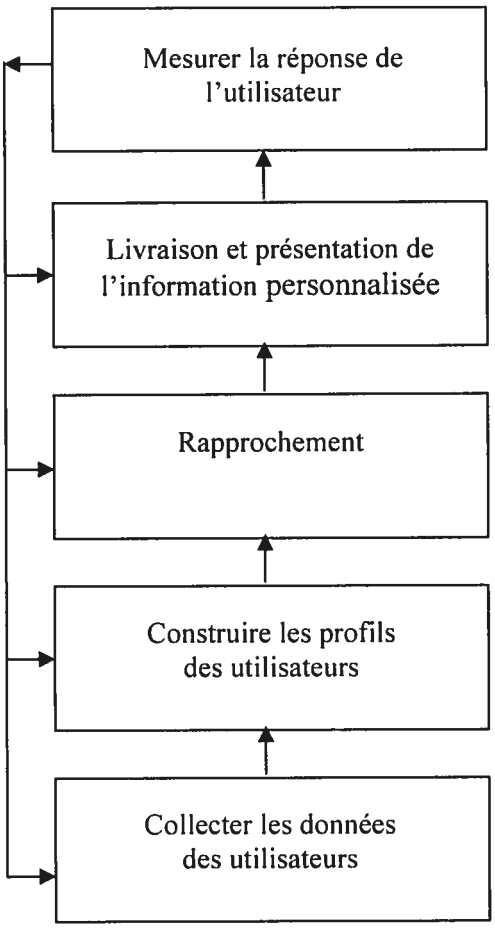


Figure 3.1 : Le processus de personnalisation [Adomavicius *et al.*, 2001]

L'avantage de cette famille de systèmes de recommandation est que tout le processus de forage de données est exécuté hors ligne [Perugini *et al.*, 2004]. Ainsi, malgré que le temps du processus d'établissement des profils puisse être long, leur exploitation durant la recommandation est rapide. Cependant, le processus de forage de données requiert un ensemble de données. Les deux chapitres suivants (*conception et implémentation*) expliquent en détail le fonctionnement de notre système de recommandation, *RARE*, basé sur la technique d'extraction des règles d'association.

3.5 La qualité de la recommandation

Vu les faiblesses des techniques utilisées par les systèmes de recommandation, il s'avère difficile d'assurer une qualité parfaite des suggestions fournies. Cependant, les techniques ne sont pas les seuls facteurs d'influence sur la qualité de la recommandation. Il est, donc, très utile d'étudier la qualité selon différents points de vue, tels que les erreurs qu'un système peut commettre, la transparence, la confiance dans le système, etc. La problématique de la qualité de la recommandation a été explorée dans une étude expérimentale à l'université de *Berkeley* en *Californie* pour une comparaison entre les recommandations fournies par des systèmes de recommandation en ligne et celles proposées par des amis [Swearingen *et al.*, 2001]. Les résultats de cette expérience ont montré que bien que les utilisateurs préfèrent les recommandations proposées par leurs amis, plus que la moitié d'entre eux expriment une grande satisfaction envers les systèmes de recommandation en ligne. Cela est du à leur utilité ainsi qu'à leur capacité de proposer des items que l'utilisateur ne connaissait pas auparavant. Toutefois, les systèmes de recommandation n'ont pas les mêmes performances. Dans la section qui vient, nous expliquons quelques critères qui doivent être vérifiés lors de la conception d'un système de recommandation pour mieux garantir la qualité des suggestions qu'il fournit et gagner ainsi la confiance de ses utilisateurs.

3.5.1 La transparence du système

Un bon système de recommandation ne doit pas être une boîte noire vis-à-vis de ses utilisateurs. Au contraire, il doit les faire vivre le processus de recommandation, leurs

donner une idée sur la logique du traitement des requêtes et leurs fournir les explications des résultats produits [Rafaeli *et al.*, 2005] [Swearingen *et al.*, 2002]. L'explication aide l'utilisateur à comprendre le raisonnement derrière une recommandation et connaitre ses forces et ses faiblesses, et accorder en conséquence le crédit à cette recommandation [Herlocker *et al.*, 2000]. La compréhension de la relation entre les données entrées et les résultats affichés permet aussi à l'utilisateur une interaction efficace avec le système. En résumé, l'obscurité d'un système l'empêche de gagner la confiance de ses utilisateurs surtout dans des domaines à risque comme le domaine des vacances. Un exemple d'une explication d'une recommandation est le suivant : *ce chemisier est cher par ce qu'il est en soie, celui ci est moins cher, il est en polyester*. Non seulement le système doit fournir les explications de ses recommandations, il peut en plus présenter à son utilisateur la pertinence des questions qu'il pose. Une telle méthode est présentée dans le système *Top Case* [McSherry, 2004].

3.5.2 Sources d'erreurs

Les erreurs d'un système de recommandation peuvent provenir de deux sources : le profil d'utilisateur et les données :

- **Erreurs dans le profil**

  Un système durant son processus de recommandation essaye de tracer le profil de l'utilisateur pour adapter la recherche à ses caractéristiques. Ce profil peut changer selon le contexte et les objectifs de la recherche, et même du tempérament de l'utilisateur. Cela peut fausser le calcul des recommandations [Herlocker *et al.*, 2000], [Adomavicius *et al.*, 2001].

- **Erreurs de données**

  Les données fournies à un système de recommandation peuvent être insuffisantes, erronées et même contradictoires. Dans le premier cas, le système lors de son démarrage procède au filtrage sur un petit ensemble de données. Les profils sont alors nouveaux et incomplets et les items sont peu évalués. Dans le cas du filtrage collaboratif, la recommandation sera basée sur un rapprochement de l'utilisateur à

d'autres partageant très peu d'intérêts avec lui. Le système produit, par conséquent, des recommandations en se basant sur un minimum de similarité. Et même avec un volume considérable d'informations, des données erronées dues à des erreurs de saisie peuvent exister. Tous ces problèmes peuvent engendrer des recommandations pauvres telles que *le faux positif* et *le faux négatif* [Cho *et al.*, 2002], [Lu, 2004] deux types de recommandations non attendus. Tandis que le premier type consiste à fournir des recommandations qui ne conviennent pas à l'utilisateur, le deuxième ignore des recommandations très pertinentes pour lui.

3.5.3 La confiance

Après une certaine expérience avec un système de recommandation, un utilisateur peut bâtir une confiance en lui ou se méfier de ses suggestions. Cette confiance peut être acquise suite à plusieurs raisons telles que:

- La similarité entre le goût de l'utilisateur et celui du système.
- La possibilité de fournir des informations sur les recommandations produites [Swearingen *et a*l., 2002].
- L'évaluation de l'historique des recommandations. En d'autres termes, si les recommandations antérieures ont été jugées positives et bénéfiques, ceci favorise le succès du système vis-à-vis de l'utilisateur et, par conséquent, augmente la confiance en lui. Et même si le système recommande plusieurs anciens items appréciés par l'utilisateur, les nouveaux items pourront également plaire à ce dernier [Swearingen *et al.*, 2001].

3.5.4 Autres facteurs

D'autres métriques peuvent être prises en compte dans la mesure de la qualité des recommandations :

- **L'utilité de la recommandation**

  Les recommandations produites doivent être pertinentes. Du point de vue utilité, elles doivent être nouvelles et porteuses afin de satisfaire le besoin requis et enrichir la connaissance de l'utilisateur dans un domaine quelconque. Selon l'application, l'utilité peut être spécifiée par l'utilisateur, à travers ses évaluations, ou calculée par l'application, comme le cas d'une fonction d'utilité basée sur le profit [Adomavicius *et al.*, 2005].

- **Le temps**

  Pour mesurer le temps global que peut prendre un processus de recommandation, on doit calculer le temps écoulé entre l'enregistrement de l'utilisateur et la production des recommandations par le système. Ce temps doit être acceptable.

- **L'interface**

  Dans la conception de l'interface, on doit tenir compte de plusieurs critères. La quantité des données que l'utilisateur doit fournir ainsi que la présentation des résultats retournés, ont un impact sur la qualité du système.

D'autres caractéristiques générales comme la navigation, l'utilisation des couleurs et le graphisme peuvent être également prises en compte par les systèmes de recommandation.

3.6 La recommandation de cours

Les livres, les ressources Web, les films et beaucoup d'autres produits représentent les domaines d'application de plusieurs systèmes de recommandation. Dans le domaine des études, les étudiants souffrent d'un manque d'information à propos des cours qui conviennent à leurs besoins et trouvent des difficultés lors de leurs choix. Au meilleur de notre connaissance, seulement quelques systèmes de recommandation de cours ont été développés. La section suivante est une revue de ces travaux.

3.6.1 Student Course Recommender (*SCR*)

Le système *SCR* a été réalisé dans le cadre d'une maîtrise à l'Université de Technologie *Lulea* en Suède [Ekdahl *et al.*, 2002]. Son objectif est d'étudier comment les méthodes d'apprentissage machine peuvent être utilisées pour créer un système capable de suggérer des cours et d'apprendre à partir des choix de cours effectués par les étudiants. Le résultat est une implémentation d'une stratégie qui utilise la modélisation par réseau bayésien. Les utilisateurs visés par le système *SCR* sont les étudiants de l'université. Les informations sur les cours suivis, le programme d'éducation et la spécialisation sont données par l'utilisateur (voir figure 3.2) et stockées dans la base des étudiants. Cette base doit être mise à jour chaque fois qu'un nouvel utilisateur accède au système, ou qu'il y ait une modification d'information de la part d'un ancien utilisateur. L'étudiant doit aussi fournir explicitement les sessions d'études et les cours non souhaités. Les informations générales nécessaires dans la description d'un cours, d'un programme d'éducation, ou d'une spécialisation sont sauvegardées dans la base des cours.

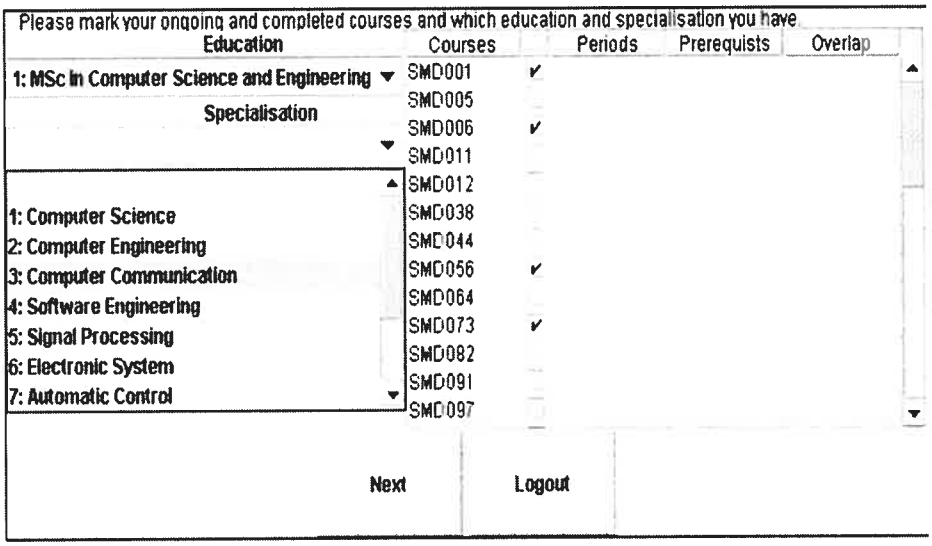


Figure 3.2 : Une interface utilisateur de *SCR* [Ekdahl *et al.*, 2002]

La modélisation bayésienne est utilisée dans la recommandation à un étudiant particulier en se basant sur des informations le concernant et des informations stockées dans la base des cours. L'étudiant peut exiger, via une interface usager, des contraintes

comme la session de cours, comme il peut avoir de nouvelles suggestions s'il n'est pas satisfait de celles déjà proposées, comme la figure 3.3 le montre. Dans le réseau utilisé par *SCR*, les variables représentent les cours et les programmes d'études de l'université. Un cours ou un programme peut être suivi ou pas par un étudiant. Donc, chaque variable peut avoir deux états possibles qui sont : *vrai* ou *faux*. Si un arc part d'un nœud de cours vers un autre, il résulte que le premier cours est un pré-requis du cours suivant. Pour chaque nœud, qui a au moins un parent, la probabilité conditionnelle de chaque état de ce nœud, étant donné toutes les combinaisons possibles d'état des nœuds parents, doit être spécifiée. Au moment de la création d'un réseau bayésien, ces valeurs sont stockées dans des paramètres dans une table de probabilité conditionnelle appelée *CPT*, pour chaque nœud fils. Si un nœud n'a aucun parent, la *CPT* contiendra les paramètres qui représentent la distribution de probabilité antérieure des états des variables du nœud.

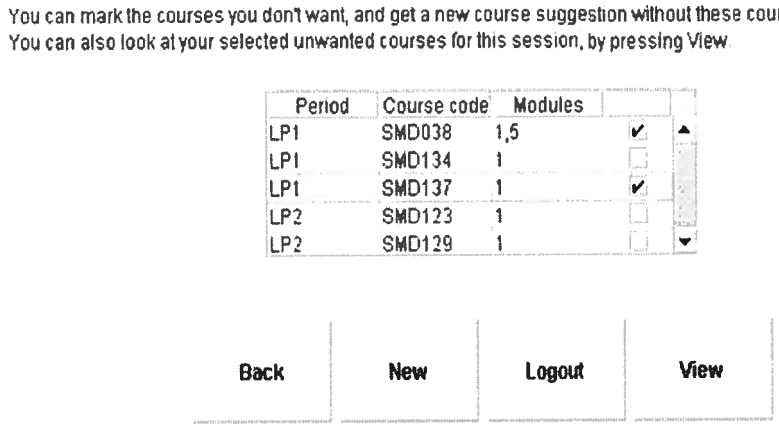


Figure 3.3 : Les recommandations de *SCR* [Ekdahl *et al.*, 2002]

Pour que le système démarre son fonctionnement, il faut que la base des étudiants contienne suffisamment de cas. Par conséquent, si un utilisateur n'a suivi aucun cours et ne poursuit aucun programme à l'université, *SCR* ne lui fournira aucune recommandation. Ainsi, plus le système est utilisé, plus il apprend à partir des nouvelles informations et arrive à faire des prédictions plus précises pouvant affiner les recommandations proposées.

3.6.2 Course recommender

Course recommender [Simbi, 2003] est un système de recommandation qui a été développé à Princeton University pour aider les étudiants à faire leurs choix de cours. Il implémente quelques algorithmes de filtrage collaboratif à savoir l'algorithme *basé-utilisateur* [Breese *et al.*, 1998], l'algorithme *basé-item* [Sarwar *et al.*, 2001], *OC1* [Murthy *et al.*, 1993], et une variante modifiée de *C4.5* [Quinlan, 1993]. Dans sa production des recommandations, ce système se base sur les évaluations des cours que fournissent les étudiants. L'évaluation consiste en un vote explicite basé sur la difficulté ainsi que l'appréciation des cours. L'échelle varie entre 1 et 5 dont la valeur 1 indique que le cours évalué est facile/ennuyeux, tandis que 5 signifie qu'il est difficile/agréable. Pour utiliser le système (voir figure 3.4), un étudiant doit d'abord s'enregistrer puis évaluer le plus de cours possible.

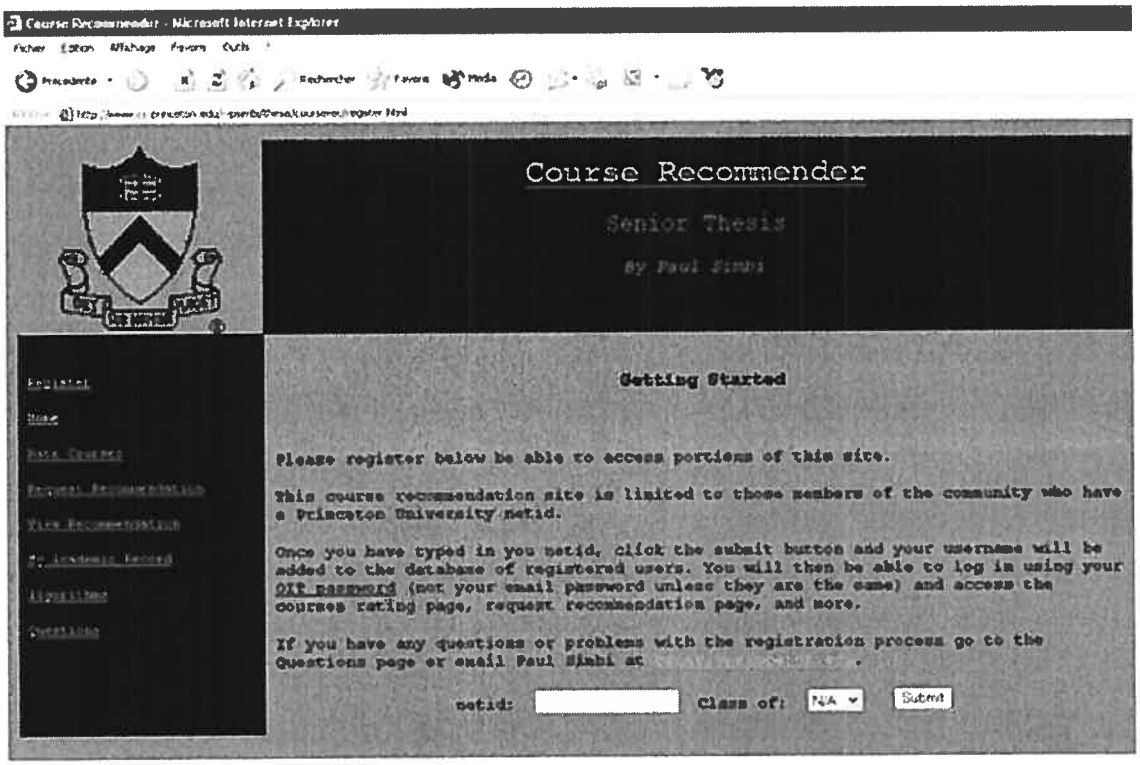


Figure 3.4 : La page Web principale de *Course Recommender* [Simbi, 2003]

Après avoir évalué au moins un cours, l'utilisateur reçoit des recommandations selon la difficulté ou l'appréciation et suivant un ordre ascendant ou descendant. Le

système ne peut pas suggérer des recommandations aux étudiants qui n'ont pas suivi de cours à l'université. Pour stocker les données, deux tables sont utilisées : la table des utilisateurs et celle des cours. La première table contient l'identification des étudiants, un champ pour chaque cours et des champs supplémentaires qui contiennent des informations requises par quelques algorithmes. Les champs des cours contiennent la difficulté et l'appréciation attribuées par les étudiants à chaque cours. Le tableau 3.3 représente cette table.

Tableau 3.3 : Table des utilisateurs [Simbi, 2003]

ID Utilisateur	*Cours 1*	*Cours 2*	...	*Cours n*	*TOTAL*	...
Utilisateur 1	3/5		...	4/4	5	...
Utilisateur 2					10	...
...						...
Utilisateur n		3/4			20	...

La table des cours, quant à elle, contient des champs dont la valeur correspond aux similarités entre deux cours par rapport à la difficulté et à l'appréciation. D'autres champs sont créés pour décrire les cours. Le tableau 3.4 illustre cette table.

Tableau 3.4 : Table des cours [Simbi, 2003]

Cours	**ECO101**	**ECO102**	...	**MAT104**	**Description**	...
ECO101		-0.18/0.35	...	0.04/-0.82	...	...
ECO102	-0.18/0.35		...	-1.00/-1.00	...	...
...			...		...	...
MAT104	0.04/-0.82	-1.00/-1.00	...		...	...

Les algorithmes de filtrage collaboratif suscités ont été testés par le système pour connaitre leurs efficacités dans la production des recommandations. Tandis que l'algorithme *basé-utilisateur* essaye de trouver les étudiants voisins de l'utilisateur cible, le but de l'algorithme *basé-item* consiste à trouver les cours qui sont similaires aux cours appréciés auparavant par l'utilisateur. Par ailleurs, les deux autres algorithmes, *C4.5* (modifié) et *OC1* essayent tout les deux de générer des modèles, en utilisant l'ensemble des évaluations, puis se servent du modèle généré pour prédire des recommandations. Dans l'implémentation de la variante de *C4.5*, par exemple, chaque cours est représenté par un arbre dont les classes représentent les cinq évaluations. La figure 3.5 illustre l'arbre de décision du cours *ECO102*.

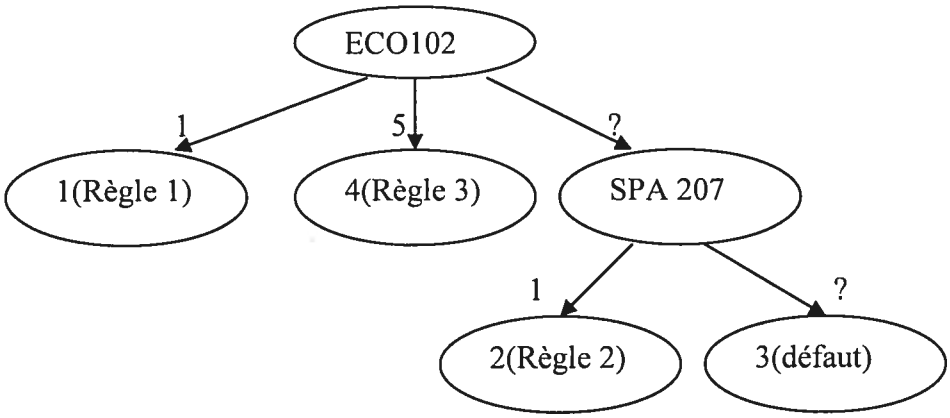


Figure 3.5 : L'arbre de décision du cours ECO102 [Simbi, 2003]

Les résultats affichés à l'utilisateur comportent les dix premières recommandations de chaque algorithme. Suite à la requête, les recommandations sont classées en ordre ascendant ou descendant et selon la difficulté ou l'appréciation. Le tableau 3.5 montre les 10 premières recommandations produites par l'un des algorithmes implémentés. Après l'évaluation des résultats fournis par chaque algorithme, l'auteur suggère *C4.5* comme étant le meilleur algorithme pour la recommandation de cours.

Tableau 3.5 : Les 10 premières recommandations [Simbi, 2003]

Algorithme X		
Cours	*Difficulté prévue*	*Appréciation prévue*
1) MOL 457	1	3.2
2) CWR 202	1	4.4
3) CEE 263	1	4.5
...		
8) SOC 200	1.1	1
9) ARC 311	1.1	1
10) ARC 403	1.1	1

3.6.3 PEL-IRT

PEL-IRT (Personalized E-Learning system using Item Response Theory) [Chen *et al.*, 2005] (Taiwan) est un système de e-learning personnalisé basé sur *la théorie des réponses aux items* (*Item Response Theory*). Il recommande les matériels de cours appropriés aux étudiants, en tenant compte de leur difficulté ainsi que des capacités des étudiants. Chaque matériel de cours est classé dans une unité prédéfinie. En utilisant *PEL-IRT*, les utilisateurs peuvent choisir les catégories et les unités de cours ou employer des mots-clés appropriés afin de rechercher les matériels dont ils ont besoin. S'il s'agit d'une première utilisation, le système fournit des matériels non-personnalisés basés sur les termes de la requête aussi bien que sur la catégorie et l'unité choisies. Après que les matériels de cours soient recommandés et consultés par les étudiants, le système leurs demande de répondre à deux questionnaires. Ce feedback explicite est employé pour réévaluer les capacités des utilisateurs et ajuster la difficulté des matériels des cours. Les nouvelles capacités d'étudiants ainsi que les difficultés modifiées des matériels de cours sont employées pour raffiner les recommandations futures. Afin de modéliser les matériels de cours, la fonction des caractéristiques d'item (*item characteristic function)* avec un paramètre simple de difficulté est employée. Le système utilise également une approche de vote collaboratif pour ajuster la difficulté du matériel de cours. Pour obtenir

une évaluation plus précise des capacités de l'étudiant, l'estimation de la vraisemblance maximale *MLE* (*Maximum Liklihood Estimation*) est appliquée en se basant sur le feedback explicite de l'étudiant.

Ainsi, le système est composé de deux parties principales qui sont : la *partie d'interface* (*front-end*) et la partie principale (*back-end*). La partie d'interface interagit avec les étudiants et enregistre leurs comportements (figure 3.6). La deuxième partie, quant à elle, analyse la capacité d'étudiant et choisi, par la suite, des matériels de cours appropriés. L'agent d'interface, qui appartient à la partie d'interface, identifie le statut des étudiants, transfère les requêtes et suggère aux étudiants des matériels de cours. Cependant, la partie principale peut être divisée en deux agents séparés : l'agent de feedback et l'agent de recommandation de matériels de cours. D'une part, le premier agent vise à rassembler le feedback, mettre à jour les capacités des étudiants, et ajuster la difficulté des matériels de cours. D'autre part, l'agent de recommandation choisit les matériels de cours appropriés aux étudiants à partir de la base des cours. *PEL-IRT* se sert de trois bases de données à savoir la *base des comptes des utilisateurs*, la *base des profils des utilisateurs* et la *base des cours*. Pour identifier le statut de l'étudiant, la base des comptes des utilisateurs enregistre les adresses électroniques des étudiants. La base des profils des utilisateurs contient les termes des requêtes des étudiants, le comportement de clique (*clicking behavior*), les réponses aux questionnaires, les capacités des étudiants et les difficultés des matériels des cours cliqués. La base des cours contient les matériaux des cours et leurs niveaux de difficulté.

Vu que la conception des matériels de cours de haute qualité prend du temps et vu qu'il est également difficile de les obtenir à partir du WEB, le système contient actuellement peu de matériels pour la plupart des unités de cours. En conséquence, le développement d'une grande quantité de matériels pour une utilisation dans ce système est un travail futur. Toutefois, les résultats de l'expérimentation montrent que l'application de la théorie des réponses aux items (IRT) par *PEL-IRT* permet une personnalisation du e-learning et une amélioration de l'efficacité de l'apprentissage des étudiants.

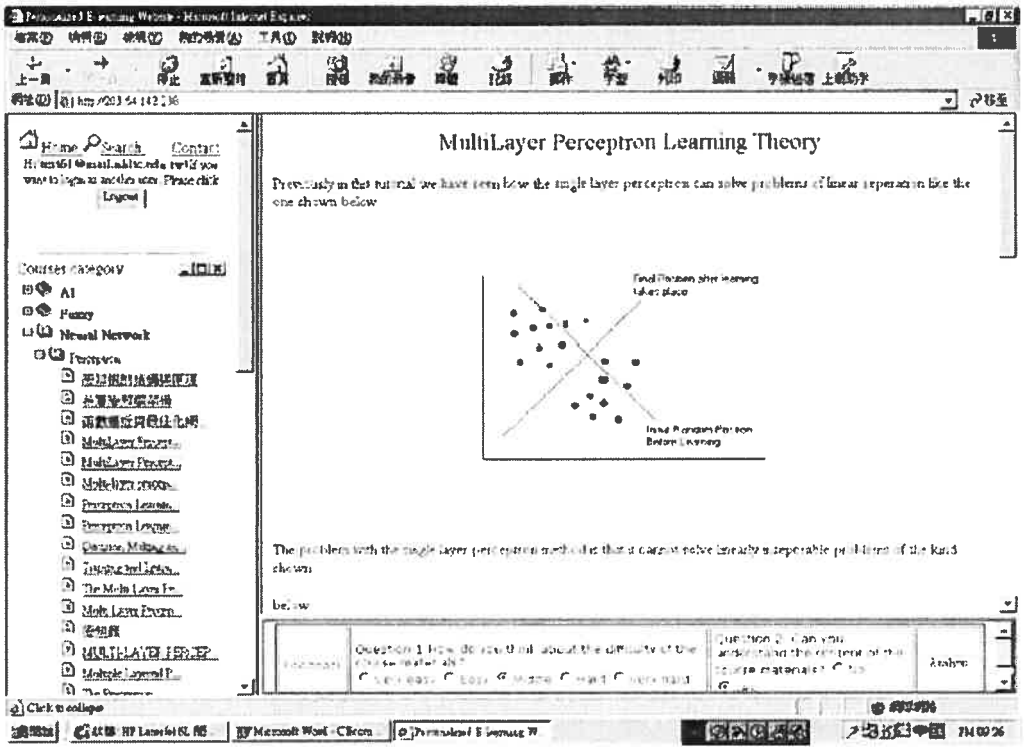


Figure 3.6 : L'interface utilisateur de *PEL-IRT* [Chen *et al.*, 2005]

3.6.4 AACORN

AACORN (Academic Advisor Course Recommendation eNgine) [Sandvig *et al.*, 2006] est un système de raisonnement à base de cas qui est actuellement en cours de développement. Il recommande des cours aux étudiants gradués de School of Computer Science, Telecommunications, and Information Systems (*CTI*) de DePaul University (Chicago). L'architecture d'*AACORN* comporte deux composantes : une composante de *raisonnement à base de cas* et une composante de *satisfaction de contrainte*. Le raisonnement à base de cas est une méthode de résolution de problème qui utilise des connaissances issues des expériences antérieures pour ajuster une solution à un nouveau problème [Aamodt *et al.*, 1994]. Ainsi, dans la composante de raisonnement à base de cas, *AACORN* suppose que les étudiants similaires ont des historiques de cours similaires. Il représente un cas comme un objet structuré contenant des informations sur un étudiant. Ainsi, un cas inclut quatre niveaux de caractéristiques à savoir: le programme académique de l'étudiant, les termes du curriculum qui définissent les conditions de graduation de l'étudiant, la moyenne globale, ainsi que l'historique de cours. Ce dernier

est un dispositif hiérarchique contenant des informations sur chaque cours suivi par l'étudiant. À partir des expériences passées des anciens étudiants, le système utilise les historiques de cours comme base dans la recommandation. Ceci est effectué par un processus de recherche des séquences similaires d'historiques de cours. Pour déterminer la similarité entre ces derniers, le système emploie une métrique appelée « *edit distance* » et utilisée généralement en bioinformatique, aussi bien que d'autres heuristiques. *Edit distance* représente le nombre minimal d'opérations exigées afin de transformer un historique spécifique de cours en un autre. Les opérations basiques qui peuvent être effectuées sont l'insertion, la suppression, et le remplacement.

Pour fournir des recommandations à ses utilisateurs, *AACORN* leurs exige un historique partiel des cours suivis. Une composante de satisfaction de contrainte est aussi intégrée comme étant un filtre de recommandations. Ainsi, les cours qui ne seront pas offerts dans la session prochaine ainsi que ceux qui ne sont pas requis dans le programme académique de l'utilisateur seront filtrés.

3.7 Conclusion

Ce chapitre a présenté une vue globale des outils de recherche et de filtrage d'information ainsi que des systèmes de recommandation. Malgré qu'une multitude de techniques de recommandation existent, nous ne pouvons toutes les présenter dans ce chapitre. Nous avons donc cité en premier lieu les techniques qui se servent du filtrage collaboratif et du filtrage basé sur le contenu. Or, chacune de ces techniques présente des avantages et souffre de faiblesses. Toutefois, la disponibilité d'un ensemble de données initiales permet à un système basé sur le forage de données d'extraire des connaissances utiles dans la recommandation et de dépasser quelques problèmes, comme notamment le problème de démarrage à froid et le problème de données à grande échelle (*scalability*), sans perdre les avantages des autres techniques de filtrage. Dans le chapitre suivant, nous présentons la conception de notre système de recommandation de cours basé sur le forage de données.

Chapitre 4 : Conception de *RARE*

Dans ce chapitre, nous discutons de la conception de notre système de recommandation de cours nommé *RARE*. Nous commençons tout d'abord par la présentation de l'approche suivie dans son développement et nous schématisons son architecture. Ensuite, nous y décrivons les deux composantes principales de l'architecture proposée, et nous expliquons les rôles des différentes bases de données utilisées par le système.

4.1 L'approche et l'architecture de *RARE*

Pour estimer les cours les plus appropriés aux besoins et aux objectifs des étudiants, nous avons développé et implémenté *RARE* (*a course Recommender system based on Association RulEs*), un système de recommandation de cours basé sur les règles d'association. Son principe est de faire profiter les étudiants d'une expérience collective. Pour ce faire, *RARE* combine l'expérience des étudiants qui ont achevé leurs études avec les évaluations des cours données par les étudiants actuels. Ainsi, l'approche suivie est constituée de deux phases : une *phase hors ligne* et une *phase en ligne*. La *phase hors ligne* représente l'élément clé de notre approche. Elle consiste en un processus de forage de données effectué sur l'historique des données réelles relatives aux choix des cours accomplis par les anciens étudiants. L'objectif de l'intégration de ce processus est d'extraire des règles qui permettront au système d'inférer les recommandations de cours pendant la phase en ligne. Ainsi, le système de recommandation *RARE* est basé sur le forage de données. Par le biais des règles extraites, le système sera capable de dépasser le problème de démarrage à froid dont la plupart des systèmes de recommandation souffrent (voir chapitre 3 : système de recommandation). Or, l'extraction des règles doit s'exécuter hors ligne vu que le forage de données est un processus semi-automatique dont seule l'étape manuelle de préparation de données requiert 60% du temps global d'exécution [Cabena *et al.*, 1998] (voir chapitre 2 : forage de données). La *phase en ligne* de *RARE*, quant à elle, s'exécute en temps réel. Elle représente l'interaction du système avec les utilisateurs. En outre, cette phase est dédiée à la recommandation qui encapsule toutes les opérations relatives au traitement des requêtes des utilisateurs. En plus, pour permettre la collaboration de ces derniers, *RARE* met en place un processus de traitement du

« feedback » qu'il reçoit de leur part. Celui-ci est fondé sur les évaluations des cours fournies au système et permettant la mise à jour des règles afin d'affiner les recommandations futures. Les deux phases forment les deux principales composantes du système qui sont, entre autres, schématisées dans la figure 4.1 et expliquées dans les sections qui suivent.

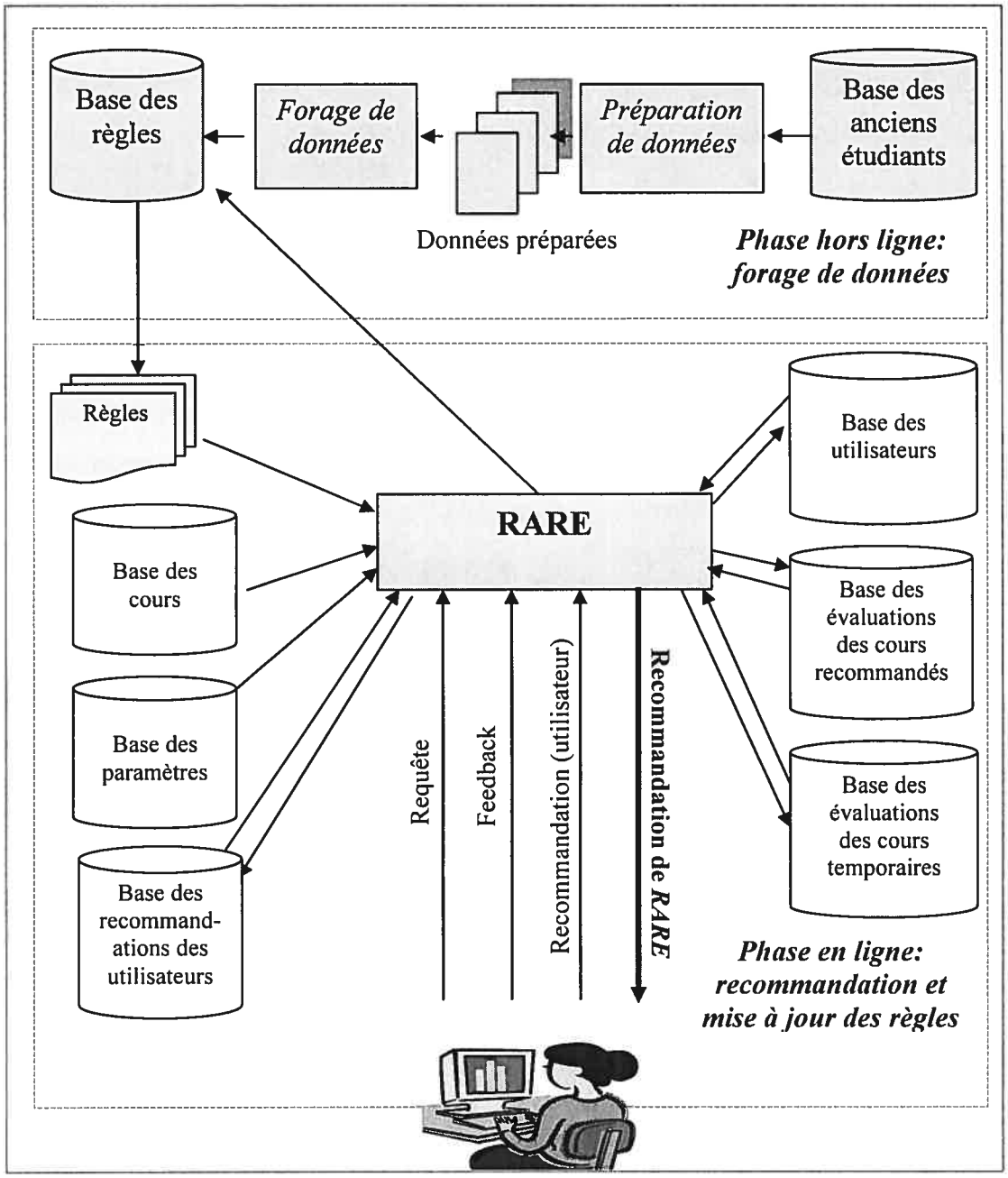


Figure 4.1 : L'architecture de *RARE*

4.2 Phase hors ligne

La phase hors ligne est fondée sur l'application des techniques de forage de données. Ainsi, cette phase encapsule les six étapes qui se rapportent au processus de forage de données à savoir : la détermination de l'objectif, la compréhension et la préparation des données, la modélisation, l'évaluation et le déploiement (voir chapitre 2 : forage de données). Ce qui suit décrit chacune de ces étapes.

4.2.1 Détermination de l'objectif (étape 1)

Afin de disposer des données réelles sur lesquelles nous effectuons le processus de forage de données, nous visons la collecte d'un ensemble de données des étudiants ayant suivi le programme de maîtrise au sein du Département d'Informatique et de Recherche Opérationnelle de l'Université de Montréal. Dans ce programme, l'étudiant doit suivre des cours et réaliser un projet de recherche dans l'un des laboratoires de recherche existants. Ainsi, la structure de ce programme permet la généralisation de notre travail à d'autres programmes d'études où une multitude de cours peut être offerte aux étudiants, et leurs choix de cours leur permettent de suivre une spécialité parmi plusieurs. Dans notre cas, les cours à recommander sont donc les cours offerts aux étudiants de maîtrise en informatique, et les spécialités à suivre constituent les laboratoires de recherche.

L'objectif que nous poursuivons lors de la mise en œuvre de notre processus de forage de données est d'extraire des règles qui décrivent, le mieux possible, les profils des anciens étudiants concernant les choix des cours qu'ils ont effectués. L'extraction des règles peut se faire suite à la classification de données, à la génération des règles d'association, à la détection automatique de segments,...etc. Les règles de classification et les règles d'association sont deux types de modèles fréquemment recherchés dans les processus de forage de données. Elles constituent des modèles compréhensibles et faciles à exploiter par l'utilisateur [Larose, 2005]. De ce fait, nous essayons d'abord de classifier les données, puis de découvrir les règles d'association. Selon les résultats obtenus à partir de la classification de données et de la recherche des règles d'association la technique la

mieux adaptée à nos besoins sera choisie et les règles obtenues seront employées dans la recommandation de cours.

4.2.2 Compréhension et préparation de données (étapes 2, 3)

Au départ, les données collectées peuvent contenir toute information se rapportant aux étudiants comme le nom, le sexe, les cours suivis, l'année d'inscription, les notes obtenues, les domaines de recherche, etc. Un exemple des données collectées concernant un étudiant peut être le suivant : « *Carl* est un étudiant du Département d'Informatique de l'Université de Montréal qui s'intéressait à la recherche en téléinformatique. Il s'est inscrit en 1996 au programme de maîtrise et a suivi les cours suivants : « *Introduction à l'algorithmique* », « *Synthèse des systèmes numériques* », « *Téléinformatique* », « *Spécification et vérification formelle* » et « *Intelligence artificielle : introduction* ». Les notes qu'il a obtenues sont respectivement : *B+, A-, A, B et A* ». Les données collectées sont sauvegardées dans une base désignée par la *base des anciens étudiants* (figure 4.2).

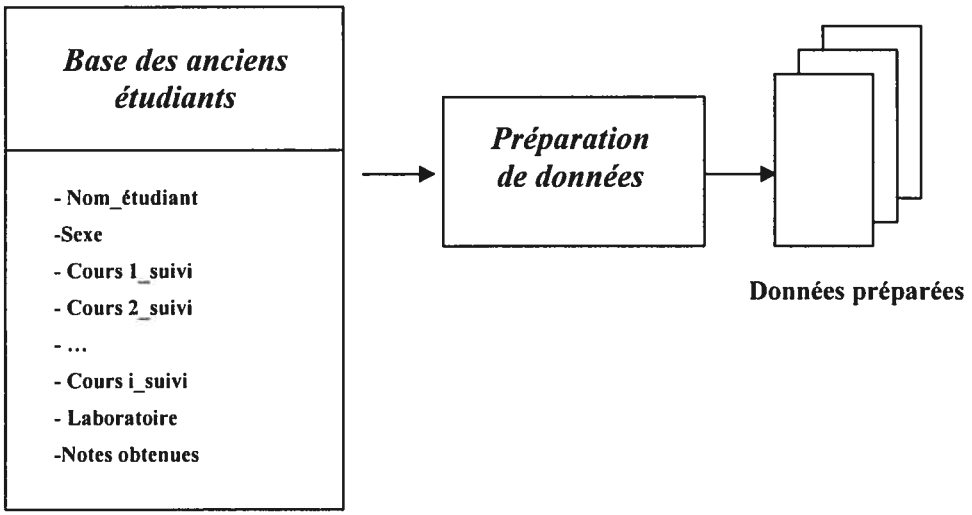


Figure 4.2 : Base des anciens étudiants

Vu que nous nous intéressons aux cours suivis par les étudiants ainsi qu'aux domaines de leur recherche, une étape de nettoyage de données est nécessaire. Son but est de supprimer toutes les données non significatives dans le contexte de notre application et

de garder les attributs qui nous révèlent des informations sur les choix des cours. Dans l'exemple précédent, les données qui peuvent nous être utiles sont les cours suivis par *Carl* ainsi que son domaine de recherche. À la suite de la préparation, un formatage de données doit être effectué pour adapter les données aux besoins des algorithmes que nous allons appliquer dans l'étape de modélisation lors de l'implémentation.

4.2.3 Modélisation (étape 4)

La modélisation des données consiste à appliquer les algorithmes de forage sur les données préparées et formatées. Le rôle des algorithmes de classification est de construire un arbre de décision ou des règles de classification qui permettent d'assigner une classification à chaque cas (c.à.d. étudiant). Vu que les données à modéliser se centralisent autour des cours suivis et des domaines de recherche, nous pouvons tracer des profils d'étudiants constituant des parcours ou des chemins. Chaque parcours part du nœud racine qui représente l'ensemble des cas, passe par des nœuds intermédiaires constituant les cours suivis et mène à une feuille marquée par un laboratoire de recherche. Les règle de classification sont de type : « **SI** *conditions* **Alors** *prédiction* ».
Dans l'exemple 1, nous montrons une règle de classification qui peut représenter un parcours dans l'arbre de décision :

Exemple 1 :
« **Si** premier_cours = *Concepts de bases de données avancées* **et** deuxième_cours = *Traitement des connaissances* **et** troisième_cours = *Génie logiciel avancé* **Alors** laboratoire = Multimédia et tutoriels intelligents (*HERON)* »

D'autre part, les règles d'association sont de type : A ==> B (*Antécédent* ==> *Conséquent*). Leur objectif est de découvrir des relations permettant de tracer les cours qui sont souvent suivis simultanément. En d'autres termes, ces règles découvrent les implications qui existent entre les cours suivis. Ainsi, nous pouvons générer des règles d'association au niveau de chaque laboratoire de recherche afin de découvrir les cours qui sont suivis simultanément par les étudiants appartenant au même domaine de recherche.

Ce qui suit représente un exemple de règle d'association qui peut être générée au niveau du laboratoire *CRT* (Centre de Recherche sur les Transports).

Exemple 2 :

« *Techniques d'optimisation 3* **et** *Programmation en nombres entiers* ==> *Simulation : aspects stochastiques* **et** *Sujets en optimisation* »

4.2.4 Évaluation (étape 5)

La recommandation par le biais des règles de classification se fait par la vérification de la similarité entre les cours suivis par l'utilisateur et les conditions des règles. Admettons qu'un utilisateur ait suivi quelques cours, le système recherche les règles dont les conditions contiennent les cours suivis. Les cours additionnels dans les conditions de ces règles seront recommandés. Le système peut également suggérer les laboratoires de recherche que constituent les prédictions des règles utilisées.

Exemple 3 :

En utilisant la règle de classification de l'exemple1, le système recommande à un étudiant qui a suivi le cours « *Concepts de bases de données avancées* », les deux cours « *Traitement des connaissances* » et « *Génie logiciel avancé* ». Le laboratoire recommandé est « Multimédia et tutoriels intelligents (*HERON)*».

En utilisant les règles d'association, la recommandation se fait par la vérification de la similarité entre les cours suivis par les étudiants et les antécédents des règles. Les conséquents des règles similaires indiqueront les cours à recommander. Ce qui suit représente un exemple de recommandation en se basant sur une règle d'association:

Exemple 4 :

Si un étudiant a pris les deux cours « *Techniques d'optimisation 3* » et « *Programmation en nombres entiers* », en utilisant la règle d'association de l'exemple 2 (« *Techniques d'optimisation 3* **et** *Programmation en nombres entiers* ==> *Simulation : aspects*

stochastiques **et** *Sujets en optimisation* »), le système recommande les deux cours : « *Génie logiciel avancé* » et « *Algorithmes d'apprentissage* ».

Pour évaluer les règles, la collecte d'un ensemble de données de test est nécessaire. Ces données doivent décrire les cours suivis par des étudiants qui ont suivi ou qui suivent le programme de maîtrise et n'appartenant pas à l'ensemble à partir duquel nous avons généré les règles. Pour chaque étudiant E_i, nous appelons S_i la liste des cours qu'il a suivis. Les deux questions principales que nous posons sont :

- Étant donnés le premier cours ou les deux premiers cours de S_i, est ce que le système arrive à recommander des cours à cet étudiant ?
- Est ce que les recommandations produites sont appropriées ? Autrement dit : sont-elles contenues dans le reste des cours de S_i (c.à.d. dans l'ensemble des cours suivis par E_i et non connus par le système) ?

En d'autres termes, nous mesurons le *recouvrement* et l'*exactitude* de la recommandation. Si nous désignons par $R(E_i)$ l'ensemble des cours recommandés à l'étudiant E_i, et nous désignons par $T(E_i)$ les cours figurant dans S_i et non connus par le système, nous définissons le recouvrement et l'exactitude de la recommandation comme suit :

➢ *Le recouvrement* mesure la capacité du système de produire tous les cours qui sont susceptibles d'être suivis par l'étudiant, c.à.d. la proportion des cours recommandés appropriés par rapport à tous les cours qui doivent être recommandés. Sa formule est définie par :

$$\text{Re}\,couvrement = \frac{|R(E_i) \cap T(E_i)|}{|T(E_i)|}$$

➢ *L'exactitude* mesure la capacité du système de fournir les recommandations adéquates, c.à.d. la proportion de recommandations appropriées par rapport à la totalité des recommandations. Sa formule est définie par :

$$Exactitude = \frac{|R(E_i) \cap T(E_i)|}{|R(E_i)|}$$

Plus le recouvrement et l'exactitude sont élevés, meilleure est la prédiction des cours à recommander par le système. Ainsi, les règles qui fournissent un meilleur recouvrement et une grande exactitude, dans cette étape d'évaluation, seront maintenues dans une base appelée *base des règles*.

Comme la montre la figure 4.3, la base des règles permet au système la production des recommandations des cours aux utilisateurs. Chaque règle, est définie par un *code*. Pour une règle de classification, la base contient la *condition* et la *prédiction* de la règle. Dans le cas d'une règle d'association, la base comporte l'*antécédent* et le *conséquent* de la règle. Toutefois, ces règles ne sont pas statiques. Elles évoluent au fur et à mesure que les utilisateurs évaluent les recommandations du système. Or, des mises à jour de la base des règles sont prévues.

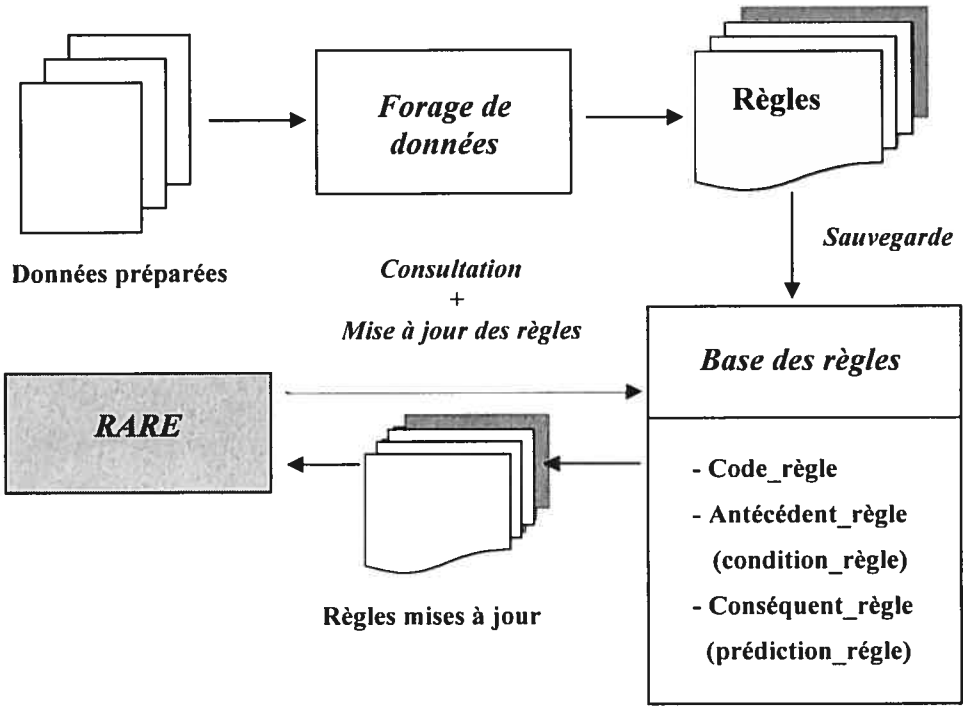


Figure 4.3 : Base des règles

À partir du moment où la phase hors ligne est achevée, *RARE* dispose des connaissances suffisantes lui permettant la réponse aux requêtes de ses utilisateurs dès le démarrage de son fonctionnement. En plus, l'avantage d'extraire les règles, à partir des données des étudiants dans une phase hors ligne, aide le système à dépasser le problème de données à grande échelle (*scalability*), car le fait de traiter une quantité de données plus volumineuse ne requiert pas plus de ressources.

L'étape de déploiement (étape 6), quant à elle, constitue la phase en ligne du système. Elle comporte l'utilisation des règles ainsi que leur mise à jour par le biais des évaluations des utilisateurs. Dans la section suivante, nous expliquons en détail cette phase.

4.3 Phase en ligne

Une fois les règles évaluées et sauvegardées dans la base des règles (voir figure 4.3), la phase en ligne s'occupe de leur usage dans le but d'assurer des recommandations à tous les utilisateurs de *RARE*. Ainsi, les règles sont utilisées de différentes manières. Ceci est expliqué en détail dans les sections suivantes. Une qualité permanente de recommandation doit également être garantie par le système. Il est donc nécessaire de mettre à jour les règles afin de les renforcer et de les améliorer. Cette opération consiste à ajouter et à supprimer des cours dans les règles. Elle tient compte des évaluations des cours, par les utilisateurs, qui permettent au système de calculer le poids de chaque cours. Ce dernier représente une valeur qui permet le maintien d'un cours dans une règle ou sa suppression. Ainsi, non seulement les règles dépendent des choix des cours des anciens étudiants, mais également des poids que leurs attribuent les étudiants actuels.

Comme illustré dans l'architecture, le système utilise une base de données nommée *base des cours* (figure 4.4). Cette base constitue un lieu de stockage de tous les cours. Nous pouvons donc avoir pour chaque cours le *titre*, le *nombre de crédits*, les *pré-requis*, et un *lien* qui mène à une page Web qui décrit le cours. Vu que nous visons le programme de maîtrise en informatique comme domaine de notre application, cette base contiendra tous les cours offerts, aux étudiants de maîtrise, par le Département

d'Informatique et de Recherche Opérationnelle de l'Université de Montréal. La figure 4.4 met en relief l'accessibilité à la base des cours par le système ainsi que par un administrateur. Le rôle de ce dernier est d'effectuer les mises à jour nécessaires dans cette base de données, comme l'ajout des cours, leur suppression, ou la modification des informations sur les cours.

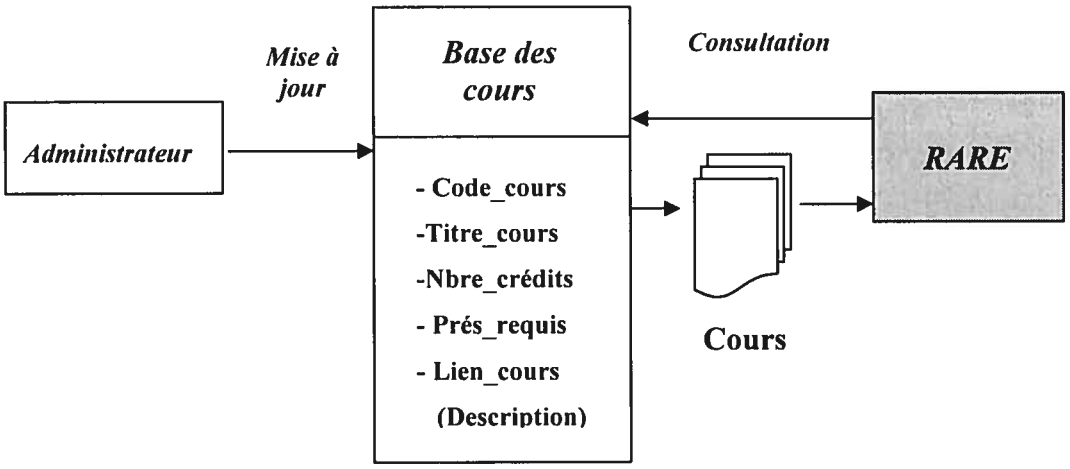


Figure 4.4 : Base des cours

Pour renforcer la collaboration des utilisateurs et pour maximiser leur contribution dans la recommandation, le système les sollicite afin de lui fournir les recommandations des cours qu'ils ont appréciés auparavant. Nous avons donc mis en place la *base des recommandations des utilisateurs* (figure 4.5). Elle sert à stocker les cours recommandés par les utilisateurs et permet d'avoir un autre type de recommandation indépendant de toute contrainte comme les cours suivis auparavant ou la spécialité visée par l'utilisateur. Dans cette base, nous attribuons à chaque cours une valeur nommée *popularité*. Cette donnée représente le nombre de fois que le cours a été recommandé par les utilisateurs. Lors de la recommandation, la popularité sert à classer les cours dans un ordre décroissant de pondération. La liste ordonnée des cours peut être consultée par tous les utilisateurs et à tout moment. La figure 4.5 montre l'interaction de la base des recommandations des utilisateurs avec le système ainsi qu'avec l'administrateur. Ce

dernier fait des mises à jour quand les codes des cours changent. Il supprime aussi les cours qui ne sont plus offerts.

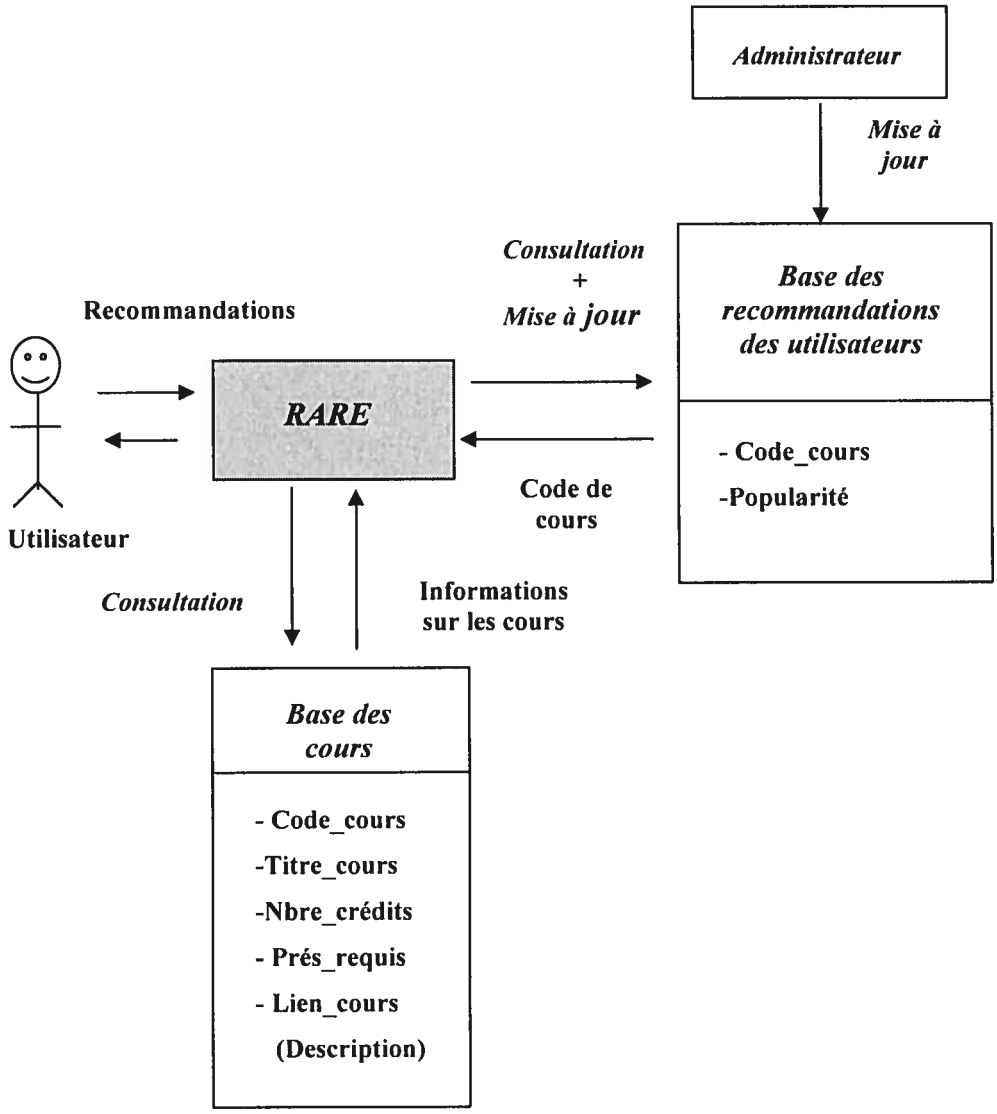


Figure 4.5 : Base des recommandations des utilisateurs

Les différentes informations sur l'utilisateur constituent son profil qui est sauvegardé dans une base nommée *base des utilisateurs*. Comme le montre la figure 4.6, cette base contient le *nom d'utilisateur*, son *mot de passe*, son *adresse électronique*, les *cours qu'il a suivis* et les *cours que lui recommande le système* lors de chaque utilisation. *RARE* attribue aussi à chaque utilisateur une *fidélité*. Cette donnée représente le nombre

de fois qu'un étudiant a utilisé le système et lui a fourni des évaluations. Elle aide donc, à identifier les utilisateurs qui sont fidèles. Ainsi, la fidélité d'un nouvel étudiant est initialisée à 1 et incrémentée (par 1), à la fin de chaque réutilisation du système. La consultation des profils des utilisateurs permet d'assister ces derniers dans leurs choix et leurs évaluations de cours et, par conséquent, d'intégrer leur collaboration dans le processus de recommandation.

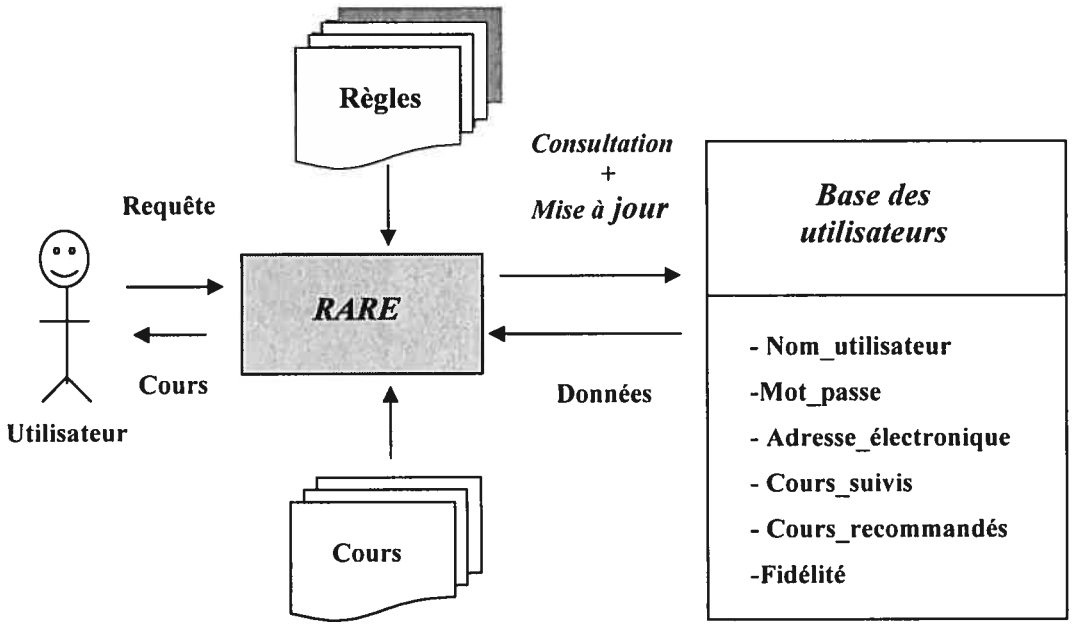


Figure 4.6 : Base des utilisateurs

Lors de l'implémentation de *RARE* (chapitre 5), nous procéderons à l'extraction des règles de classification et des règles d'association. Ensuite, nous évaluerons les règles obtenues en suivant la méthode proposée dans la section 4.2.4. Nous déciderons ainsi du type des règles (règles de classification ou règles d'association) qui conviendra le mieux à la recommandation. De ce fait, nous proposons dans les sections qui suivent, la conception de la phase en ligne selon deux modes de fonctionnement : un *fonctionnement basé sur les règles de classification* et un *fonctionnement basé sur les règles d'association.*

4.3.1 La recommandation

Pour que *RARE* puisse tirer avantage des règles extraites, il offre à ses utilisateurs une multitude de choix leurs permettant d'avoir plusieurs types de recommandation. Lorsqu'un utilisateur envoie sa requête, le système lance une action de recherche de règles qui va tenter de répondre à cette requête de différentes façons. Si l'utilisateur a déjà suivi des cours, le rôle du système est de produire des recommandations basées sur les cours suivis. Sinon, l'utilisateur peut faire une simple recherche de recommandations par sélection de laboratoire (domaine) de recherche. La vocation du système cette fois-ci, consiste essentiellement à trouver l'ensemble des règles qui sont en relation avec le laboratoire sélectionné. L'utilisateur peut lancer aussi une requête lui permettant la consultation des recommandations des autres utilisateurs, comme il peut, à son tour, suggérer des recommandations au système. Ce qui suit décrit en détail les opérations suscitées, et ceci dans les deux modes de fonctionnement : basé sur les règles de classification et basé sur les règles d'association.

4.3.1.1 La recommandation basée sur les règles de classification

Comme nous l'avons déjà mentionné, les règles de classification prennent la forme suivante : « **Si** *conditions* **Alors** *prédiction* ». Ce qui suit explique comment le système les utilise selon les différents types de recommandations suivants :

- ***La recommandation par cours suivis***

Si l'utilisateur a déjà pris des cours, il peut les fournir au système. Ce dernier parcouru les règles à la recherche de celles dont les conditions contiennent les cours suivis. En effet, plusieurs cas se présentent :

- ***Cas 1*** **:** le système trouve des règles dont les conditions couvrent tous les cours suivis par l'utilisateur. Il recommande alors, tous les cours additionnels dans les conditions des règles retrouvées. Pour un étudiant qui a suivi le cours « *Traitement des connaissances* », par exemple, le tableau 4.1 indique qu'il y a exactement 5 recommandations en réponse à sa requête.

Tableau 4.1 : Les recommandations basées sur les règles de classification (*cas 1*)

Règles de classification	***Cours recommandés***
Si premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Concepts de bases de données avancées* **Alors** laboratoire = *Multimédia et tutoriels intelligents (HERON)*	*1. Concepts de bases de données avancées*
Si premier_cours = *Intelligence artificielle: introduction* **et** deuxième_cours = *Traitement des connaissances* **et** troisième_cours = *Recherche d'information* **Alors** laboratoire = *Linguistique informatique (RALI)*	*1. Intelligence artificielle : introduction;* *2. Recherche d'information*
Si premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Génie logiciel avancé* **et** troisième_cours = *Recherche d'information* **et** quatrième_cours = *Sujets en génie logiciel* **Alors** laboratoire = *Génie logiciel (GEODES)*	*1. Génie logiciel avancé* *2. Recherche d'information* *3. Sujets en génie logiciel*
...	

- ***Cas 2*** **:** les cours suivis par l'utilisateur ne figurent pas ensemble dans aucune des conditions des règles. Le système lance une recherche des règles dont les conditions contiennent au moins un des cours suivis. Il recommande alors, tous les cours additionnels dans les conditions des règles retrouvés. Supposons qu'un étudiant ait suivi les deux cours « *Traitement des connaissances* » et « *Synthèse d'images* », le système trouve des règles contenant dans leurs conditions chacun des deux cours à part. Les règles utilisées et les cours recommandés sont représentés dans le tableau 4.2.

Tableau 4.2 : Les recommandations basées sur les règles de classification (*cas 2*)

Règles de classification	***Cours recommandés***
Si premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Concepts de bases de données avancées* **Alors** laboratoire = *Multimédia et tutoriels intelligents (HERON)*	*1. Concepts de bases de données avancées*
Si premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Intelligence artificielle : introduction* **et** troisième_cours = *Recherche d'information* **Alors** laboratoire = *Linguistique informatique (RALI)*	*1. Intelligence artificielle : introduction;* *2. Recherche d'information*
Si premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Génie logiciel avancé* **et** troisième_cours = *Recherche d'information* **et** quatrième_cours = *Sujets en génie logiciel* **Alors** laboratoire = *Génie logiciel (GEODES)*	*1. Génie logiciel avancé* *2. Recherche d'information* *3. Sujets en génie logiciel*
Si premier_cours = *Synthèse d'images* **et** deuxième_cours = *Traitement d'images* **Alors** laboratoire = *Traitement d'images (IMAGE)*	*1. Traitement d'images*
Si premier_cours = *Traitement d'images* **et** deuxième_cours = *Synthèse d'images* **et** troisième_cours = *Modélisation de solides* **et** quatrième_cours = *Algorithmes d'apprentissage* **Alors** laboratoire = *Traitement d'images (IMAGE)*	*1. Traitement d'images* *2. Modélisation de solides* *3. Algorithmes d'apprentissage*
...	

- ***Cas 3*** **:** les cours suivis par l'utilisateur ne figurent dans aucune des conditions des règles. Par conséquent, le système propose à l'utilisateur de sélectionner un laboratoire (un domaine) de recherche à partir d'une liste de laboratoires. Le critère de recherche cette fois-ci consiste à trouver toutes les règles ayant pour prédiction le laboratoire sélectionné. Ceci permet d'avoir comme recommandations tous les cours contenus dans les conditions des règles retrouvées. Le tableau 4.3 montre quelques règles liées au domaine des transports (le laboratoire *CRT*) choisi par un étudiant.

Conclusion

Dans les cas 1, le fait d'avoir des recommandations basées sur tous les cours suivis n'empêche pas l'utilisateur d'obtenir d'autres types de recommandations. Le système utilise, comme dans le cas 2, les règles dont les conditions contiennent au moins un des cours suivi, et recommande tous les cours additionnels dans les

conditions de toutes les règles retrouvées. D'autre par, l'utilisateur dans le cas 1 et le cas 2, peut aussi sélectionner un laboratoire de recherche. Grâce aux règles dont la prédiction est égale au laboratoire choisi, il reçoit des recommandations comme dans le cas 3.

Tableau 4.3 : Les recommandations basées sur les règles de classification (*cas 3*)

Règles de classification	***Cours recommandés***
Si premier_cours = *Techniques d'optimisation 3* **et** deuxième_cours = *Programmation en nombres entiers* **Alors** laboratoire = *Centre de recherche sur les transports (CRT)*	*1. Techniques d'optimisation 3* *2. Programmation en nombres entiers*
Si premier_cours = *Programmation en nombres entiers* **et** deuxième_cours = *Techniques d'optimisation 3* **et** troisième_cours = *Simulation : aspects stochastiques* **Alors** laboratoire = *Centre de recherche sur les transports (CRT)*	*1. Programmation en nombres entiers* *2. Techniques d'optimisation 3* *3. Simulation : aspects stochastiques*
Si premier_cours = *Sujets en optimisation* **et** deuxième_cours = *Algorithmes d'apprentissage* **Alors** laboratoire = *Centre de recherche sur les transports (CRT)*	*1. Sujets en optimisation* *2. Algorithmes d'apprentissage*
...	

- ***La recommandation par laboratoire de recherche***

  En revanche, si l'utilisateur n'a suivi aucun cours, il n'a qu'à choisir un laboratoire de recherche. Par conséquent, toutes les règles du laboratoire sélectionné seront utilisées et les cours contenus dans leurs conditions seront recommandés.

- ***La recommandation de l'utilisateur***

  RARE invite son utilisateur à suggérer aux autres les cours qu'il a appréciés auparavant. Ceux-ci sont sauvegardés dans la base des recommandations des utilisateurs (voir figure 4.5). Pour chaque cours, le système met à jour la popularité.

- ***Les recommandations des autres utilisateurs***

  Le système offre à l'utilisateur la possibilité de visualiser la liste de tous les cours recommandés par les autres utilisateurs. Sur cette liste, le système affiche, entre autre,

la popularité des cours et ceci en ordre décroissant. Ce type de recommandation est indépendant des règles utilisées par *RARE*.

4.3.1.2 La recommandation basée sur les règles d'association

Admettons cette foi-ci que *RARE* utilise le fonctionnement basé sur les règles d'association. Comme nous avons mentionné dans la section 4.2.3 (la modélisation), ces règles sont de type : « *Antécédent* ==> *Conséquent* ». Ce qui suit explique comment ce type de règles est utilisé par le système selon les différents scénarios de recommandation mentionnés dans le mode de fonctionnement basé sur les règles de classification.

- ***La recommandation par cours suivis***

Si un utilisateur a déjà suivi des cours, il peut les indiquer au système. Ce dernier effectue une recherche des règles dont les antécédents sont équivalents aux cours suivis. Les conséquents des règle retrouvées détermineront les cours que *RARE* recommandera. Plusieurs cas peuvent se présenter :

➢ ***Cas 1*** **:** le système trouve des règles dont les antécédents sont équivalents aux cours suivis par l'utilisateur. Il recommande alors, tous les cours des conséquents des règles retrouvées. Admettons qu'un étudiant ait suivi le cours « *Traitement des connaissances* ». Par le biais des règles d'association retrouvées, le système arrivera à lui recommander les cours suivants (tableau 4.4) :

Tableau 4.4 : Les recommandations basées sur les règles d'association (*cas 1*)

Règles d'association	***Cours recommandés***
Traitement des connaissances ==> Concepts de bases de données avancées	*1. Concepts de bases de données avancées*
Traitement des connaissances ==> Concepts de bases de données avancées **et** *Génie logiciel avancé*	*1. Concepts de bases de données avancées* *2.. Génie logiciel avancé*
Traitement des connaissances ==> Intelligence artificielle: introduction **et** *Téléinformatique*	*1. Intelligence artificielle : introduction;* *2. Téléinformatique*
...	

- ***Cas 2*** **:** les cours suivis par l'utilisateur ne paraissent pas ensemble dans les antécédents des règles. Le système cherche les règles dont les antécédents contiennent au moins un des cours suivis, et recommande tous les cours contenus dans les conséquents des règles trouvées. Pour un étudiant qui a suivi les deux cours « *Traitement des connaissances* » et « *Synthèse d'images*», par exemple, le système trouve des règles contenant dans leurs antécédents l'un des deux cours (voir tableau 4.5).

Tableau 4.5 : Les recommandations basées sur les règles d'association (*cas 2*)

Règles d'association	***Cours recommandés***
Traitement des connaissances ==> Concepts de bases de données avancées	*1. Concepts de bases de données avancées*
Traitement des connaissances ==> Génie logiciel avancé	*1. Génie logiciel avancé*
Traitement des connaissances ==> Intelligence artificielle: introduction **et** *Téléinformatique*	*1. Intelligence artificielle : introduction;* *2. Téléinformatique*
...	
Synthèse d'images ==> Traitement d'images	*1. Traitement d'images*
Synthèse d'images ==> Traitement d'images **et** *Modélisation de solides* **et** *Algorithmes d'apprentissage*	*1. Traitement d'images* *2. Modélisation de solides* *3. Algorithmes d'apprentissage*
...	

- ***Cas 3*** **:** le système n'arrive pas à trouver les cours suivis par l'utilisateur dans aucun des antécédents des règles. Alors, l'utilisateur est invité à choisir un laboratoire de recherche. Toutes les règles générées à partir des données des étudiants appartenant au laboratoire choisi sont utilisées et les cours contenus dans leurs conséquents sont recommandés. Le tableau 4.6 montre quelques règles utilisées dans la recommandation ainsi que les cours suggérés par celles-ci à un étudiant qui a choisi le domaine des transports (le laboratoire *CRT*).

Conclusion

Dans les cas 1 (où le système trouve des règles dont les antécédents sont équivalents aux cours suivis par l'utilisateur), le système suggère à l'utilisateur des

recommandations supplémentaires en utilisant, comme dans le cas 2, les règles dont les antécédents contiennent au moins un des cours suivi. Il recommande ensuite tous les cours des conséquents des règles retrouvées. Dans les cas 1 et 2, les utilisateurs sont sollicités à sélectionner un laboratoire de recherche pour avoir des recommandations comme dans le cas 3 suscité.

Tableau 4.6 : Les recommandations basées sur les règles d'association (*cas 3*)

Règles d'association	***Cours recommandés***
Techniques d'optimisation 3 ==> Programmation en nombres entiers	*1. Programmation en nombres entiers*
Programmation en nombres entiers ==> Techniques d'optimisation 3 **et** *Simulation : aspects stochastiques*	*1.Techniques d'optimisation 3* *2. Simulation : aspects stochastiques*
Techniques d'optimisation 3 **et** *Simulation : aspects stochastiques ==> Sujets en optimisation* **et** *Algorithmes d'apprentissage*	*1. Sujets en optimisation* *2. Algorithmes d'apprentissage*
...	

- ***La recommandation par laboratoire de recherche***

  Si l'utilisateur n'a suivi aucun cours, nous nous retrouvons avec aucune règle en réponse à la recommandation basée sur les cours suivis. Pour faire face à ce problème, le système ne requiert que le nom d'un laboratoire de recherche (voir le cas 3). Par conséquent, toutes les règles du laboratoire sélectionné sont utilisées et les cours contenus dans les conséquents des règles sont recommandés.

- ***La recommandation de l'utilisateur***

  Cette fonctionnalité est la même dans les deux modes de fonctionnement, le fonctionnement basé sur les règles de classification et le fonctionnement basé sur les règles d'association. Ainsi, le système invite l'utilisateur à recommander aux autres les cours qu'il juge intéressants. Les cours recommandés sont stockés dans la base des recommandations des utilisateurs (voir figure 4.5), et leurs popularités sont incrémentées.

- ***Les recommandations des autres utilisateurs***

  À tout moment, l'utilisateur est sollicité à consulter les cours recommandés par tous les utilisateurs du système et cela dans les deux modes de fonctionnement de *RARE*. Lors de leur affichage, les cours sont triés par ordre décroissant de popularité.

4.3.2 La mise à jour des bases de données

Après chaque recommandation, la mise à jour des bases de données suivantes est nécessaire.

4.3.2.1 La mise à jour de la base des utilisateurs

À la fin de la recommandation, *RARE* sauvegarde les cours suivis par l'utilisateur ainsi que les cours qui lui ont été recommandés dans la base des utilisateurs. Chacun des cours recommandés doit être associé à chacune des règles utilisées dans sa recommandation. Un exemple d'association « *cours_ règle* » est le suivant :

Exemple 5 :

Prenons l'exemple de *Joe*, un étudiant ayant suivi le cours « *Traitement des connaissances* ». Afin de lui suggérer des recommandations, le système dans un fonctionnement basé sur les règles d'association, utilise les règles suivantes (les règles du tableau 4.4) :

Règles d'association

- *Traitement des connaissances ==> Concepts de bases de données avancées* ***(R50)***
- *Traitement des connaissances ==> Concepts de bases de données avancées* ***et*** *Génie logiciel avancé* ***(R53)***
- *Traitement des connaissances ==> Intelligence artificielle: introduction* ***et*** *Téléinformatique* ***(R61)***

Supposons que les codes des règles d'association précédentes sont respectivement *R50*, *R53* et *R61*, et étant donné que « *IFT6243* » est le code du cours « *Concepts de*

bases de données avancées », « *IFT6310* » est le code du cours « *Génie logiciel avancé* », « *IFT6330* » est le code du cours « *Intelligence artificielle: introduction* » et « *IFT6320* » est le code du cours « *Téléinformatique* ». Le système associe donc, les cours recommandés aux codes des règles utilisées comme suit : « *R50_IFT6243* », « *R53_IFT6243* », « *R53_IFT6310* », «*R61_IFT6330* » *et* « *R61_IFT6320* ». Nous pouvons constater que le cours « *Concepts de bases de données avancées (IFT6243)* » est recommandé deux fois, et cela par le biais des règles *R50* et *R53*. Ce cours doit paraître une seule fois dans la liste des recommandations pour l'utilisateur *Joe*. Par contre, il doit être associé à chacun des codes des deux règles (*R50* et *R53*) utilisées dans sa recommandation. Les différentes associations « *cours_règle* » sont sauvegardées dans le champ « *Cours_recommandés* » de l'utilisateur *Joe* dans la base des utilisateurs comme le tableau 4.7 l'illustre. Comme il s'agit de la première utilisation du système par *Joe*, sa fidélité est initialisée à 1.

Tableau 4.7 : La mise à jour de la base des utilisateurs

Nom d'utilisateur	***Mot de passe***	***Adresse électronique***	***Cours suivis***	***Cours recommandés***	***Fidélité***
...					
Joe	*******	[redacted]	*IFT6261 (Traitement des connaissances)*	*R50_IFT6243, R53_IFT6243, R53_IFT6310, R61_IFT6330, R61_IFT6320*	1
...					

Le système effectue les mêmes opérations dans un fonctionnement basé sur les règles de classification.

Règles de classification

- **Si** premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Concepts de bases de données avancées* **Alors** laboratoire = *Multimédia et tutoriels intelligents (HERON)* ***(R20)***

- **Si** premier_cours = *Intelligence artificielle: introduction* **et** deuxième_cours = *Traitement des connaissances* **et** troisième_cours = *Recherche d'information* **Alors** laboratoire = *Linguistique informatique (RALI)* ***(R33)***
- **Si** premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Génie logiciel avancé* **et** troisième_cours = *Recherche d'information* **et** quatrième_cours = *Sujets en génie logiciel* **Alors** laboratoire = *Génie logiciel (GEODES)* ***(R80)***

Les cours recommandés à *Joe* dans ce cas sont : « *Concepts de bases de données avancées (IFT6243)* », « *Intelligence artificielle: introduction (IFT6330)*», « *Recherche d'information (IFT6255)* », « *Génie logiciel avancé* (*IFT6310*) » et « *Sujets en génie logiciel (IFT6251)* ». En supposant que les codes des règles sont respectivement : *R20*, *R33* et *R80*, les associations « *cours_règle* » sont : « *R20_IFT6243* », « *R33_IFT6330* », « *R33_IFT6255* », « *R80_IFT6310* », « *R80_IFT6255* » et « *R80_IFT6252* ». La mise à jour de la base des utilisateurs effectuée dans le fonctionnement basé sur les règles d'association est aussi accomplie dans le fonctionnement basé sur les règles de classification.

4.3.2.2 La mise à jour de la base des évaluations des cours recommandés

L'interaction du système avec la base des évaluations des cours recommandés est illustrée dans la figure 4.7. Cette base sert à conserver les différentes données nécessaires dans le calcul d'une valeur nommée *poids de cours*, et que nous associons à chaque cours recommandé. Cette valeur permet au système de maintenir ce cours dans les règles qui l'ont recommandé ou de l'y supprimer. Le maintien et la suppression d'un cours se fait par comparaison de son poids avec un *seuil d'acceptation*, et ceci après un *nombre d'évaluations requis*. Les deux paramètres, seuil d'acceptation et nombre d'évaluations requis, sont prédéterminés et sauvegardés dans la *base des paramètres* que nous expliquons plus tard. La suppression et l'ajout des cours dans les règles sont expliqués dans la mise à jour des règles (voir section 4.3.4).

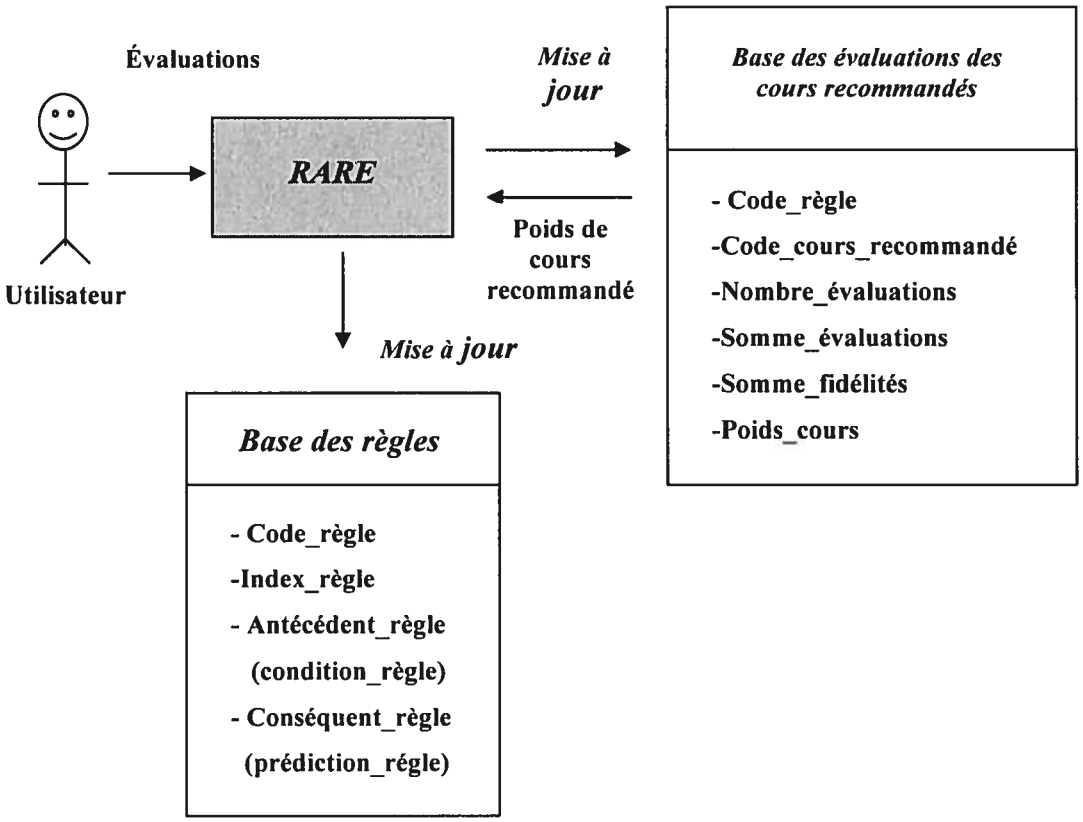


Figure 4.7 : Base des évaluations des cours recommandés

Après avoir associé tous les cours recommandés aux règles utilisées (voir section 4.3.2.1), le système consulte la base des évaluations des cours recommandés à la recherche de toutes les associations « *cours_règle* ». Si un cours recommandé associé à une règle ne figure pas dans cette base, le système l'y insère. Vu que ce cours n'est pas encore évalué, le système initialise à 0 le nombre et la somme de ses évaluations, la somme des fidélités des utilisateurs qui l'ont évalué, ainsi que son poids.

Exemple 6 :

Dans l'exemple 5 ci-dessus (voir section 4.3.2.1), le système a associé le cours recommandé « *Concepts de bases de données avancées (IFT6243)* aux deux règles d'association *R50* et *R53* comme suit : «*R50_IFT6243* », «*R53_IFT6243*». Ainsi, le système recherche ces deux associations dans la base des évaluations des cours recommandés et ne les trouve pas. Il les insère et met à 0 les autres valeurs.

Au cas où une association « *cours_règles* » existe déjà dans la base des évaluations des cours recommandés, aucune mise à jour de cette base de données n'est nécessaire par rapport à cette association.

4.3.3 Les algorithmes de recommandation

Les deux algorithmes présentés ci-dessous (figure 4.8 et figure 4.9) récapitulent toutes les opérations effectuées par *RARE* dans son processus de recommandation basé respectivement sur les règles de classification et les règles d'association.

L'algorithme de la recommandation basée sur les règles de classification

RARE demande à son utilisateur *U* de sélectionner un choix ;

Si *U a suivi des cours* **alors**

  Si *RARE trouve des règles dont les conditions contiennent tous les cours suivis* **alors**

    RARE recommande les cours additionnels dans les conditions des règles retrouvées;

    Pour *tous les cours recommandés* **faire**

      Si *un cours recommandé ne figure pas dans la base des évaluations des cours recommandés* **alors**

        RARE l'insère dans cette base en l'associant à chacune des règles utilisées dans sa recommandation;

        RARE initialise à 0 le nombre et la somme des évaluations, la somme des fidélités ainsi que le poids du cours;

      Fin

    Fin

  Sinon (* *RARE ne trouve aucune règle dont la condition contient tous les cours suivis* *)

    Si *RARE trouve des règles dont les conditions contiennent au moins un des cours suivis* **alors**

      RARE recommande tous les cours additionnels dans les conditions des règles retrouvées;

    Sinon (**les cours suivis ne figurent dans aucune des conditions des règles***)

      RARE propose à *U* de sélectionner un laboratoire de recherche;

      RARE utilise les règles dont la prédiction est égale au laboratoire sélectionné et recommande tous les cours contenus dans leurs conditions;

    Fin

  Fin

Sinon (**U n'a suivi aucun cours***)

  RARE propose à *U* de sélectionner un laboratoire de recherche;

  RARE utilise les règles dont la prédiction est égale au laboratoire sélectionné et recommande tous les cours contenus dans leurs conditions;

Fin

RARE propose à *U* de recommander à d'autres utilisateurs les cours qu'il a appréciés auparavant;

RARE permet à *U* de consulter à tout moment les cours recommandés par les autres utilisateurs, triés par ordre décroissant de popularité;

RARE sauvegarde le profil de *U* dans la base des utilisateurs ;

Figure 4.8 : L'algorithme de la recommandation basée sur les règles de classification

L'algorithme de la recommandation basée sur les règles d'association

RARE demande à son utilisateur *U* de sélectionner un choix ;

Si *U a suivi quelques cours* **alors**

  Si *RARE trouve des règles dont les antécédents sont équivalents aux cours suivis* **alors**

    RARE recommande les cours contenus dans les conséquents des règles;

    Pour *tous les cours recommandés* **faire**

      Si *un cours recommandé ne figure pas dans la base des évaluations des cours recommandés* **alors**

        RARE l'insère dans cette base en l'associant à chacune des règles utilisées dans sa recommandation;

        RARE initialise à 0 le nombre et la somme des évaluations, la somme des fidélités ainsi que le poids du cours;

      Fin

    Fin

  Sinon (**aucune règle dont l'antécédent est équivalent aux cours suivis n'est trouvée**)

    Si *RARE trouve des règles dont les antécédents contiennent au moins un des cours suivis* **alors**

      RARE recommande tous les cours contenus dans les conséquents de ces règles;

    Sinon (**les cours suivis ne figurent dans aucun des antécédents des règles**)

      RARE propose à *U* de sélectionner un laboratoire de recherche;

      RARE utilise les règles générées au niveau du laboratoire choisi et recommande tous les cours contenus dans leurs conséquents;

    Fin

  Fin

Sinon (**U n'a suivi aucun cours**)

  RARE propose à *U* de sélectionner un laboratoire de recherche;

  RARE utilise les règles générées au niveau du laboratoire choisi et recommande tous les cours contenus dans leurs conséquents;

Fin

RARE propose à *U* de recommander à d'autres utilisateurs les cours qu'il a appréciés auparavant;

RARE permet à *U* de consulter à tout moment les cours recommandés par les autres utilisateurs, triés par ordre décroissant de popularité;

RARE sauvegarde le profil de *U* dans la base des utilisateurs ;

Figure 4.9 : L'algorithme de la recommandation basée sur les règles d'association

4.3.4 La mise à jour des règles

RARE effectue un suivi des recommandations qu'il fournit à ses utilisateurs. Lors de l'ouverture d'une session, il identifie l'utilisateur fidèle à travers son profil sauvegardé dans la base des utilisateurs (voir figure 4.6). Pour avoir des recommandations basées sur les cours suivis, l'utilisateur doit indiquer tous les cours qu'il a pris après sa dernière utilisation du système. L'utilisateur est également sollicité à évaluer chacun des cours suivis. L'échelle d'évaluation varie entre 1 et 10. Tandis que la valeur 1 représente un cours non apprécié par l'utilisateur, la valeur 10 indique la parfaite satisfaction de l'utilisateur à propos d'un cours.

Le système reçoit en entrées les évaluations des utilisateurs pour calculer les poids des cours. Ainsi, il stocke ces informations et suit les changements des appréciations des cours par les utilisateurs afin d'adapter les règles et les mettre à jour. Ceci repose sur le maintien, la suppression et l'ajout des cours dans les règles. Au fur et à mesure que les évaluations progressent, le système parvient à améliorer la recommandation. La mise à jour porte sur certains paramètres prédéfinis qui sont *le nombre d'évaluations requis* et *le seuil d'acceptation d'un cours*. Le premier paramètre constitue le nombre minimum de fois qu'un cours a été évalué pour que le système déclenche la vérification de son poids. D'autre part, le deuxième paramètre consiste en une valeur fixe à laquelle le poids d'un cours est comparé pour décider de son maintien ou de sa suppression des règles. Ainsi, la comparaison permet d'éliminer, des règles, tout cours ayant atteint un poids inférieur au seuil. Les deux paramètres sont maintenus dans la base des paramètres illustrée dans la figure 4.8. Cependant, ils peuvent être modifiés par l'administrateur du système afin de les convenir à des besoins et des conditions futures de ce travail. Il faut noter que dans notre application et suivant les conseils des professeurs expérimentés, nous avons fixé leurs valeurs à :

- Le nombre d'évaluations requis: ***NBRE_ÉVAL_REQUIS = 10***
- Le seuil d'acceptation d'un cours: ***SEUIL_ACC = 4***

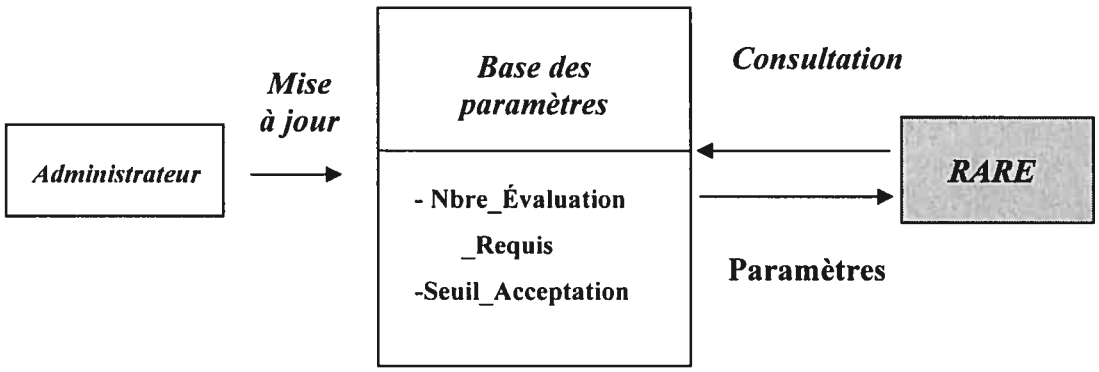


Figure 4.8 : Base des paramètres

Après avoir parcouru tous les cours suivis et évalués par un utilisateur, *RARE* arrive à différencier entre deux types de cours suivis : les cours qui ont été recommandés à cet utilisateur par le système, et les cours non recommandés. Ceux-ci sont désignés par le nom « *cours temporaires* ». Le traitement des deux types de cours, recommandés et temporaires, est défini dans ce qui suit.

4.3.4.1 Les cours recommandés

À chaque fois qu'un utilisateur évalue un cours, le système vérifie à travers son profil s'il s'agit d'un cours qui lui a été recommandé. Si c'est le cas, le système calcule le nouveau poids de ce cours en utilisant les équations 1 et 2 ci-dessous. Il se sert, dans son calcul, de l'évaluation courante du cours ainsi que de ses anciennes évaluations sauvegardées dans la base des évaluations des cours recommandés.

$$\textit{Somme des évaluations du cours} = \sum_{\text{utilisateur}} (\text{Évaluation d'utilisateur} \times \text{fidélité d'utilisateur}) \qquad \textit{Équation (1)}$$

$$\textit{Poids du cours} = \frac{\text{Somme des évaluations du cours}}{\sum_{\text{utilisateur}} (\text{fidélité d'utilisateur})} \qquad \textit{Équation (2)}$$

À partir du nombre d'évaluations requis (***NBRE_ÉVAL_REQUIS = 10***), le poids de ce cours est comparé au seuil d'acceptation (***SEUIL_ACC = 4***). S'il est inférieur à ce dernier, il sera supprimé des règles utilisées dans sa recommandation ainsi que de la base des évaluations des cours recommandés. Dans l'exemple qui vient, nous présentons en détail ces opérations.

Exemple 7 :

Lors de sa première utilisation du système, l'utilisateur *Joe* (cité dans l'exemple 5) a suivi le cours « *Traitement des connaissances (IFT6261)* ». Pour lui fournir des recommandations, le système a utilisé les règles d'association suivantes :

- *Traitement des connaissances ==> Concepts de bases de données avancées (IFT6243)* ***(R50)***
- *Traitement des connaissances ==> Concepts de bases de données avancées (IFT6243)* ***et*** *Génie logiciel avancé (IFT6310)* ***(R53)***
- *Traitement des connaissances ==> Intelligence artificielle: introduction (IFT6330)* ***et*** *Téléinformatique (IFT6320)* ***(R61)***

Comme le montre l'exemple 6, le système a sauvegardé les cours recommandés à *Joe* dans la base des utilisateurs (voir tableau 4.7) ainsi que dans la base des évaluations des cours recommandés en les associant aux règles utilisées dans leurs recommandations. Sur la liste des recommandations, figure le cours « *Concepts de bases de données avancées (IFT6243)* » qui a été associé aux deux règles *R50* et *R53*.

Admettons que *Joe* sollicite les suggestions du système une deuxième fois. Les cours qu'il a suivis cette fois-ci sont «*Concepts de bases de données avancées* (*IFT6243*)» et « *Algorithmes d'apprentissage* (*IFT6266*) ». Il les évalue respectivement à 9 et à 7. Le système compare ces cours avec les cours recommandés enregistrés dans le profil de *Joe* dans la base des utilisateurs. Comme le cours «*Concepts de bases de données avancées* (*IFT6243*)» lui a été recommandé, et a été associé aux règles *R50* et *R53*, le système incrémente par 1 le nombre d'évaluations de ce cours dans la base des évaluations des cours recommandés comme le montre le tableau 4.8. Le

nombre d'évaluations qui était égal à 0 prend la valeur 1. Lors de sa première utilisation du système, la fidélité de *Joe* a été initialisée à 1. En tenant compte de cette valeur, le système calcule la somme des évaluations ainsi que le poids du cours comme illustré ci-dessous. Il incrémente, par la suite, la fidélité de *Joe* qui prend la valeur 2.

$$\textbf{\textit{Somme des évaluations du cours}} = \sum_{\text{utilisateur}} (\text{Évaluation d'utilisateur} \times \text{ fidélité d'utilisateur})$$

$$\textbf{\textit{Somme des évaluations du cours}} = 0 + 1 \times 9 = 9$$

$$\textbf{\textit{Poids du cours}} = \frac{\text{Somme des évaluations du cours}}{\sum_{\text{utilisateur}} (\text{fidélité d'utilisateur})}$$

$$\textbf{\textit{Poids du cours}} = \frac{9}{0 + 1} = 9$$

Avant de mettre à jour le poids d'un cours recommandé dans la base des évaluations des cours recommandés, le nombre de ses évaluations est comparé à *NBRE_ÉVAL_REQUIS*. S'il est inférieur à ce nombre, le système ne vérifie pas son poids. Il effectue par contre toutes les mises à jour nécessaires des valeurs dans la base des évaluations à savoir le nombre et la somme des évaluations, la somme des fidélités des utilisateurs ayant évalué ce cours, ainsi que son poids actuel. En contre partie, il est nécessaire de contrôler le poids si le nombre des évaluations est supérieur ou égal à *NBRE_ÉVAL_REQUIS*. Dans ce cas, un poids supérieur ou égal à *SEUIL_ACC* permet le maintien du cours dans les règles utilisées. Le cas contraire permet de supprimer ce cours des règles ainsi que de la base des évaluations des cours recommandés. Dans l'exemple courant (exemple 7), le nombre d'évaluations du cours recommandé « *Concepts de bases de données avancées (IFT6243)* » est égal à 1. Ainsi, aucune vérification de poids n'est effectuée et les mises à jour requises dans la base des évaluations des cours recommandés sont exécutées.

Tableau 4.8 : La mise à jour des évaluations des cours recommandés

Code règle	...	*R50*	...	*R53*	...
Code cours recommandé		*IFT6243 (Concepts de bases de données avancées)*		*IFT6243 (Concepts de bases de données avancées)*	
Nombre des évaluations		0+1=1		0+1=1	
Somme des évaluations		$0 + 1 \times 9 = 9$		$0 + 1 \times 9 = 9$	
Somme des fidélités		0+1=1		0+1=1	
Poids du cours		9/1=9		9/1=9	

Supposons maintenant que le cours « *Génie logiciel avancé (IFT6310)* » recommandé par la règle d'association ***R53*** : (« *Traitement des connaissances* ==> *Concepts de bases de données avancées (IFT6243)* ***et*** *Génie logiciel avancé (IFT6310)* ») ait été suivi et évalué par *Joe*, et que le système met à jour son poids et le trouve inférieur à *SEUIL_ACC*. Le système vérifie également le nombre des utilisateurs qui ont évalué ce cours et se rend compte qu'il a atteint le nombre d'évaluations requis (*NBRE_ÉVAL_REQUIS =10)*. Ce cours doit être supprimé de la base des évaluations des cours recommandés ainsi que de la règle ***R53***. Par conséquent, la règle *R53* suivante : « *Traitement des connaissances* ==> *Concepts de bases de données avancées (IFT6243)* ***et*** *Génie logiciel avancé (IFT6310)* ***(R53)*** » devient :

- « *Traitement des connaissances* ==> *Concepts de bases de données avancées (IFT6243)* ***(R53)*** »

Désormais, le cours « *Génie logiciel avancé (IFT6310)* » ne sera plus recommandé par la règle ***R53***. Toutefois il peut être recommandé par d'autres règles.

Dans le cas des règles de classification, la suppression d'un cours recommandé s'effectue au niveau des conditions des règles. Après avoir supprimé le cours « *Génie logiciel avancé (IFT6310)* » de la règle de classification suivante : « **Si** premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Génie logiciel avancé* **et** troisième_cours = *Recherche d'information* **et** quatrième_cours = *Sujets en génie*

logiciel **Alors** laboratoire = *Génie logiciel (GEODES)* ***(R80)*** », cette règle se transforme en :

- **Si** premier_cours = *Traitement des connaissances* **et** deuxième_cours = *Recherche d'information* **et** troisième_cours = *Sujets en génie logiciel* **Alors** laboratoire = *Génie logiciel (GEODES)* ***(R80)***

4.3.4.2 Les cours temporaires

Les cours non recommandés par le système et suivis par les utilisateurs sont désignés par le nom « *cours temporaires* ». Le système tient compte des évaluations de ces cours et les sauvegarde dans une base de données nommée *la base des évaluations des cours temporaires.* Comme nous pouvons remarquer dans la figure 4.9, cette base est fondée sur le même principe de la base des évaluations des cours recommandés. Cependant, elle stocke les différentes données requises dans le calcul des poids des cours temporaires. Le système consulte le profil de l'utilisateur pour connaitre les règles qui ont été utilisées afin de lui fournir des recommandations lors de sa dernière utilisation. Chaque cours temporaire que l'utilisateur a préféré suivre doit être associé à chacune de ces règles, puis sauvegardé dans la base des évaluations des cours temporaires. En se servant des équations 1 et 2 présentées dans la section précédente, le système calcule aussi les poids de tous ces cours. L'optique est d'intégrer un cours temporaire dans les règles, auxquelles il a été associé, une fois son poids atteint le *seuil d'acceptation* (*SEUIL_ACC=4*) et à condition que le nombre des utilisateurs qui l'ont évalué soit égal à *NBRE_ÉVAL_REQUIS* (voir l'exemple 8).

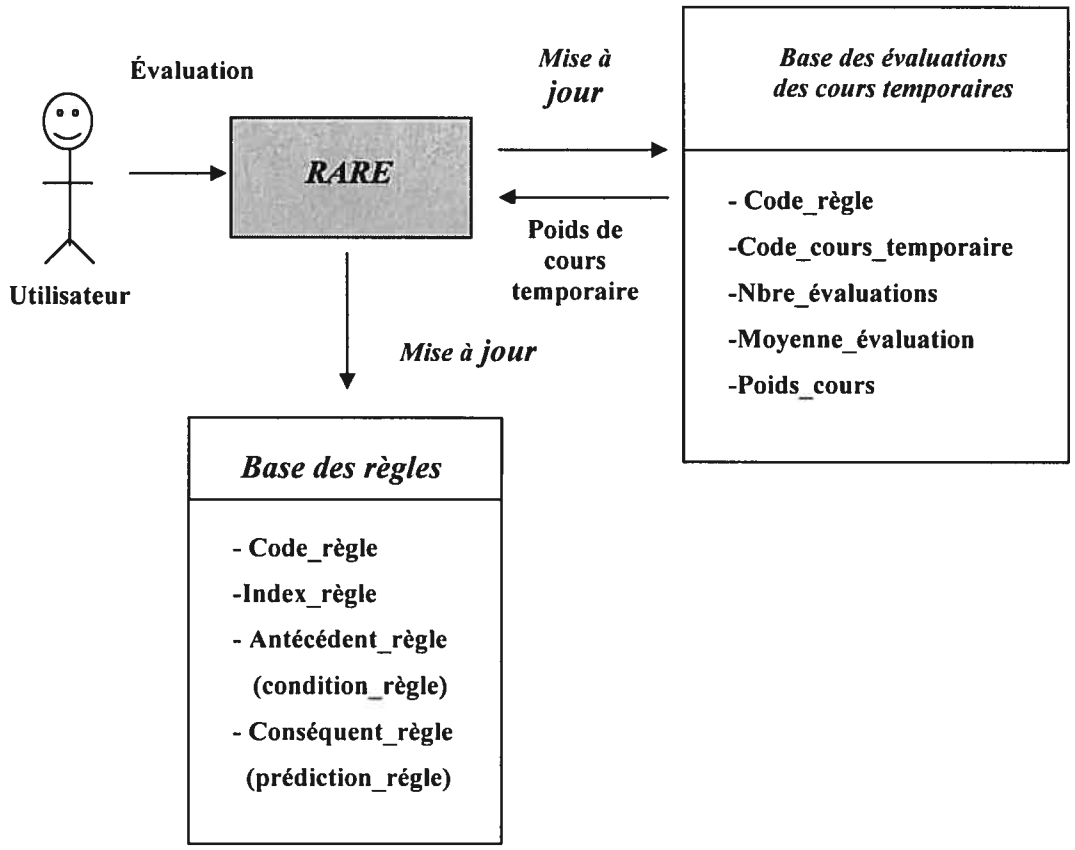


Figure 4.9 : Base des évaluations des cours temporaires

Exemple 8 :

Lors de sa dernière utilisation, *Joe* a indiqué au système qu'il a suivi les deux cours « *Concepts de bases de données avancées (IFT6243)* » et « *Algorithmes d'apprentissage (IFT6266)* ». Contrairement au premier cours, *Joe* a préféré suivre le cours « *Algorithmes d'apprentissage (IFT6266)* » qui n'a pas été recommandé. Par conséquent, *IFT6266* représente un cours temporaire. Le système vérifie les règles utilisées pour la production des recommandations à *Joe*, elles sont ***R50***, ***R53*** et ***R61***. Le système vérifie si les associations du cours « *Algorithmes d'apprentissage (IFT6266)* » avec les règles *R50*, *R53* et *R61* figurent dans la base des évaluations des cours temporaires illustrée dans le tableau 4.9.

Tableau 4.9 : Table des évaluations des cours temporaires

Code de la règle	...	*R50*	...	*R53*	...	*R61*	...
Cours temporaire		IFT6266		IFT6266		IFT6266	
Nombre des évaluations		9		9		9	
Somme des évaluations		149		149		149	
Somme des fidélités		19		19		19	
Poids du cours		7.84		7.84		7.84	

Les trois associations sont déjà présentes. Le système incrémente ainsi le nombre d'évaluations de ce cours qui était égal à 9. Comme illustré dans le tableau 4.9, la somme des fidélités des utilisateurs qui ont évalué ce cours est égale à 19 et la somme de leurs évaluations est égale à 149. Comme *Joe* a évalué ce cours à 7 (voir exemple 7), le calcul du nouveau poids est illustré ci-dessous.

Somme des évaluations du cours = $\sum_{\text{utilisateur}}$ (Évaluation d'utilisateur × fidélité d'utilisateur)

Somme des évaluations = 149 + **1×7 = 156**

$$\textit{Poids du cours} = \frac{\text{Somme des évaluations du cours}}{\sum_{\text{utilisateur}} (\text{fidélité d'utilisateur})}$$

$$\textit{Poids du cours} = \frac{\mathbf{156}}{\mathbf{19+1}} = \frac{\mathbf{156}}{\mathbf{20}} = \mathbf{7.8}$$

Le système procède à la vérification du nombre d'évaluations avant les mises à jour dans la base des évaluations des cours temporaires. Dans l'exemple actuel, le nombre d'évaluations du cours temporaire « *Algorithmes d'apprentissage (IFT6266)* » a atteint la valeur 10. Le système constate que son poids est égal à 7.8 qui est supérieur à *SEUIL_ACC*. Ainsi, ce cours associé aux règles *R50*, *R53* et *R61* est supprimé de la base des évaluations des cours temporaires. En revanche, il est intégré au niveau de la base des règles et les règles *R50*, *R53* et *R61* deviennent :

- *Traitement des connaissances (IFT6261) ==> Concepts de bases de données avancées (IFT6243)* **et** ***Algorithmes d'apprentissage (IFT6266)*** ***(R50)***
- *Traitement des connaissances (IFT6261) ==> Concepts de bases de données avancées (IFT6243)* **et** ***Algorithmes d'apprentissage (IFT6266)*** ***(R53)***
- *Traitement des connaissances (IFT6261) ==> Intelligence artificielle: introduction (IFT6330)* **et** *Téléinformatique (IFT6320)* **et** ***Algorithmes d'apprentissage (IFT6266)*** ***(R61)***

Dans le cas des règles de classification, l'ajout du cours temporaire « *Algorithmes d'apprentissage (IFT6266)* » s'exécute dans les conditions des règles comme suit :

- **Si** premier_cours = *Traitement des connaissances (IFT6261)* **et** deuxième_cours = *Concepts de bases de données avancées (IFT6243)* **et** troisième_cours = ***Algorithmes d'apprentissage (IFT6266)*** **Alors** laboratoire = *Multimédia et tutoriels intelligents (HERON)* ***(R20)***
- **Si** premier_cours = *Intelligence artificielle: introduction (IFT6330)* **et** deuxième_cours = *Traitement des connaissances (IFT6261)* **et** troisième_cours = *Recherche d'information (IFT6255)* **et** quatrième_cours = ***Algorithmes d'apprentissage (IFT6266)*** **Alors** laboratoire = *Linguistique informatique (RALI)* ***(R33)***
- **Si** premier_cours = *Traitement des connaissances (IFT6261)* **et** deuxième_cours = *Recherche d'information (IFT6255)* **et** troisième_cours = *Sujets en génie logiciel (IFT6251)* **et** quatrième_cours = ***Algorithmes d'apprentissage (IFT6266)*** **Alors** laboratoire = *Génie logiciel (GEODES)* ***(R80)***

4.3.5 Les algorithmes de mise à jour des règles

Les deux algorithmes suivants représentent toutes les opérations que *RARE* exécute afin de mettre à jour et d'enrichir les règles selon le mode de fonctionnement basé sur les règles de classification et le fonctionnement basé sur les règles d'association.

L'algorithme de la mise à jour des règles de classification

RARE reconnaît un utilisateur fidèle *U* en regardant son profil;
RARE détecte si *U* a suivi les cours qu'il lui a recommandés;
Pour *tous les cours suivis par U* **faire**
    RARE demande à *U* l'évaluation du cours;
    Si *le cours a été recommandé* **alors**
        RARE met à jour son poids (équations 1, 2) dans la base des évaluations des cours recommandés;
        Si *après un nombre d'évaluations requis (NBRE_ÉVAL_Requis), le poids d'un cours est inférieur au seuil d'acceptation (SEUIL_ACC)* **alors**
            RARE élimine ce cours des conditions des règles utilisées dans sa recommandation ;
            RARE élimine ce cours de la base des évaluations des cours recommandés ;
        Fin
    Sinon (**le cours n'a pas été recommandé* *)
        Si *ce cours ne figure pas dans la base des évaluations des cours temporaires* **alors**
            RARE l'insère dans cette base en l'associant à chacune des règles utilisées dans les recommandations précédentes fournies à *U*;
            RARE calcule le poids du cours (équations 1, 2) et le lui attribue dans la base des évaluations des cours temporaires;
        Sinon (**le cours figure dans la base des évaluations des cours temporaires**)
            RARE met à jour son poids (équations 1, 2) dans la base des évaluations des cours temporaires;
            Si *après un nombre d'évaluations requis (NBRE_ÉVAL_Requis), le poids d'un cours Temporaire atteint le seuil d'acceptation (SEUIL_ACC)* **alors**
                RARE ajoute ce cours aux conditions des règles auxquelles il a été associé;
                RARE élimine ce cours de la base des évaluations des cours temporaires;
            Fin
        Fin
    Fin
Fin

Figure 5.7 : L'algorithme de la mise à jour des règles de classification

L'algorithme de la mise à jour des règles d'association

RARE reconnaît un utilisateur fidèle *U* en regardant son profil;

RARE détecte si *U* a suivi les cours qu'il lui a recommandés;

Pour *tous les cours suivis par U* **faire**

RARE demande à *U* l'évaluation du cours;

Si *le cours a été recommandé* **alors**

RARE met à jour son poids (équations 1, 2) dans la base des évaluations des cours recommandés;

Si *après un nombre d'évaluations requis (NBRE_ÉVAL_Requis), le poids d'un cours est inférieur au seuil d'acceptation (SEUIL_ACC)* **alors**

RARE élimine ce cours des règles utilisées dans sa recommandation ;

RARE élimine ce cours de la base des évaluations des cours recommandés ;

Fin

Sinon (**le cours n'a pas été recommandé* *)

Si *ce cours ne figure pas dans la base des évaluations des cours temporaires* **alors**

RARE l'insère dans cette base en l'associant à chacune des règles utilisées dans les recommandations précédentes fournies à *U*;

RARE calcule le poids du cours (équations 1, 2) et le lui attribue dans la base des évaluations des cours temporaires;

Sinon (**le cours figure dans la base des évaluations des cours temporaires**)

RARE met à jour son poids (équations 1, 2) dans la base des évaluations des cours temporaires;

Si *après un nombre d'évaluation requis (NBRE_ÉVAL_Requis), le poids d'un cours temporaire atteint le seuil d'acceptation (SEUIL_ACC)* **alors**

RARE ajoute ce cours aux règles auxquelles il a été associé;

RARE élimine ce cours de la base des évaluations des cours temporaires;

Fin

Fin

Fin

Fin

Figure 5.7 : l'algorithme de la mise à jour des règles d'association

4.4 Conclusion

Au cours de ce chapitre, nous avons présenté la conception et l'architecture de *RARE*, un système de recommandation de cours basé sur le forage de données. *RARE* analyse dans sa phase hors ligne le comportement relatif aux choix de cours effectués par les anciens étudiants, et ceci afin d'extraire des règles qui décrivent les cours qu'ils ont suivis. Ces règles permettent de prévoir des recommandations de cours pour les nouveaux étudiants. Ainsi, le forage de données fournit au système des connaissances utiles pour la production des recommandations et, par conséquent, la prévention du problème de démarrage à froid. En plus, le forage de données qui constitue la phase hors ligne du système, aide d'une part, à dépasser le problème de données à grande échelle (*scalability*) car le traitement d'un volume de données plus grand ne nécessite pas plus de ressources. D'autre part, plus la quantité de données est grande meilleure est la qualité des connaissances extraites. Ensuite, *RARE* combine l'approche de forage de données avec l'approche de collaboration des étudiants. Celle-ci se fait en ligne et permet aux utilisateurs de fournir leur feedback à travers les évaluations des cours qu'ils suivent. Le traitement de ces évaluations permet au système le suivi de ses recommandations ainsi que l'enrichissement des règles utilisées. Nous avons aussi proposé dans ce chapitre deux modes de fonctionnement du système : un fonctionnement basé sur les règles de classification et un fonctionnement basé sur les règles d'association.

Dans le chapitre suivant, nous présentons en détails l'implémentation de *RARE*. Nous expliquons comment nous avons extrait et évalué les deux types de règles (les règles de classification et les règles d'association) pour que *RARE* adopte en fin de compte le fonctionnement basé sur les règles d'association.

Chapitre 5 : Implémentation et évaluation

Dans ce chapitre, nous commençons par la présentation des outils et des langages de programmation utilisés dans l'implémentation de *RARE*. Ensuite, nous développons les étapes suivies dans notre processus de forage de données (phase hors ligne), et nous expliquons le fonctionnement détaillé de la phase en ligne à travers un scénario d'utilisation. À la fin de chacune des deux phases, hors ligne et en ligne, nous évaluons les résultats obtenus.

5.1 Outils d'implémentation

Dans l'implémentation de *RARE*, nous avons développé en premier lieu la phase hors ligne. Nous avons utilisé le package WEKA (Waikato Environment for Knowledge Analysis) [Eibe et *al*, 2000] développé à l'Université de *Waikato* en *Nouvelle-Zélande*. Ce package contient une collection d'algorithmes implémentés en *JAVA* et disponibles sur le Web, pour la résolution des problèmes de forage de données. Via son interface (voir figure 5.1), nous pouvons utiliser plusieurs techniques, comme notamment la classification et l'extraction des règles d'association auxquelles nous nous sommes intéressés. Nous nous sommes servis respectivement de l'algorithme *C4.5* (*J48* dans WEKA) et l'algorithme *Apriori*. WEKA peut fonctionner sous les systèmes d'exploitation Linux, Windows et Macintosh.

La phase en ligne a été développée sous le système d'exploitation Windows XP. Comme *RARE* est présenté par une interface Web, nous avons utilisé le serveur Web Apache Tomcat 5.0.30. Nous avons également utilisé quelques technologies et langages de programmation liés au Web, à savoir JSP (*Java Server Pages*) version 2.0, JavaBeans, HTML et Javascript. Ces techniques et langages nous ont permis la conception, le développement et la gestion de *RARE*. Dans les pages JSP, nous pouvons utiliser du code JAVA et du code HTML en même temps, ce qui permet le développement des applications client-serveur dynamiques. Pour implémenter les interfaces, nous avons employé le code HTML ainsi que Javascript.

Pour le stockage des données, nous avons utilisé le serveur de bases de données MySQL. Parallèlement, le pilote JDBC (*Java DataBase Connectivity*) nous a permis la connexion à la base de données pour faire des requêtes SQL.

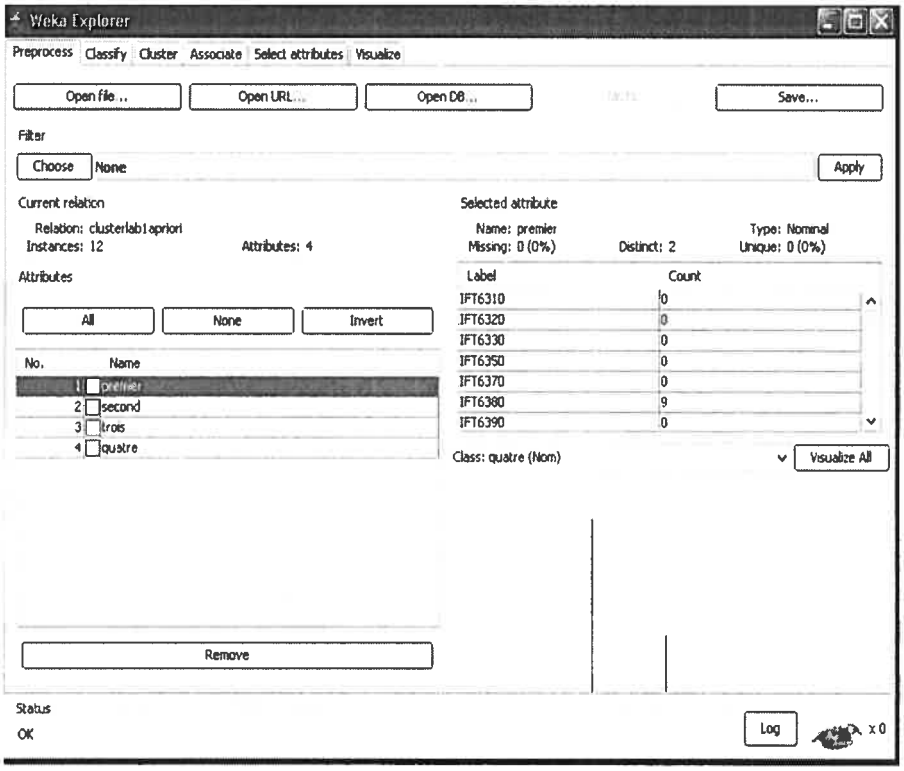


Figure 5.1 : Interface de WEKA

5.2 Implémentation de la phase hors ligne

Rappelons que l'objectif principal de la phase hors ligne est d'effectuer un processus de forage de données sur un ensemble de données réelles. Pour ce faire, nous avons utilisé des données qui décrivent 230 étudiants ayant suivi le programme de maîtrise entre l'année 1990 et 1999 au sein du Département d'Informatique et de Recherche opérationnelle de l'Université de Montréal (*DIRO*). Afin d'assurer la confidentialité des informations sur les étudiants, les données qui nous ont été fournies étaient anonymes. Pour chaque cas (c.à.d. chaque étudiant), nous avons le *sexe* de l'étudiant, le *nom de son directeur de recherche*, les *codes des cours suivis*, les *sessions* et les *années d'études*. La

figure 5.2 montre un échantillon de ces données. Les sessions *automne, hiver* et *été* sont représentées respectivement par les chiffres 1, 2 et 3.

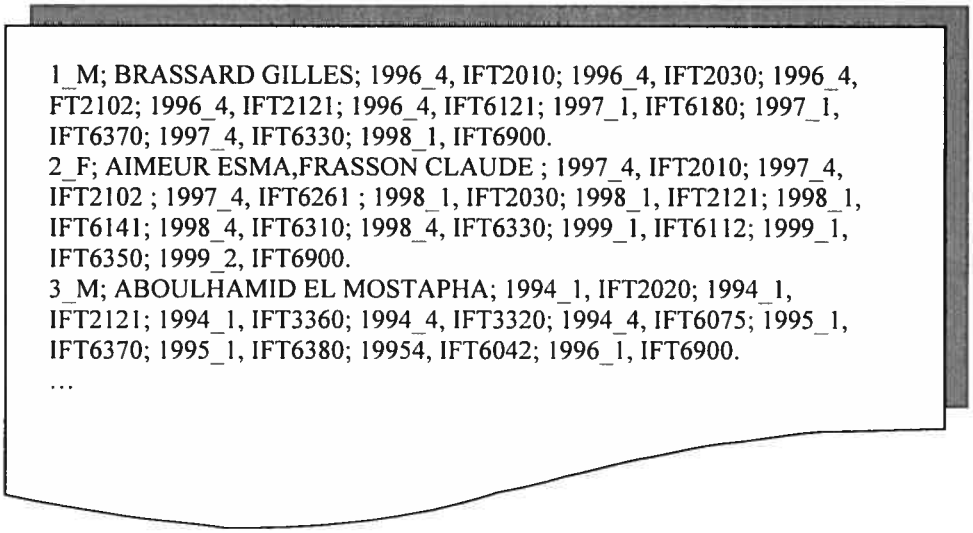


Figure 5.2 : Données collectées

Par exemple, l'étudiant 1 est masculin. Son directeur de recherche s'appelle *Gilles Brassard.* Chaque cours suivi est précédé d'une combinaison *Année_Session*, telle que 1996_2, où (1996) représente l'année, et (2) représente la session d'hiver. Des exemples de cours suivi sont *IFT2121* et *IFT6180.* Ils représentent respectivement les deux cours intitulés *Introduction à l'algorithmique* et *Cryptologie: théorie et applications.*

Dans la section qui suit, nous présentons les détails des six étapes de notre processus de forage de données selon le standard « *CRISP-DM* » présenté dans le chapitre 2 (forage de données), et qui consiste à : déterminer notre objectif, comprendre et préparer nos données, les modéliser, évaluer les résultats et enfin les exploiter.

5.2.1 Détermination de l'objectif (étape 1)

En essayant de formuler l'objectif de notre application sous l'une des tâches de forage de données, nous avons opté dans le chapitre 4 (conception de *RARE*) pour la classification et l'extraction des règles d'association. Ce qui suit explique en détail l'implémentation

des deux tâches. L'étape de l'évaluation présentée dans la section 5.2.4, détermine le type de règles qui convient mieux à notre application.

5.2.2 Compréhension et préparation de données (étapes 2, 3)

À la fin de la phase de compréhension et de préparation, l'ensemble de données final qui sera utilisé dans la découverte des règles doit être construit. Pour ce faire, nous avons procédé dans un premier temps à un nettoyage de données. Il en résulte une élimination de plusieurs données, comme les cours hors programme et les cours qui ne sont pas donnés à l'Université de Montréal. Pour les cours qui ne se donnent plus, ou les codes de cours qui ont changé, nous procédons à une substitution par le cours ou le code équivalent. Par exemple, le code *IFT6250* d'un cours intitulé « S*ujets spéciaux en génie logiciel* » est remplacé par le code *IFT6251* qui représente le même cours. Vu que nous nous intéressons aux cours suivis par les étudiants ainsi qu'aux domaines de leur recherche, le sexe de l'étudiant, les années et les sessions des cours ont été également supprimés. Ainsi, nous avons gardé uniquement les valeurs des cours suivis et celles des directeurs de recherche. Toutefois, l'information sur le directeur de recherche peut révéler l'appartenance d'un étudiant à un laboratoire. Or, nous avons associé tous les directeurs de recherche à leurs laboratoires. Cela nous a permis de construire un nouvel attribut qui indique le *laboratoire de recherche* de chaque étudiant. Dans le programme de la maîtrise en informatique à l'Université de Montréal, un étudiant doit suivre quatre cours et réaliser un projet de recherche au niveau d'un laboratoire. Nous avons donc structuré les données en cinq attributs: le *premier cours*, le *deuxième cours*, le *troisième cours*, le *quatrième cours* et le *laboratoire de recherche*. Chacun des quatre premiers attributs peut prendre le code du cours suivi qui appartient à l'ensemble des cours gradués : {IFT6310, IFT6320, IFT6330, IFT6350, ..., IFT6690}. Cependant, le cinquième attribut est équivalent au nom de l'un des laboratoires de recherche existants : *CRT*, *GEODES*, *GRITI*, *HERON*, *IMAGE*, etc.

Les données sont ensuite mises sous le format *ARFF* (*Attribute-Relation File Format*) [Witten *et al.*, 2005], et sauvegardées dans le fichier *Course.ARFF* comme le montre la figure 5.3. Ce format est adopté pour adapter les données au besoin des

algorithmes appliqués dans la phase de modélisation de données. Cette phase est expliquée dans la section suivante.

```
@relation course

@attribute first {IFT6310, IFT6320, IFT6330, IFT6350,
                  IFT6370, IFT6380,…, IFT6490, IFT6690}
@attribute second {IFT6310, IFT6320, IFT6330, IFT6350,
                  IFT6370,IFT6380,…, IFT6490, IFT6690}
@attribute third {IFT6310, IFT6320, IFT6330, IFT6350,
                  IFT6370, IFT6380,…, IFT6490, IFT6690}
@attribute fourth{IFT6310, IFT6320, IFT6330, IFT6350,
                  IFT6370,IFT6380,…, IFT6490, IFT6690}
@attribute fifth {CRT, GEODES, GRITI, HERON, IMAGE,…}

@data

IFT6380,IFT6075,IFT6370,IFT6042, LASSO
IFT6261,IFT6320,IFT6015,IFT6255, HERON
IFT6222,IFT6221,IFT6165,IFT6251, GEODES
…
```

Figure 5.3 : Fichier sous le format *ARFF*

5.2.3 Modélisation (étape 4)

Dans l'étape de la modélisation de données, nous avons utilisé le package WEKA (*Waikato Environment for Knowledge Analysis*) [Witten *et al.*, 2005] pour extraire des règles de classification et d'association. Nous allons à présent donner le détail de l'implémentation de ces deux techniques.

5.2.3.1 Classification

Pour la classification des données, les cours constituent les variables indépendantes et le laboratoire représente la classe. Nous avons appliqué l'algorithme *C4.5* [Quinlan, 1993] (voir chapitre 2), ou *J48* dans WEKA, sur les données du fichier *Course.ARFF* pour construire un arbre de décision ou des règles de classification. Sur celui-ci, nous devons avoir plusieurs parcours partant du nœud racine qui représente l'ensemble des cas. Ensuite, chaque parcours doit passer par des nœuds intermédiaires constituant les cours suivis. Au niveau de chaque nœud, l'algorithme *C4.5* teste les différentes valeurs de

cours qui permettent le passage au cours suivant. Les différents parcours des cours suivis aboutissent aux laboratoires de recherche.

L'arbre obtenu n'était pas assez profond ; il a tout au plus deux niveaux. D'une façon générale, en classifiant les cas à partir du nœud racine en bas de l'arbre, seulement un test est effectué sur les attributs. Il consiste à vérifier la valeur du premier cours et donner en conclusion la valeur de la classe, c.à.d. du laboratoire. Étant donné qu'il y a quatre cours à suivre, nous justifions ces résultats de la façon suivante. En regardant le niveau global des données, une multitude de classifications est possible vu le nombre important de cours qui peuvent être choisis au départ. Par conséquent, beaucoup de chemins dérivent du nœud racine. Dans les meilleurs cas, les étudiants ont deux cours en commun qui se rapportent généralement à une spécialité ou un laboratoire de recherche. Ils ont rarement trois cours en commun. Ainsi, après avoir traité deux cours, *C4.5* ne trouve aucune information indicative sur le troisième ou le quatrième cours. Par conséquent, un chemin atteint une feuille seulement après un ou deux cours et mène directement au laboratoire de recherche. En appliquant ce type d'approche, il n'est pas possible de prévoir les valeurs des troisième et quatrième cours à suivre. Par conséquent, l'utilisation de cet arbre de classification produit une recommandation pauvre. De ce fait, la technique de classification ne peut as être adopté par notre système de recommandation.

5.2.3.2 Règles d'association

Les quatre premiers attributs ont été maintenus dans le fichier *Course.ARFF*. Nous avons procédé à la découverte des règles d'association en appliquant l'algorithme *Apriori* [Agrawal *et al.*, 1994] (voir chapitre 2). Cet algorithme essaye de découvrir les *itemsets* (ensembles d'items) fréquents qui devraient être les plus grands possibles. Autrement dit, ils doivent idéalement contenir beaucoup de cours communs entre les étudiants. Nous rappelons que pour générer les règles d'association, l'algorithme *Apriori* utilise deux paramètres pour mesurer l'importance des règles générées. Ces paramètres sont le support et la confiance (voir chapitre 2, section 2.6.2) qui reflètent respectivement

l'utilité et la certitude des règles. Pour une règle d'association (A==>B) où A et B sont deux ensembles d'items, les formules de support et de confiance sont les suivantes :

$$Support(A \Rightarrow B) = P(A \cup B)$$

$$Confiance(A \Rightarrow B) = P(B \mid A)$$

Vu que les étudiants ne suivent pas toujours les mêmes cours, cet algorithme ne pouvait pas découvrir des *itemsets* de taille importante à partir de l'ensemble global de données. Pour toute valeur de support et de confiance, *Apriori* produit tout au plus quatre règles d'association. Pour remédier à ce problème, nous avons segmenté les données dans des groupes homogènes, chacun représentant les étudiants d'un laboratoire de recherche. Le but est de maximiser la similarité entre les cours suivis par les étudiants du même laboratoire. En fixant à chaque fois les seuils minima de support et de confiance, *Apriori* génère plusieurs règles d'association par laboratoire. La figure 5.4 montre quelques règles d'association générées par l'algorithme *Apriori*, avec un support égal à 15% et une confiance égale à 60%, au niveau du laboratoire de recherche « *HERON : Multimédia et tutoriels intelligents* ». Un exemple d'association entre les cours est le suivant : « les étudiants qui suivent le cours *IFT6243 (Concepts de bases de données avancées)* et le cours *IFT6310 (Génie logiciel avancé)* suivent aussi le cours *IFT6261 (Traitement des connaissances)* ». Contrairement aux cours du premier cycle, la majorité des cours gradués ne nécessitent pas des pré-requis. Il est donc possible à un étudiant de commencer sa maîtrise en session d'automne ou d'hiver. Vu que les cours offerts dans chacune des sessions sont différents, nous tenons compte des cours dans les antécédents et les conséquents des règles d'association extraites. L'ordre des cours, par contre, n'est pas considéré.

Les règles d'association extraites au niveau des différents laboratoires de recherche et qui vont peupler la base des règles doivent d'abord être évaluées. Dans la section suivante, nous expliquons comment nous avons procédé pour leur évaluation.

```
 === Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.6 -D 0.05 -U
1.0 -M 0.1 -S -1.0
Relation:     heron
Instances:    26
Attributes:   4
              first
              second
              third
              fourth
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.15
Minimum metric <confidence>: 0.6
Number of cycles performed: 17

Generated sets of large itemsets:
Size of set of large itemsets L(1): 8
Size of set of large itemsets L(2): 6
Size of set of large itemsets L(3): 1

Best rules found:
 1. second=IFT6243 7 ==> first=IFT6261 7    conf:(1)
 2. second=IFT6243 third=IFT6310 5 ==> first=IFT6261 5    conf:(1)
 3. third=IFT6251 4 ==> first=IFT6243 4    conf:(1)
 4. second=IFT6075 4 ==> first=IFT6243 4    conf:(1)
 5. second=IFT6330 4 ==> first=IFT6243 4    conf:(1)
 6. third=IFT6310 7 ==> first=IFT6261 6    conf:(0.86)
 7. first=IFT6261 third=IFT6310 6 ==> second=IFT6243 5    conf:(0.83)
 8. second=IFT6243 7 ==> first=IFT6261 third=IFT6310 5    conf:(0.71)
 9. third=IFT6310 7 ==> first=IFT6261 second=IFT6243 5    conf:(0.71)
10. first=IFT6261 second=IFT6243 7 ==> third=IFT6310 5    conf:(0.71)
...
```

Figure 5.4 : Règles d'association générées par *Apriori*

5.2.4 Évaluation expérimentale des règles (étape 5)

Notre première évaluation consiste à expérimenter les règles. L'objectif est de mesurer leur efficacité dans la recommandation. Nous avons collecté un ensemble de données de test représentant les cours suivis par 40 étudiants choisis arbitrairement. Ces étudiants ont été enregistrés dans le programme de maîtrise en informatique après l'année 2000 et ont achevé leurs cours. Comme expliqué dans le chapitre 4, l'inférence des recommandations par le système se fait par la vérification de l'équivalence entre les cours suivis et les antécédents des règles. Les conséquents des règles déterminent les cours à recommander.

Nous rappelons également que notre méthode d'évaluation, comme décrite dans le chapitre 4 (section 4.2.4), consiste à mesurer le *recouvrement* et l'*exactitude* des

recommandations fournies aux étudiants par le biais des règles. Pour chaque étudiant E_i, nous appelons S_i la liste des cours qu'il a suivis. Si nous désignons par $R(E_i)$ l'ensemble des cours recommandés à l'étudiant E_i, et nous désignons par $T(E_i)$ les cours figurant dans S_i et non connus par le système, les formules de recouvrement et d'exactitude, comme définies dans le chapitre 4, sont les suivantes :

$$\text{Re}couvrement = \frac{|R(E_i) \cap T(E_i)|}{|T(E_i)|}$$

$$Exactitude = \frac{|R(E_i) \cap T(E_i)|}{|R(E_i)|}$$

Exemple :

Prenons l'exemple de l'étudiant E_6 ayant suivi les cours : « *IFT6243 (Concepts de bases de données avancées)*, *IFT6310 (Génie logiciel avancé)*, *IFT6261 (Traitement des connaissances)* et *IFT6320 (Téléinformatique)* ». Ainsi, nous avons :

$S_6 = \{IFT6243, IFT6310, IFT6261, IFT6320\}$.

Si nous fournissons au système les deux premiers cours suivis par E_6, c.à.d. *IFT6243* et *IFT6310*, $T(E_6)$ est ainsi la suivante : $T(E_6) = \{IFT6261, IFT6320\}$.

Admettons que le système recommande la liste des cours suivante :

$R(E_6) = \{IFT6255, IFT6261, IFT6251, IFT6075, IFT6150\}$. Le recouvrement et l'exactitude de cette recommandation sont les suivants :

$$\text{Recouvrement} = \frac{|R(E_6) \cap T(E_6)|}{|T(E_6)|} = \frac{1}{2}$$

$$\text{Exactitude} = \frac{|R(E_6) \cap T(E_6)|}{|R(E_6)|} = \frac{1}{5}$$

Comme nous l'avons déjà mentionné, l'algorithme *Apriori* utilise deux paramètres qui sont le support et la confiance. Nous tenons compte, dans l'évaluation des règles, de

la variation de ces deux paramètres. Comme point de départ, nous fixons le support à 10% [Zaïane, 2004]. Ensuite plusieurs valeurs de support sont testées. Pour ces différentes valeurs, nous tenons compte des règles générées avec une confiance supérieure à 50%. Si pour une certaine valeur de support le nombre des règles est trop petit, nous considérons les règles générées avec une confiance égale à 50%.

Dans un premier temps, nous donnons au système le premier cours suivi par chacun des 40 étudiants et nous calculons le recouvrement et l'exactitude de la recommandation générée par les différents ensembles de règles d'association. Les résultats montrent que la variation du support de l'algorithme *Apriori* influe sur le recouvrement ainsi que sur l'exactitude. Comme l'illustre la figure 5.5, quant le support augmente, la valeur de recouvrement diminue. Contrairement à ce dernier, l'exactitude augmente avec l'augmentation du support. En d'autres termes, le recouvrement diminue quand nous essayons d'augmenter l'exactitude. Cela peut être constaté à partir de la figure 5.6. La raison est qu'en diminuant le support, plus de règles sont générées. Le système propose donc plus de recommandations, d'où résulte la diminution dans l'exactitude.

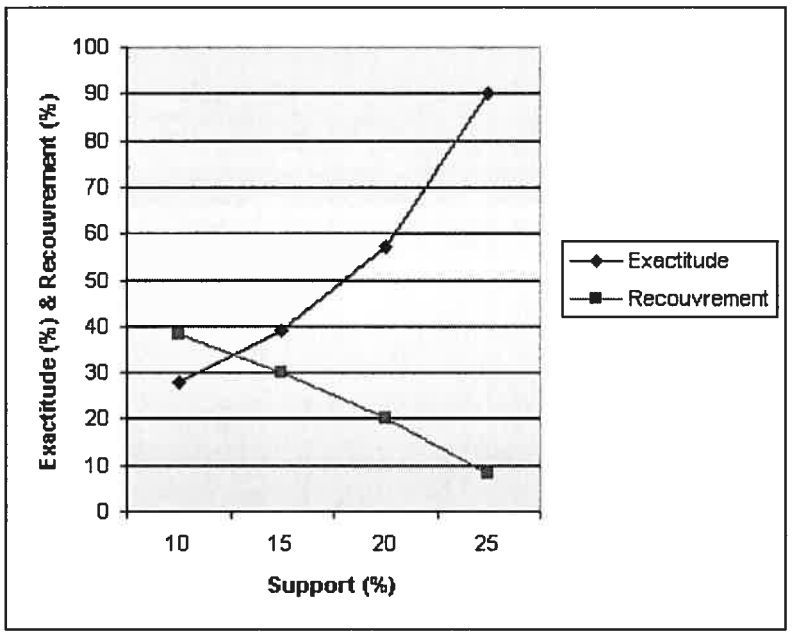


Figure 5.5 : Le recouvrement et l'exactitude de la recommandation (pour un cours suivi)

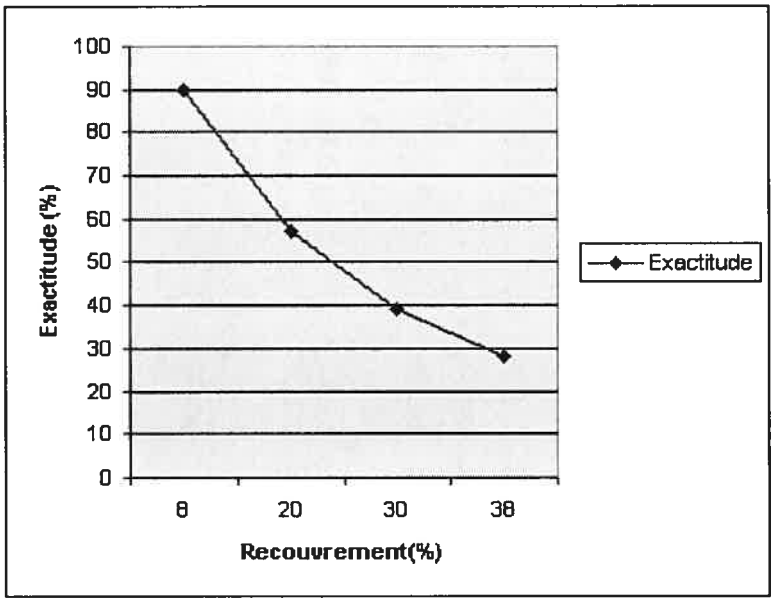


Figure 5.6 : Recouvrement versus exactitude

Nous fournissons ensuite au système les deux premiers cours suivis par les étudiants. Les résultats du calcul de recouvrement et d'exactitude sont représentés sur la figure 5.7. Nous constatons que le recouvrement change de la même façon que sur la figure 5.5. L'exactitude, quant à elle, commence à diminuer quand la valeur du support va au-dessus de 15%. L'explication la plus plausible pour ceci est la suivante : étant donné que la population des étudiants gradués est petite et diverse, les étudiants ont généralement deux cours en commun et rarement trois. Par conséquent, les règles de la forme (*A* et *B* ==> *C*) sont rares et moins significatives. *A* et *B* se rapportent généralement aux cours qui sont communs dans une spécialité (ou un laboratoire de recherche) et sont très instructifs, alors que *C* est un cours lié, par exemple, au domaine d'application ou au domaine d'intérêt de l'étudiant et révèle donc moins d'information. Au delà d'une valeur de support égale à 15%, peu de règles d'association qui contiennent deux cours dans leurs antécédents existent. À un support minimum égal à 25%, *Apriori* ne génère aucune règle contenant deux cours ce qui fait que le recouvrement et l'exactitude sont nuls.

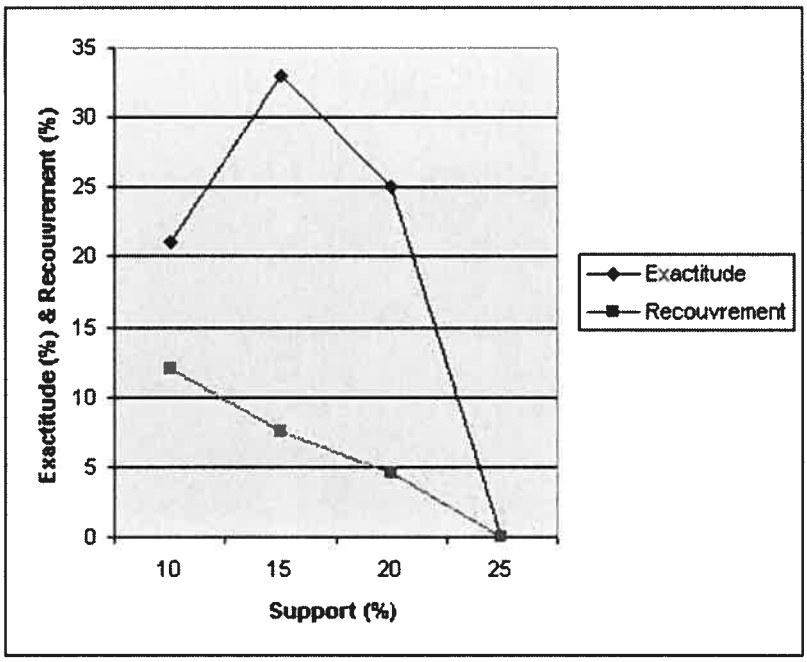


Figure 5.7 : Le recouvrement et l'exactitude de la recommandation (pour deux cours suivi)

Une meilleure performance de recommandation nécessite un compromis entre le recouvrement et l'exactitude. Plus ces deux facteurs sont élevés, meilleure est la prédiction des cours à recommander. À partir des figures 5.5 et 5.7, nous constatons qu'un support qui varie entre 10% et 15% donne des résultats atteignant la meilleure performance. Ainsi, pour recommander les cours, nous optons pour les règles extraites dans cet intervalle. Celles-ci constituent 253 règles générées au niveau de tous les laboratoires et stockées dans la base des règles. Ainsi, *RARE* se base dans son fonctionnement sur les règles d'association.

La phase en ligne présentée ci-dessous, montre comment ces règles sont utilisées et enrichies afin de les transformer en connaissances utiles.

5.3 Implémentation de la phase en ligne

Comme son nom l'indique, la phase en ligne est dédiée au traitement des requêtes des utilisateurs afin de leur fournir des recommandations de cours. En plus, *RARE* met en place, dans cette phase, une fonction qui permet d'examiner le feedback des utilisateurs

qui va être exploité dans la mise à jour et l'enrichissement des règles. Dans l'objectif d'illustrer l'implémentation de cette phase, nous présentons un scénario d'utilisation agrémenté de captures d'écran de quelques interfaces du système.

5.3.1 Scénario de recommandation

Prenons l'exemple de *Tifanny*, une étudiante au Département d'Informatique et de Recherche Opérationnelle à l'Université de Montréal. Lors de sa première utilisation du système, *Tifanny* doit s'enregistrer. La figure 5.8 présente la page Web principale de *RARE*, c.à.d. sa page d'enregistrement et d'identification. Pour s'enregistrer, il suffit de remplir un formulaire et ceci en cliquant sur « *S'enregistrer* ». Les informations demandées sont le nom de l'utilisateur, son mot de passe et son adresse électronique. Cette dernière est très utile au cas où l'utilisateur oublie son mot de passe lors de l'ouverture des prochaines sessions. Dans ce cas, il n'a qu'à cliquer sur «*Mot de passe oublié* ». Le système l'invite à entrer son nom d'utilisateur et son adresse électronique qu'il a fournis lors de son enregistrement. Ainsi, l'utilisateur recoit un courrier électronique contenant son mot de passe. Quant à la fonctionnalité d'identification, l'utilisateur ne peut l'utiliser qu'après avoir été enregistré. Dans ce cas, il n'a qu'à ouvrir une session à l'aide de son nom d'utilisateur et son mot de passe.

Une fois enregistrée, le système offre à *Tifanny* quatre choix illustrés dans la figure 5.9. Dans le premier choix, le système demande comme entrées les cours suivis par l'utilisateur pour lui produire des recommandations par cours suivis. Si aucun cours n'a été pris, l'utilisateur peut sélectionner son laboratoire de recherche préféré. Les deux options qui restent sont la consultation des recommandations des autres utilisateurs du système, ou la proposition des cours qu'il souhaite recommander lui même. En haut du menu proposé à l'utilisateur, le système affiche une courte description de son fonctionnement.

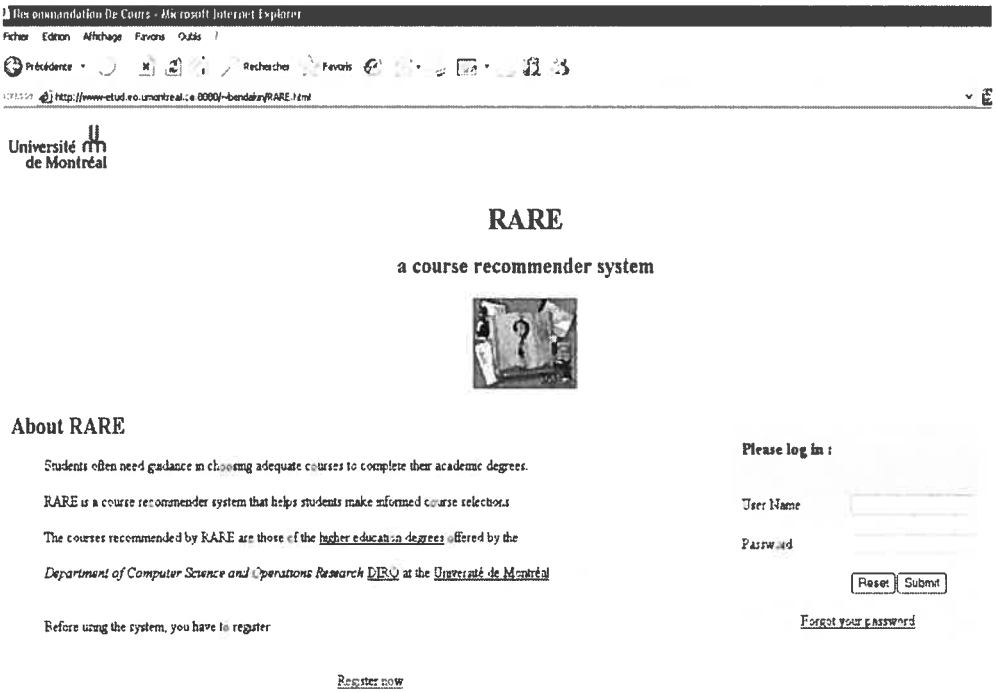


Figure 5.8 : La page Web principale de *RARE*

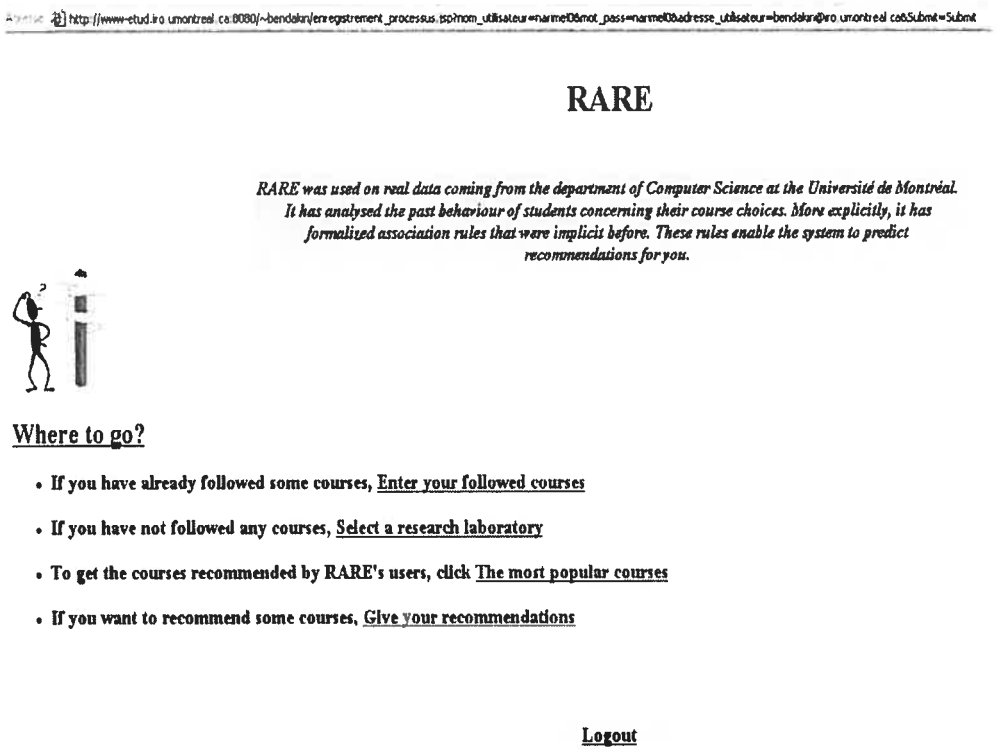


Figure 5.9 : Les fonctionnalités offertes par *RARE*

Supposons que *Tifanny* opte pour le premier choix afin d'indiquer à *RARE* les cours qu'elle a déjà pris. Le système lui affiche l'interface illustrée dans la figure 5.10, sur laquelle figurent tous les cours de maîtrise en informatique. *Tiffany* sélectionne les cours *IFT6042* et *IFT6150* intitulés respectivement « *Synthèse d'images* (*Images synthesis*) » et « *Traitement d'images* (*Image processing*) ».

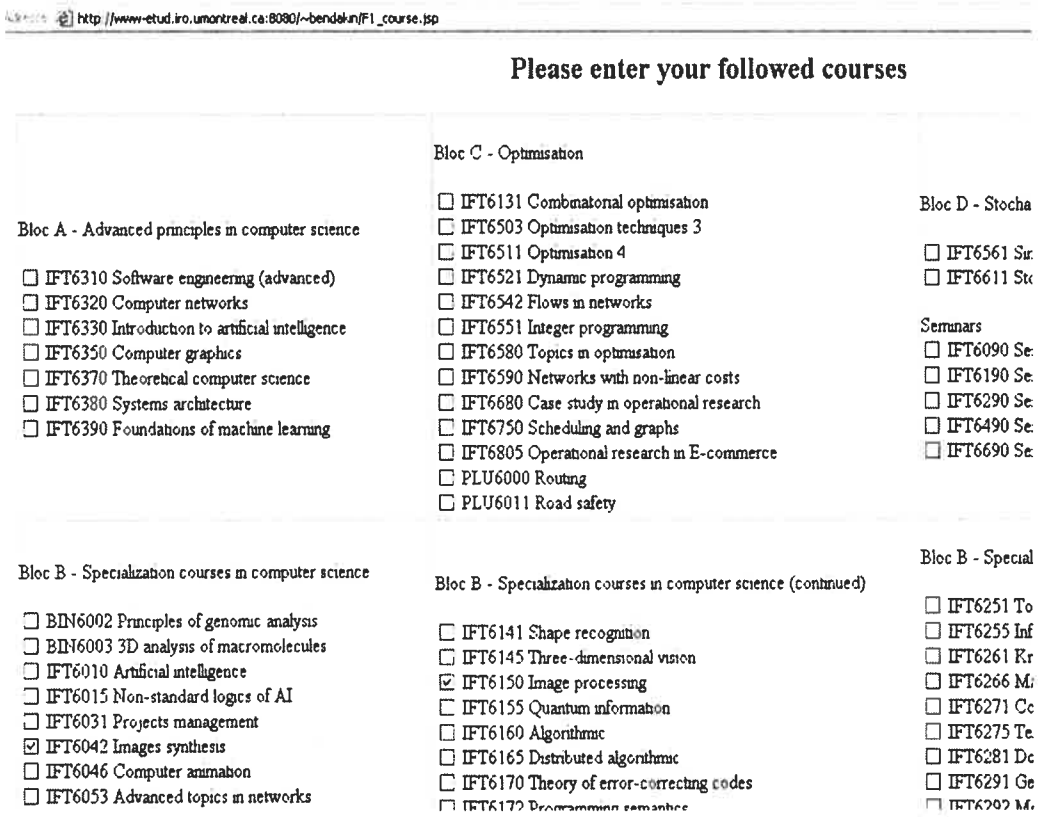
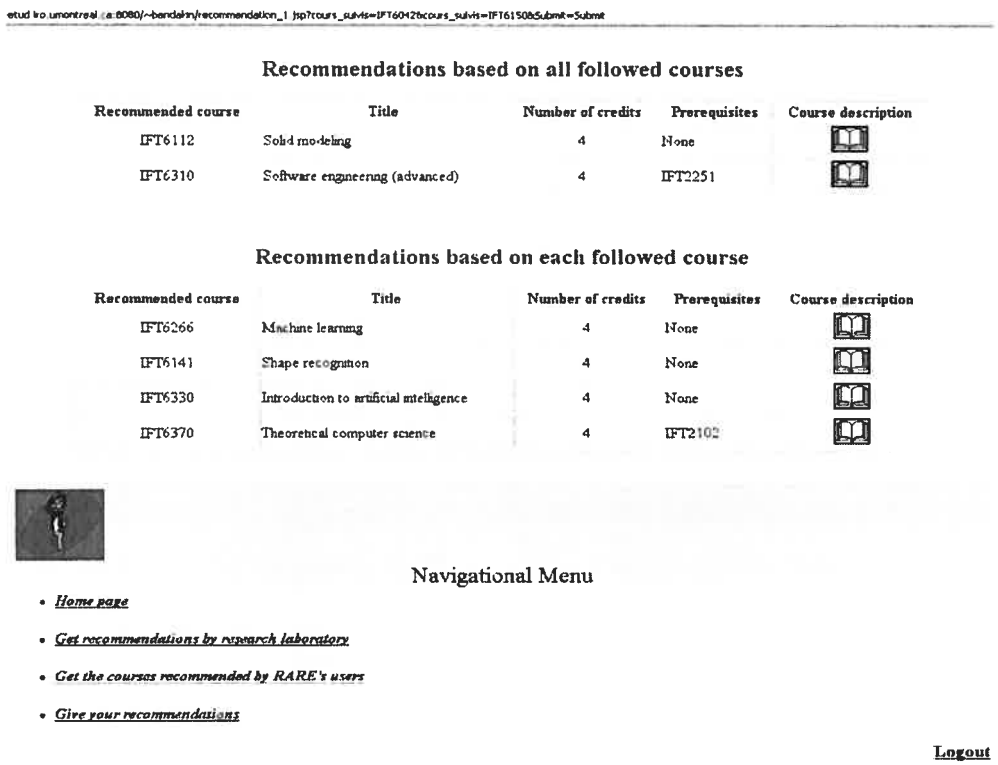


Figure 5.10 : La sélection des cours

Le système interroge la base des règles pour une recherche de similarité entre leurs antécédents et les cours suivis par *Tiffany*. Les antécédents recherchés doivent comprendre les deux cours ensemble. Sinon, il faut qu'au moins le cours *IFT6042* ou le cours *IFT6150* paraissent dans les antécédents, c'est le principe de la recherche décrit dans la section 4.3.1. La figure 5.11 indique qu'il y a exactement 2 recommandations en réponse à la recherche des règles qui contiennent les deux cours ensemble. Ce type de résultats, s'il y'a lieu, est appelé « *les recommandations basées sur tous les cours suivis* ». En ce qui concerne le deuxième type de résultats, basé sur l'un des deux cours

suivis, le système affiche 4 cours à *Tiffany*. Chaque recommandation contient le titre, le nombre de crédits ainsi que les pré-requis du cours. En plus, cliquer sur le lien d'un cours permet à l'utilisateur d'accéder à une page Web qui décrit son contenu (voir figure 5.12).



Recommendations based on all followed courses

Recommended course	Title	Number of credits	Prerequisites	Course description
IFT6112	Solid modeling	4	None	
IFT6310	Software engineering (advanced)	4	IFT2251	

Recommendations based on each followed course

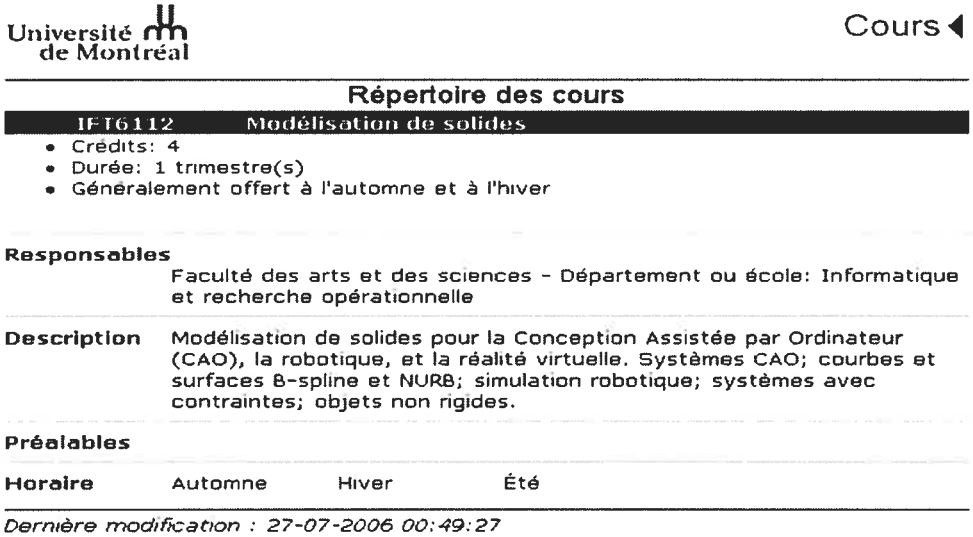
Recommended course	Title	Number of credits	Prerequisites	Course description
IFT6266	Machine learning	4	None	
IFT6141	Shape recognition	4	None	
IFT6330	Introduction to artificial intelligence	4	None	
IFT6370	Theoretical computer science	4	IFT2102	

Navigational Menu

- Home page
- Get recommendations by research laboratory
- Get the courses recommended by RARE's users
- Give your recommendations

Logout

Figure 5.11 : La liste des recommandations



Université de Montréal

Cours

Répertoire des cours

IFT6112 Modélisation de solides

- Crédits: 4
- Durée: 1 trimestre(s)
- Généralement offert à l'automne et à l'hiver

Responsables
Faculté des arts et des sciences - Département ou école: Informatique et recherche opérationnelle

Description Modélisation de solides pour la Conception Assistée par Ordinateur (CAO), la robotique, et la réalité virtuelle. Systèmes CAO; courbes et surfaces B-spline et NURB; simulation robotique; systèmes avec contraintes; objets non rigides.

Préalables

Horaire Automne Hiver Été

Dernière modification : 27-07-2006 00:49:27

Figure 5.12 : Exemple de page de description de cours

Pour que l'utilisateur puisse tirer avantage de toutes les fonctionnalités du système, celui-ci lui affiche, en plus des recommandations, un menu qui facilite la navigation sur le site Web. Ceci offre la possibilité d'avoir des suggestions de cours rattachées à l'un des laboratoires de recherche et permet ainsi de couvrir un domaine spécifique. Les suggestions des autres utilisateurs peuvent également être consultées. Le navigateur sollicite aussi l'utilisateur à recommander des cours. *Tiffany* souhaite recevoir des recommandations liées aux laboratoires (domaines) de recherche et le système affiche l'interface présentée dans la figure 5.13 et qui ne requiert que le nom du laboratoire. Cette fonctionnalité vise à simplifier la recherche des cours appropriés en offrant à l'utilisateur des recommandations sans devoir suivre des cours auparavant.

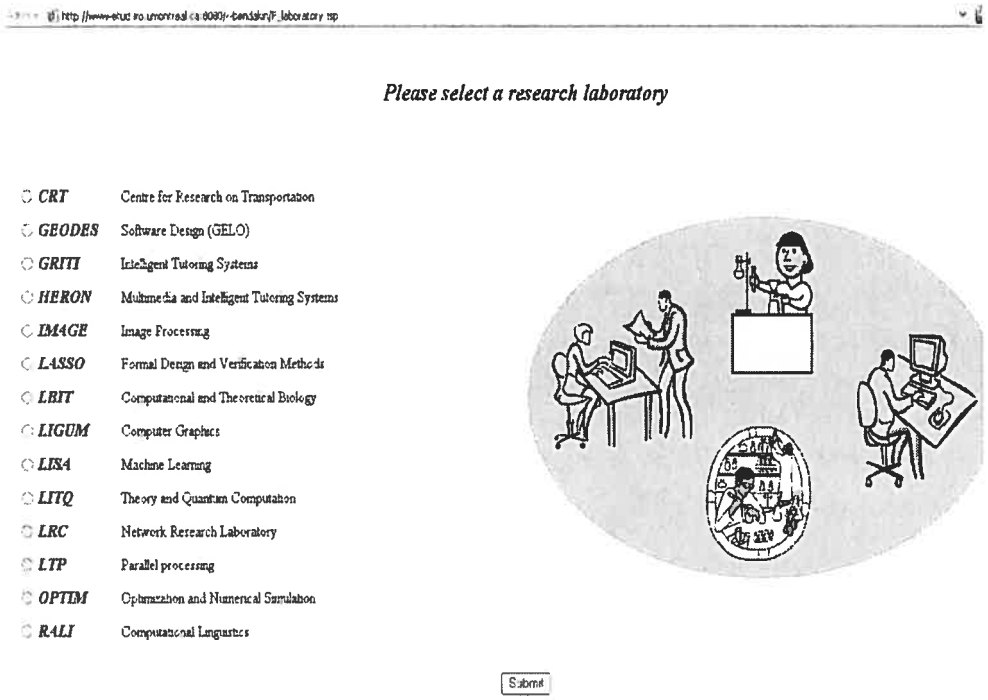


Figure 5.13 : L'interface de sélection de laboratoire de recherche

Tiffany sélectionne le laboratoire « *Image* » et le système se sert, cette fois-ci, de toutes les règles générées au niveau dudit laboratoire. Il affiche alors les recommandations suivantes (figure 5.14).

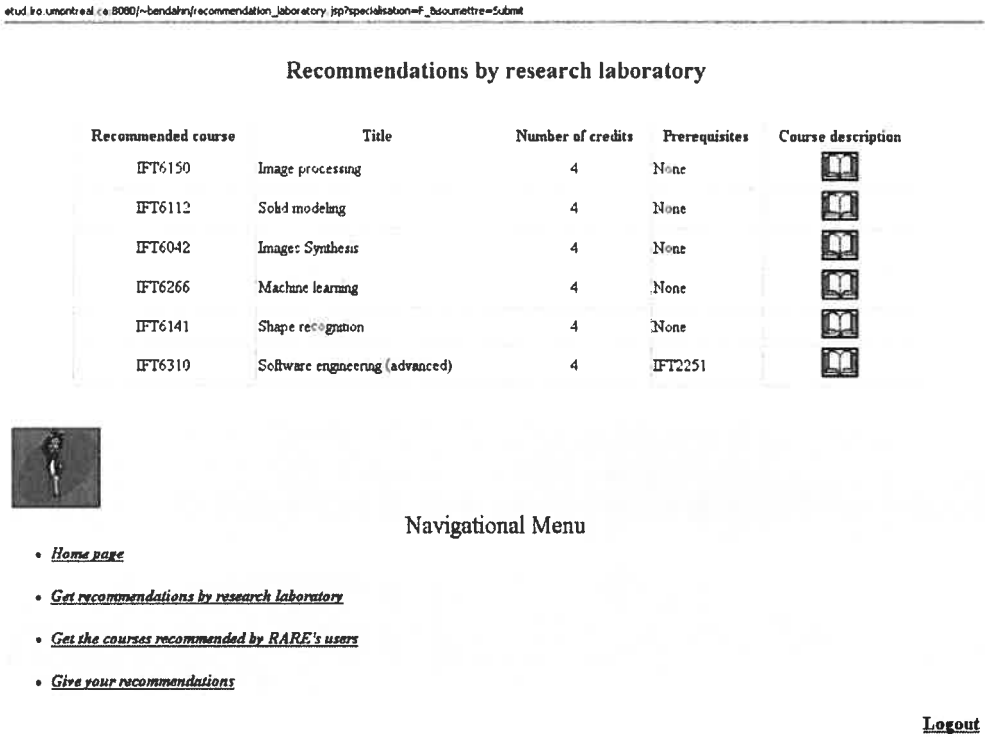


Figure 5.14 : La liste des recommandations par laboratoire de recherche

Afin de consulter les suggestions de cours proposées par les autres utilisateurs, *Tiffany* utilise une deuxième fois le navigateur du site et le système lui suggère une liste de recommandations triées par ordre décroissant de popularité (figure 5.15).

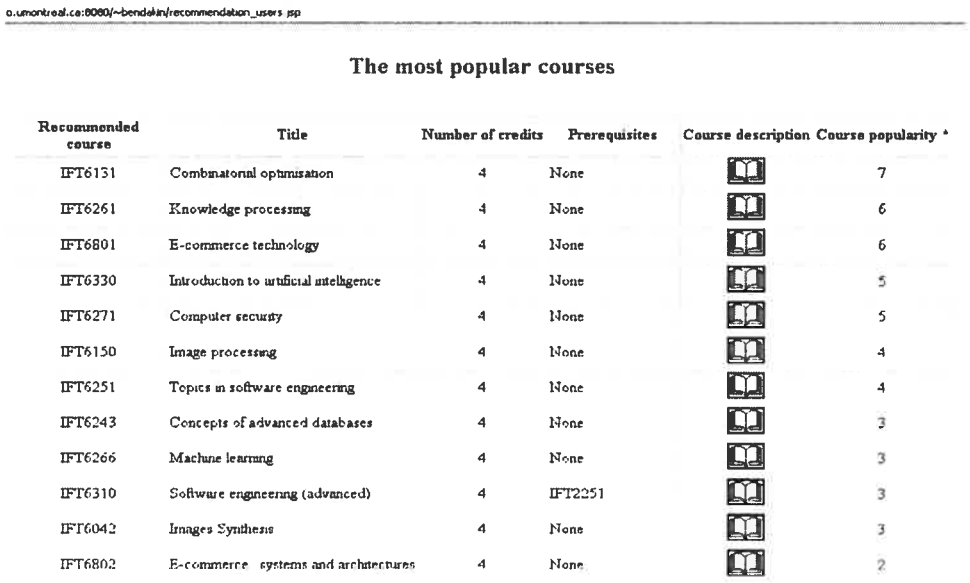


Figure 5.15 : Les recommandations des utilisateurs de *RARE*

À la fin de la session de travail, *RARE* sauvegarde dans la base des utilisateurs le profil de *Tiffany*. En plus de ses informations d'enregistrement, son profil contient les cours qu'elle a suivis, de même que les cours qui lui ont été recommandés. Chacun de ces derniers est associé aux règles utilisées dans sa recommandation. La base des évaluations des cours recommandés est également consultée par le système pour la recherche des cours recommandés à *Tiffany*. Si le système constate que les cours recherchés, ainsi que les règles auxquelles ils doivent être associés, figurent dans cette base, leur insertion ne sera pas nécessaire.

Supposons que *Tiffany* utilise *RARE* une deuxième fois. Elle doit s'identifier. *RARE* lui rappelle qu'elle a déjà suivi des cours et lui offre les quatre options à sélectionner (voir figure 5.9). *Tiffany* choisit les recommandations par cours suivis et *RARE* lui affiche cette fois-ci une autre interface pour la sélection et l'évaluation des cours (figure 5.16). En effet, sur cette interface, chaque utilisateur a la possibilité de voir les cours qu'il a déjà suivis, et qu'il a indiqués au système lors de ses utilisations précédentes. Par conséquent, ces cours ne figurent plus parmi les cours affichés. L'utilisateur doit alors sélectionner les cours qu'il a pris après sa dernière utilisation du système. De même, *RARE* montre aux étudiants l'utilité de leurs évaluations de cours et ceci pour solliciter leur collaboration. Or, chacun des cours doit être évalué en lui attribuant une note entre 1 et 10, où 1 représente un cours non apprécié et 10 indique un cours très apprécié. *Tiffany* choisit le cours *IFT6112* intitulé « *Modélisation de solides* (*Solid modeling*) » et le cours *IFT6370* intitulé « *Informatique théorique* (*Theoretical computer science*) », et les évalue respectivement à 8 et à 7. En comparant les deux cours suivis aux cours recommandés à *Tiffany*, *RARE* déduit que seul le cours *IFT6112* lui a été recommandé. Toutefois le cours *IFT6370* est considéré comme étant un cours temporaire (voir chapitre 4, section 4.3.4.2).

RARE ne trouve pas des règles dont les antécédents contiennent les quatre cours suivis par *Tiffany*. Il suggère alors des recommandations en se basant sur au moins l'un des cours suivis (figure 5.17).

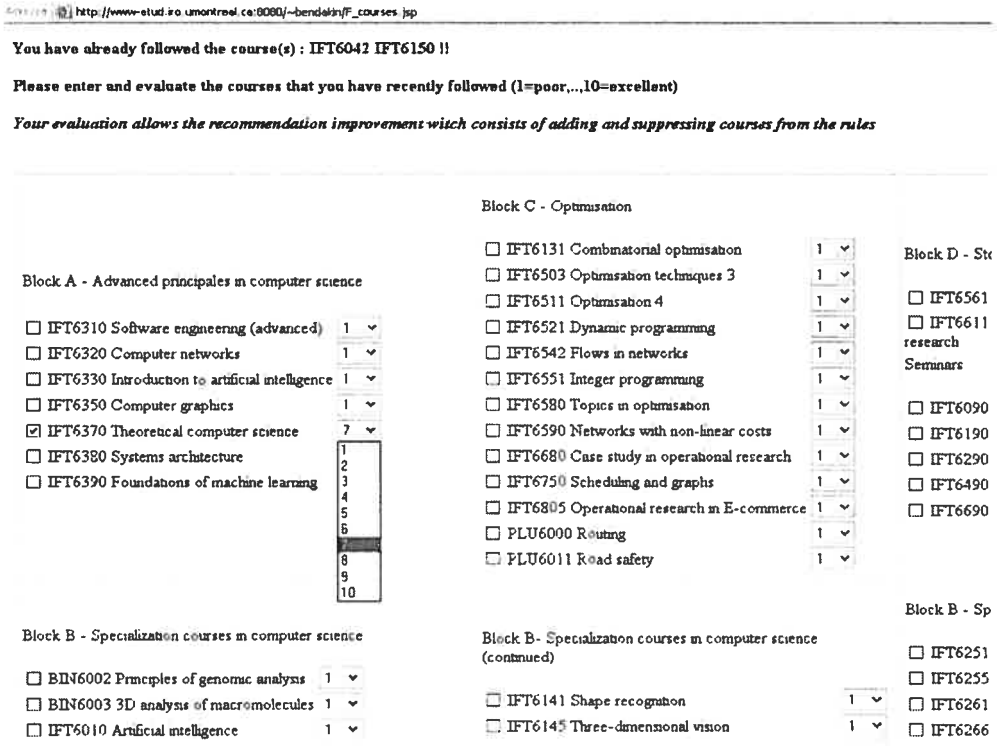


Figure 5.16 : La sélection et l'évaluation des cours

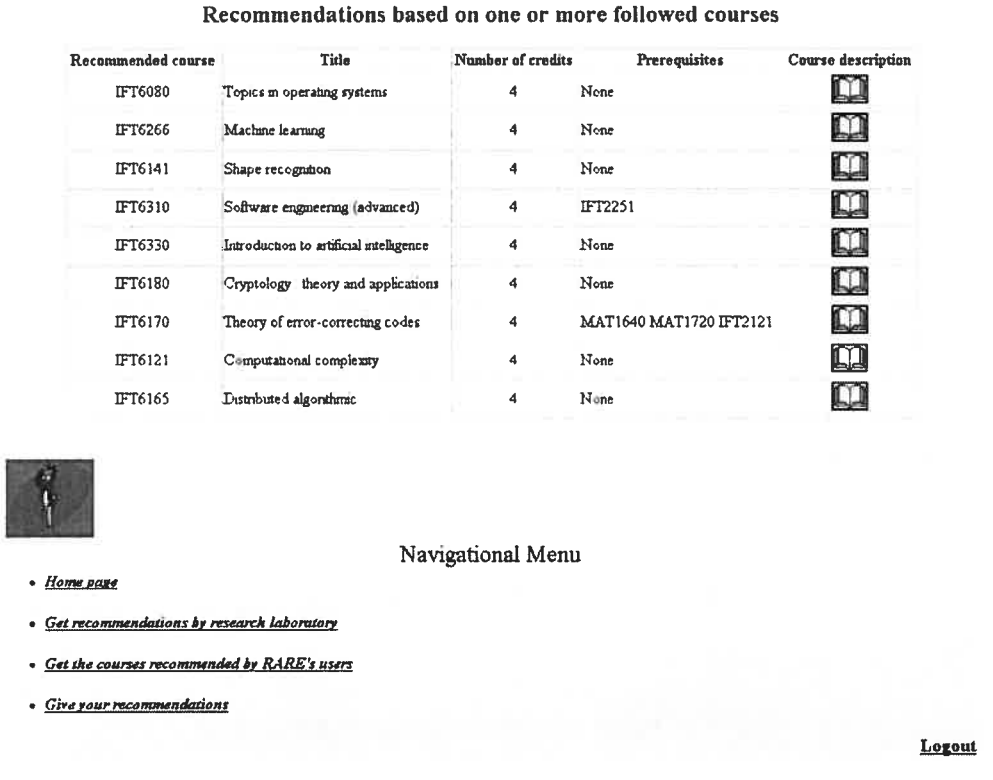


Figure 5.17 : La liste des recommandations lors de la deuxième utilisation de *RARE*

Bien que nous ayons évalué les règles, avant même que *RARE* ne les utilise dans son processus de recommandation (voir section 5.2.4), nous avons procédé à une autre évaluation qui porte sur quelques aspects en rapport avec la recommandation. Dans la section suivante, nous analysons et discutons des résultats de cette évaluation.

5.4 Évaluation générale

Rappelons que notre objectif de départ était de réaliser un système de recommandation de cours capable de faire profiter ses utilisateurs des expériences passées et récentes des étudiants. À partir du moment où l'implémentation de *RARE* a été achevée, nous avons procédé à son évaluation.

Nous pensons que notre système peut être bénéfique pour les étudiants lors de leurs choix de cours. Nous estimons également que *RARE* peut fournir à ses utilisateurs des recommandations appropriées et qu'il peut offrir une solution adéquate pour partager l'information entre les étudiants. Afin de vérifier notre hypothèse et de savoir si nous avons atteint notre objectif, 52 étudiants de maîtrise appartenant aux différents laboratoires de recherche du Département d'Informatique et de Recherche Opérationnelle ont participé à son évaluation. Chaque participant est invité à tester *RARE* via l'adresse : http://wwww-etud.iro.umontreal.ca:8080/~bendakin
Une fois le test terminé, il est invité à répondre à un questionnaire accessible via une page Web du système. Le questionnaire est constitué des trois parties décrites ci-dessous.

5.4.1 Première partie de l'évaluation

La première partie de l'évaluation consiste à poser des questions dans le but de confirmer l'existence de *la problématique du choix de cours* que rencontrent les étudiants. Nous avons demandé aux participants s'ils trouvent difficile de choisir les cours vu la diversité des sujets et de la matière qu'ils offrent. Les résultats présentés dans la figure 5.18 montrent que 92.30% des participants ont confirmé l'existence de ce problème.

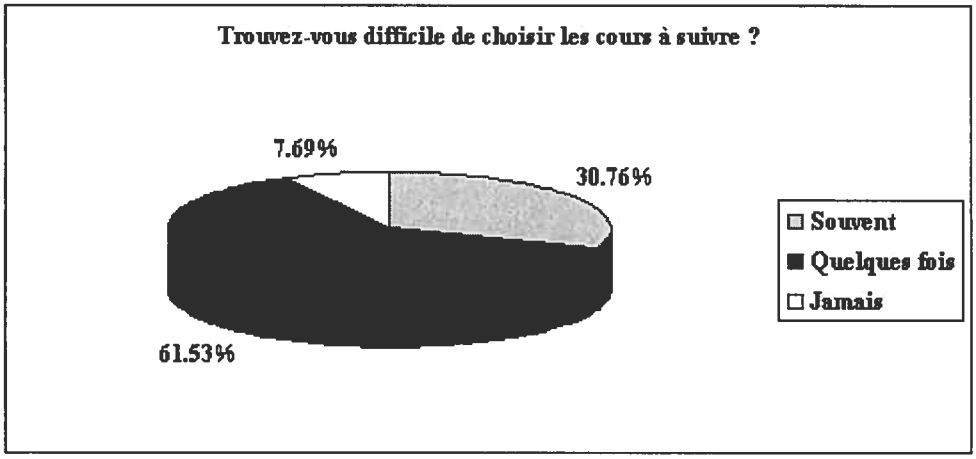


Figure 5.18 : La difficulté de choisir les cours à suivre

La difficulté du choix de cours peut être due à l'*insuffisance d'information relative aux descriptifs des cours* (sur les pages Web de l'université, sur le babillard, etc.). Pour tester cette hypothèse, nous avons interrogé les participants afin de connaître leurs opinions. En effet, la majorité d'entre eux (plus de 63%) ont dit que ces informations ne sont pas suffisantes pour les aider à choisir les cours qui leur conviennent le plus (figure 5.19).

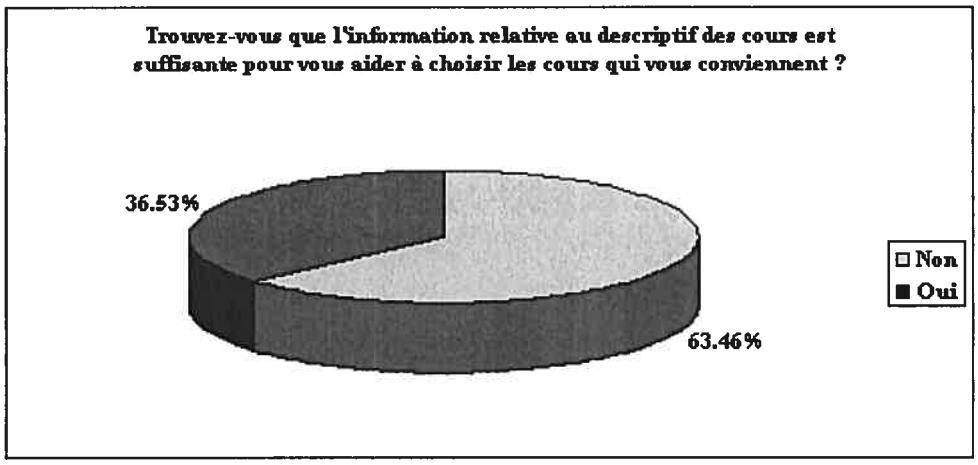


Figure 5.19 : L'insuffisance d'information relative aux descriptifs des cours

Il est à noter que les étudiants ont des tuteurs désignés qu'ils peuvent consulter, de même qu'il existe une séance d'accueil pour leur donner des informations sur les cours offerts.

Nous avons ensuite demandé aux participants s'ils requièrent l'aide ou les recommandations de leurs collègues, professeurs, ou amis avant de choisir les cours. D'après les résultats dans la figure 5.20, nous constatons que 67.30% des participants le font souvent, 26.92% le font quelques fois, alors qu'uniquement 5.76% d'entre eux n'ont jamais demandé de suggestions à une autre personne.

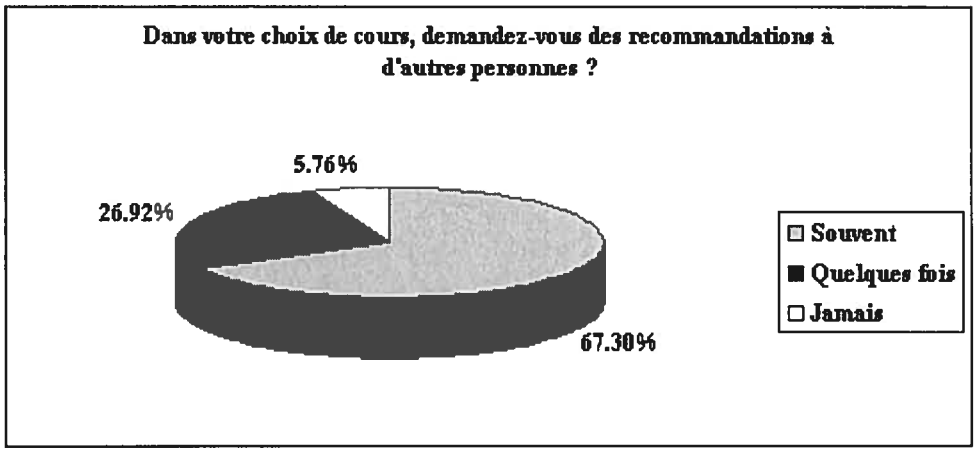


Figure 5.20 : Les recommandations issues des professeurs, collègues et amis

5.4.2 Deuxième partie de l'évaluation

La deuxième partie de l'évaluation vise à connaître les avis des 52 utilisateurs sur le système à savoir la *clarté de sa problématique*, la *pertinence de ses recommandations* ainsi que son *utilité dans l'orientation des étudiants*.

La réponse à la première interrogation était très positive. La totalité des participants (100%) ont affirmé que *RARE* traite une problématique claire, bien ciblée et très importante.

En outre, nous avons demandé aux étudiants : « *est-ce que RARE vous a aidé à repérer les cours qui pourraient vous intéresser* ? ». En réponse à cette question, 80.76% des étudiants ont confirmé que sur la liste des recommandations qu'ils ont eues, figuraient des cours qu'ils ont suivis ou que la plupart de leurs collègues du laboratoire de recherche ont pris (figure 5.21). Par conséquent, les participants ont exprimé une confiance envers le système. Les 19.23% des étudiants qui ont répondu à cette question par « peut être », ont dit qu'ils ne s'attendaient pas à certaines recommandations. Malgré cela, ils sont

arrivés à donner des explications à ces résultats à savoir : la diversité des intérêts des étudiants et la multitude des relations qui peuvent lier les différents domaines de recherche.

Par exemple, dans le cas des étudiants qui ont suivi le cours *IFT6310* (*Génie logiciel avancé* », *RARE* a recommandé le cours *IFT6042* (*Synthèse d'images*). Malgré que quelques uns n'aient pas été intéressés par cette recommandation, ils ont admis que ce cours peut être utile dans un volet de recherche en génie logiciel comme dans le cas de la représentation graphique des données numériques. Celle-ci sert à visualiser des métriques de logiciels comme le couplage de classes, la complexité du code, etc.

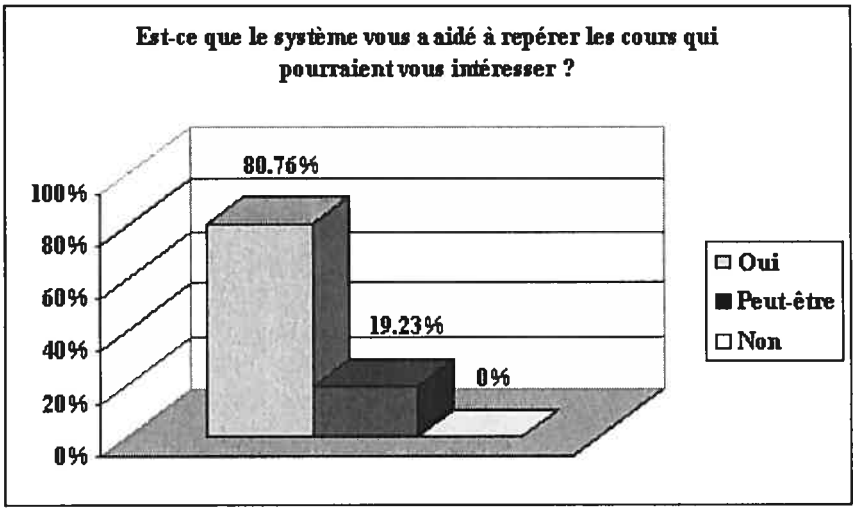


Figure 5.21 : La pertinence des recommandations de *RARE*

Pour avoir plus d'informations sur la satisfaction des participants, nous avons posé la question : « *est-ce que le système est utile dans l'orientation des étudiants ?* ». Une bonne majorité (71.15%) des participants a répondu par oui. 23.07% d'entre eux ont trouvé que ce système peut être utile dans certains cas comme, par exemple, le cas des nouveaux étudiants. 5.76% de l'ensemble des participants trouvent que le système n'est pas d'une grande utilité aux étudiants, vu que ces derniers sont encadrés par leurs professeurs et leurs directeurs de recherche (figure 5.22).

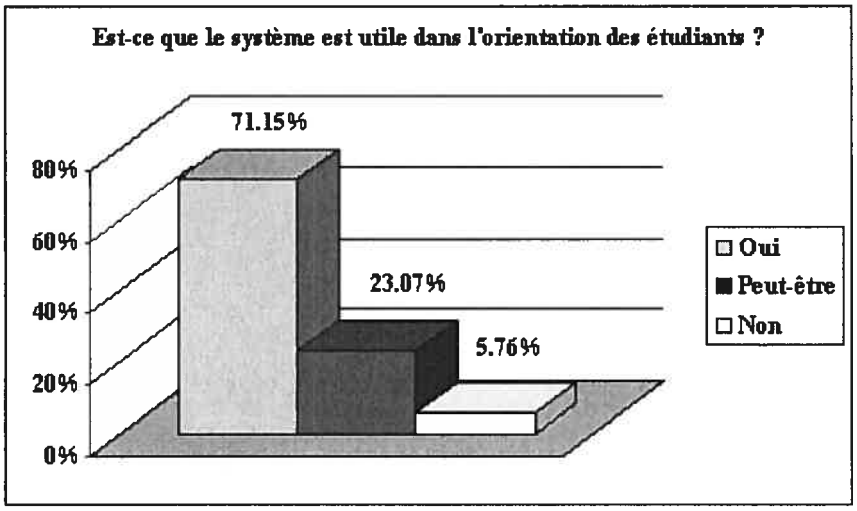


Figure 5.22 : L'utilité de *RARE*

5.4.3 Troisième partie de l'évaluation

La troisième partie, quant à elle, traite d'autres facteurs qui portent sur la *qualité de l'interface*, le *temps de la recommandation* et la *présentation des résultats*. L'utilisateur est invité à attribuer une note entre 1 et 10 à chaque facteur. La note 10 exprime la satisfaction parfaite d'un étudiant.

Vu que la facilité d'utilisation du système est liée à la qualité de son interface ainsi qu'à la facilité de navigation, nous avons interrogé les étudiants à ce propos. 71.15% des participants ont trouvé que la navigation sur le site est simple et son utilisation n'est pas compliquée. Ils ont alors attribué une note entre 8 et 10 à ces facteurs, et la moyenne de toutes les évaluations a été de 8.09 (figure 5.23).

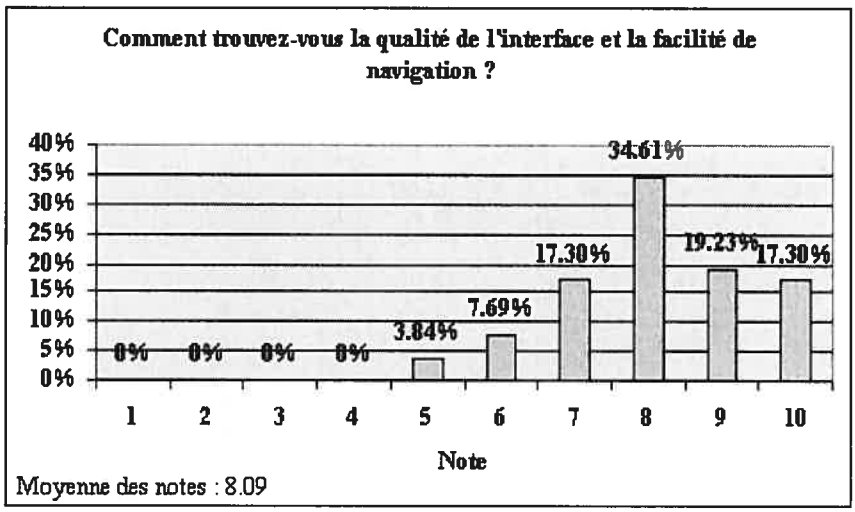


Figure 5.23 : La qualité de l'interface de *RARE* et la facilité de sa navigation

Un aspect intéressant pour les étudiants était le facteur *temps*. Ils ont confirmé qu'ils ne devaient pas passer beaucoup de temps à fournir des informations à *RARE* pour recevoir des résultats, et que le processus de production des recommandations était rapide. Ainsi, plus de leur moitié (59.61%) ont noté le facteur temps à 10, et ceci en réponse à la question : « *est-ce que le temps requis pour la production des recommandations est convenable ?* » (figure 5.24).

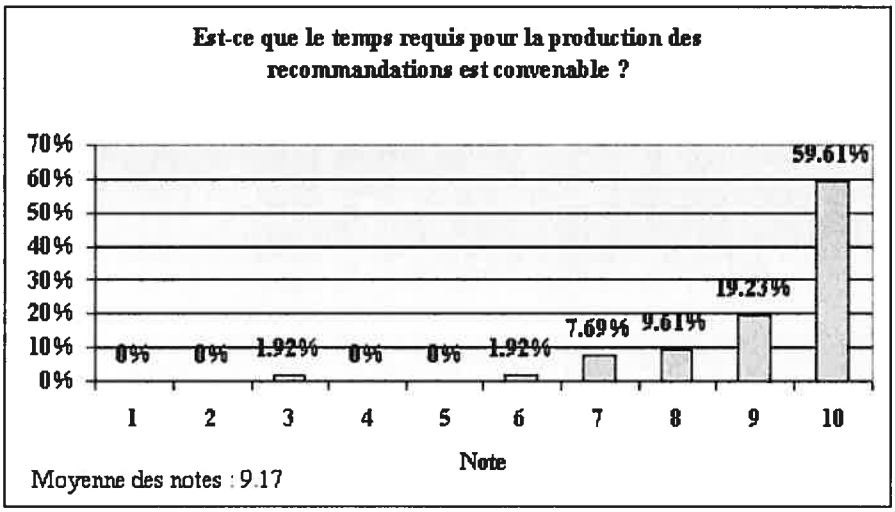


Figure 5.24 : Le temps requis par *RARE* pour la recommandation

Côté présentation des recommandations à savoir : l'utilisation des tableaux, des images et des couleurs, ainsi que les informations présentées pour chaque cours, nous pouvons voir dans la figure 5.25 que la moyenne des évaluations de ce facteur a été de 8.51.

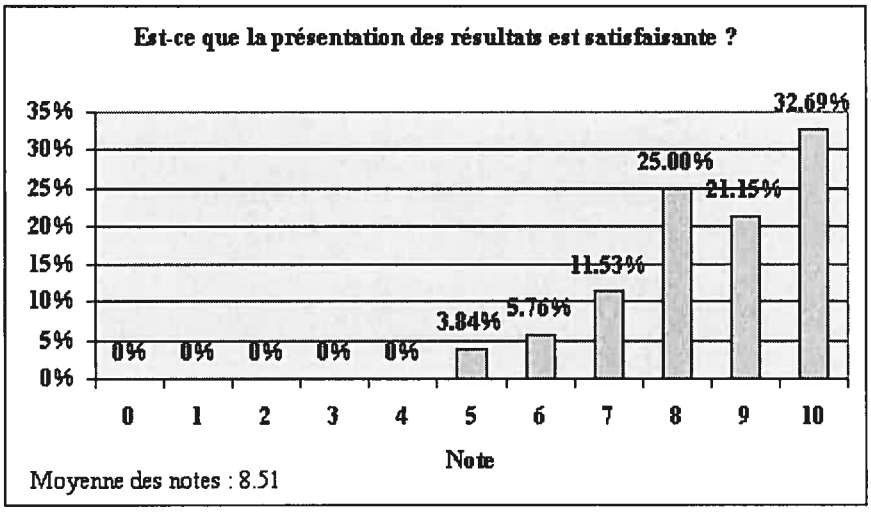


Figure 5.25 : La présentation des résultats de *RARE*

En résumé, nous avons demandé aux étudiants s'il est profitable d'intégrer un tel système dans les procédures administratives. Là aussi, 98.07% des réponses étaient positives et les étudiants ont trouvé l'idée de disposer d'un système de recommandation de cours très intéressante, et ceci afin de leur offrir plus d'informations sur les cours ainsi qu'un meilleur service (figure 5.26).

À la fin de l'évaluation, nous avons permis aux participants de donner leurs commentaires et leurs suggestions sur le système pour apporter des corrections s'il ya lieu.

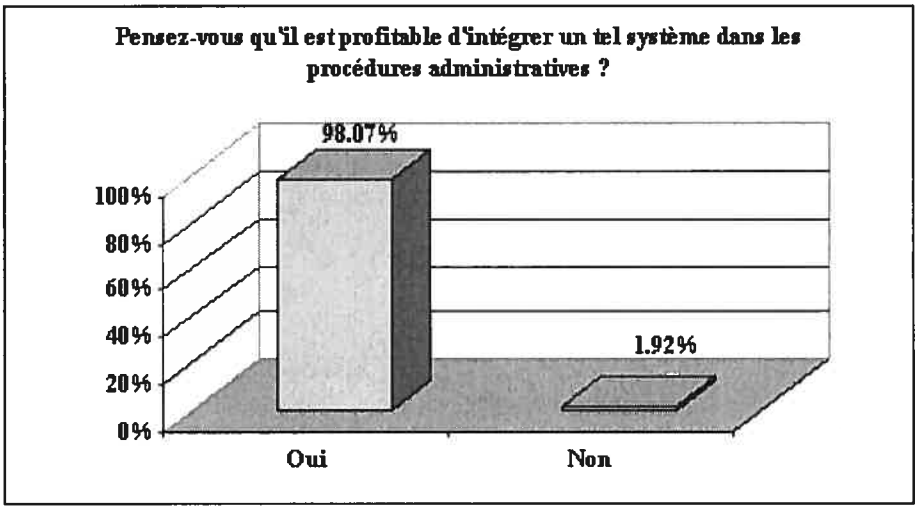


Figure 5.26 : L'apport de *RARE* aux étudiants

Cette évaluation montre que notre système, *RARE*, a répondu au problème de choix de cours auquel les étudiants font souvent face. En effet, le système a pu aider les utilisateurs à repérer les cours appropriés. De ce fait, en disposant d'un système de recommandation de cours comme *RARE*, les utilisateurs peuvent économiser le temps consacré à la recherche des informations sur les cours, et ceci en profitant de l'expérience des autres étudiants.

5.5 Conclusion

Au cours de ce chapitre, nous avons décrit la réalisation de *RARE* et ceci à travers la présentation de tous les détails de son implémentation. Nous avons également effectué une évaluation expérimentale des règles obtenues dans la phase de forage de données (voir section 5.2.4) avant de les intégrer dans le processus de recommandation. Cette première évaluation nous a permis de sélectionner l'ensemble de règles qui permettent un bon recouvrement et une meilleure exactitude de recommandation. Les différentes fonctionnalités de *RARE* ont été présentées à travers un scénario de recommandation. Selon les différentes façons proposées dans l'utilisation des règles, le système peut répondre aux besoins des différents types d'utilisateurs. De même, les utilisateurs jouent un rôle très important dans l'enrichissement des recommandations et l'amélioration de la performance du système. À la fin de l'implémentation, une autre évaluation de *RARE* a

été effectuée pour s'assurer de l'efficacité du fonctionnement général du système. Celle-ci a montré que les étudiants qui ont testé *RARE* l'ont jugé d'une grande utilité et ont trouvé ses recommandations adéquates. Le chapitre suivant résume notre travail et discute de ses forces et faiblesses.

Chapitre 6 : Conclusion

Tout au long de ce mémoire, notre objectif principal était de concevoir et d'implémenter un système de recommandation pour faire face au problème de choix de cours qui touche les étudiants. Afin d'atteindre cet objectif, nous avons opté pour la recommandation basée sur le forage de données. Ainsi, dans une première étape, nous nous sommes intéressés aux techniques de forage de données afin d'avoir la capacité de comprendre aisément les données des étudiants inscrits, et d'employer efficacement les informations qu'elles contiennent. Nous avons collecté des données réelles de 230 étudiants qui se sont inscrits au programme de maîtrise du Département d'Informatique de l'Université de Montréal entre les années 1990 et 1999. Ces données nous ont permis d'extraire des règles d'association qui décrivent le comportement de ces étudiants dans leurs choix de cours.

Dans une première évaluation expérimentale à laquelle 40 étudiants ont participé, nous avons testé l'efficacité des règles en calculant leur recouvrement et leur exactitude dans les recommandations qu'elles ont produites pour les participants. Ainsi, parmi toutes les règles extraites, nous avons pu choisir celles qui permettent le recouvrement et l'exactitude les plus élevés. Nous sommes conscients que nous ne disposions pas d'assez de données et que les règles que nous avons extraites ne sont pas les plus optimales. Toutefois nous pensons que ce système constitue un premier pas dans l'optique de l'application du forage de données dans la recommandation de cours.

Bien que les règles permettent de fournir des recommandations bénéfiques, les utilisateurs de *RARE* jouent un rôle très important dans l'amélioration de sa recommandation. En effet, ceux-ci peuvent évaluer tous les cours qu'ils suivent. Le traitement de évaluations aide *RARE* à ajuster les règles et adapter les recommandations aux préférences des étudiants. Par conséquent, les règles d'association utilisées par *RARE* ne sont plus basées exclusivement sur les deux critères de support et de confiance sur lesquels s'appui l'algorithme *Apriori*. Bien au contraire, elles sont consolidées au fur et à mesure par l'intégration de certains critères additionnels à savoir la fidélité des

utilisateurs et leurs évaluations qui permettent le calcul des poids des cours. L'amélioration des règles permet également l'intégration des nouveaux cours dans les règles, ainsi que leur recommandation s'ils sont bien reçus par les étudiants. Si tous les cours dans le conséquent d'une règle sont supprimés, la règle sera éliminée de la base des règles. Grâce à toutes ces opérations, la recommandation de *RARE* peut s'enrichir et devenir de plus en plus robuste.

Après avoir achevé ce travail, 52 étudiants de l'Université de Montréal ont pris part à l'évaluation générale du système. En réponse aux questions que nous avons posées, 92.30% des participants ont confirmé avoir eu une certaine difficulté à choisir les cours à suivre pendant leurs études et ont, par conséquent, demandé l'aide ou la recommandation de cours à d'autres personnes. Ainsi, ces participants ont confirmé que *RARE* est un outil nécessaire pour l'orientation des étudiants. En plus, l'évaluation a abouti à des résultats prometteurs. Elle a montré que les techniques mises en œuvre dans ce système lui ont permis de générer des recommandations de cours adéquates et utiles aux utilisateurs. En effet, avec ce système, les étudiants pourraient économiser le temps consacré à la recherche d'information sur les cours en profitant de l'expérience des anciens et des nouveaux étudiants. Ainsi, *RARE* constitue une solution efficace pour partager l'information entre les étudiants. Pour ceux qui voudraient le tester, *RARE* est disponible sur l'adresse suivante : http://wwww-etud.iro.umontreal.ca:8080/~bendakin

6.1 Comparaisons

Dans le chapitre 3, nous avons présenté un certain nombre de systèmes de recommandation de cours : *SCR* [Ekdahl *et al.*, 2002], *Course Recommender* [Simbi, 2003], *PEL-IRT* [Chen *et al.*, 2005] et *AACORN* [Sandvig *et al.*, 2006]. Afin de mettre en évidence les forces et les faiblesses de *RARE*, nous le comparons à ces systèmes dans le tableau 6.1, et ceci en se basant sur quelques critères en rapport avec la recommandation.

Tableau 6.1 : Comparaison des systèmes de recommandation de cours

Caractéristique	Système de recommandation				
	SCR	***Course recommender***	***PEL-IRT***	***AACORN***	***RARE***
Méthode utilisée	La modélisation bayésienne.	Algorithmes de filtrage collaboratif.	Théorie des réponses aux items.	Raisonnement à base de cas.	Une méthode hybride de forage de données et des évaluations des étudiants.
Problème de démarrage à froid	Oui	Oui	Lors de la première session de travail d'un utilisateur, *PEL-IRT* lui fournit des recommandations non-personnalisées.	Exige un historique partiel de cours suivis par l'utilisateur.	Non
Problème de données à grande échelle (scalability)	Oui	Oui	Oui	Oui	Non
Facteurs considérés dans la recommandation	Le programme d'étude, la spécialisation, les sessions des cours à recommander, le filtrage des cours non désirés par l'utilisateur.	Les votes des utilisateurs.	La difficulté des matériels des cours et les capacités des étudiants.	La similarité entre les cours suivis et les cas (dans la base des cas), le filtrage des cours non offerts dans la session prochaine et les cours non requis dans le programme de l'utilisateur.	La similarité entre les cours suivis et les règles d'association (dans la base des règles), les domaines de recherche et les évaluations des utilisateurs.
Traitement du feedback des utilisateurs	Non	Non	Oui	Non	Oui

6.1.1 Forces

À travers ce tableau, nous constatons que notre système remplit la plupart des critères de comparaison :

- Alors que les trois systèmes *SCR*, *Course Recommender* et *PEL-IRT* nécessitent une période d'apprentissage pour suggérer des recommandations appropriées, *RARE* se base sur les règles extraites durant le processus de forage de données. La force de cette approche réside dans la capacité de recommander des cours dès le démarrage du système (*RARE*). Ceci veut dire que ce dernier possède une meilleure performance vis-à-vis du problème de démarrage à froid.

- Contrairement à *SCR*, *Course Recommender* et *AACORN*, *RARE* fournit des recommandations aux nouveaux étudiants ne disposant pas d'un historique partiel de cours suivis.

- Un autre problème dont souffrent les autres systèmes de recommandation de cours est le problème de données à grande échelle (*scalability*). Quand la quantité de données à traiter est significative, *SCR*, *Course Recommender*, *PEL-IRT* et *AACORN* demandent plus de ressources que *RARE*. Par exemple, pendant une recommandation en temps réel, *Course Recommender* dans une des méthodes qu'il a implémentées, doit calculer la similarité entre l'utilisateur actuel et tous les autres utilisateurs dans la base des utilisateurs. Dans le cas de *RARE*, la totalité du processus de forage de données est exécutée hors ligne. Par conséquent, le traitement d'une quantité de données plus volumineuse ne requiert pas plus de ressources. Nous avons mentionné d'ailleurs, que la réussite des techniques de forage de données dépend potentiellement de leur volume, et plus celui-ci croît, plus les connaissances découvertes sont intéressantes.

- Il n'y a que *PEL-IRT* et *RARE* qui permettent à leurs utilisateurs d'évaluer explicitement les recommandations. Ce feedback est par la suite intégré dans le calcul

des recommandations, ce qui permet d'améliorer la qualité et d'augmenter l'utilité des résultats futurs.

- Un autre avantage de *RARE* est que le temps de calcul des recommandations est plus court. Ceci peut être expliqué par le fait que la recommandation ne nécessite qu'un balayage de la base des règles. Bien que le nombre des règles d'association à extraire se développe exponentiellement avec le nombre des items [Schafer, 2005], c.à.d. le nombre des cours dans le cas de notre système, il est possible de produire un ensemble compact de règles significatives en ajustant convenablement le support et la confiance, sur lesquels se base l'algorithme *Apriori*. Ainsi, malgré le fait que la dérivation des règles pendant la phase hors ligne puisse être longue, leur balayage durant la recommandation se fait efficacement.

6.1.2 Faiblesses

Vu que *SCR*, *PEL-IRT* et *AACORN* tiennent compte d'autres critères à savoir la disponibilité des cours dans une session, les cours non désirés par les utilisateurs et les catégories des cours, nous projetons l'amélioration de la performance de *RARE* en tenant compte de ces facteurs.

Comme nous l'avons déjà mentionné, notre approche dépend de la disponibilité d'un volume de données réelles. Plus ce volume est grand plus la recommandation devient intéressante. Cela peut se justifier par le fait que l'algorithme *Apriori* nécessite d'énormes jeux de données pour générer des règles d'association plus fortes.

6.2 Travaux futurs

Pour remédier aux faiblesses mentionnées, nous proposons quelques pistes de solutions comme travaux futurs. Nous pensons améliorer la performance du système en exploitant les informations sur les étudiants actuels (les utilisateurs du système). Il sera question d'appliquer l'algorithme *Apriori* sur les données de ces étudiants pour extraire davantage de règles. La base des utilisateurs actuels stocke toutes les informations nécessaires en prévision de ce travail futur. Le principe de la recommandation consistera alors, à mixer

les recommandations courantes avec celles issues des nouvelles règles. Toutefois, toutes les recommandations de cours devront être évaluées par les utilisateurs afin d'effectuer la mise à jour et l'enrichissement des règles. Comme nous l'avons déjà mentionné, nous prévoyons aussi tenir compte lors de la recommandation de la disponibilité des cours dans les sessions. Nous projetons également de permettre aux utilisateurs de fournir au système leurs explications ou leurs commentaires lors des évaluations de cours. Les commentaires sur un cours feront partie des informations fournies par le système lors de sa recommandation.

Nous suggérons la réutilisation de notre méthode dans la recommandation de cours pour d'autres programmes d'études (certificat, baccalauréat, etc.) à condition que des données des anciens étudiants soient disponibles. Nous pouvons appliquer l'approche telle quelle pour les programmes qui offrent plusieurs spécialités ou laboratoires de recherche. La segmentation des données permet d'identifier des groupes d'étudiants qui semblent avoir des préférences semblables et gagner, en conséquence, plus de précision dans l'information recherchée. Pour les programmes qui ne disposent pas de spécialités, la segmentation des données n'est pas nécessaire. Toutefois, l'algorithme *Apriori* peut être appliqué sur l'ensemble entier de données et les règles extraites peuvent être employées par la même méthode utilisée dans *RARE*.

Bibliographie

[Aamodt *et al.*, 1994] Aamodt, A., Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, vol.7, no.1, 39-59.

[Adomavicius *et al.*, 1999a] Adomavicius, G., Tuzhilin, A. (1999). User Profiling in Personalization Applications through Rule Discovery and Validation. In *Proceedings of the Fifth International Conference on Data Mining and Knowledge Discovery, San Diego, California*, 377-381.

[Adomavicius *et al.*, 1999b] Adomavicius, G., Tuzhilin, A. (May 1999). Integrating User Behavior and Collaborative Methods in Recommender Systems. In *CHI'99 Workshop on Interacting with Recommender Systems, Pittsburgh, PA, USA.*

[Adomavicius *et al.*, 2001] Adomavicius, G., Tuzhilin, A. (2001). Using Data Mining Methods to Build Customer Profiles. *IEEE Computer*, vol. 34, no. 2, 74-82.

[Adomavicius *et al.*, 2005] Adomavicius, G., Tuzhilin, A. (2005). Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, 734-749.

[Agrawal *et al.*, 1993] Agrawal, R., Imielinski, T., Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Int'l Conf. on Management of Data (ACM SIGMOD '93)*, Washington, USA, 207-216.

[Agrawal *et al.*, 1994] Agrawal, R., Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of 20th International Conference on Very Large Data Bases*, Santiago, Chile, 487-499.

[Baeza-Yates *et al.*, 1999] Baeza-Yates, R., Ribeiro Neto, B. (May 1999). Modern Information Retrieval. *ACM Press / Addison-Wesley, USA*, 262-267.

[Balabanovic *et al.*, 1997] Balabanovic, M., Shoham, Y. (1997). Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, vol. 40, no. 3, 66-72.

[Belkin *et al.*, 1992] Belkin, N. J., Croft, W. B. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, vol.35, no. 12, 29-38.

[Bendakir *et al.*, 2006] Bendakir, N., Aïmeur, E. (2006). Using Association Rules for Course Recommendation. *Workshop on Educational Data Mining. Conference AAAI-2006 (Twenty-first National Conference on Artificial Intelligence)*, Boston, Massachusetts USA, 31- 40.

[Berkhin, 2002] Berkhin, P. (2002). Survey of clustering data mining techniques. *Technical report, Accrue Software*, San Jose, CA.

[Berrut *et al.*, 2003] Berrut, C., Denos, N. (2003). Filtrage collaboratif. *Assistance intelligente à la recherche d'information*, Hermes - Lavoisier, chapitre 8.

[Berry *et al.*, 2004] Berry, M. J. A., Linoff, G. S. (2004). Data Mining Techniques For Marketing, Sales, and Customer Relationship Management. *Edition WILEY.*

[Berson *et al.*, 2000] Berson, A., Smith, S., Thearling, K. (1999). Building Data Mining Applications For CRM, *McGraw-Hill.*

[Billsus *et al.*, 1997] Billsus, D., Pazzani, M. J. (1997). Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, vol. 27, no. 3, 313-331.

[Billsus *et al.*, 1998] Billsus, D., Pazzani, M. J. (1998). Learning Collaborative Information Filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 46-54.

[Bottraud *et al.*, 2003] Bottraud, J. C., Bisson, G., Bruandet, M.F. (2003). An Adaptive Information Research Personal assistant. In *Workshop AI2IA (Artificial Intelligence, Information Access and Mobile Computing) of IJCAI (International Joint Conference on Artificial Intelligence)*, Acapulco, Mexico, 48-58.

[Breese *et al.*, 1998] Breese, J., Heckerman, D., Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty inArtificial Intelligence*, Madison, USA, 43-52.

[Breiman *et al.*, 1984] Breiman, L., Friedman, J. H., Olshen, R., Stone, C. (1984). Classification and Regression Trees. *Wadsworth International Group*, Monterey, CA.

[Burke, 2002] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User Adapted Interaction*, vol. 12, no 4, 331-370.

[Cabena *et al.*, 1998] Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., Zanasi, A. (1998). Discovering Data Mining: From Concept to Implementation. *Prentice Hall*, Upper Saddle River, NJ.

[Candillier, 2004] Candillier, L. (2004) La classification non supervisée. *Équipe GRAppA*, Lille 3, Pertinence. http://www.grappa.univ-lille3.fr/~candillier/

[Chen, 2002] Chen, Z. (2002). Intelligent Data Warehousing: From Data Preparation to Data Mining. *CRC Press LLC.*

[Chen *et al.*, 2004] Chen, S. Y., Liu, X. (2004). The contribution of data mining to information science. *Journal of Information Science*, vol. 30, no. 6, 550-558.

[Chen *et al.*, 2005] Chen, C. M., Lee, H. M., and Chen, Y. H. (2005). Personalized E-Learning System Using Item Response Theory. *Computers and Education*, vol. 44, no.3, 237-255.

[Cho *et al.*, 2002] Cho, T. H., Kin, J. K., Kim, S. H. (2002). A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, vol. 23, no.3, 329-342.

[Clifton *et al.*, 2001] Clifton, C., Thuraisingham, B. (2001). Emerging standards for data mining. *Computer Standards and Interfaces*, Elsevier Science, Amsterdam, vol. 23, no. 3, 187-193.

[Edelstein, 2000] Edelstein, H. (2000). Building profitable customer relationships with data mining. *Two Crows Corporation, SPSS white paper-executive briefing.* http://www.twocrows.com.

[Edelstein, 2001] Edelstein, H. (March 2001). Pan For Gold In The Clickstream. *Information Week.* http://www.informationweek.com/828/prmining.htm

[Edelstein, 2003] Edelstein, H. (March 2003). Data Mining In Depth: Description is Not Prediction. *DM Review.* http://www.dmreview.com/article_sub.cfm?articleId=6388

[Ekdahl *et al.*, 2002] Ekdahl, M., Lindström, S., Svensson, C. (2002). A Student Course Recommender. Master of Science Programme, LULEA University of Technology, ECHNOLOGY, Department of Computer Science and Electrical Engineering / Division of Computer Science and Networking.

[Fayyad *et al.*, 1996] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, vol. 39, no.11, 27-34.

[Fayyad *et al.*, 2002] Fayyad, U., Uthurusamy, R.(2002). Evolving data mining into solutions for insights: Introduction. *Communications of the ACM*, vol. 45, no. 8, 28-31.

[Gallardo-lopez, 2002] Gallardo-lopez, L. (2002). Une première approche de la qualité dans les systèmes de filtrage collaboratif. In *INFORSID* , Nantes, 427-428.

[Géry *et al.*, 2003] Géry, M., Haddad, H. (2003). Evaluation of Web Usage Mining approaches for user's next request prediction. In *Fifth International Workshop on Web Information and Data Management*, Lousiane, USA, 74-81.

[Goebel *et al.*, 1999] Goebel, M., Gruenwald, L. (1999). A survey of data mining and knowledge discovery software tools. *SIGKDD Explorations*, vol. 1, no. 1, 20–33.

[Goldberg *et al.*, 1992] Goldberg, D., Nichols, D., Oki, B. M., Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, vol. 35, no. 12, 61-70.

[Groth, 1999] Groth, R. Data Mining: Building Competitive Advantage. (1999). *Prentice Hall.*

[Han et al., 2001] Han, J., Kamber, M. Data Mining: Concepts and Techniques. (2001). *Morgan Kaufmann Publishers.*

[Hartigan, 1975] Hartigan, J. A. (1975). Clustring Algorithms. *John Wiley & Sons*, NY.

[Herlocker *et al.*, 1997] Herlocker, J. L., Konstant, J.A., Miller, B. N., Maltz, D., Gordon, L. R., Riedl, J. (1997). GroupLens: Applying collaborative filtering to usenet new, *Communications of the ACM*, vol.3, no. 40, 77-87.

[Herlocker *et al.*, 2000] Herlocker, J. L., Konstan, J. A., Riedl, J. T. (2000). Explaining Collaborative Filtering Recommandations. In *Proceedings of the ACM 2000 conference on Computer supported cooperative work (CSCW'00)*, Philadelphie, Pennsylvanie, 241-250.

[Kantardzic, 2002] Kantardzic, M. (2002). Data Mining: Concepts, Models, Methods, and Algorithms. *IEEE Press & John Wiley.*

[Karypis, 2001] Karypis, G. (2001). Evaluation of Item-Based Top-*N* Recommendation Algorithms. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM)*, Atlanta, GA, 247-254.

[Larose, 2005] Larose, D. T. (2005). Discovering Knowledge In Data: An Introduction to Data Mining. *Edition WILEY.*

[Lewis, 1991] Lewis, D. (1991). Learning in intelligent information retrieval. In *Proceedings of the Eighth International Workshop on Machine Learning*, Chicago, 235-239.

[Li *et al.*, 2004] Jia Li, J., Zaïane, O. R. (2004). Combining Usage, Content and Structure Data to Improve Web Site Recommendation. In *5th International Conference on Electronic Commerce and Web Technologies (EC-Web 2004)*, 305-315.

[Lu, 2004] Lu, J. (2004). A Personalized e-Learning Material Recommender System. In *Proceedings of the 2nd International Conference on Information Technology for Application (ICITA).* Macquarie Scientific Publishing , Harbin, China , 374-379.

[Mena, 1998] Mena, J. (January 1998). Data Mining FAQs. *DM Review.* http://www.dmreview.com/article_sub.cfm?articleId=792

[Mendonca et al., 1999] Mendonca, M., Sunderhaft, N. L. (1999). Mining Software Engineering Data: A Survey. *A DACS State-of-the-Art Report*, Rome, NY. http://www.thedacs.com/techs/datamining/

[McSherry, 2004] McSherry, D. (2004). Explanation in Recommender Systems. In *Proceedings of the ECCBR 2004 Workshops*, Universidad Complutense Madrid, 125-134.

[Mitra et al., 2003] Mitra, S., Acharya, T. (2003). Data Mining: Multimedia, Soft Computing, and Bioinformatics. *John Wiley*, New York.

[Mooney *et al.*, 2000] Mooney, R. J., Roy, J. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, San Antonio, USA, 195-204.

[Murthy *et al.*, 1993] Murthy, S., Kasif, S., Salzberg, S., Beigel, R. (1993). OC1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, MenloPark, MIT Press, 322-327.

[Noonan, 2000] Noonan, J. (July 2000). Data Mining Strategies. *DM Review*. http://www.dmreview.com/article_sub.cfm?articleId=2367

[Owen, 1998] Owen, C. (November 1998). Data Modeling, Data Warehousing, and Data Mining: How To Make Your Data Work For You Like Never Before! *DM Review*. http://www.dmreview.com/article_sub.cfm?articleId=297

[Pazzani, 1999] Pazzani, M. J. (1999). A Framework for Collaborative, Content-based and Demographic Filtering. *Artificial Intelligence Review*, vol. 13, no. 5-6, 393-408.

[Perugini *et al.*, 2004] Perugini, S., Gonçalves, M, A., Fox, E. A. (2004). Recommender Systems Research: A Connection-Centric Survey. *Journal of Intelligent Information Systems*, vol. 23, no. 2, 107-143.

[Quinlan, 1993] Quinlan, J, R. (1993). C4.5: Programs for Machine Learning, *Morgan Kaufman Publishers*.

[Quinlan, 1986] Quinlan, J, R.(1986). Induction of Decision Trees. *Machine Learning*, 81-106.

[Rafaeli *et al.*, 2005] Rafaeli, S., Dan-Gur, Y., and Barak, M. (2005). Social Recommender Systems: Recommendations in Support of E-Learning. *Journal of Distance Education Technologies*, vol.3, no. 2, 29-45.

[Resnick *et al.*, 1997] Resnick, P., Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, vol. 40, no. 3, 56-58.

[Sandvig *et al.*, 2006] Sandvig, J., and Burke, R. (2006). AACORN: A CBR Recommender for Academic Advising. School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, USA. (Currently in development).

[Sarwar *et al.*, 2001] Sarwar, B. M. , Karypis, G., Konstan, J. A., Riedl, J. T. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, 285-295.

[Shannon, 1948] C. E. Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, vol. 27, 379-423 and 623-656.

[Shearer, 2000] Shearer, C. (2000).The CRISP-DM Model: The new Blueprint for Data Mining. *Journal of Data Warehousing*, vol. 5, no. 4, 13-22.

[Schafer, 2005] Schafer, J.B. (2005). The Application of Data Mining to Recommender Systems, Published in *Encyclopaedia of Data Warehousing and Mining*, Wang, J. (Editor), Information Science Publishing, 44-48.

[Schwaighofer *et al.*, 2003] Schwaighofer, A., Yu, K., Tresp, V., Ma,W-Y., Zhang, H. (2003). Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical Bayes. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, 616-623.

[Simbi, 2003] Simbi, P. (2003). Course Recommender. Senior Thesis, Princeton University.

[Stadnyk *et al.*, 1992] Stadnyk, I., Kass, R. (1992). Modeling Users' Interests in Information Filters. *Communication of the ACM*, vol. 35, no.12, 49-50.

[Swearingen *et al.*, 2001] Swearingen, K., Sinha, R. (2001). Comparing Recommendations Made by Online Systems and Friends. In *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems*, Dublin, Ireland, 18-20.

[Swearingen *et al.*, 2002] Swearingen, K., Sinha, R. (2002). The Role of Transparency in Recommender Systems. In *Proceedings of ACM CHI 02 Conference on Human Factors in Computing Systems Conference Companion*, 830-831.

[Tchienehom, 2004] Tchienehom, P. (2004). Architecture de Recherche et de Recommandation d'Information à base de Profils : définitions, acquisitions et usages de profils. Dans : *22ième Congrès National Inforsid'04, Biarritz*, Editions Inforsid, 143-159.

[Thearling, 2000] Thearling, K (December 2000). Data Warehousing. *Published on hr.com*. http://www.thearling.com/text/hrdotcom/dw.htm.

[Thearling, 2005a] Thearling, K. An Introduction to Data Mining: Discovering hidden value in your data warehouse. http://www.thearling.com/text/dmwhite/dmwhite.htm

[Thearling, 2005b] Thearling, K. Scoring Your Customers. http://www.thearling.com/text/scoring/scoring.htm.

[Tollari, 2003] Tollari, S. (2003). Rehaussement de la classification textuelle d'une base de données photographiques par son contenu visuel. Mémoire de DEA, Université du Sud Toulon-Var, Laboratoire LSIS - Equipe INCOD, France.

[Witten et al, 2005] Witten, I., Frank, E. 2005. Data Mining: Practical machine learning tools and techniques (2nd Edition). Morgan Kaufmann.

[Wu, 2000] Wu, J. (August 2000). Business Intelligence: What is Data Mining? *DM Review*. http://www.dmreview.com/article_sub.cfm?articleId=2582

[Zaima, 2003] Zaima, A (2003). Five myths of data mining. vol. 15, May 2003. www.dw-institute.com/research/display.asp?id=6674

[Zaima et al., 2003] Zaima, A., Kashner, J. (2003). Data Mining Primer for the Data Warehouse Professional. *Business Intelligence Journal*, vol. 8, no. 2.

[Zaïane, 2004] Zaïane, O. R. (2004). Communication personnelle.

URLs

[URL1] *Data Mining Techniques*. http://www.statsoftinc.com/textbook/stdatmin.html

[URL2] Brand, E., Gerritsen, R. 1998. *Data Mining and Knowledge Discovery*. http://www.exclusiveore.com/

[URL3] Two Crows Corporation, 1999. *Introduction to Data Mining and Knowledge Discovery*. http://www.twocrows.com/booklet.htm

[URL4, 2005] http://www.olapreport.com

[URL5] *CRoss Industry Standard Process for Data Mining* (DEC 1999). http://www.crisp-dm.org

[URL6] http://dms.irb.hr/index.php

[URL7] http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/4_dtrees1.html

[URL8] http://www2.cs.uregina.ca/~dbd/cs831/notes/itemsets/itemset.html