Using Association Rules for Course Recommendation

Narimel Bendakir and **Esma Aïmeur**
Département d'informatique et de recherche opérationnelle, Université de Montréal
Pavillon André-Aisenstadt, CP 6128 succ. Centre-Ville
Montréal (Québec), H3C 3J7, Canada
{bendakin, aimeur}@iro.umontreal.ca

Abstract

Students often need guidance in choosing adequate courses to complete their academic degrees. 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, which incorporates a data mining process together with user ratings in recommendation. Starting from a history of real data, it discovers significant rules that associate academic courses followed by former students. These rules are later used to infer recommendations. In order to benefit from the current students' opinions, *RARE* also offers to users the possibility to rate the recommendations, 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 recommend the most relevant courses to its users.

Introduction

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 friends and colleagues' recommendations that the majority of them choose their courses and register. It would be useful to help students in finding courses of interest by the intermediary of a recommender system. The proposed system is based on the same principle that consists of taking advantage of the collaborative experience of the students who have finished their studies. Since the volume of data concerning registered students keeps increasing, applying data mining to interpret this data can reveal hidden relations between courses followed by students. Once interesting results are discovered, a course recommender system can use them to predict the most appropriate courses for current students. As the users rate the recommendations thus provided, system performances can be improved.

This paper presents *RARE*, a course recommender system, and focuses on the effectiveness of the incorporation of data mining in course recommendation. The remainder of this paper is structured as follows. We start by presenting a state of the art in course recommender systems, and then define the term data mining and depict the steps to be followed in its process. Afterwards, we illustrate the architecture of *RARE* and provide its implementation details, including an evaluation of the rules used in recommendation inference. We then give a recommendation scenario. Finally, we compare *RARE* to other course recommender systems before concluding the paper.

Recommender Systems

The most common recommender systems base their recommendations on user ratings for items (Adomavicius & Tuzhilin 2005). In order to estimate the usefulness of unrated items for a given user, several filtering techniques are used. In particular, we can use *content-based filtering* (Perugini, Gonçalves, & Fox 2004), *collaborative filtering* (Herlocker, Konstan, & Riedl 2000) and *hybrid filtering* (Burke 2002). Content-based filtering systems recommend items to the user based on his past preferences. In contrast, collaborative filtering systems base their recommendation for a given user on the past preferences of other users who share his tastes. In the first stage of the recommendation process, both the content-based filtering and the collaborative filtering systems suffer from the *cold start problem* (Adomavicius & Tuzhilin 2005; Rafaeli, Dan-Gur, & Barak 2005). For the former, accurate recommendations depend on a training period in order to detect user preferences. For the latter, successful recommendations depend on the availability of a critical mass of users. In order to leverage the benefits of both techniques, hybrid techniques, which combine content-based and collaborative filtering, have been suggested. Another approach to recommendation is *data mining based recommendation* (Adomavicius & Tuzhilin 2001; Schafer 2005). Instead of using user ratings, this approach learns behavioural rules or models from a history of stored data and makes recommendations based on the learned rules.

To the best of our knowledge, only a few course recommender systems have been developed: ***SCR*** (Ekdahl, Lindström, & Svensson 2002), ***Course Recommender*** (Simbi 2003), ***AACORN*** (Sandvig & Burke 2006), and ***PEL-IRT***

(Chen, Lee, & Chen 2005). ***SCR***, which is an acronym for *Student Course Recommender*, suggests courses by using a strategy based on *Bayesian Network modelling*. The *SCR* network learns from the information stored about the students who have used the system. It requires the presence of enough cases in the student database. Therefore, if a user has not started or completed any courses, and is not pursuing any degree at the university, *SCR* cannot give him any course recommendation. The ***Course Recommender*** system is based on the following collaborative filtering algorithms: *user-based* (Breese, Heckerman, & Kadie 1998), *item-based* (Sarwar *et al.* 2001), *OC1* (Murthy *et al.* 1993), and a modified variant of *C4.5* (Quinlan 1993). The system can predict the usefulness of courses to a particular student based on other users' course ratings. To get accurate recommendations, one must evaluate as many courses as possible. Based on the evaluation results, the author suggests *C4.5* as the best algorithm for course recommendation. The system cannot predict recommendations for students who have not taken any courses at the University. The ***AACORN*** system, *Academic Advisor COurse Recommendation eNgine*, is currently in development. It is a case-based reasoning system that recommends courses to graduate students at DePaul CTI. The system uses the experience of past students and their course histories as the basis for course advising. In order to determine the similarities between course histories, the system uses a metric commonly used in bio-informatics called the edit distance. The system requires a partial history of the courses followed by a student before it can provide useful recommendations. Finally, ***PEL-IRT*** stands for *Personalized E-Learning system using Item Response Theory*. It recommends appropriate course material to students, taking into account both course material difficulty and student ability. When using *PEL-IRT*, students can select course categories and units and can use appropriate keywords to search interesting course material. Once the course material has been recommended to students and they have browsed through, the system asks them to answer two questionnaires. This explicit feedback is used by *PEL-IRT* to re-evaluate the students' abilities and adjust the course material difficulty used in the recommendation.

Data Mining

Data mining is a process that uses a variety of data analysis tools to discover patterns and relationships in the data, which can in turn be used to make predictions (Edelstein 2003). The CRoss Industry Standard Process for Data Mining (CRISP-DM) (Shearer 2000) breaks down the life cycle of a data mining project into six steps: *business understanding* (*objective determination*), *data understanding*, *data preparation*, *modeling*, *evaluation*, and *deployment*. The business understanding (objective determination) consists of defining the objectives of the application, converting this knowledge into a data mining task (Berry & Linoff 2004), and developing a preliminary plan to achieve the goals. The data understanding phase consists of the collection, description and exploration of data, and of data quality verification. The data preparation phase involves the selection, cleaning, construction, integration and formatting of the data. The modelling phase consists of selecting a technique, such as classification and association rules discovery, or a suitable combination of techniques, in order to establish, examine and choose the models. The evaluation phase serves as a checkpoint to ascertain that the model properly achieves the objectives. The final phase is the deployment phase, in which the knowledge gained is organized and presented in a way that makes sense to the user.

The next section describes the *RARE* system and explains how it uses data mining.

Architecture

Figure 1 highlights the architecture of *RARE*. This architecture is composed of two phases: an *off-line* phase, which consists of a data mining process, and an *on-line* phase, which constitutes the interaction of *RARE* with its users.

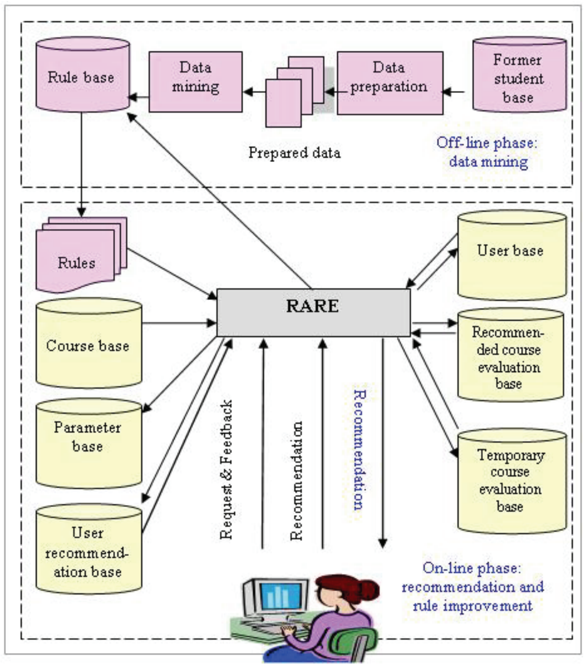


Figure 1: *RARE* architecture

The objective of the off-line phase is to extract rules from data that describes previous course selections from students. Once these rules are extracted, the on-line phase uses them to infer course recommendations. By applying these rules for recommendation, *RARE* avoids the cold start problem mentioned in the previous section. *RARE* stores the student profiles in a *User base*. These profiles enable the system to recognize users when they come back, and to ask them to evaluate previous recommendations. As more evaluations are provided, *RARE* enhances and rule updates. Several databases are used by *RARE*. The *Former Student base* contains the collected data prepared for the data mining process. The *Rule base* stores the rules generated by the data mining process. The *User base* keeps student information such as: the courses they followed, the courses that were recommended to them and their *fidelity*. Fidelity represents the number of times the user has returned to the system and provided evaluations. The fidelity of a new student is set to 1 and increases each time he reuses *RARE* and evaluates courses. The *Recommended course evaluation base* is used to keep the various values needed in the calculation of recommended course weights. The *Temporary course evaluation base* contains the various values needed in the calculation of temporary course weights. A temporary course is not a course recommended by the system, but it is a course followed frequently by students. The system asks for its evaluation. If it is well evaluated, it will be integrated into the set of the rules that are used. *RARE* uses predefined parameters, kept in the *Parameter base*, in order to integrate temporary courses or to remove recommended courses from the rules. The *Course base* stores course information and weights. The *User recommendation base* is used to store the courses recommended by users.

Implementation

This section gives more detail about the implementation of the off-line and the on-line phases of *RARE*.

Off-line Phase

To help us undertake this stage, the department of Computer Science at the Université de Montréal provided us with a set of anonymous real data. This data describes the graduate courses followed by 230 former students registered in the Master of Computer Science program between 1990 and 1999. Figure 2 shows a sample of the data. For each case (i.e. each student), the sex, the supervisor name, the courses followed, the years and the sessions of studies were given. For example, student 1 is male, with supervisor *Gilles Brassard*. Each course he followed is preceded by a combination of Year_Session, such as 1996_2, where (1996) stands for the year, and (2) for the winter session. Examples of courses he followed are: IFT2121 and IFT6180. They represent courses titled *Algorithmics* (*introduction*), and *Cryptology: theory and applications*, respectively.

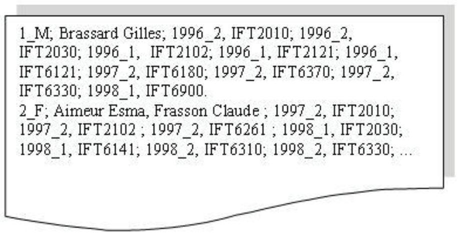

1_M; Brassard Gilles; 1996_2, IFT2010; 1996_2,
IFT2030; 1996_1, IFT2102; 1996_1, IFT2121; 1996_1,
IFT6121; 1997_2, IFT6180; 1997_2, IFT6370; 1997_2,
IFT6330; 1998_1, IFT6900.
2_F; Aimeur Esma, Frasson Claude ; 1997_2, IFT2010;
1997_2, IFT2102 ; 1997_2, IFT6261 ; 1998_1, IFT2030;
1998_1, IFT6141; 1998_2, IFT6310; 1998_2, IFT6330; ...

Figure 2: The collected data

Next, we provide detail of the steps of our data mining process according to the CRISP-DM standard (see data mining section).

Objective Determination (Step 1) There exist several ways to formulate the objectives of an application into a data mining task, such as classification, association rules discovery, clustering, etc. The classification and association rules are the frequently requested models in a data mining process. They constitute comprehensible and easily exploitable models by users (Larose 2005). Therefore, we first tried to classify data, and then to discover association rules.

Classification makes it possible to build a decision tree (or classification rules) that classifies each student. Since the information on the gathered data is centralized around the followed courses, we can build student profiles that represent the different paths of the chosen courses. Each path starts from the root, goes through nodes representing courses and leads to a leaf labelled with a research laboratory. The recommendation is performed by following the paths in the tree according to the courses followed by a student. If only one course has been followed, we look for the paths which start with this followed course. The remaining courses of these paths as well as the laboratories are recommended. If two courses have been followed by a student, we look for the paths which begin with these two courses. Then, we recommend the third and the fourth course as well as the laboratories. The same technique is used when three courses or more have been followed. If we have for example a path represented by the following classification rule:
IF course_one = IFT6243 (*Advanced database concepts*) **AND** course_two = IFT6261 (*Knowledge management*) **AND** course_tree = IFT6310 (*Advanced software engineering*) **THEN** heron_laboratory; then, the course recommendation for a student who has followed the course IFT6243 would be: *IFT6261* and *IFT6310*. The laboratory recommendation would be *heron_laboratory*.

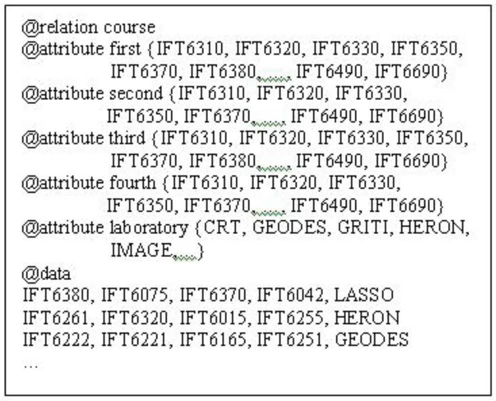
Discovering association rules enables us to know which courses are often followed simultaneously. In other words, the ultimate objective is to assess all the implications that can exist between the courses that are followed. The following depicts an example of an association rule:

$$IFT6243 \; and \; IFT6261 \; and \; IFT6310 \Rightarrow IFT6251$$

When using association rules, we infer recommendation by checking the similarity between the courses followed by students and the antecedents of the rules. The rules' consequents produce the courses which will be recommended.

According to the results obtained from the classification and the association rule discovery, the technique best adapted to our needs is selected and the rules thus obtained are used in course recommendation. We are aware than the rules extracted may not be the optimal ones, but they constitute a starting point as well as a way to circumvent the cold start problem. Moreover, these rules will be improved as more and more users interact with the system and evaluate recommendations.

Data Understanding and Preparation (Steps 2, 3) The first step of data cleaning results in the elimination of some data, such as the student's sex and the courses that are not given at the Université de Montréal. Courses that are not offered any more or those whose codes have been changed are substituted, if possible, by equivalent courses or equivalent codes. For example, the code *IFT6250* is substituted by the code *IFT6251*. The names of the followed courses and those of the supervisors are kept. However, information on the supervisors can reveal the student's membership in a laboratory. Thus, we build, starting from this information, a new attribute representing the students' research laboratories. In the Masters of Computer Science program at the Université de Montréal, a student has to follow four courses. Accordingly, the data is structured into five attributes, of which the first four are: *first course*, *second course*, *third course* and *fourth course*. Each one can take the name of the followed course belonging to the graduate course set: {IFT6310, IFT6320,..., IFT6690}. The fifth attribute represents the *research laboratory*. The data is stored in the *ARFF* format (Attribute-Relation File Format) (Witten & Frank 2005) in the *Course.ARFF* file. The ARFF format was adopted to present the data according to the needs of the applied algorithms in the modelling stage. Figure 3 illustrates the ARFF format file.



```
@relation course
@attribute first {IFT6310, IFT6320, IFT6330, IFT6350,
               IFT6370, IFT6380,..., IFT6490, IFT6690}
@attribute second {IFT6310, IFT6320, IFT6330,
              IFT6350, IFT6370,..., IFT6490, IFT6690}
@attribute third {IFT6310, IFT6320, IFT6330, IFT6350,
               IFT6370, IFT6380,..., IFT6490, IFT6690}
@attribute fourth {IFT6310, IFT6320, IFT6330,
              IFT6350, IFT6370,..., IFT6490, IFT6690}
@attribute laboratory {CRT, GEODES, GRITI, HERON,
               IMAGE,...}
@data
IFT6380, IFT6075, IFT6370, IFT6042, LASSO
IFT6261, IFT6320, IFT6015, IFT6255, HERON
IFT6222, IFT6221, IFT6165, IFT6251, GEODES
...
```

Figure 3: The ARFF format file

Modelling (Step 4) We used the *WEKA* (Waikato Environment for Knowledge Analysis) software (Witten & Frank 2005). WEKA is an open source package for machine learning and data mining developed at the University of Waikato in New Zealand. For data classification, the last attribute laboratory constitutes the class. We applied the *C4.5* algorithm (Quinlan 1993) (*J48* in WEKA) in order to build a decision tree that assigns a classification to each case. The tree we obtained had at most two levels and therefore was not deep enough. Generally, when traversing the cases down the tree by starting from the root node, only one test was performed on the overall attributes. It consisted in checking the value of the first course, giving immediately as a result the class value, i.e. the laboratory. By noting that the number of courses to be followed during the masters program is four, it is possible to give the following explanation to these results. When looking at the data from a global level, several different classifications are possible due to the important number of courses that can be chosen at the beginning. As a result, many paths exist that leave from the root node. In the best cases, the students have two courses in common (and very rarely three). Therefore, after having processed two courses, *C4.5* does not find any meaningful information on the third or fourth course. As a consequence, the path reaches a leaf only after one or two courses. This leads directly to the research laboratory. By applying this type of approach, it is therefore not possible to predict the value of the third and fourth courses. It follows that using this classification tree will provide poor recommendation.

Afterwards, the first four attributes were kept in the file Course.ARFF. We proceeded to the association rule discovery by applying the *Apriori* algorithm (Agrawal & Srikant 1994). This algorithm has two parameters, the *support* and the *confidence*, which influence the discovered rules. They represent two measures of rule interestingness. They reflect the usefulness and certainty of discovered rules, respectively (Han & Kamber 2001). The rules are extracted from a database composed of records, each record representing a collection of items. For association rules of the form $A \Rightarrow B$ where A and B are sets of items, support and confidence formulas are respectively defined as:

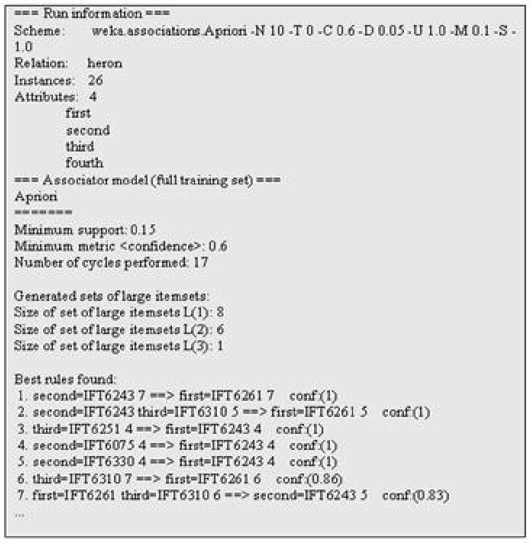
$$Supp(A \Rightarrow B) = \frac{\#\ records\ containing\ both\ A\ and\ B}{total\ \#\ of\ records}$$

$$Conf(A \Rightarrow B) = \frac{\#\ records\ containing\ both\ A\ and\ B}{\#\ records\ containing\ A}$$

Association rules that satisfy both a specified minimum confidence threshold and a specified minimum support threshold are referred to as strong association rules and are considered interesting (Han & Kamber 2001).

Apriori tries to discover the frequent itemsets. These itemsets should be the largest possible, meaning that ideally they must contain many courses that are common within the student population. However, since students do not always follow the same courses, this algorithm was not able to discover any itemsets of meaningful size. For any given support and confidence, *Apriori* generated at most four association rules from the overall cases. To resolve this problem, the data were segmented into homogeneous segments, each one representing research laboratory students. The aim was to maximize the similarity between the courses followed by the students of the same laboratory. By fixing each time the minimum support and confidence, *Apriori* generated different numbers of association rules per laboratory. Giving a minimum support=15% and a minimum confidence=60%, *Apriori* generated from the *HERON* laboratory data, for example, the rules illustrated in figure 4. As an example, the second rule means: the students who took the two courses *Advanced concepts of database* and *Advanced software engineering* also followed the course *Knowledge management*.

The 253 rules generated from various laboratory data were gathered in order to be used during the recommendation process. Contrary to undergraduate courses, the majority of the graduate ones do not have any prerequisites. Moreover, it is possible for a student to start his Masters either during the autumn or the winter session and the courses offered differ for each session. Consequently, we have just considered the values of the courses in the antecedents and the consequents of the rules and not the order of the courses.



```
=== 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)
...
```

Figure 4: Example of association rules generated by *Apriori*

Experimental Evaluation (Step 5) The recommendation inference is done by checking the equivalence between the courses followed by students and the antecedents of the rules. The rules' consequents determine the courses to be recommended. In order to evaluate the rule efficiency in recommendation, we collected test data representing the courses followed by 40 students chosen arbitrarily. These students were registered in the Masters of Computer Science program at the Université de Montréal after the year 2000, and have finished their courses. For each student $\mathbf{S}_i$, we call $\mathbf{F}_i$, the set of courses followed by this student. The main questions are: can *RARE* recommend courses to that student knowing the first course, or the two first courses of $\mathbf{F}_i$? And, also, are these course recommendations relevant (i.e. are they contained in the remaining courses of $\mathbf{F}_i$)? In other words, we measure the recommendation *Coverage* and *Accuracy*. If we denote by $\mathbf{R(S}_i\mathbf{)}$ the set of the courses recommended to a student $\mathbf{S}_i$, and we denote by $\mathbf{T(S}_i\mathbf{)}$ the remaining courses followed in $\mathbf{F}_i$, we define coverage and accuracy as follows: coverage measures the ability of the system to produce all courses that are likely to be followed by the student (i.e., the proportion of relevant recommendations to all the courses that should be recommended). Its formula is defined as:

$$Coverage = \frac{|R(s_i) \cap T(s_i)|}{|T(s_i)|}$$

Accuracy measures the system's ability to provide correct recommendations (i.e., the proportion of relevant recommendations to the total number of recommendations). Its formula is defined as:

$$Accuracy = \frac{|R(s_i) \cap T(s_i)|}{|R(s_i)|}$$

If *RARE* is given only the first course taken by the students, our results show that varying the value of the support has an impact on recommendation coverage and accuracy. As figure 5 depicts, a low support gives the best coverage in contrast to the accuracy, which is better when the support value is high. Confidence, however, is set higher than 50%. If there are very few rules to apply, we consider rules with confidence decreased to 50%. In other words, the coverage decreases when we try to increase the accuracy. The reason is that when decreasing the support, more association rules are discovered. The system then proposes more recommendations, which explains the decrease in their accuracy.

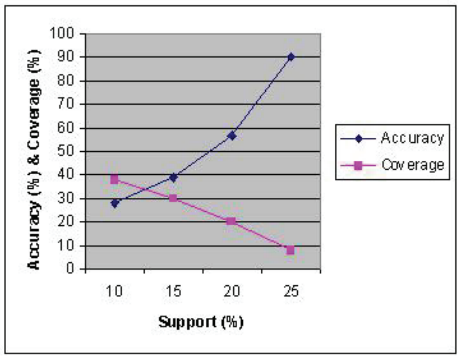


Figure 5: Recommendation Accuracy and Coverage
(When *RARE* is given only one course followed by students)

If *RARE* is given two courses followed by the students, the results are illustrated in figure 6. The coverage values vary in a way similar to that of figure 5. Accuracy, on the other hand, starts decreasing when the support value increases above 15%. Since the population of graduate students is small and diverse, the students have only two courses in common in the best case. They have rarely three courses in common. Consequently, rules of the form $A\ and\ B \Rightarrow C$ are much less significant and rare. For instance, A and B generally refer to courses that are common within a speciality (or a research laboratory) and are very informative, while C is a course related, for example, to the application domain of the student and therefore carries less information. Beyond a support value of 15%, few association rules that contain two courses in their antecedents exist. At minimum support=25%, no association rules exist at all, and coverage and accuracy are null.

To achieve maximum performance with *RARE*, we need a compromise between recommendation coverage and accuracy. The higher the coverage and accuracy, the better *RARE* is at predicting courses. From figures 5 and 6, we can see that a support threshold between 10% and 15% gives results that achieve our objective. Thus, to recommend courses, the generated rules in this interval are kept in the rule base. If our approach is applied to another domain, the setting of support and confidence thresholds depends upon the application and the data we deal with. Typically, as a starting point for an application like recommender systems, the initial support can be set to 10%. Confidence, however, should be set higher than 50%. To tune these settings, we need to see how many strong rules we get. In situations where there is no rule to apply, we need to decrease confidence, but we do not go below 50% (Zaiane 2004).

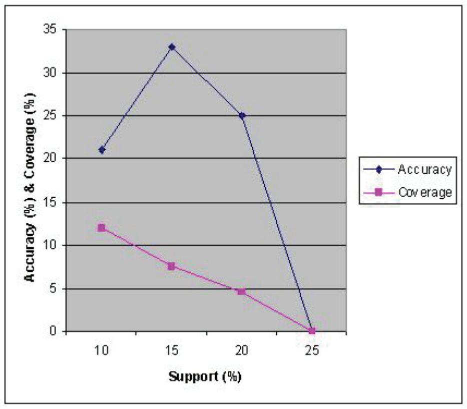


Figure 6: Recommendation Accuracy and Coverage
(When *RARE* is given two courses followed by students)

The next section, which constitutes the deployment step, discusses how rules are used and improved in order to transform them into useful knowledge by *RARE*.

On-line Phase

To maximize benefit of the discovered rules, *RARE* uses them in different ways. So, whether a student *S* has already followed some courses or not, *RARE* will be able to predict suitable recommendations to him. In order to reinforce the rules, *RARE* improves them. The rule improvement method consists of adding and suppressing courses from rules. *RARE* computes the course weights using *course evaluation* and *user fidelity*. After a *required number of evaluations* of a course, its weight is compared to the *course acceptance threshold* to decide if the course will be kept in the rules or removed. We chose the values of the used parameters by following the advice of experienced professors. These parameters are:

- *Required number of evaluations*: ***Req_num_eval = 10***
- *Course acceptance threshold*: ***Acc_thrd = 4***
- *Course evaluation*: ***Eval: 1..10***
- *User fidelity*: ***fid*** $\geq$ ***1***

They are kept in the parameter base and can only be modified by the system administrator. Therefore, these parameters can be adapted to fit future application requirements.

The two following subsections explain how *RARE* performs the recommendation and the rule improvement.

Recommendation When using *RARE* for the first time, a student *S* has to provide initial information so that *RARE* can determine whether or not *S* has followed courses. If *S* has, *RARE* will look at the similarity of those courses with the antecedents of the rules. The rule consequents determine the additional courses that *S* did not follow yet, and that *RARE* has to recommend. Indeed, several cases arise:

Case 1: the system finds some rules whose antecedents match exactly the courses followed by the student. Thus, the system recommends to *S* all the courses that appear in the consequents of those rules.

Case 2: the courses followed by *S* do not appear together in the antecedents of any rule. In this case, *RARE* seeks for the rules whose antecedents contain at least one of the courses followed by *S*. It recommends thereafter all the courses appearing in the consequents of those rules.

Case 3: the courses followed by *S* do not appear in any rule antecedents. Then, *RARE* invites *S* to select his preferred research laboratory. *RARE* uses the rules generated from the preferred laboratory's data and recommends to *S* all the courses that appear in the consequents of those rules.

On the other hand, if *S* has not yet followed any courses at all, *RARE* allows him to select his preferred laboratory as in case 3. *RARE* also gives *S* the possibility to recommend to other users the courses that he has enjoyed before. These courses are sorted by popularity, and can be consulted by all the users at any time. At the end of the session, *RARE* keeps track of the courses followed by *S* and the courses recommended to him in the user base. It also updates some values in the recommended course evaluation base.

Algorithm 1 summarizes the recommendation process. For demonstration purpose, a recommendation scenario is given later.

```
RARE asks a student S for initial information;
if S has followed some courses then
    RARE looks at the equivalence between the courses
    followed by S and the rules' antecedents;
    RARE recommends the courses contained in the
    rules' consequents;
    for all the recommended courses do
        if the course is not present in the recommended
        course evaluation base then
            RARE adds this course to this base and
            associates it with the rules used in its
            recommendation;
            RARE initializes the course weight to 0;
        end
    end
end
else
    RARE invites S to select his preferred research
    laboratory;
    RARE recommends to S all the courses that appear
    in the consequents of the laboratory rules;
end
RARE allows S to recommend to the other users the
courses that he has enjoyed before;
RARE allows S to consult the courses recommended by
the other users, sorted by popularity;
RARE keeps the profile of S in the user base;
```
Algorithm 1: Recommendation

Rule Improvement The system follows up the recommendations it provides and asks the faithful students (see architecture section) to evaluate the courses recommended to them. Evaluations are used to calculate course weights. After the required number of evaluations (Req_num_eval), the calculated weight of a course is compared to the course acceptance threshold (Acc_thrd). If the weight is less than the threshold, *RARE* removes this course from the rules used in its recommendation and from the recommended course evaluation base. If not, the course is kept. Equations 1 and 2 serve in evaluating the course weight.

$$Eval\_Sum = \sum_{user} (user's\, evaluation \times user's\, fidelity) \tag{1}$$

$$Weight = \frac{Eval\_Sum}{\sum_{user}(user's\, fidelity)} \tag{2}$$

Every time a student follows a course that was not recommended to him, *RARE* keeps this course as a temporary course (see architecture section) in the temporary course evaluation base. *RARE* associates this course with all the rules used in the recommendations provided to that user. *RARE* also calculates a weight for all the temporary courses. After the required number of evaluations, the calculated weight of a temporary course is compared to the course acceptance threshold. This makes it possible to add this course in the rules associated with it, provided its weight reaches the threshold. This course is then removed from the temporary course evaluation base. For the weight calculation of a temporary course, equations 1 and 2 are employed.

RARE updates the rules according to the algorithm 2. For a demonstration, see the recommendation scenario given in the next section.

Recommendation Scenario

Let us take the example of *Lina*, a student from the department of Computer Science at the Université de Montréal. *Lina* must register before using *RARE* for the first time. Figure 7 illustrates the registration and login screen. *RARE* asks her for initial information, and knows from her history that *Lina* has followed the course IFT6042. Figure 8 shows the courses recommended to her.

When *Lina* uses *RARE* for a second time, she is invited to check the courses that she followed recently, and give their evaluations via the interface depicted in figure 9. *Lina* followed the courses IFT6370 and IFT6150; she evaluates them respectively as 7/10 and 8/10.

The course IFT6150 was recommended to *Lina* by the use of two rules, which are:

Rule 17: $IFT6042 \Rightarrow IFT6150\; and\; IFT6112$

Rule 43: $IFT6042 \Rightarrow IFT6150\; and\; IFT6310$

Using equations 1 and 2, *RARE* updates the weight of this course in the recommended course evaluation base as illustrated in table 1. This course is associated with the rules 17 and 43 used in its recommendation. Since the number of evaluations of this course is less than 10, no weight verification is performed by *RARE*.

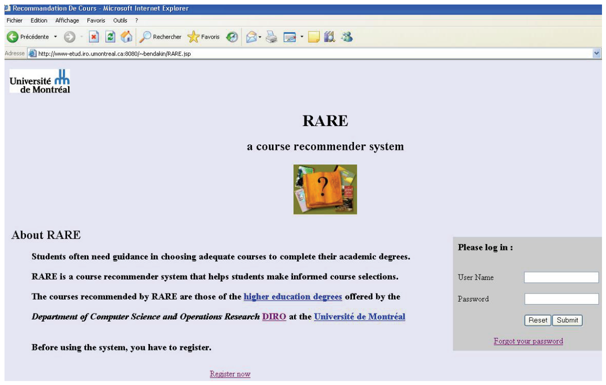
The course IFT6370, however, was not recommended to *Lina*. By use of equations 1 and 2, *RARE* updates this course weight in the temporary course evaluation base. In this base, the course IFT6370 is associated with the two rules 17 and 43 used in the recommendation. As table 2 depicts, the calculated weight is higher than the acceptance threshold (Acc-thrd = 4) and the number of evaluations is equal to 10 (Req-num-eval = 10). Therefore *RARE* adds this course to the two rules 17 and 43. The improved two rules are:

$$IFT6042 \Rightarrow IFT6150\; and\; IFT6112\; and\; IFT6370$$

$$IFT6042 \Rightarrow IFT6150\; and\; IFT6310\; and\; IFT6370$$

```
RARE recognizes the faithful user (fid ≥ 1);
RARE detects if a faithful user S has followed the
courses recommended to him;
for all the courses followed by S do
    if the course was recommended then
        RARE asks S for the evaluation of this course;
        RARE updates the course weight in the
        recommended course evaluation base
        (Eqs. 1-2);
        if after the required number of evaluations
        (Req_num_eval), the course weight is less than
        the course acceptance threshold (Acc_thrd)
        then
            RARE removes this course from the rules
            used in its recommendation;
            RARE removes it from the recommended
            course evaluation base;
        end
    end
    else
        RARE asks S for the evaluation of this course;
        if the course is not present in the temporary
        course evaluation base then
            RARE adds this course to the temporary
            course evaluation base associated with the
            rules used in all the recommendations
            provided to S;
            RARE initializes the course weight to 0;
        end
        else
            RARE updates this course weight
            (Eqs. 1-2);
            if after the required number of evaluations
            (Req_num_eval), the weight of this
            temporary course reaches the course
            acceptance threshold (Acc_thrd) then
                RARE adds this course to the rules
                associated with it;
                RARE removes this course from the
                temporary course evaluation base;
            end
        end
    end
end
```

Algorithm 2: Rule improvement


Recommandation De Cours - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente
Rechercher
Favoris
Adresse http://www-etud.iro.umontreal.ca:8080/~bendakin/RARE.jsp
Université de Montréal

RARE

a course recommender system

About RARE

Students often need guidance in choosing adequate courses to complete their academic degrees.

RARE is a course recommender system that helps students make informed course selections.

The courses recommended by RARE are those of the higher education degrees offered by the *Department of Computer Science and Operations Research* DIRO at the Université de Montréal

Before using the system, you have to register.

Register now

Please log in :

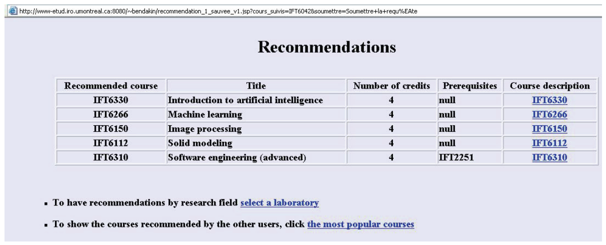
User Name

Password

Reset Submit

Forgot your password

Figure 7: *RARE* registration and login screen

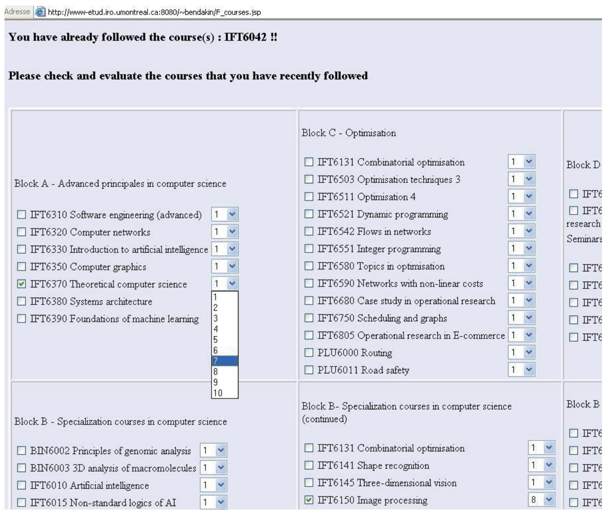

http://www-etud.iro.umontreal.ca:8080/~bendakin/recommendation_1_sauvee_v1.jsp?cours_suivis=IFT6042&soumettre=Soumettre+la+requ%EAte

Recommendations

Recommended course	Title	Number of credits	Prerequisites	Course description
IFT6330	Introduction to artificial intelligence	4	null	IFT6330
IFT6266	Machine learning	4	null	IFT6266
IFT6150	Image processing	4	null	IFT6150
IFT6112	Solid modeling	4	null	IFT6112
IFT6310	Software engineering (advanced)	4	IFT2251	IFT6310

- To have recommendations by research field select a laboratory
- To show the courses recommended by the other users, click the most popular courses

Figure 8: Courses recommended by *RARE*


Adresse http://www-etud.iro.umontreal.ca:8080/~bendakin/F_courses.jsp

You have already followed the course(s) : IFT6042 !!

Please check and evaluate the courses that you have recently followed

Block A - Advanced principales in computer science

- ☐ IFT6310 Software engineering (advanced) 1
- ☐ IFT6320 Computer networks 1
- ☐ IFT6330 Introduction to artificial intelligence 1
- ☐ IFT6350 Computer graphics 1
- ☑ IFT6370 Theoretical computer science 1
- ☐ IFT6380 Systems architecture
- ☐ IFT6390 Foundations of machine learning

1 2 3 4 5 6 7 8 9 10

Block C - Optimisation

- ☐ IFT6131 Combinatorial optimisation 1
- ☐ IFT6503 Optimisation techniques 3 1
- ☐ IFT6511 Optimisation 4 1
- ☐ IFT6521 Dynamic programming 1
- ☐ IFT6542 Flows in networks 1
- ☐ IFT6551 Integer programming 1
- ☐ IFT6580 Topics in optimisation 1
- ☐ IFT6590 Networks with non-linear costs 1
- ☐ IFT6680 Case study in operational research 1
- ☐ IFT6750 Scheduling and graphs 1
- ☐ IFT6805 Operational research in E-commerce 1
- ☐ PLU6000 Routing 1
- ☐ PLU6011 Road safety 1

Block B - Specialization courses in computer science

- ☐ BIN6002 Principles of genomic analysis 1
- ☐ BIN6003 3D analysis of macromolecules 1
- ☐ IFT6010 Artificial intelligence 1
- ☐ IFT6015 Non-standard logics of AI 1

Block B- Specialization courses in computer science (continued)

- ☐ IFT6131 Combinatorial optimisation 1
- ☐ IFT6141 Shape recognition 1
- ☐ IFT6145 Three-dimensional vision 1
- ☑ IFT6150 Image processing 8

Figure 9: *RARE* interface allowing course selection and evaluation

Comparison With Other Systems

Whereas the two systems *Course Recommender* and *SCR* (mentioned in section on recommender systems) require a training period before they can enhance their predictions, *RARE* benefits from the basic knowledge that constitutes the extracted rules. Therefore, *RARE* displays a better performance under the cold start problem compared to the two former systems. Furthermore, *RARE* can provide recommendations to students without a course history, contrary to *Course Recommender*, *SCR* and *AACORN*. Another important issue is scalability. When the amount of data is significant, *SCR*, *Course Recommender*, *AACORN* and *PEL IRT* are more resource demanding than *RARE*. For example, during a real-time recommendation, *Course Recommender* has to compute the similarity between the current student and all the other students, or between the rated courses and all the other courses. In the case of *RARE*, all the mining process is performed off-line. Although the number of rules that are generated grows exponentially with the number of items (Schafer 2005), it is possible to produce a compact set of highly significant rules by appropriately tuning the support and the confidence. Therefore, deriving the rules during the off-line phase can be time-consuming but scanning them during recommendation time is done quickly. Whereas *SCR*, *AACORN*, and *PEL-IRT* take into account some criteria such as course availability, the undesired courses, and the categories of courses, *RARE* has some limitations when providing its recommendations. For instance, it does not yet consider student backgrounds and course availability.

Table 1: Recommended course weight updating

Rule number	...	Rule 17	...	Rule 43
Recommended course	...	IFT6150	...	IFT6150
Number of course evaluations	...	8	...	8
Evaluation sum	...	169	...	169
Sum(users's fidelity)	...	23	...	23
Weight	...	169/23 **=7.34**	...	169/23 **=7.34**

Table 2: Temporary course weight updating

Rule number	...	Rule 17	...	Rule 43
Temporary course	...	IFT6370	...	IFT6370
Number of course evaluations	...	10	...	10
Evaluation sum	...	219	...	219
Sum(users's fidelity)	...	29	...	29
Weight	...	219/29 **=7.55**	...	219/29 **=7.55**

Conclusion

This paper presented *RARE*, a *course recommender* system based on association rules. *RARE* was used on real data coming from the department of Computer Science at the Université de Montréal. It analyses the past behaviour of students concerning their course choices. More explicitly, it formalizes association rules that were implicit before. These rules enable the system to predict recommendations for new students. A solution to the cold start problem, which is a central question in recommender systems, is proposed.

Moreover, data mining helps us overcome the problem of scalability because the more the volume of data increases the better is the quality of extracted knowledge. Since the student course appreciation changes over time, *RARE* asks for the evaluation of its recommendations and thus improves the quality of the rules. Therefore, the extracted rules are no longer based only on the two criteria of support and of confidence, which respectively represent the utility and the certainty of the rules. Instead, *RARE* further consolidates the rules by the integration of certain additional criteria, which are the course evaluation, the student fidelity, and the course weight, as demonstrated in the recommendation scenario. The rule improvement also allows the integration of new courses into the rules, as well as their recommendation if they are well received by students. If all the courses in the consequent of a rule have been removed, the rule will also be deleted from the rule base. Since *RARE* does not yet consider the student backgrounds and the course availability, we plan to enhance its performance by taking into account these factors. Future work will also include the use of information extraction techniques to enable *RARE* to deduce course similarity from course descriptions. Accordingly, the courses belonging to the same category will be recommended and the system will display a higher recommendation coverage and accuracy. Currently, we are performing an experimental evaluation of the system. The evaluation consists of tree parts. The first one consists of collecting more data, and then training the system on a subset of the data and testing it on the remaining subset. The second part corresponds to asking the opinions of laboratory directors and professors regarding the quality of the extracted rules. Finally, the third requires having students experimenting the system and collecting their feedback. We also plan to conduct an experimental comparison between *RARE* and the other course recommender systems. We believe that the *RARE* approach could be re-used easily for other programs of study, provided that a large amount of former student data is available. We can apply the approach as such for programs that offer several specialities or research laboratories. For programs in which no specialities exist, the segmentation of the data is not necessary. In this case, the algorithm can be applied on the whole dataset and we can use the extracted rules in the same manner.

Acknowledgments

We would like to acknowledge the help and give our deepest thanks to Sébastien Gambs for all the useful discussions we had with him and for all the work he has done on reviewing earlier versions of this paper. The authors are supported in parts by the Natural Sciences and Engineering Research Council of Canada.

References

Adomavicius, G., and Tuzhilin, A. 2001. Using data mining methods to build customer profiles. *IEEE Computer* 34(2):74–82.

Adomavicius, G., and 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* 17(6):734–749.

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Berry, M. J. A., and Linoff, G. S. 2004. *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management*. Wiley.

Breese, J.; Heckerman, D.; and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 43–52.

Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Chen, C. M.; Lee, H. M.; and Chen, Y. H. 2005. Personalized e-learning system using item response theory. *Computers and Education* 44(3):237–255.

Edelstein, H. 2003. Data mining in depth: Description is not prediction. *DM Review*.

Ekdahl, M.; Lindström, S.; and Svensson, C. 2002. A student course recommender. Master of Science Programme, Lulea University of Technology, Department of Computer Science and Electrical Engineering / Division of Computer Science and Networking.

Han, J., and Kamber, M. 2001. *Data Mining: Concepts and Technique*. Morgan Kaufmann.

Herlocker, J.; Konstan, J.; and Riedl, J. 2000. Explaining collaborative filtering recommandations. In *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work (CSCW'00)*, 241–250.

Larose, D. T. 2005. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley.

Murthy, S.; Kasif, S.; Salzberg, S.; and Beigel, R. 1993. OC1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 322–327.

Perugini, S.; Gonçalves, M. A.; and Fox, E. A. 2004. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems* 23(2):107–143.

Quinlan, J. R. 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc.

Rafaeli, S.; Dan-Gur, Y.; and Barak, M. 2005. Social recommender systems: Recommendations in support of e-learning. *Journal of Distance Education Technologies* 3(2):29–45.

Sandvig, J., and Burke, R. 2006. AACORN: A CBR recommender for academic advising. Currently in development, School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, USA.

Sarwar, B. M.; Karypis, G.; Konstan, J. A.; and Riedl, J. T. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference*, 285–295.

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.

Shearer, C. 2000. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing* 5(4):13–22.

Simbi, P. 2003. Course recommender. Senior Thesis, Princeton University.

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

Zaiane, O. R. 2004. Personal communication.