Intelligent Tutor Recommender System for On-Line Educational Environments

Marian Cristian Mihăescu
University of Craiova
Department of Computer Science
Craiova, Romania
mihaescu@software.ucv.ro

Paul Ştefan Popescu
University of Craiova
Department of Computer Science
Craiova, Romania
sppopescu@gmail.com

Costel Ionaşcu
University of Craiova
Faculty of Economics and Business Administration
Craiova, Romania
icostelm@yahoo.com

ABSTRACT

This paper presents a method of using a classification procedure for retrieval of the most appropriate tutors in online educational environments. The main goal is assisting learners to find the most suitable colleagues that can provide them help. The retrieval is based on a user model built from experiences of previous generations of students. Performed activities represent the raw input data(i.e., the experiences), that one obtained from on-line educational environment. The goal of the developed system is to provide a list of colleagues that are willing and able to provide help. The student that is looking for a tutor will be aware of his weakness, his place among his colleagues and get some intuition regarding needed future activities that may improve effectively his knowledge level. The data processing for the retrieval mechanism is based on a classical classification engine that is custom designed for fulfilling the presented goal.

Keywords

Decision Tree, Classification Induction, Recommender System, e-Learning

1. INTRODUCTION

This paper addresses the problem of improving the knowledge level of a student that uses an online educational environment by using algorithms for indexing and retrieval mechanism. The proposed approach contains two main modules: the server side module where the indexing model is created and the client side where visualization and retrieval of a set of learners(i.e. prospective tutors) takes place. This set of colleagues represent the most appropriate options according to several predefined criteria specified by retrieval mechanism.

The first prerequisite for building a reliable recommender system is gathering high quality data in order to properly train the classifier, as main data processing unit within indexing mechanism. The activity related assets (e.g., database, log files, etc.) of the e-Learning environment are queried in order to obtain the training dataset. The assets must provide enough information such that all needed features that define students are computed and stored into the training dataset.

Once the input data for analysis is available the aim is to design a machine learning based recommender system that trains a classifier which acts as a core processing unit for the indexing mechanism. The next steps are choosing: the appropriate algorithm type (e.g., supervised, unsupervised, rule based, etc.), the algorithm itself, the features (e.g., name, meaning, type, values, etc.) and the overall setup necessary for obtaining a solution. In our case, we use supervised learning algorithms (e.g., classifier) and more exactly, decision trees [6]. The algorithm is used to classify new items, which in our educational context are represented by learners. The research issue of this paper regards designing and implementing a tutor recommender system. Addressing of this issue is accomplished by two means: properly designing a custom data analysis pipeline and building a tool that retrieves tutors in the practical context of Tesys [2] e-Learning platform.

For prototyping a general purpose classifier is used, i.e. a decision tree that is implemented in Weka [5] and this implementation is used for experiments. The key issues that are addressed regard properly setting up the general purpose classifier in a context of a practical problem in e-Learning application domain. Main ingredients for concept proof description and tool prototyping are presented in this paper along with detailed description of choices and their expected, observed and validated impact.

Once the recommender system is built, it can be used to obtain tutors for current or new learners by providing input only their computed values for the chosen features. On the client side, the learner is able to log in the application and access the tutor search utility application. The student first sees his class label in the existing model (i.e., his actual class) and then his target class which gathers the best suited tutors for him.

2. RELATED WORKS

The paper folds at boundaries between domains of machine learning and information retrieval as part of EDM and recommender systems. Educational Data Mining is an emerg-

ing discipline, concerned with developing methods for improving the relationship and interaction level between learners and professors. Since 2005 when the workshop referred to as 'Educational Data Mining' AAAI'05-EDM took place in Pittsburg, USA [4], followed by several related work-shops and the establishment of an annual international conference first held in 2008 in Montreal [1] many work has been done in this area [12] [11] .

Building recommender systems for e-Learning gathered many research efforts due to large number of practical usages within this application domain. Since e-Learning is a highly interaction domain, it is very appropriate for using Intelligent Data Analysis techniques for building various types of recommender systems. E-Learning personalization [8] represents one of the most common and general issues in e-Learning. Within this area of research issues like adapting the presentation and navigation [3], smart recommender in e-Learning [14] and various other commercial systems [9] proposes different input data, user modelling strategies or prediction techniques for reaching various business goals. Among the most used data analysis techniques there are content-based or item-based filtering, collaborative filtering, rule-based filtering, etc [10]. The general machine learning strategy of learning and predicting, information retrieval strategy of indexing and retrieval become in recommender systems for e-learning modelling and recommendation. Modelling regards thus users, content(i.e. questions, chapters, etc) and recommendation implies the existence of a implicit or explicit query. Once all these ingredients are put together in a consistent data analysis pipeline the output takes the form of a single recommended set [13].

Regarding involved technologies this paper uses Weka (Waikato Environment for Knowledge Analysis) as a popular suite of machine learning, data mining and information retrieval algorithms written in Java. The implemented algorithms are very flexible and can be used into the analyzing process of different kinds of data(from different domains). From Weka we have used J48 which is the implementation of the C4.5 [7] algorithm in Weka, a data analysis algorithm which generates a decision tree in order to classify data.

3. FRAMEWORK FOR INTELLIGENT TUTOR RETRIEVAL

The recommender module has the task of matching the query of a learner for a tutor against the existing data model. From software perspective the recommender module is a client for model builder module. Its main task is to produce results in such a way they may be intuitively displayed by the thin client application used by learner in his attempt to find a suitable tutor. Therefore the learner will see a tree-like structure due to native shape of the decision trees with actual class of the learner marked in red and with target classes marked in shades of green. In fact the green classes represent set of learners that are suitable for being tutors for learner that is querying the system.

In Fig.1 presents how the tutor recommender mechanism is designed as a data workflow. From interaction point of view students have to query the system that is integrated within Tesys-Web and after performing necessary operations on the server side they obtain the decision tree filled with

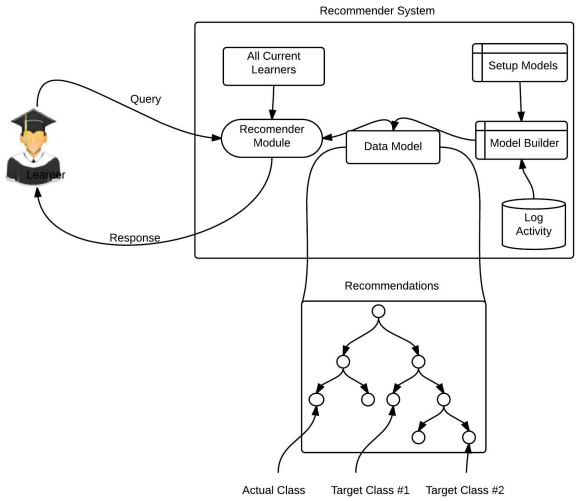

Figure 1: Activity use case diagram

prospective tutors. On the server side we have the business logic of the recommender system. Here the training dataset is built, thereafter the data model and the output as an xml file that can be displayed on the client side.

3.1 Description of Tasks within Recommender System

The main tasks performed within the recommender system regard preliminary offline data model building, setup of the recommender system, indexing currently existing learners into already created model and computing and extracting relevant tutors by applying the already setup recommendation strategy.

3.1.1 Learners Modelling Phase

We apply machine learning (i.e., decision trees) techniques to build learners profiles by using already existing implicit performed activities from usage sessions of learners that used the system in previous years. This data represents the training data and the output is represented by a set of classes (i.e. leaves in the decision tree) such that each class corresponds to a learners profile. Once sessions and corresponding activity data are delimited in such a way that all features describing a learner are processed, we can use the decision tree builder to obtain the baseline data model. Once the data model is created the currently existing learners within the e-Learning platform are placed in their corresponding leaves. At this moment currently existing learners are indexed in the decision tree data structure and are ready to be queried.

3.1.2 Tutors Recommendation Phase

The query of a learner for a tutor is regarded as a parametrized implicit query. The parameters aim tuning the retrieval mechanism such that optimal solutions are returned from solution space. The solution space is regarded as a set of classes(i.e., leaves) each class containing a set of prospective tutors. The set of classes need to fulfill one basic requirement, which is to be labeled with a "better" class label that

the one in which the querying learner resides. A total order set of classes is ergonomically computed with the first item containing learners "close" to the querying learner and the last item containing learners "further" to the querying learner. In this context parameter tuning will manage to decide a certain number of prospective tutors that are picked up from one of the target classes. Intuitively, choosing tutors from a class that is "closer" to the querying learners' class will return tutors with a profile that is better but similar. On the other hand, choosing tutors from a class that is "further" to the querying class will return tutors with the best profiles among all colleagues.

3.2 Description of the Data Analysis Process

The main concept considered in the model is the "Learner", which is also a "Tutor". Once the data model is built from the training data the current set of learners $L=\{L_1,L_2,L_3...L_n\}$ is distributed in corresponding classes according with the key feature values $f_{i,k}$. For current prototype implementation the features are not weighted since the decision tree itself provides a ranking in feature selection.

All classes of learners are considered as resources for which an *"affinity"* function needs to be defined in order to retrieve the most suitable tutors. Defining the *affinity* function needs to take into consideration several criteria such a better overall knowledge weight, specific values in communication related features(i.e., messaging activity, forum activity, etc) and demographic features. Due to its specific topology of the decision tree also ranks the leaves in classes. A normal distribution function is defined such that the lowest ranked class is assigned 0 knowledge weight and highest ranked class is assigned a value of 1 knowledge weight. All in between classes get a knowledge weight ranking between 0 an 1.

Thus the data analysis task is to identify the actual class of the learner who is querying for a tutor and to provide most suitable options from the subsequent classes in obtained ranking of the current learners. With this approach the tutor retrieval becomes a matter of properly specifying querying parameters. The proposed mechanism offers the possibility of obtaining any of the feasible solutions, somewhere between the very next learner which resides in the class with the next knowledge weight value up to the top class learners in ranking. From this perspective several parameters are defined. One manages the proximity of the class from which tutors are retrieved.

4. EXPERIMENTAL RESULTS

The main input of the server application is the activity repository. This raw data taken from the database is converted to an *.arff* file, which is used to train the classifier.

In Fig. 2 are presents the meaning of the attributes from the *.arff* file and Fig.3 presents the obtained decision tree based on the training dataset. Several functionalities were developed in order to be able to load and parse the decision trees. One of them is successor computation. Because of our dataset structure, the decision tree contains ordered leaves. This functionality is successfully used for fulfilling specific user constraints. For example, if the student does not want

userid	The identification number of the student
nrHours	The total number of hours spent on the platform
avgMark	The average mark obtained at a discipline
messagingActivity	This feature represents a discretization of the on-line involvement regarding sent and received messages. A messaging activity greater than 50% of the average number of sent and received messages means YES and smaller means NO.
noOfTests	The total number of tests taken on the platform
avgMessageLength	This feature represents a discretization of the average message length. An average length greater than 160 characters means the student sends LONG messages and smaller means SHORT messages. This value has been chosen with respect to the limitation of the SMS message.
class	The class that the user belongs to, which can be low or high. Low/high values were assigned based on the final result obtained by the student.

Figure 2: Features

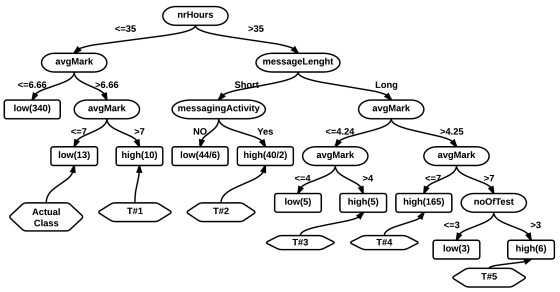

Figure 3: Tree example

to have a step by step progress, he will be able to use this feature to retrieve tutors which are i steps ahead of him.

Here is the pseudocode for the method used to locate the classified student's actual and target class. This recursive method takes two parameters: a node element containing information from the xml and a boolean variable, stating whether the parent has been marked or not. A node is marked when the student meets the requirement stated inside the node (for example if the "messageLenght" is "LONG").

```
function studentSearch(Node parent, boolean isMarked)
{
   SET nodeList to the list of child nodes of "parent"
   FOR each node in nodeList
      IF is element node THEN
         SET atr to the value of the attribute "attribute"
         IF atr is null THEN
            IF isMarked THEN
               add new atrribute to the node
            END IF
            IF "decision" attribute is "high" THEN
               SET target to the current node
            END IF
         ELSE
            SET expresie to ""
            SET belongs to false
            CASE atr IS
               "avgMark":
                  generate the expression to be evaluated
                  SET belongs to the result of the evaluation
               "nrHours":
                  generate the expression to be evaluated
```

```
                SET belongs to the result of the evaluation
               ..............................................
            END CASE
            IF belongs AND isMarked THEN
               add new atrribute to the node
               CALL studentSearch WITH current node, true
            ELSE
               CALL studentSearch WITH current node, false
            END IS
         END IF
      ELSE
         REMOVE node FROM nodeList
      END IF
   END FOR
}
```

The thin client automatically computes the class of the student who is searching for a tutor. These values are used to determine the actual class of the student. The actual class of the student is marked with red and the target class is marked with green.

In Fig. 4, the inspection of the green node reveals to the student a list on colleagues that may help him as tutors. The student has a messaging system to his disposal for contacting his recommended tutors in an attempt to find answers from the right persons.

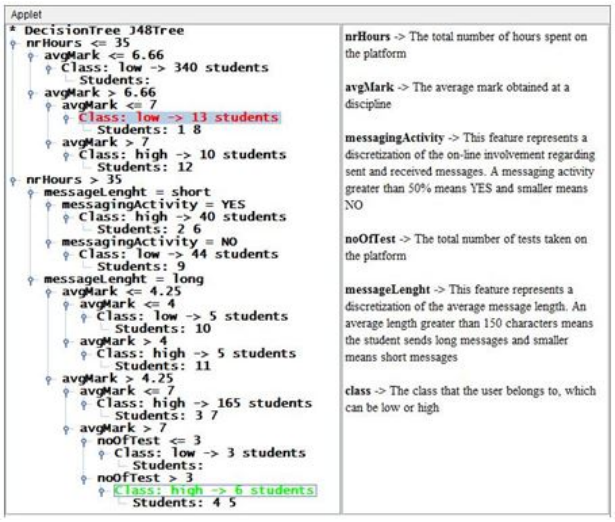

Figure 4: Prototype of the tutor retrieval system

Let us consider student S1, which has already used the e-learning platform and there is enough data logged about him, including the following values for the attributes: nrHours = 30, avgMark = 6.80, messagingActivity = No, noOfTests = 2, avgMessageLength = SHORT. After running the tool, he finds out that his actual leaf/class is the second one when parsing the tree from left to right, so he has to put some effort to catch up with his colleagues. Assuming he wants to spend the necessary time to gradually learn from his colleagues, the system can recommend him tutors belonging to the third leaf of the tree (1^{st} level successors). He will therefore receive the contact details of the students from that leaf. After managing to improve his performances and become himself part of that leaf, he will be able to continue to the next step. If he however wants to try to learn from the best directly, the system will be able to provide him with the contacts of the best tutors available (belonging to the green leaf in Fig. 4).

5. CONCLUSIONS

The paper presents a custom approach for providing assistance to learners in on-line educational environments. The assistance regards the option of finding colleagues that may offer guidance in respect to activities that need to be performed for improving the student's knowledge level.

Each student will benefit by using this tool from the perspective of improving their knowledge. It will be easier for students to communicate with someone their own age, ask questions about the things they don't understand, and get more clarification and feedback.

6. REFERENCES

[1] T. B. J. Baker R.S.J.d.; Barnes. Educational data mining 2008 :1st international conference on educational data mining proceedings. *Educational Data Mining*, 1:20–21, 2008.

[2] M. M. C. Software architectures of applications used for enhancing on-line educational environments. *Manuscript submitted for publication.*

[3] H. Chorfi and M. Jemni. Perso : Towards an adaptative e-learning system. *Journal of Interactive Learning Research*, 15(4):433–447, 2004.

[4] B. J. Proceedings of aaai2005 workshop on educational data mining. 2005.

[5] W. I. F. E. T. L. H. M. H. G. C. S. Jo. Weka: Practical machine learning tools and techniques with java implementations. *University of Waikato, Department of Computer Science*, 1999.

[6] J.R.Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[7] J.R.Quinlan. C4.5: Programs for machine learning. *San Mateo, CA*, 1993.

[8] M. Khribi M. K.; Jemni and O. Nasraoui. Recommender system for predicting student performance. *Educational Technology & Society*, 12(4):30–42, 2009.

[9] B. Mobasher. Data mining for web personalization. *The Adaptive Web: Methods and Strategies of Web Personalization*, Springer-Verlag, Berlin-Heidelberg, 2006.

[10] O. Nasraoui. World wide web personalization. *Encyclopedia of Data Mining and Data Warehousing*, 2005.

[11] C. Romero and S. Ventura. Data mining in e-learning (advances in management information). *WIT Pr Computational Mechanics*, 2006.

[12] B. R. S. and Y. K. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining (JEDM)*, 1:3–17, 2009.

[13] O. N. Z. Z. E. Saka. Web recommender system implementations in multiple flavors: Fast and (care) free for all. *Proceedings of the ACM-SIGIR Open Source Information Retrieval*, 2006.

[14] O. Zaiane. Building a recommender agent for e-learning systems. *Proc. of the 7th International Conference on Computers in Education,*, 3-6:55–59, 2002.