Computers & Education 53 (2009) 828–840

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Applying Web usage mining for personalizing hyperlinks in Web-based adaptive educational systems

Cristóbal Romero [a,*], Sebastián Ventura [a], Amelia Zafra [a], Paul de Bra [b]

[a] Department of Computer Sciences and Numerical Analysis, University of Córdoba, 14071 Córdoba, Spain
[b] Department of Computer Sciences, Eindhoven University of Technology, PO Box 513, Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 8 January 2009
Received in revised form 4 May 2009
Accepted 4 May 2009

Keywords:
Education and e-learning
Adaptive hypermedia
Recommender system
Data mining
Web mining

ABSTRACT

Nowadays, the application of Web mining techniques in e-learning and Web-based adaptive educational systems is increasing exponentially. In this paper, we propose an advanced architecture for a personalization system to facilitate Web mining. A specific Web mining tool is developed and a recommender engine is integrated into the AHA! system in order to help the instructor to carry out the whole Web mining process. Our objective is to be able to recommend to a student the most appropriate links/Web pages within the AHA! system to visit next. Several experiments are carried out with real data provided by Eindhoven University of Technology students in order to test both the architecture proposed and the algorithms used. Finally, we have also described the meaning of several recommendations, starting from the rules discovered by the Web mining algorithms.

1. Introduction

The task of delivering personalized content is often framed in terms of a recommendation task in which a system recommends items to an active user (Mobasher, 2007). A recommender system is a program that observes what a user is doing and tries to recommend actions, objects, etc. it thinks would be beneficial to that user (Schafer, 2005). Such systems have become powerful tools in many domains ranging from electronic commerce to digital libraries and knowledge management. Recommender systems can use data mining techniques for making recommendations using knowledge learnt from the actions and attributes of users (Schafer, 2005). The objective of the data mining process is to discover new, interesting and useful knowledge using a variety of techniques such as prediction, classification, clustering, association rule mining and sequential pattern discovery. In the case that the used data are obtained from the Web, the process is called Web mining instead of data mining. Web mining can be defined as the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World-Wide Web (Liu, 2007). There are roughly three knowledge discovery domains that pertain to Web mining: Web content mining, Web structure mining, and Web usage mining. Web content mining is the process of extracting knowledge from the content of documents or their descriptions. Web document text mining, resource discovery based on concepts indexing or agent-based technology may also fall in this category. Web structure mining is the process of inferring knowledge from the World-Wide Web organization and links between references and referents in the Web. Finally, Web usage mining, also known as Web log mining, is the process of extracting interesting patterns in Web access logs. An interesting application of Web usage mining is link recommender systems (LRS). Their purpose is to facilitate the navigation of the users on a Web site and to help them not to get lost when they are browsing through hypertext documents (Delort & Bouchon, 2002).

On the other hand, there is an increasing interest in applying data mining to educational systems, making educational data mining (EDM) a new and growing research community (Romero & Ventura, 2006) (Romero & Ventura, 2007) (Romero, Ventura, & Salcines, 2008). EDM is an emerging discipline, concerned with developing methods for exploring the unique types of data that are obtained from different types of educational contexts. On the one hand, there are traditional face-to-face classroom environments such as special education (Tsantis & Castellani, 2001) and higher education (Luan, 2002). On the other hand, there are computer-based education and Web-based education such as well-known learning management systems (Pahl & Donnellan, 2003) the examples of which are WebCT (WebCT, 2009), BlackBoard (BlackBoard, 2009) and Moodle (Moodle, 2009); intelligent tutoring systems (Mostow & Beck, 2006) the

* Corresponding author. Tel.: +34 957 218630.
E-mail address: cromero@uco.es (C. Romero).

0360-1315/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2009.05.003

examples of which are SCHOLAR (Carbonell, 1970), PUMP Algebra Tutor (Anderson, Corbett, Koedinger, & Pelletier, 1995) and SHERLOCK (Lesgold, Lajoie, Bunzo, & Eggan, 1992); and Web-based adaptive hypermedia systems (Koutri, Avouris, & Daskalaki, 2005) the examples of which are ELM-ART (Weber & Brusilovsky, 2001), AHA! (De Bra & Calvi, 1998) and KBS-Hyperbook (Henze & Nejdl, 2001). All these learning environments accumulate a vast amount of data which are very valuable for analyzing students' behavior and could create a gold mine of educational data. The main difference between them is the data available in each system. Traditional classrooms only have information about student attendance, course information, curriculum goals and individualized plan data. However, computer and Web-based education has much more information available because these systems can record all the information about students' actions and interactions onto log files and databases. Finally, the data from intelligent tutoring systems (ITS) and adaptive Web-based educational systems (AWBES) are semantically richer and can lead to more diagnostic analysis than data from traditional Web-based education system (Merceron & Yacef, 2004).

In this work, we are going to describe the use of Web usage mining techniques for links recommendation in AWBES. The task of links recommendation in Web-based education can be seen as a special type of adaptive navigation support due to the fact that they share the same goal of helping students to find an optimal path through the learning material (Brusilovsky & Peylo, 2003). Adaptive educational hypermedia systems can adaptively sort, annotate, or partly hide the links to make it easier to choose or to recommend to the students where they should go from a certain point. This technology is one of the most popular in AWBES and there are a lot of systems that use it. In fact, we have integrated our recommender system based on Web usage mining into the well-known AHA! system. The proposed system uses different recommendation techniques (clustering and sequence mining) in order to suggest links based on other students with similar characteristics. The originality of our personalized recommender system consists of the use of Web usage mining together with hyperlink adaptation. Our mining approach uses all the available usage information about students (profile and log information) in order to learn user routes or browsing pathways for personalized link recommendation.

This paper is arranged in the following way: first we describe the related background and the AHA! system. Next, we explain the proposed architecture for link recommendation based on Web usage mining. Then, we describe the Web mining tool and links recommender engine that we have developed and integrated into the AHA! system. Finally, we describe the experiments that we have carried out, conclusions and future work.

2. Background

The work described in this paper is an intersection of several research areas: personalization, recommendation, data mining and adaptive Web-based educational systems. So, in this section we are going to do a survey of these related subjects.

2.1. Personalization, recommendation and data mining

The ability of a personalization system to tailor content and recommend items implies that it must be able to anticipate the needs of users and provide them with recommendations of products or items that they might appreciate based on previous or current interactions with that user, and possibly other users. The personalization task can therefore be viewed as a prediction problem: the system must attempt to predict the user's level of interest in, or the utility of, specific content categories, pages, or items, and then rank these according to their predicted values (Brusilovsky & Millán, 2007). Recommendation and personalization techniques are usually classified into three different categories (Pazzani & Billsus, 2007): rule-based filtering systems, content-based filtering systems and collaborative filtering systems. Rule-based filtering systems rely on manually or automatically generated decision rules that are used to recommend items to users. Content-based filtering systems (Mobasher, 2007) recommend items that are considered sufficiently similar to the content descriptions in the user profile. Collaborative filtering systems (Schafer, Frankowski, Herlocker, & Shilad, 2007), also referred to as social filtering, attempt to identify groups of people with similar tastes to those of the user and recommend items that are not yet rated or seen using the opinion of these similar users. It is also interesting to comment that there are hybrid systems (Burke, 2007) that combine these techniques in an attempt to achieve better performance by overcoming the weaknesses that are present in each method.

There are a considerable number of data mining approaches applied to personalization (Frias-Martinez, Chen, & Liu, 2006). The essential motivation for using these techniques is that they have demonstrated to work really well for modeling preferences of users in recommendation systems (Frias-Martinez et al., 2006). Two of their main strengths are the flexibility for leveraging different data channels in a comprehensive manner and the possibility to better integrate the personalization tasks with other existing applications. Most data mining approaches that are applied to personalization are based on collaborative filtering. In these approaches the pattern discovery algorithms take the historical rating or navigation profiles of past users as input and generate aggregated user models. The most common data mining techniques for recommender applications are:

- Clustering is a process of grouping objects into classes of similar objects (Jain, Murty, & Flynn, 1999). It is a partitioning of patterns (observations, data items, or feature vectors) into groups or subsets (clusters). This technique groups records based on their location and connectivity within an n-dimensional space. The principle of clustering is maximizing the similarity inside an object group and minimizing the similarity between the object groups. There are many clustering methods (Jain et al., 1999), including hierarchical and function-based algorithms. One of the most well known and commonly used is the k-means algorithm (MacQueen, 1967) that tries to minimize the distance of the objects to the centroid or mean point of each cluster.
- Association rules mining (Agrawal, Imielinski, & Swami, 1993) addresses the problem of discovering association rules in data. The original problem was the market basket analysis that seeks to find relationships between purchases. Such rules associate one or more attributes of a dataset with another attribute, producing if–then statements concerning attribute values. There are a lot of association rule mining algorithms: Apriori was the first and it opened a brand new family of algorithms (Ceglar & Roddick, 2006) such as Apriori-TID, DIC, Eclat, and FP-Growth.
- Sequential modeling or sequential pattern mining (Han, Pei, & Yan, 2005) discovers inter-session patterns. A sequential pattern is a restrictive form of association rule (Agrawal et al., 1993) in which the order of the accessed items is taken into account (the association rule discovers all the relationships without restrictions). Sequential pattern mining was first introduced into the study of customer

purchase sequences (Agrawal & Srikant, 1995). Given a set of sequences, where each sequence consists of a list of elements and each element consists of items, and given a user-specified minimum support threshold, sequential pattern mining tries to find out all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than the minimum support. In Web applications a Web server log file is used to discover sequences of resource requests. The problem of mining sequences in Web navigational patterns refers to the identification of those Web document references which are shared through time by a large number of user sequences, where a user sequence is a time-ordered set of visits. There are several popular pattern discovery algorithms (Han et al., 2005) such as AprioriAll, GSP, SPADE, PrefixSpan, CloSpan and FreSpan.

2.2. Adaptive Web-based educational systems

Adaptive Web-based educational systems (AWBES) provide an alternative to the traditional "just-put-it-on-the-Web" approach in the development of Web-based educational courseware (Brusilovsky & Peylo, 2003). AWBES attempt to be more adaptive by building a model of the goals, preferences and knowledge of each individual student and by using this model throughout the interaction with the student in order to adapt to the needs of that student. That is, a student will be given a presentation that is adapted especially to his or her knowledge of the subject and a suggestion is made of the most relevant links to proceed further. AWBES inherit from traditional intelligent tutoring systems (ITSs) and adaptive hypermedia systems (AHSs). ITSs typically partition the information space in knowledge about the domain, knowledge about the user and teaching strategies to support individualized learning. Adaptive hypermedia systems usually enable content and navigation adaptation, by altering the link structure and the node contents of the hypertext that contains the educational material. AWBES can use different techniques and methods in order to add adaptive functionality and educational system to (Brusilovsky, 1996):

- Curriculum Sequencing (or instructional planning): it provides the learner with the most suitable individually planned sequence of knowledge units and learning tasks.
- Intelligent analysis of a student's solutions: it identifies exactly what is wrong or incomplete and which missing or incorrect knowledge may be responsible for the error.
- Interactive problem solving support: it provides the student with intelligent help on each step of the problem solving process from giving a hint to executing the next step for the student.
- Example-based problem solving: it helps students by suggesting the most relevant cases (examples previously explained or problems already solved by the students).
- Adaptive presentation technology: it adapts the content of a hypermedia page to the user's goals, knowledge and other information stored in the user model.
- Adaptive collaboration support: it uses the system's knowledge about different users (stored in user models) to form matching collaboration groups.
- Adaptive navigation support technology: it supports the student navigation and orientation in hyperspace by changing the appearance of visible links. In particular, the system can adaptively sort, annotate, or partly hide the links in the current page to facilitate the choice of the next link.

There are a lot of adaptive educational hypermedia systems developed since 1996 (Brusilovsky, 2001) such as ELM-ART, InterBook, PT, 2L670, Medtech, AST, ADI, HysM, MetaLinks, RATH, TANGOW, Arthur, CAMELEON, KBS-Hyperbook, SKILL, ACE, ART-Web, and AHA!.

2.3. Recommendation in Web-based educational systems using Web usage mining

Although personalized recommendation approaches that use data mining techniques were first proposed and applied in e-commerce for product purchase, there are also several works about the application of different data mining techniques within recommender systems in e-learning. Examples grouped by technique:

- The extraction of sequential patterns has been used to find patterns in the process of recommending relevant concepts to students, mainly for curriculum sequencing (Ksristofic, 2005). Based on a student's current learning session and discovered patterns, the recommender system is able to recommend a sequence of concepts, which the student should study next. Sequential rules have also been used to guide a learning resource recommendation service based on the simple sequencing specification (Shen & Shen, 2004). This intelligent recommendation system organizes learning contents into small atomic units called Learning Objects (LOs) so that they could be used and reused effectively together with their ontology (description of the LOs and relationships).
- Association rules have been used to provide feedback to the courseware author and to recommend how to improve the courses (García, Romero, Ventura, & Castro, 2006)(Romero, Ventura, & De Bra, 2004). They use a cyclical methodology with a rule discovery algorithm without parameters (García et al., 2006), grammar-based genetic programming and multi-objective genetic algorithms (Romero et al., 2004). Fuzzy association rules and a multi-attribute evaluation method have been used in a personalized learning material recommender system (Lu, 2004). That system helps students to find learning materials they would need to read. Association rules have also been used in a recommender agent for suggesting online learning activities or shortcuts on a course Web site based on a learner's access history (Li & Zaïane, 2004)(Zaïane, 2002). Their objective is to improve course material navigation as well as to assist the online learning process. This e-learning task recommender uses the tasks already done by the learner and their success and the tasks completed by other similar learners.
- Clustering has been used to find groups of students with similar learning characteristics and to promote group-based collaborative learning in a research paper recommender system (Tiffany & Gordon, 2003). That research proposes an evolving Web-based learning system which can adapt its content based on the system's observation of its learners and the accumulated ratings given by them. Clustering and association rules have been used together for recommending a list of Web pages on an e-learning Web site (Wang & Shao, 2004). The site

clusters students based on the time-framed navigation sessions and applies association mining to establish a recommendation model for similar students in the future.

3. The AHA! system

AHA! (De Bra & Calvi, 1998) is a well-known open source general-purpose adaptive hypermedia system, available from the Eindhoven University of Technology, at http://aha.win.tue.nl/. AHA! offers low-level facilities for creating the exact look-and-feel for each application and for fine-tuning the adaptation; and it offers high-level facilities for creating the conceptual structure of an application, using concepts and concept relationships. Since AHA! is essentially an adaptive client and server at the same time, it can be used as a component in the content delivery pipeline and thus integrated into other server environments. AHA! was originally developed to support an online course with some user guidance through conditional (extra) explanations and conditional link hiding. In particular, AHA! has some adaptive hypermedia methods and techniques that are especially useful for educational applications:

- A user model based on concepts: Each time a student visits a concept in an AHA! course the name of the concept is passed to the adaptation engine which updates the user model. A user model consists of concepts that have attributes. A typical example of an AHA! action is that visiting a concept may increase a knowledge attribute for that concept. This knowledge update may propagate to the knowledge attribute of other concepts, perhaps corresponding to a section or chapter of a textbook. In AHA! a concept can have arbitrarily many attributes of types Boolean, integer or string. A concept may also have an associated resource which is a page to be presented to the user.
- Adaptive link hiding or link annotation: The suitability of link destinations (pages) is determined by an author-defined requirement. This is a (Boolean) expression using arbitrary user model values. The requirements can express the common prerequisite relationships between concepts but can be used for any other condition that can be expressed through such a Boolean expression. When a page is generated, links marked as conditional are displayed differently depending on the suitability of the link destination. If the expression is true the link is shown in blue (unvisited) or purple (visited), and when the expression is false the link is shown in black, and not underlined. This results in hiding the unsuitable or undesired links. The color scheme can also be altered by the end-user to make all links visible, in different colors.

AHA! is implemented as a typical Java-based Web application, using Servlets in combination with a Java-based server like Tomcat. Fig. 1 shows the different files or databases used by the AHA! engine.

In Fig. 1 we see that the AHA! engine uses three types of local information, and can also access pages that reside on other servers on the Web. The combined domain and adaptation model (DM/AM) represents the conceptual structure of the application. It consists of concepts and concept relationships. In AHA! every page that can be presented to the end-user must have a corresponding concept. All of this information is retrieved when the end-user logs in. As the user interacts with the application, a user model (UM) is used and is constantly updated. The UM consists of a set of concepts with attributes that are stored in student profile files. This model contains an overlay; for every concept in DM there is a concept in UM. All the interaction with the students is saved in student logs files. The AM is what drives the adaptation engine. It defines how user actions are translated into user model updates and into the generation of an adapted presentation of a requested page. In AHA! the presentation of a course uses a layout model to define how concepts are presented. A layout is basically a HTML frames structure; apart from a frame that shows a page there can be frames that show part of a table of contents, concepts of which the knowledge is increased by reading the current page, prerequisite concepts (and their knowledge level), etc. Through the powerful layout model AHA! applications can be made to look very similar to applications in other adaptive educational hypermedia platforms. Fig. 2 shows the AHA! tutorial as an AHA! course, using just one of many ways in which the presentation is possible.

As we can see in Fig. 2, AHA! tutorial interface has five layouts (the same as other AHA! courses normally have). In the left, the Index layout contains direct links to the main concepts, shown as a summary table of content of the course. In the top is the Header layout with information and links about the read pages and level of knowledge of the students. In the middle is the Content layout that contains the current concept (Web page) to show to the students. In the bottom at right, the Footer layout contains information about the copyright of the course. And in the bottom at left, the Application layout contains direct links to other useful application such as glossary, full table of contents, etc.

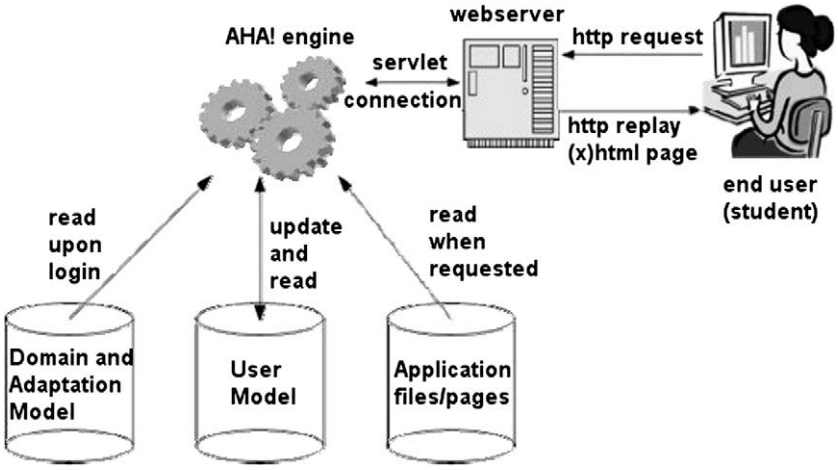

Fig. 1. AHA! architecture and data stores used by the AHA! engine.

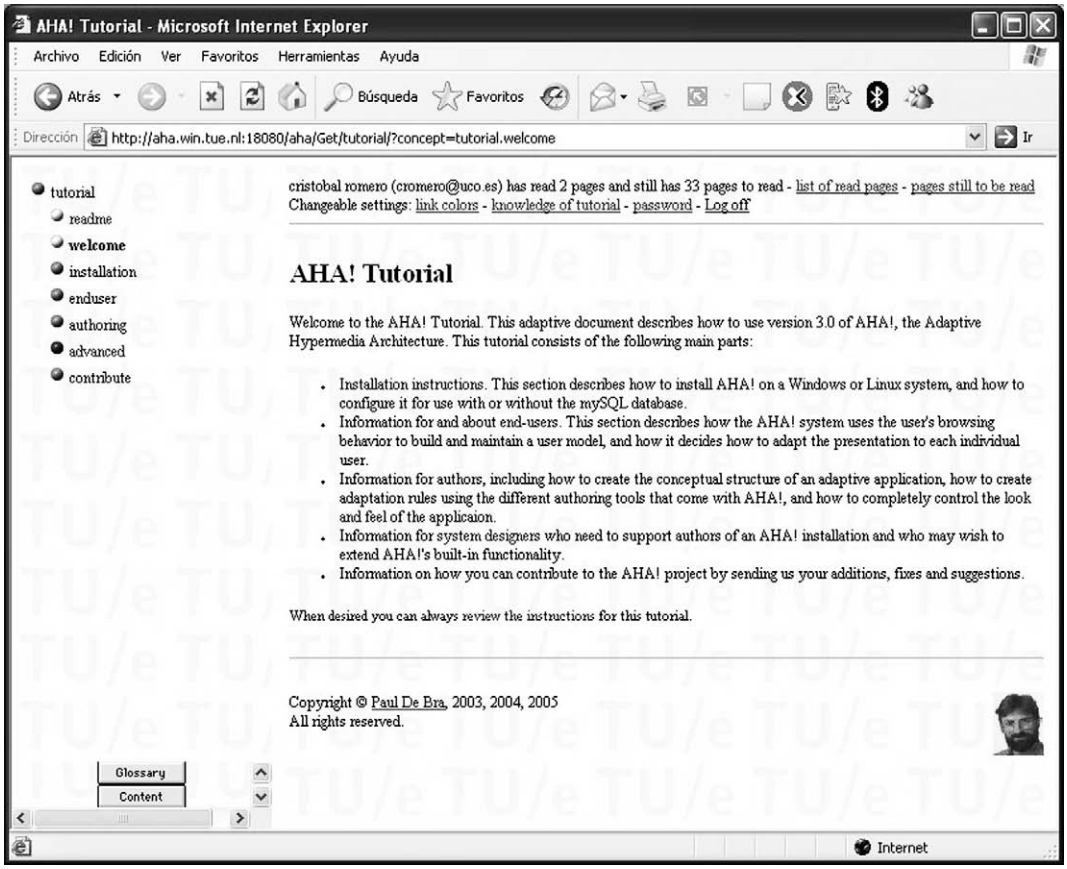

Fig. 2. AHA! tutorial.

4. AHA! Web mining system for personalized links recommendation

Most of the current data mining tools such as DBMiner (DBMiner, 2009), SPSS Clementine (Clementine, 2009) and Weka (Witten & Frank, 2005) can be too complex for educators to use and their features go well beyond the scope of what an educator may want to do. A specific tool should have easier-to-use interfaces to simplify the algorithm configuration and execution, and specialized visualization facilities to make its results meaningful to educators and courseware designers (Romero & Ventura, 2007). For this reason, we have developed a specific Web mining tool in order to help the teacher to carry out the Web usage mining process. We have integrated this tool and its corresponding recommendation engine into the well-known AHA! system (De Bra & Calvi, 1998) (Adaptive Hypermedia Architecture). In this way the whole process can be carried out within a single e-learning system, and the feedback and results obtained can be directly applied to the courses, that is, the new links can be automatically added into the course.

4.1. Architecture of the Web-based recommender system

The overall process of Web personalization based on Web usage mining generally consists of three phases: data preparation, pattern discovery and recommendation. The first two phases are performed off-line and the last phase is performed online (Mobasher, 2007). In our case, data preparation will transform Web log files and profiles into data with the appropriate format. Pattern discovery will use a data mining technique, such as clustering, sequential pattern and association rule mining. Finally, recommendation will use the discovered patterns to provide personalized links or contents.

We have designed a flexible architecture for Web-based recommendation (see Fig. 3) with two possible ways of using it: a basic mode or basic architecture (recommender engine only with sequential pattern mining) and an advanced mode or advanced architecture (with group matching and student clustering).

- The basic architecture only uses the student's information stored in Web log files. In this case, the system uses only one mining algorithm, usually a sequential mining algorithm, over all user navigation sessions to discover the most frequent navigational pattern that can predict the student's navigation and next page request. One problem of this type of basic architecture is that a new student obtains poor recommendations because they can be based solely on his or her current navigation (which is still very limited in the beginning of his first session).
- The advanced architecture also uses additional information about the students (such as profiles). This architecture uses several mining algorithms, in our case clustering and sequential pattern mining. In this way we can discover clusters of students showing their common general behavior and/or knowledge; we can then discover the sequential patterns of each cluster. This type of recommender can

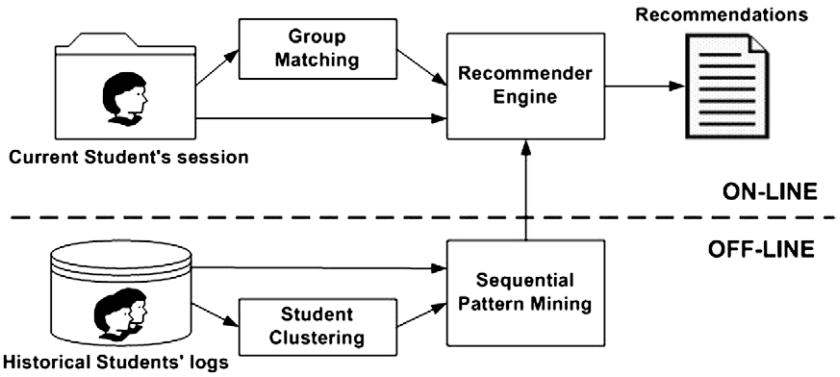

Fig. 3. Architecture of recommender system based on web usage mining.

personalize the recommendations. First, it classifies the new students in one of the groups of students (clusters). Then, it only uses the sequential patterns of the corresponding group to personalize the recommendations based on other similar students and his or her current navigation.

We have implemented this architecture into the AHA! system (De Bra & Calvi, 1998). We can see in Fig. 4 that the implemented system consists of two applications: AHA! mining tool (off-line) and recommender engine (online).

As we can see in Fig. 4, both the user model data (students log files and students profile files) and the learning model data (recommendation file and cluster file) are stored in XML (Extensible Markup Language) files into the AHA! server file system. And there are two main modules; the off-line module (mining tool) and the online module (recommender engine) that we are going to describe in detail in the next two sub-sections. It is important to notice that this system can work as both the basic and advanced architecture described previously.

4.2. AHA! mining tool

The AHA! mining tool is a Java Applet, just like the standard AHA! authoring tools (De Bra & Calvi, 1998) such as Graph Editor and Test Editor. The author of the course can execute it when enough information from new students has been collected. The user interface of the mining tool is simple, easy to use and specifically oriented to discover sequential patterns and to recommend personalized links. Its main window consists of a menu and two information areas (see Fig. 5). At the top, we can see the information panel that shows general information about the data and algorithm execution. At the bottom, we can see the sequential pattern panel, where the discovered sequences are shown.

First, the author must create a new data file starting from the student's log files. We have to preprocess the AHA! log files in order to group them into a single data file with the most appropriate format to be mined. In our case, it is not necessary to do user and session identification since all users must log in using their unique ID. AHA! also stores the session information in log files. AHA! stores all log information for each student in one XML file (the date and time at which the page was accessed, the session identification and the name of the Web page). The author only has to select one of his/her courses in AHA! and one of the three available methods for selecting students (automatic, manual and clustering) in order to create a data file:

- The totally automatic method selects all the students in the course.
- The manual method shows a list of all the students in the courses so that the author can select a specific group of students.
- The clustering method automatically creates several data files instead of only one. We have used the *k*-means algorithm (MacQueen, 1967), which is the most popular clustering algorithm; it is easy to use, as the instructor only has to specify the number of clusters (*k*) to find. In order to do clustering, we have used two of the students' variables: the number of pages visited and the average knowledge obtained from these pages. This information has been obtained from the AHA! XML User Model profile files (also one per student, containing visited and knowledge attributes for each concept of the course). It is important to note that in the used AHA! courses each con-

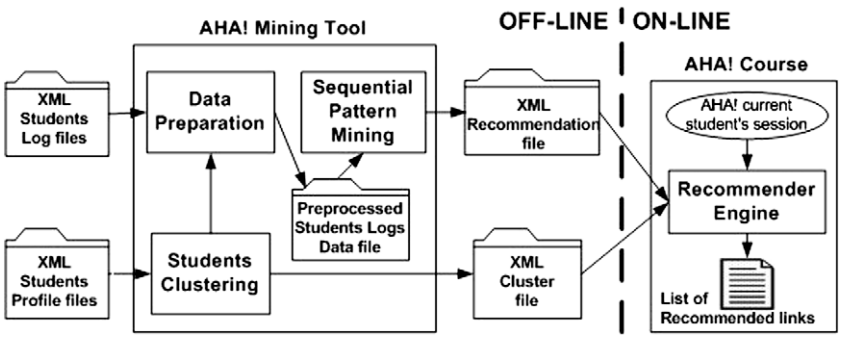

Fig. 4. AHA! mining tool and recommender engine integrated into AHA! system.

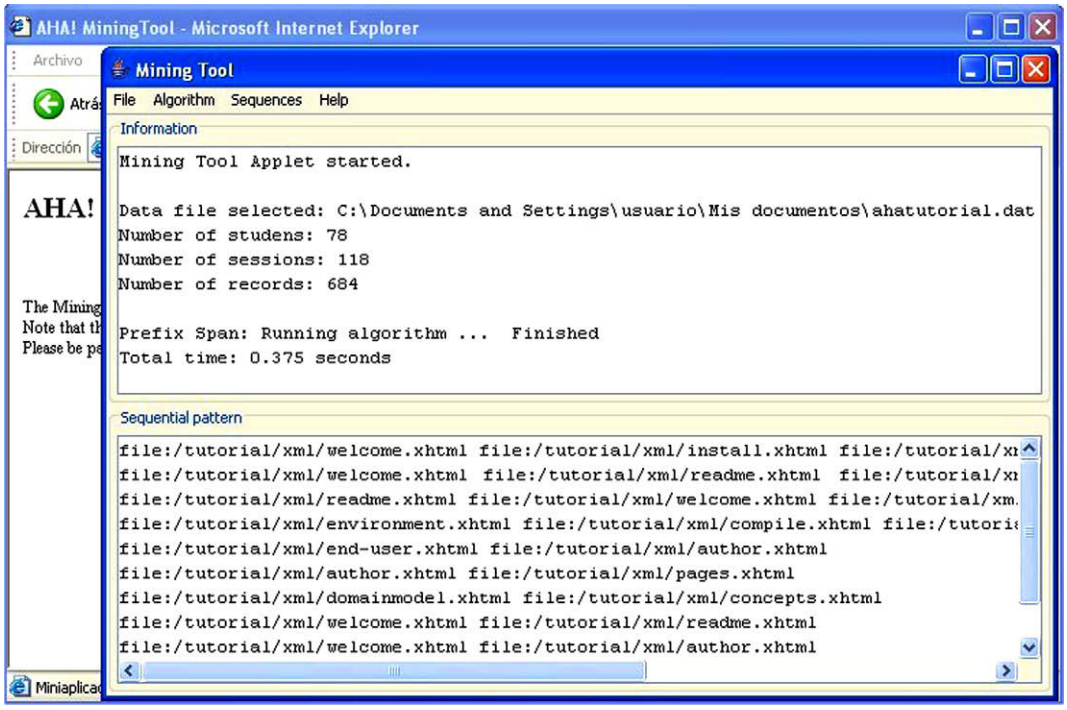

Fig. 5. Main window of the AHA!-based mining tool.

cept has an associated XHTML Web page. Each of the clusters obtained corresponds to a specific student's model and is stored in an XML file. In this file we store the centroid of each cluster (in some sense representing a typical user of the cluster). In our case we store the number of pages visited and the average knowledge of the centroid, as we can see in the next XML example file with two clusters: cluster 0 that represents sporadic students (low level of knowledge and very few visited pages) and cluster 1 that represents good students (high level of knowledge and more visited pages).

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfClusters NumberOfClusters="2">
  <cluster NumberOfPagesVisited="2"
    AverageLevelOfKnowledge="45">0</cluster>
  <cluster NumberOfPagesVisited="24"
    AverageLevelOfKnowledge="99">1</cluster>
</ListOfClusters>
```

Finally, one data file (for all automatic and manual methods) or several data files (for the clustering method) are created in the well-known ARFF Weka format (Witten & Frank, 2005) containing all the students logs information. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. The log information of each student is grouped together in this file or these files according to the clusters in which they have been classified. For each student, these log files contain the information about all the accessed Web pages grouped by session, that is, all the pages visited during a maximum period of time.

Then, the author can select one of these data files in order to execute a sequential pattern mining algorithm. We have several algorithms available, such as AprioriAll (Agrawal & Srikant, 1995), GSP (Srikant & Agrawal, 1996) and PrefixSpan (Pei et al., 2001), which are some of the most popular pattern discovery algorithms. The author can execute the selected algorithm directly or can change its default parameters values first. AprioriAll and PrefixSpan algorithms only have one parameter (minimum support threshold; that is, the minimum number of sessions in which the rule has to appear). The GSP algorithm has a second parameter (maximum number of gaps – the maximum number of gaps between two links to be considered in the same sequence).

When the algorithm finishes its execution, the discovered sequences are shown in the sequential pattern panel of the main tool window (see Fig. 3 down). Each sequence is shown in one line. The user has to use the horizontal and vertical scroll controls in order to see all the sequences (horizontal scroll) completely (vertical scroll). As we can see, a sequence is an ordered list of pages (only separated by a blank space), in which the full name of a page has the next format: file:/CourseName/Directory/PageName. These sequences can be saved into a text file, they can also be visualized using the sequence view window (see Fig. 6). The sequence view show sequences as nodes (Web pages visited) and arcs (to access from a page to other page). By analyzing these sequences, the teacher can get an idea about what the most general students' browsing behavior or trails during their learning process are. Trails are plans or routes that learners follow within curriculum elements, in our case the Web pages of the course. The teacher can easily see what are the most- and least-visited sequences or trails. This

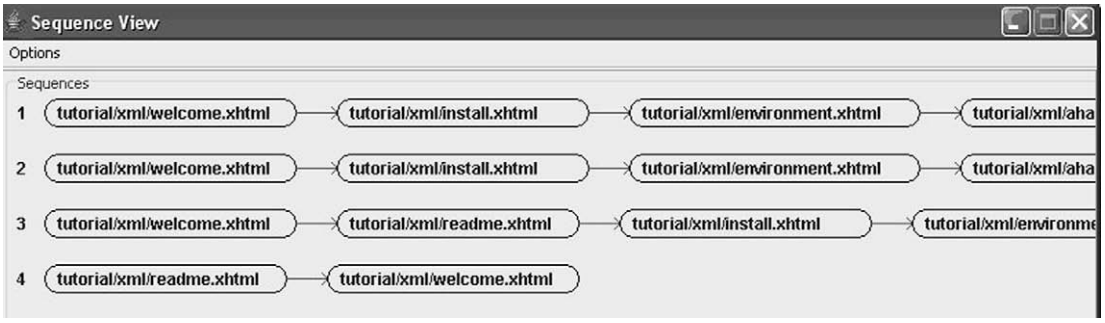

Fig. 6. Example of sequences.

information can be useful to a learning designer, providing a deeper understanding of the learning paths effectively followed by the students. For example, comparing this information with expected a priori paths allows the designer to refine the sequencing strategy.

Finally, the tool can recommend links starting from the obtained sequences. In order to do so, we have first split all the longer sequences into length 2 sub-sequences (ordered pairs) or rules using two different methods: path recommendation or shortcut recommendation (Ishikawa, Ohta, Yokoyama, Nakayama, & Katayama, 2002). Path recommendation splits the sequences into all the possible rules (every ordered pair of pages directly connected in the sequence) and the shortcut recommendation splits the sequence in a single rule (the first and the last page in the sequence). In Fig. 7, we show an example of sequence and the two splits methods.

So, a recommendation link is composed of an ordered pair considered as a rule with only one element in the antecedent and one in the consequent (the antecedent represents the page in which the recommendation will be shown and the consequent is the link destination recommended to the student). The support and confidence measures of each rule are also calculated starting from the student's usage data and are shown to the user. The support (value between 0 and 1) is a percentage of the students who visit the antecedent and the consequent together. The confidence (value between 0 and 1) is the conditional probability that students visit the consequent will occur given the occurrence of the antecedent. Then, all the generated recommendation links are shown to the author so that he/she can validate them and select which recommendations will be used by the recommender engine (see Fig. 8).

The author has to select links/rules (all, none or a specific group) in order to filter the most appropriate recommendations depending on the antecedent and consequent concepts, and the confidence and support values. The author can order all the rules by the value of the support or confidence by double-clicking in the corresponding column title. The confidence of the rules indicates how strong the rules are, whereas the support of the rules indicates their coverage. Finally, all the selected recommendations are saved into the extended AHA! system in an XML file as shown in the following example file:

```
<?xml version="1.0" encoding="UTF-8"?>
<RecommendedLinks>
  <Concept name="tutorial.welcome">
    <Recommendation text="installation"
     autogenerated="true" group="all" color="blue"
     interest="57">tutorial.installation
</Recommendation>
    <Recommendation text="readme"
     autogenerated="true" group="all" color="blue"
     interest="14">tutorial.readme</Recommendation>
  </Concept>
</RecommendedLinks>
```

We can see in the previous example file that there are two recommended links for the concept/page "welcome" of the course "tutorial". The value of the "recommendation" label indicates the name of the destination Web page/concept ("installation" and "readme", respectively). And the meanings of the attributes are: "text" (text of the hyperlink, by default this is the concept of the rule consequent), "autogenerated" (boolean value that indicates if the recommendation has been generated by data mining or not), "group" (indicates if the recommendation is for all students or for specific clusters), "interest" (the confidence value of the rule).

This recommendation file is prepared to be used by the recommender engine in online mode; meanwhile the student's use the AHA! system as is explained in the next section.

4.3. Recommender engine

We have developed our recommendation engine as a new AHA! View class (De Bra & Calvi, 1998), just like the other Views such as MainView, TOCView, and ConceptbarView. (Each HTML frame in an AHA! application shows one view. Typically there is a MainView which shows the course page, and one or more other views that provide navigation aids such as a partial table of contents.) Our recommender engine (RecommendedLinksView class) is activated each time that when the student visits a Web page (each concept corresponds to a Web page).

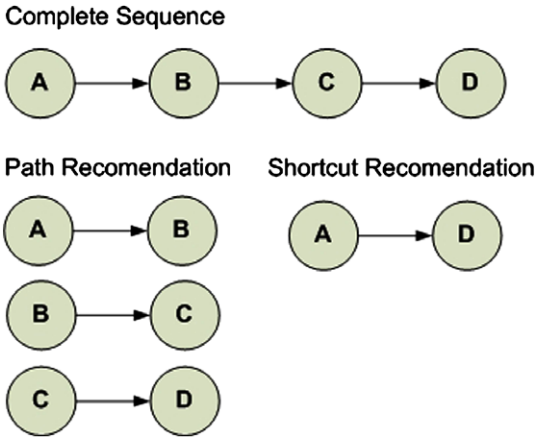

Fig. 7. Methods for splitting sequences.

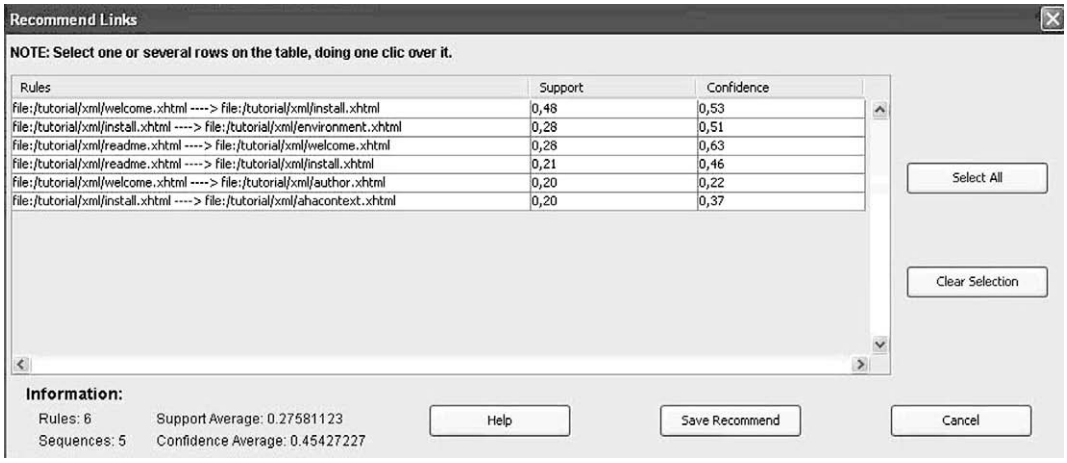

Fig. 8. Recommendation links window.

The recommendation engine considers both the active student in conjunction with the XML recommendation file to provide personalized recommendations. First, if there are clusters in the XML recommendation file, then the engine has to classify the current student to determine the most likely cluster. We have to communicate with the AHA! engine to obtain the current student profile (to know the current number of pages visited and average knowledge of the student). Then, we use the centroid minimum distance method (MacQueen, 1967) for assigning the student to the cluster whose centroid is closest to that student (XML cluster file). Finally, we make the recommendation according to the rules in the cluster. So, only the rules of the corresponding cluster (or all the rules if there are no clusters) are used to match the current Web page (concept) in order to obtain the current list of recommended links.

In Fig. 9 we can see the interface of the AHA! tutorial with a list of recommended links (indicated by a circle and label) in which the student "cristobal" is located on the "welcome" page and the recommender engine recommends going to the "installation" page (strong recommendation) and to the "readme" page (normal recommendation). We can see that we have adaptively sorted, annotated and partly hidden (Brusilovsky & Peylo, 2003) the list of recommended links (Farzan & Brusilovsky, 2006). First, we only select the links that match with the current student's state and the current page. Next, these links are sorted depending on their confidence value (on a decreasing scale). Then, we annotate the links with triangular icons that can vary in color depending on which data have been used to obtain them (blue for all the data, green to a specific data cluster) and we can vary their number depending on the value of the confidence (1 triangle or normal recommendation to values lower than 0.33, 2 triangles or strong recommendation to values higher than 0.33 and lower than 0.66, and 3 triangles or very strong recommendation to values over 0.66).

5. Experimental results

We have carried out several experiments in order to test the architecture proposed and the algorithms used. The data used in this study are collected from the online AHA! tutorial (http://aha.win.tue.nl/tutorial/) that consists of 43 Web pages. We have used a total number of 78 students with 118 sessions and 684 records. These students are mainly TU/e (Eindhoven University of Technology) students taking a traditional course in adaptive hypermedia and some other Internet users interested in the AHA! system and taking the tutorial online (rather than installing AHA! first and then using their local version). So, all students used in this work are familiar with adaptive hypermedia. We have carried out three experiments that we describe in the following section.

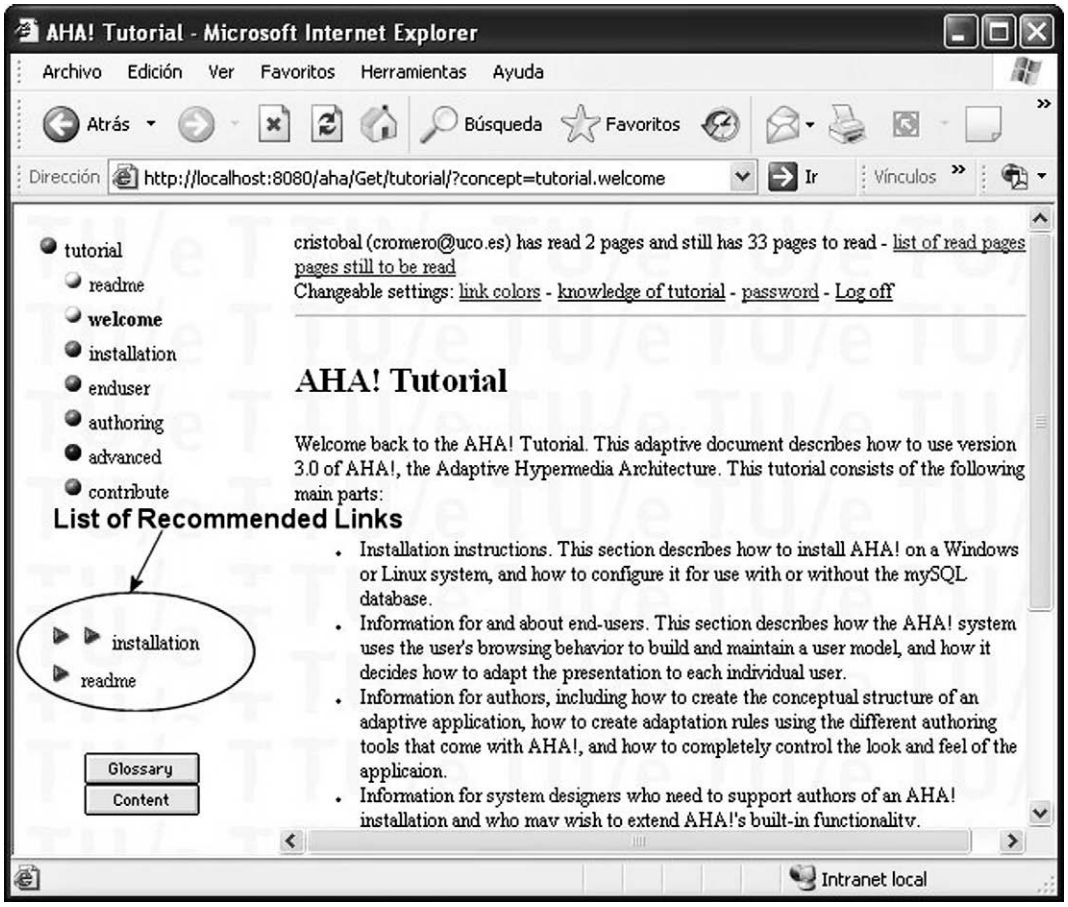

Fig. 9. AHA! tutorial with recommended links added.

In the first experiment, we have executed the three available sequence mining algorithms: AprioriAll (Agrawal & Srikant, 1995), GSP (Srikant & Agrawal, 1996) and PrefixSpan (Pei et al., 2001), using all the data available in order to find out which is the best for our problem. We have compared the shortcut recommendation rules discovered (number of rules discovered and the average value of the support and confidence of the rules) by the three algorithms varying the minimum support threshold (from 0.3 to 0.03).

As we can see in Table 1, there are not many differences between the three algorithms (specifically with higher minimum support values). However, GSP and PrefixSpan discover a lower number of rules with higher support and confidence values. In our problem of links recommendation, it is important to show to the students only good links (a small number of rules with a higher value of support and confidence). So, although the three algorithms show similar results, the GSP and PrefixSpan are a bit better than the AprioriAll algorithm. And as far as the minimum support threshold is concerned, we can see that in order not to obtain a lot of rules, a good range is between 0.3 and 0.1.

In the second experiment, we have executed the clustering algorithm in order to find out the number of clusters that are more appropriated to use with our data. We have executed the *k*-means algorithm (Li & Zaïane, 2004) varying the *k* value (number of clusters) from 2 to 5. Table 2 shows the number of students, sessions and records that are obtained in each one of the clusters.

In our problem of grouping students into different clusters, it is important for the obtained data to be balanced (equal number in each cluster). That is, the number of students, sessions and records must be uniform in all the clusters. In this way, we can later obtain a similar number of sequence rules for each data cluster. We can see in Table 2 that the data are more balanced when we use a low number of clusters (2 and 3 clusters). But when we increase the number of clusters (4, 5 and more) then there are more differences between clusters (some clusters have a lot of data and others have very few data). So, two or three clusters give us well-balanced data.

In the third experiment, we have done a comparison study between the basic architecture (only sequential mining) and the advanced architecture (clustering and sequential mining). In order to do it, we have compared the shortcut recommendation rules discovered (the number of rules discovered and the average value of the support and confidence of the rules) by the PrefixSpan algorithm varying the min-

Table 1
Number/average support/average confidence of rules discovered using all data.

	Minimum support = 0.3	Minimum support = 0.15	Minimum support = 0.07	Minimum support = 0.03
AprioriAll	3/0.35/0.55	7/0.24/0.43	22/0.13/0.40	70/0.07/0.32
GSP(gap = 1)	2/0.39/0.55	6/0.24/0.43	20/0.16/0.43	62/0.11/0.40
PrefixSpan	2/0.39/0.55	6/0.24/0.43	14/0.18/0.43	61/0.12/0.41

Table 2
Number of students/sessions/records in each data cluster.

No clusters	All data				
Number of stud./sess./records	78/118/684				
Cluster $K = 2$	Cluster 1	Cluster 2			
Number of stud./sess./records	48/60/383	30/58/301			
Cluster $K = 3$	Cluster 1	Cluster 2	Cluster 3		
Number of stud./sess./records	18/34/120	32/45/299	28/39/265		
Cluster $K = 4$	Cluster 1	Cluster 2	Cluster 3	Cluster 4	
Number of stud./sess./records	23/38/233	12/16/99	11/21/87	32/43/265	
Cluster $K = 5$	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Number of stud./sess./records	14/30/106	12/25/81	6/14/40	4/12/26	42/47/449

Table 3
Number/average support/average confidence of rules discovered using PrefixSpan over all data and over different numbers of clusters.

	Minimum support = 0.3	Minimum support = 0.15	Minimum support = 0.1
No clusters (all data)	2/0.39/0.55	6/0.22/0.43	12/0.20/0.39
$k = 2$ (Cluster 1)	1/0.40/0.57	6/0.28/0.46	10/0.24/0.47
$k = 2$ (Cluster 2)	1/0.40/0.63	8/0.23/0.48	14/0.26/0.45
$k = 3$ (Cluster 1)	2/0.39/0.64	7/0.31/0.53	16/0.21/0.51
$k = 3$ (Cluster 2)	1/0.39/0.55	6/0.32/0.55	11/0.26/0.53
$k = 3$ (Cluster 3)	2/0.40/0.62	8/0.32/0.56	13/0.25/0.49

Table 4
Examples of rules discovered.

Rule number	Antecedent = >	Consequent	Support	Confidence	Cluster number
1	Readme	Install	0.23	0.41	1
2	Domainmodel	Concept	0.25	0.60	2
3	Author	Pages	0.27	0.32	1
3	Author	Pages	0.21	0.42	2
4	Welcome	Install	0.48	0.63	1
4	Welcome	Install	0.40	0.52	2

imum support threshold (from 0.3 to 0.1), using all data, and on the other hand, the same algorithm using the data obtained from the *k*-means algorithm for 2 and 3 clusters.

We can see in Table 3 that the number of discovered rules using each data cluster is not always less (sometimes even more) than using all the data, as we might have expected. However, their support and confidence values are always higher and this is very important in our problem. So, the advanced architecture can discover a similar number of rules as the basic architecture but with higher values of confidence and support.

Finally, we are going to describe the meaning of several rules regarding our AHA! tutorial course that we discovered using two data clusters ($k = 2$) and the PrefixSpan algorithm (minimum support = 0.15). The objective is to describe some typical recommendation and also to check if there are differences in the rules obtained from different data clusters.

We can see in Table 4 that there are some rules that only appear in one cluster (rules 1 and 2), and there are other rules that appear in both clusters (rules 3 and 4) but with different support and confidence values. Cluster number 1 represents sporadic students who only want to sort out one question about AHA! (its centroid has NumberOfPagesVisited = 2 and AverageLevelOfKnowledge = 45) and cluster number 2 represents active students really interested in reading the complete AHA! tutorial (its centroid has NumberOfPagesVisited = 16 and AverageLevelOfKnowledge = 99). Rule number 1 shows that sporadic students go from the "readme" Web page to "install" Web page (these students are looking for some specific question about the AHA! installation, and perhaps after figuring this out they continue with an AHA! server and tutorial they install locally). Rule number 2 shows that active students go from the "domainmodel" Web page to the "concept" Web page (these students are reading/learning about the AHA! core). Rule number 3 shows that both types of students go from the "author" Web page to the "pages" Web page, but this is a higher number for the active students and the confidence is higher too (these students are reading/learning about the AHA! page format) than for sporadic students. Rule number 4 shows that both types of students go from the "welcome" Web page to the "install" Web page, but a higher number of sporadic students do so and with higher confidence than active students.

6. Conclusions and future work

In this paper, we have proposed the architecture of a recommender system that utilizes Web usage mining to recommend the links to visit next in an adaptive Web-based educational system. A specific mining tool and a recommender engine have been developed to help the instructor to carry out the Web mining process. Although we have integrated both the Web mining tool and the recommender engine into the AHA! System, it can, in principle, also be used in other Web-based educational systems. To do so, it is necessary to modify mainly two

things. Firstly, the input data of the Web mining tool have to be changed in order to be able to handle other types of data format used by other ASWEs. Currently, the Web mining tool only works with AHA! data stores (XML student logs and profile files). So, it is necessary to modify/extend part of the pre-processing functions in order to add other input data. And secondly, a new recommendation engine has to be developed in order to be able to communicate the recommendations with the new ASWE. Currently, the recommendation engine is integrated into the AHA! system. Therefore it is necessary to develop a specific recommendation engine integrated into the new ASWE that uses the output data (XML recommendation and cluster files) of the Web mining tool in the same way the current recommendation engine does.

Several experiments with real user data from the online AHA! tutorial have been carried out to show the performance of the different algorithms implemented. These indicate the suitability of an advanced recommender system that uses clustering and sequential pattern mining algorithms together to discover personalized recommendation links. In fact, we have shown that an advanced architecture can discover as many rules as a basic architecture but with higher values of confidence and support, which is very important to our problem. Moreover, the advanced architecture can personalize the recommendations because it classifies the students in one of the clusters and only uses the rules of the corresponding cluster to make the recommendations. Finally, we have shown the meaning of some examples of the recommendations obtained from the AHA! tutorial course. However, no tests have yet been run to find out how much students actually use these recommendations. In order to do so, we plan to evaluate the quality of the recommendations based on feedback from students. A new group of students will be used, who will again do the course but with the recommended links added in order to measure how much the proposed links are really used.

In the future, we also would like to carry out further experiments, using more courses, still larger numbers of students and more information about the students' profiles, all in order to do clustering. Other sequence mining algorithms (Han et al., 2005) can also be integrated, such as SPADE, FreeSpan, CloSpan and PSP, along with other clustering algorithms that do not require the user to specify any parameters.

Finally, it would be very useful to develop a real-time feedback loop between Web mining and the recommendation engine. For example, we could use intelligent agents to do online Web mining automatically. In this way the recommender system could work completely autonomously. That is, the agents can apply the mining algorithm data automatically when they detect enough volume of new usage data. In this way, the authors do not have to pre-process and apply mining algorithms manually; but only accept or reject the recommendation in order to add new links to the course interface. A virtual character (Reategui, Boff, & Campbell, 2008) could also be added in order to show the recommendations in a more user-friendly way.

Acknowledgement

The authors gratefully acknowledge the financial support provided by the Spanish department of Research under TIN2008-06681-C06-03 and P08-TIC-3720 Projects.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering, Taipei, Taiwan* (pp. 3–14).
Agrawal, R., Imielinski, T., & Swami, A. N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD international conference on management of data, Washington, DC, USA* (pp. 207–216).
Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*, 167–207.
Blackboard (2009). <http://www.blackboard.com>.
Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction, 6*(2–3), 87–129.
Brusilovsky, P. (2001). Adaptive educational hypermedia. In *Proceedings of the tenth international PEG conference, Finland* (pp. 8–12).
Brusilovsky, P., & Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. In *The adaptive web* (pp. 3–53). LNCS 4321, Springer-Verlag.
Brusilovsky, P., & Peylo, C. (2003). Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education, 13*, 156–169.
Burke, R. (2007). Hybrid web recommender systems. In *Proceedings of the adaptive web* (pp. 377–408). LNCS 4321, Springer-Verlag.
Carbonell, J. R. (1970). Al in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man–Machine Systems, 11*, 190–202.
Ceglar, A., & Roddick, J. F. (2006). Association mining. *ACM Computing Surveys, 38*(2), 1–5.
Clementine (2009). <http://www.spss.com/clementine/>.
DBMiner (2009). <http://www.dbminer.com/>.
De Bra, P., & Calvi, L. (1998). AHA! An open adaptive hypermedia architecture. *The New Review of Hypermedia and Multimedia, 4*, 115–139.
Delort, J., & Bouchon, B. (2002). Facing uncertainty in link recommender systems. In *Proceedings of the eleventh international world wide web conference, Honolulu* pp. 252–253.
Farzan, R., & Brusilovsky, P. (2006). Social navigation support in a course recommendation system. In *Proceedings of the fourth international conference on adaptive hypermedia and adaptive web-based systems, Dublin* (pp. 91–100).
Frias-Martinez, E., Chen, S. Y., & Liu, X. (2006). Survey of data mining approaches to user modeling for adaptive hypermedia. *IEEE Transactions on Systems, Man and Cybernetics. Part C: Applications and Reviews, 36*(2), 734–748.
García, E., Romero, C., Ventura, S., & Castro, C. (2006). Using rules discovery for the continuous improvement of e-learning courses. In *Proceedings of the conference intelligent data engineering and automated learning, Burgos, Spain* (pp.887–895).
Han, J., Pei, J., & Yan, X. (2005). Sequential pattern mining by pattern-growth: Principles and extensions. *Studies in Fuzziness and Soft Computing, 180*, 183–220.
Henze, N., & Nejdl, W. (2001). Adaptation in open corpus hypermedia. *International Journal of Artificial Intelligence in Education, 12*(4), 325–350.
Ishikawa, H., Ohta, M., Yokoyama, S., Nakayama, J., & Katayama, K. (2002) On the effectiveness of web usage mining for page recommendation and restructuring. In *Proceedings of the web, web-services and database systems, Erfurt, Germany* (pp. 253–267).
Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264–323.
Koutri, M., Avouris, N., & Daskalaki, S. (2005). *A survey on Web usage mining techniques for Web-based adaptive hypermedia systems. Adaptable and adaptive hypermedia systems* (pp. 125–149). IRM Press.
Ksristofic, A. (2005). Recommender system for adaptive hypermedia applications. In *Proceedings of the informatics and information technology student research conference, Bratislava* (pp. 229–234).
Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). *SHERLOCK: A coached practice environment for an electronics troubleshooting jog. Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*. Lawrence Erlbaum.
Li, J., & Zaïane, O. (2004). Combining usage, content, and structure data to improve web site recommendation. In *Proceedings of the international conference on e-commerce and web technologies, Beijing, China* (pp. 305–315).
Liu, B. (2007). *Web data mining: Exploring hyperlinks, contents and usage data*. Springer.
Lu, J. (2004). Personalized e-learning material recommender system. In *Proceedings of the international conference on information technology for application, Harbin, China* (pp. 374–379).
Luan, J. (2002). Data mining, knowledge management in higher education, potential applications. In *Workshop associate of institutional research international conference, Toronto* (pp. 1–18).
MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings fifth Berkeley symposium on mathematical statistics and probability, California, USA* (pp. 281–297).

Merceron, A., & Yacef, K. (2004). Mining student data captured from a webbased tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research, 15*(4), 319–346.
Mobasher, B. (2007). Data mining for personalization. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web personalization* (pp. 1–46). Berlin, Heidelberg: Springer.
Moodle (2009). <http://moodle.org/>.
Mostow, J., & Beck, J. (2006). Some useful tactics to modify, map and mine data from intelligent tutors. *Natural Language Engineering, 12*(2), 195–208.
Pahl, C., & Donnellan, C. (2003). Data mining technology for the evaluation of Web-based teaching and learning systems. In *Proceedings of the congress e-learning, Montreal, Canada* (pp. 1–7).
Pazzani, M. J., Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325–334). LNCS 4321, Springer-Verlag.
Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., et al. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the seventeenth international conference on data engineering, Heidelberg, Germany* (pp. 2215–224).
Reategui, E., Boff, E., & Campbell, J. A. (2008). Personalization in an interactive learning environment through a virtual character. *Computers and Education, 51*, 530–544.
Romero, C., & Ventura, S. (2006). *Data mining in e-learning*. Wit Press.
Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications, 1*(33), 135–146.
Romero, C., Ventura, S., & De Bra, P. (2004). Knowledge discovery with genetic programming for providing feedback to courseware author, user modeling and user-adapted interaction. *The Journal of Personalization Research, 14*(5), 425–464.
Romero, C., Ventura, S., & Salcines, E. (2008). Data mining in course management systems: Moodle case study and tutorial. *Computer and Education, 51*(1), 368–384.
Schafer, J. B. (2005). The application of data-mining to recommender systems. In J. Wang (Ed.), *Encyclopedia of data warehousing and minin* (pp. 44–48). Hershey, PA: Idea Group.
Schafer, J. B., Frankowski, D., Herlocker, J., & Shilad, S. (2007). Collaborative filtering recommender systems. In *The adaptive web* (pp. 291–324). LNCS 4321, Springer-Verlag.
Shen, L.P., & Shen, R.M. (2004). Learning content recommendation service based-on simple sequencing specification. In *Proceedings advanced in web-based learning, Beijing, China* (pp. 363–370).
Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the international conference on extending database technology, Avignon, France* (pp. 3–17).
Tiffany, Y. T., & Gordon, M. (2003). Smart recommendation for an evolving e-learning system. In *Proceedings of the workshop on technologies for electronic documents for supporting learning, Sydney, Australia* (pp. 699–710).
Tsantis, L., & Castellani, J. (2001). Enhancing learning environments through solution-based knowledge discovery tools. *Journal of Special Education Technology, 16*(4), 1–35.
Wang, F. H., & Shao, H. M. (2004). Effective personalized recommendation based on time-framed navigation clustering and association mining. *Expert System with Applications, 27*, 365–377.
WebCT (2009). <http://www.webct.com/>.
Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education, 12*(4), 351–384.
Witten, I. H., Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*, Morgan Kaufmann.
Zaïane, O. (2002). Building a recommender agent for e-learning systems. In *Proceedings of the international conference in education, New Zealand* (pp. 55–59).