Personalized Links Recommendation Based on Data Mining in Adaptive Educational Hypermedia Systems

Cristóbal Romero[1], Sebastián Ventura[1], Jose Antonio Delgado[1], Paul De Bra[2]

[1]Córdoba University, Campus Universitario de Rabanales, 14071, Córdoba, Spain
{cromero,sventura, i92deosj}@uco.es
[2]Eindhoven University of Technology (TU/e), PO Box 513, Eindhoven, The Netherlands
debra@win.tue.nl

Abstract. In this paper, we describe a personalized recommender system that uses web mining techniques for recommending a student which (next) links to visit within an adaptable educational hypermedia system. We present a specific mining tool and a recommender engine that we have integrated in the AHA! system in order to help the teacher to carry out the whole web mining process. We report on several experiments with real data in order to show the suitability of using both clustering and sequential pattern mining algorithms together for discovering personalized recommendation links.

1 Introduction

Adaptive and intelligent web-based educational systems (AIWBES) provide an alternative to the traditional just-put-it-on-the-web approach in the development of web-based educational courseware [4]. Their main objective is to adapt and personalize learning to the needs of each student. The task of delivering personalized content is often framed in terms of a recommendation task in which the system recommends items to an active user [17]. Recommender systems help users find and evaluate items of interest. Such systems have become powerful tools in many domains from electronic commerce to digital libraries and knowledge management [23]. Some recommender systems have also been applied to AIWBES for recommending lessons (learning objects or concepts) that students should study next [19] or for providing course recommendation about courses offered that contribute to the student's progress towards career goals [8].

Recommender systems can use data mining techniques for making recommendations using knowledge learnt from the action and attributes of users [23]. The objective of data mining 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. Currently, there is an increasing interest in data mining and educational systems, making educational data mining a new and growing research community [20][21]. The data mining approach to personalization uses all the available information about users/students on the web site (in the web course) in order to learn user models and to use these models for personalization. These systems can use different recommendation techniques in order to suggest online learning activities or optimal browsing pathways to students, based on their preferences, knowledge and the browsing history of other students with similar characteristics.

In this work, we are going to describe the use of data mining techniques for links recommendation in AIWBES. 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 [4]. 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 AIWBES and there are a lot of systems that use it, such as ELM-ART [28] (and its descendents), AHA! [7], KBS-Hyperbook [11], etc. The originality of our personalized recommender system consists in the use of data mining together with hyperlink adaptation. Only a few other recommender systems use data mining for recommending links [8].

This paper is arranged in the following way: first we describe the related background and two architectures for personalization based on-web usage mining. Then, we describe the data 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

Recommendation and personalization techniques can be classified into three different categories [17]: rule-based filtering systems, content-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 recommend items that are considered sufficiently similar to the content descriptions in the user profile. Collaborative filtering systems, also referred to as social filtering, match the rating of a current user for items with those of similar users in order to produce recommendations for items not yet rated or seen. Some recent techniques used in collaborative filtering are based on data mining in order to infer recommendation rules or build recommendation models from large data sets [23]. Some of the most common data mining techniques in these recommender applications are clustering, sequence and association mining.

- Clustering is a process of grouping objects into classes of similar objects [13]. It is an unsupervised classification or partitioning of patterns (observations, data items, or feature vectors) into groups or subsets (clusters). This technique groups records together 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 [13], including hierarchical and function-based algorithms. One of the most well-known and commonly used is the k-means algorithm [16] that tries to minimize the distance of the objects to the centroid or mean point of each cluster.

- Sequential modeling or sequential pattern mining [10] discovers inter-session patterns. It is a more restrictive form of association rule [1] in which the accessed items' order 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, as follows [2]: 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 all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than the minimum support. Normally, 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 [10] such as AprioriAll, GSP, SPADE, PrefixSpan, CloSpan and FreSpan.

Although personalized recommendation approaches that use data mining techniques are 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. The extraction of sequential patterns has been used to find patterns that are used in the process of recommending relevant concepts to students [19]. Sequential rules can also guide a learning resource recommendation service based on simple sequencing specification [24]. Clustering can be used to find clusters of students with similar learning characteristics and to promote group-based collaborative learning in a research paper recommender system [26]. Association rules and clustering methods have been used for recommending a list of web pages on an e-learning web site [27]. A recommender agent which uses association rules has been used to recommend online learning activities or shortcuts on a course web site based on a learner's access history [14][30]. Fuzzy association rules have been used in a personalized e-learning material recommender system [15]. Finally, association rules have been used to provide feedback to the courseware author and to recommend how to improve the courses [9][22].

3 Architectures for personalized recommendation systems

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 on-line [17]. Data preparation transforms web log files and profiles into data with the appropriate format. Pattern discovery uses a data mining technique, such as clustering, sequential pattern and association rule mining. Finally, recommendation uses the discovered patterns to provide personalized links or contents.

In this work, we distinguish between two different architectures of recommender systems based on web usage mining (see Figure 1).

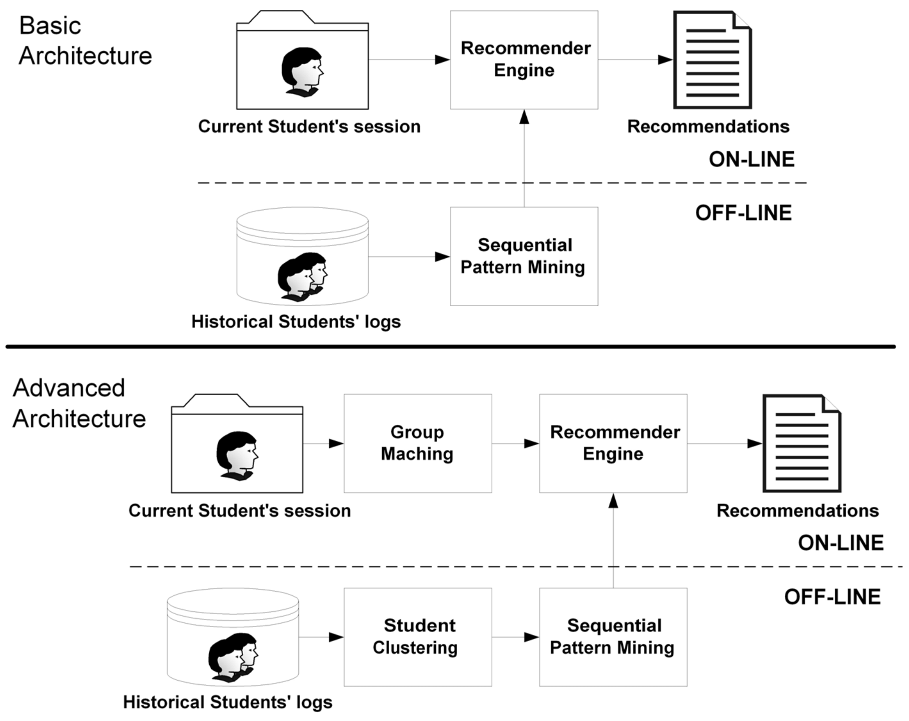


Fig. 1. Architectures for recommender systems based on web usage mining.

- **Basic Architecture of Web-based Recommender Systems.** This is a simple architecture of a recommender system that only uses the student's information stored in web log files. These systems use only one mining algorithm, usually a sequential mining algorithm (see Figure 1 above), 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 system is that the new student obtains the same recommendations based solely on his current navigation.
- **Advanced Architecture of Web-based Recommender Systems.** This is a more advanced architecture of recommender systems that also uses additional information about the students (such as profiles). These systems use several mining algorithms (see Figure 1 down), for example, clustering and sequential pattern mining. In this way they can discover clusters of students showing common behavior and/or knowledge and then they can discover the sequential patterns of each cluster. This type of recommender can 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 current navigation.

4 The AHA!-based Mining and Recommender System

Most of the current data mining tools such as DBMiner [6], SPSS Clementine [5] and Weka [29] can be too complex for educators to use and their features go well beyond the scope of what an educator may want to do. These tools should have more easy-to-use interfaces to simplify the algorithm configuration and execution, and they have provided specialized visualization facilities to make their results meaningful to educators and courseware designers [20]. For this reason, we have developed a specific data mining tool in order to help the teacher to carry out the web mining process. We have integrated this tool and its corresponding recommendation engine into the well known AHA! [7] (Adaptive Hypermedia Architecture). In this way the whole process can be carried out in a same e-learning system, and the feedback and results obtained can be directly applied to the courses (see Figure 2).

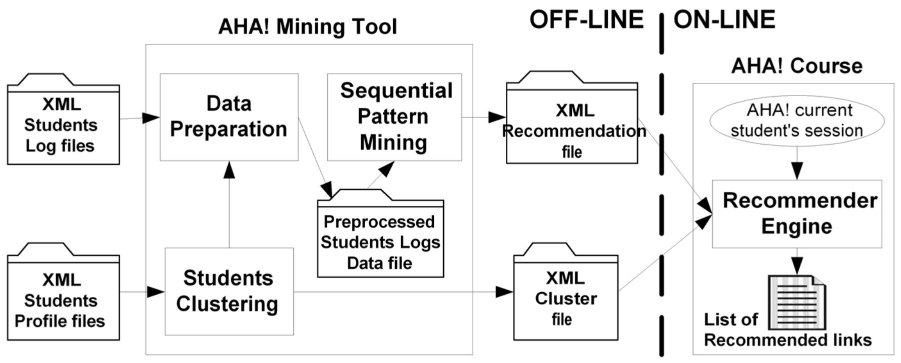
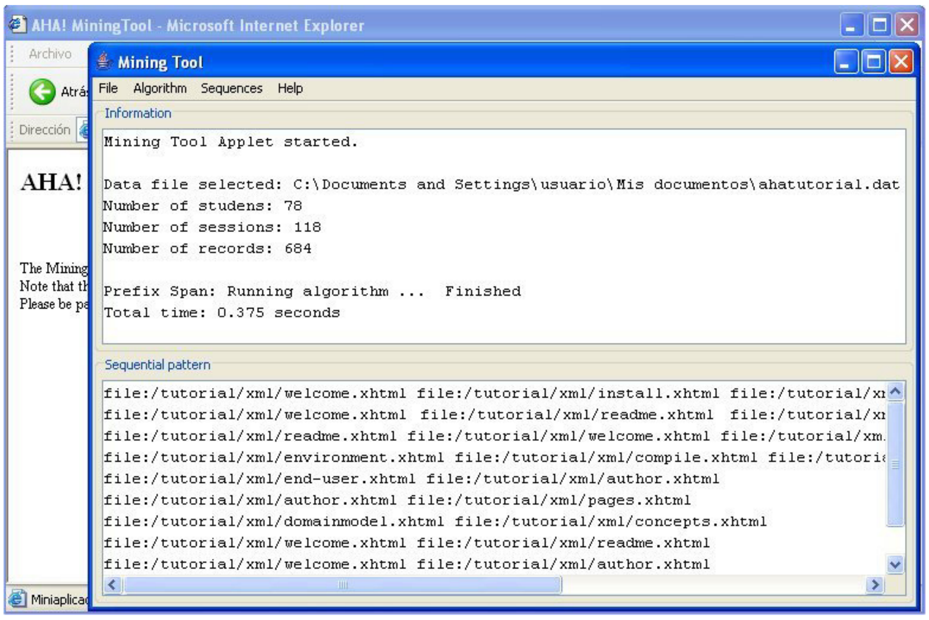


Fig. 2. Data mining tool and recommender engine integrated into AHA! system.

As we can see in Figure 2, both the user's data (student log and profile files) and the learning model data (recommendation and cluster files) are stored in XML files in the AHA! server file system. This system can work as both the basic architecture and advanced architecture described in the previous section. And there are two main modules; the off-line module (mining tool) and the on-line module (recommender links engine) that we are going to describe in detail in the next two sub-sections.

4.1 Mining Tool

The mining tool is a Java Applet, just like other AHA! authoring tools [7] such as Concept Editor, Test Editor, etc. 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 Figure 3). At the top, we can see the information panel that shows general information about the program and algorithms execution. At the bottom, we can see the sequential pattern panel, where the discovered sequences are shown.



```
Mining Tool Applet started.

Data file selected: C:\Documents and Settings\usuario\Mis documentos\ahatutorial.dat
Number of studens: 78
Number of sessions: 118
Number of records: 684

Prefix Span: Running algorithm ...  Finished
Total time: 0.375 seconds
```

```
file:/tutorial/xml/welcome.xhtml file:/tutorial/xml/install.xhtml file:/tutorial/x
file:/tutorial/xml/welcome.xhtml  file:/tutorial/xml/readme.xhtml  file:/tutorial/x
file:/tutorial/xml/readme.xhtml file:/tutorial/xml/welcome.xhtml file:/tutorial/xm
file:/tutorial/xml/environment.xhtml file:/tutorial/xml/compile.xhtml file:/tutori
file:/tutorial/xml/end-user.xhtml file:/tutorial/xml/author.xhtml
file:/tutorial/xml/author.xhtml file:/tutorial/xml/pages.xhtml
file:/tutorial/xml/domainmodel.xhtml file:/tutorial/xml/concepts.xhtml
file:/tutorial/xml/welcome.xhtml file:/tutorial/xml/readme.xhtml
file:/tutorial/xml/welcome.xhtml file:/tutorial/xml/author.xhtml
```

Fig. 3. Main window of the AHA! mining tool.

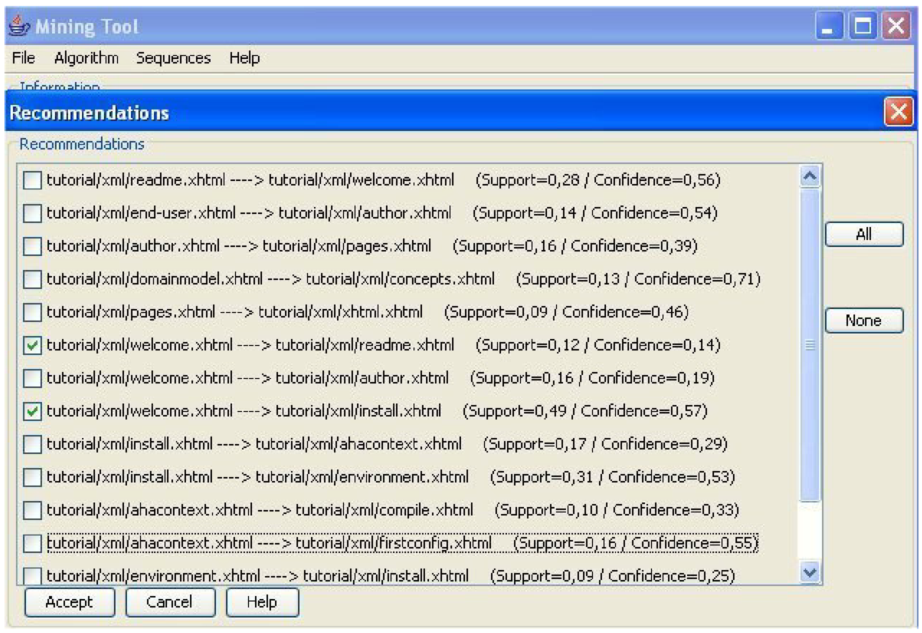
First, the author has to create a new data file starting from the student's log files. We have to preprocess the AHA! log files in order to group them into a single data file with the most appropriate format to be mined. In our case, it is not necessary to do user and session identification since all users must log in using their unique ID, and 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 method for selecting students (all automatically, 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 with all the students in the courses so that the author can select a group of specific students. The clustering method automatically creates several data files instead of only one data file. We have used the k-means algorithm [16], which is the most popular clustering algorithm and where the user 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 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 concept 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 file example with two clusters:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfClusters NumberOfClusters="2">
  <cluster NumberOfPagesVisited="2"
      AverageLevelOfKnowledge="45">0</cluster>
  <cluster NumberOfPagesVisited="16"
      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 KEEL format [3]. This is a text file format that is similar to and compatible with the well-known Weka format [29]. 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.

Then, the author can select one data file in order to execute sequential pattern mining algorithms. There are several algorithms available such as AprioriAll [2], GSP [25] and PrefixSpan [18], which are some of the most popular pattern discovering algorithms. The author can execute the selected algorithm directly or, if he wants, he can change its default parameters values. 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 that is the maximum number of gaps between two links to be considered in the same sequence). When the algorithm finishes its execution, the sequences discovered are shown in the sequential pattern panel of the main tool window (see Figure 3). These sequences can be saved into a text file and they can also be visualized better using the sequence view window. Analyzing these sequences, the teacher can have an idea about what the most general students' browsing behavior during their learning process is.

Finally, the author can recommend links starting from the sequences obtained. In order to do so, we have first split all the sequences with lengths over two in 2-length sub-sequences or rules using two different methods: path recommendation or shortcut recommendation [12]. Path recommendation splits the sequences in all the possible rules (every two pages directly connected in the sequence) and the shortcut recommendation splits the sequence in only one rule (the first and the last page in the sequence). So, a recommendation link is composed of a 2-length sequence 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 recommended to the student). 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 Figure 4). 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 confidence of the rules indicates how strong the rules are, whereas the support of the rules indicates their coverage.



```
☐ tutorial/xml/readme.xhtml ----> tutorial/xml/welcome.xhtml   (Support=0,28 / Confidence=0,56)
☐ tutorial/xml/end-user.xhtml ----> tutorial/xml/author.xhtml   (Support=0,14 / Confidence=0,54)
☐ tutorial/xml/author.xhtml ----> tutorial/xml/pages.xhtml   (Support=0,16 / Confidence=0,39)
☐ tutorial/xml/domainmodel.xhtml ----> tutorial/xml/concepts.xhtml   (Support=0,13 / Confidence=0,71)
☐ tutorial/xml/pages.xhtml ----> tutorial/xml/xhtml.xhtml   (Support=0,09 / Confidence=0,46)
✓ tutorial/xml/welcome.xhtml ----> tutorial/xml/readme.xhtml   (Support=0,12 / Confidence=0,14)
☐ tutorial/xml/welcome.xhtml ----> tutorial/xml/author.xhtml   (Support=0,16 / Confidence=0,19)
✓ tutorial/xml/welcome.xhtml ----> tutorial/xml/install.xhtml   (Support=0,49 / Confidence=0,57)
☐ tutorial/xml/install.xhtml ----> tutorial/xml/ahacontext.xhtml   (Support=0,17 / Confidence=0,29)
☐ tutorial/xml/install.xhtml ----> tutorial/xml/environment.xhtml   (Support=0,31 / Confidence=0,53)
☐ tutorial/xml/ahacontext.xhtml ----> tutorial/xml/compile.xhtml   (Support=0,10 / Confidence=0,33)
☐ tutorial/xml/ahacontext.xhtml ----> tutorial/xml/firstconfig.xhtml   (Support=0,16 / Confidence=0,55)
☐ tutorial/xml/environment.xhtml ----> tutorial/xml/install.xhtml   (Support=0,09 / Confidence=0,25)
```

Fig. 4. Recommendation links window.

Finally, the selected recommendations are saved into the AHA! system in a XML file as in the following example file:

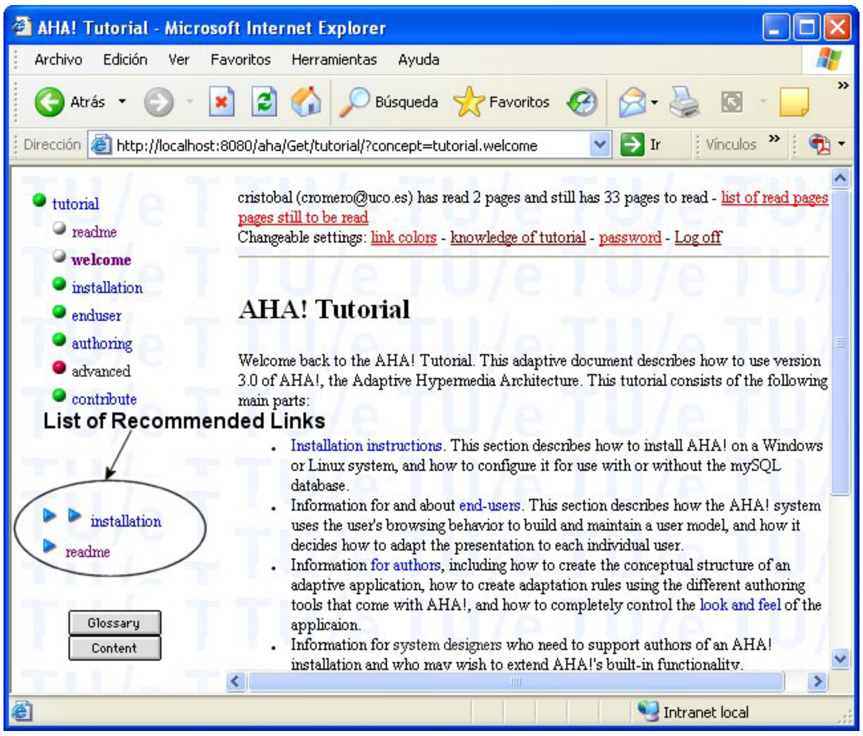
```
<?xml version="1.0" encoding="UTF-8"?>
<RecommendedLinks>
 <Concept name="tutorial.welcome">
   <Recommendation text="installation"
    autogenerated="true" group="all" color="blue"
    interest="57">tutorial.installation</Recommendation>
   <Recommendation text="readme"
    autogenerated="true"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

We can see in the previous example file that there two recommended links to 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 meaning of the attributes are: “text” (text of the hyperlink, by default 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 a specific cluster), “color” (color used by the a triangular image, for example: blue color for all students and red color for clusters), “interest” (the confidence value of the rule).

4.2 Recommendation Engine

We have developed our recommendation engine as another AHA! View class [7], just like the other Views such as MainView, TOCView, ConceptbarView, etc. So, in order for a course to be able to use the new RecommendedLinksView, it is necessary to add it in the corresponding LayoutConfig.xml file of the course. Then, when a student logs in to the AHA! Course, the recommender engine is activated each time that the student visits a web page (concept).

The recommendation engine considers the active students 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 [16] 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 aren't clusters) are used to match the current web page (concept) in order to obtain the current list of recommended links.


AHA! Tutorial - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás
Búsqueda
Favoritos
Dirección http://localhost:8080/aha/Get/tutorial/?concept=tutorial.welcome
Ir
Vínculos
tutorial
readme
welcome
installation
enduser
authoring
advanced
contribute
List of Recommended Links
installation
readme
Glossary
Content
Intranet local

cristobal (cromero@uco.es) has read 2 pages and still has 33 pages to read - list of read pages pages still to be read
Changeable settings: link colors - knowledge of tutorial - password - Log off

AHA! Tutorial

Welcome back to the AHA! Tutorial. This adaptive document describes how to use version 3.0 of AHA!, the Adaptive Hypermedia Architecture. This tutorial consists of the following main parts:

- Installation instructions. This section describes how to install AHA! on a Windows or Linux system, and how to configure it for use with or without the mySQL database.
- Information for and about end-users. This section describes how the AHA! system uses the user's browsing behavior to build and maintain a user model, and how it decides how to adapt the presentation to each individual user.
- Information for authors, including how to create the conceptual structure of an adaptive application, how to create adaptation rules using the different authoring tools that come with AHA!, and how to completely control the look and feel of the applicaion.
- Information for system designers who need to support authors of an AHA! installation and who may wish to extend AHA!'s built-in functionality.

Fig. 5. AHA! Tutorial with recommended links added.

In Figure 5 we can see the interface of the AHA! tutorial with a list of recommended links 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 [4] the list of recommended links. First, we only show the links that the current student matches on the current page. Next, the 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 what data have been used to obtain them (blue for all the data, green to a specific data cluster) and 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

The data used in this study are real data collected from the on-line AHA! Tutorial (http://aha.win.tue.nl/tutorial/) that consists of 34 web pages or concepts. Although we have the usage data of about two hundred users available, we have selected only a group of good users (users who read a significant part of the tutorial). We have used a total number of 78 students with 118 sessions and 684 records in total. These students are mainly TU/E (Eindhoven University of Technology) students taking a course in adaptive hypermedia and some other Internet users interested in the AHA! system. 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.

In the first experiment, we have executed the three available sequential mining algorithms (AprioriAll [2], GSP [24] and PrefixSpan [17]), using all the data 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).

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

	Min.Sup.=0.3	**Min.Sup.=0.15**	**Min.Sup.=0.07**	**Min.Sup.=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

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 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 the best number of clusters with our data. We have executed the k-means algorithm [14] 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.

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

No clusters	**All data**				
Number of st./se./r.	78/118/684				
Cluster K = 2	**Cluster 1**	**Cluster 2**			
Number of st./se./r.	48/60/383	30/58/301			
Cluster K = 3	**Cluster 1**	**Cluster 2**	**Cluster 3**		
Number of st./se./r.	18/34/120	32/45/299	28/39/265		
Cluster K = 4	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	
Number of st./se./r.	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 st./se./r.	14/30/106	12/25/81	6/14/40	4/12/26	42/47/449

In our problem of grouping students in 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 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). 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 PrefixSpan algorithm varying the minimum support threshold (from 0.3 to 0.1),on one hand using all data, and on the other hand the same algorithm using the data obtained from k-means algorithm for 2 and 3 clusters.

Table 3. Number/Average Support/ Average Confidence of rules discovered using PrefixSpan over all data and over different number of clusters.

	Min.Sup.=0.3	**Min.Sup.=0.15**	**Min.Sup.=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

We can see in Table 3 that the number of rules discovered 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 to basic architecture but with higher values of confidence and support. Finally, in order to see if there are differences in the rules obtained from data clusters we are going to show and describe some examples of rules discovered using two data clusters (k=2) and the PrefixSpan algorithm (Min.Sup.=0.15).

Table 4. Examples of rules discovered.

Num	**Antecedent =>**	**Consequence**	**Support**	**Confidence**	**Cluster**
1	readme	install	0.23	0.41	1
2	domainmodel	concept	0.25	0.60	2
3	author	pages	0.27	0.32	1
	author	pages	0.21	0.42	2
4	welcome	install	0.48	0.63	1
	welcome	install	0.40	0.52	2

We can see in Table 4 that there are some rules that only appear in one cluster (rule 1 and 2), and there are other rules that appear in both clusters (rule 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 all the AHA! Tutorial (its centroid has NumberOfPagesVisited=16 and AverageLevelOfKnowledge=99). Rule number 1 shows that sporadic students go from "readme" web page to "install" web page (these students are looking for some specific question about the AHA! installation). 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 type of students go from the "author" web page to the "pages" web page, but it 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 (they are looking for some specific question about the AHA! installation) than active students.

6 Conclusions and Future Work

In this paper, we have described a personalized recommender system that uses web mining to recommend the next links to visit in an AIWBES. We have developed a specific mining tool and a recommender engine in order to help the teacher to carry out the web mining process. Although we have integrated the tools in the AHA! system [7] with minor modifications (mainly to handle the file and data format and to communicate with the engine of the system), it can in principle also be used in other web-based educational systems. We have carried out several experiments with real user data from the on-line AHA! tutorial in order to show the performance of the implemented algorithms. And we have shown the suitability of an advanced recommender system that uses the clustering and sequential pattern mining algorithms together to discover personalized recommendation links.

In the future, we want to carry out more experiments with a still larger number of students and using more information about the students' profiles for doing clustering. We can also integrate other sequence mining algorithms [10] such as SPADE, FreeSpan, CloSpan and PSP, and other clustering algorithm without requiring the user to specify any parameter. We plan to evaluate the quality of the recommendations based on feedback from students as well as on results using a testing set of data. Finally, it would be very useful to develop a real-time feedback loop between data mining and the recommendation system. We can use, for example, intelligent agents for doing on-line data mining automatically and for communicating with the recommender systems. In this way the system could work completely autonomously. The agents can mine data only when they detect enough volume of new data. And the authors do not have to preprocess and apply mining algorithms, they only have to supervise the new recommender links if they want.

Acknowledgments. The authors[1] gratefully acknowledge the financial subsidy provided by the Spanish Department of Research under TIN2005-08386-C05-02 Project.

References

1. Agrawal R., Imielinski, T., Swami, A.N.: Mining Association Rules between Sets of Items in Large Databases. In Proceeding SIGMOD (1993) 207-16.
2. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In Proceedings of the Eleventh International Conference on Data Engineering (1995) 3-14.
3. Alcalá, J., del Jesús, M.J., Garrell, J.M., Herrera, F., Hervás, C., Sánchez, L.: Proyecto KEEL: Desarrollo de una Herramienta para el Análisis e Implementación de Algoritmos de Extracción de Conocimiento Evolutivos. Tendencias de la Minería de Datos en España, Eds. J. Giraldez, J.C. Riquelme, J.S. Aguilar (2004) 413-423.
4. Brusilovsky, P., Peylo, C.: Adaptive and Intelligent Web-based Educational Systems. International Journal of Artificial Intelligence in Education. 13 (2003) 156–169.
5. Clementine. (2007). http://www.spss.com/clementine/
6. DBMiner. (2007) http://www.dbminer.com
7. De Bra, P., Calvi, L.: AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia. 4 (1998) 115-139.
8. Farzan, R., Brusilovsky, P.: Social Navigation Support in a Course Recommendation System. In proceedings of 4th International Conference on Adaptive Hypermedia and Adaptive Web-based Systems. Dublin, (2006). 91-100.
9. García, E., Romero, C., Ventura, S., Castro, C.: Using rules discovery for the continuous improvement of e-learning courses. In International Conference Intellligent Data Engineering and Automated Learning. Burgos, Spain (2006) 887-895.
10. Han, J., Pei, J., Yan, X.: Sequential Pattern Mining by Pattern-Growth: Principles and Extensions. StudFuzz 180 (2005) 183–220.

11. Henze, N., Nejdl, W. : Adaptation in open corpus hypermedia. International Journal of Artificial Intelligence in Education, 12:4 (2001) 325-350.
12. Ishikawa, H., Ohta, M., Yokoyama, S., Nakayama, J., Katayama K.: On the Effectiveness of Web Usage Mining for Page Recommendation and Restructuring. In Proceeding of Web, Web-Services, and Database Systems (2002) 253-267.
13. Jain, A.K., Murty M.N., and Flynn P.J.: Data Clustering: A Review, ACM Computing Surveys, 31:3 (1999) 264-323.
14. Li, J., Zaïane, O.: Combining Usage, Content, and Structure Data to Improve Web Site Recommendation. In Proceedings of International Conference on e-commerce and Web Technologies (2004) 305–315.
15. Lu, J.: Personalized E-learning Material Recommender System. In Proceedings of International Conference on Information Technology for Application (2004) 374–379.
16. MacQueen, J. B.: Some Methods for classification and Analysis of Multivariate Observations. In Proceedings of of 5-th Berkeley Symposium on Mathematical Statistics and Probability, (1967) 281-297.
17. Mobasher, B.: Data Mining for Personalization. In The Adaptive Web: Methods and Strategies of Web Personalization, Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Springer-Verlag, Berlin Heidelberg (2007) 1-46.
18. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. Hsu, M.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In Proceedings of the Seventeenth International Conference on Data Engineering (2001) 2215-224.
19. Ksristofic, A.: Recommender System for Adaptive Hypermedia Applications. In Proceeding of Informatics and Information Technology Student Research Conference, Bratislava (2005) 229-234.
20. Romero, C., Ventura, S.: Educational Data Mining: a Survey from 1995 to 2005. Expert Systems with Applications. Elsevier 1:33 (2007) 135-146.
21. Romero, C., Ventura, S.: Data mining in e-learning. Wit Press (2006).
22. Romero, C., Ventura, S., Bra, P.D.: Knowledge discovery with genetic programming for providing feedback to courseware author. User Modeling and User-Adapted Interaction: The Journal of Personalization Research, 14 :5 (2004) 425–464.
23. Schafer, J.B.: The application of data-mining to recommender systems. In J. Wang (Ed.), Encyclopedia of data warehousing and mining. Hershey, PA. Idea Group (2005) 44-48.
24. Shen, L.P., Shen, R.M.: Learning Content Recommendation Service Based-on Simple Sequencing Specification. In Proceedings of Advanced in Web-based Learning (2004) 363-370.
25. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In Proceedings International Conference on Extending Database Technology (1996) 3-17.
26. Tiffany, Y.T., Gordon M.: Smart Recommendation for an Evolving E-Learning System. In Workshop on Technologies for Electronic Documents for Supporting Learning, Australia (2003) 699-710.
27. Wang, F.H., Shao, H.M.: Effective personalized recommendation based on time-framed navigation clustering and association mining. Expert System with Applications. Elsevier 27 (2004) 365-377.
28. Weber, G., Brusilovsky, P.: ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education. 12:4 (2001) 351-384.
29. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2005).
30. Zaïane, O.: Building A Recommender Agent for e-Learning Systems. In Proceedings of the International Conference in Education, New Zealand (2002) 55-59.